Binary Context-Free Grammars

Sherzod Turaev ^{1,*} , Rawad Abdulghafor ² , Ali Amer Alwan ² , Ali Abd Almisreb ³
and Yonis Gulzar ⁴

¹ Department of Computer Science & Software Engineering, College of Information Technology, United Arab Emirates University, Al Ain 15551, UAE

² Department of Computer Science, Faculty of Information and Communication Technology, International Islamic University Malaysia, Gombak, Selangor 53100, Malaysia; rawad@iiu.edu.my (R.A.); aliamer@iiu.edu.my (A.A.A.)

³ Faculty of Engineering and Natural Sciences, International University of Sarajevo, 71210 Sarajevo, Bosnia and Herzegovina; aalmisreb@ius.edu.ba

⁴ Department of Management Information Systems, King Faisal University, Al-Ahsa 31982, Saudi Arabia; ygulzar@kfu.edu.sa

* Correspondence: sherzod@uaeu.ac.ae; Tel.: +971-58-567-6013

Received: 17 June 2020; Accepted: 21 July 2020; Published: 24 July 2020

Abstract: A *binary grammar* is a relational grammar with two nonterminal alphabets, two terminal alphabets, a set of pairs of productions and the pair of the initial nonterminals that generates the binary relation, i.e., the set of pairs of strings over the terminal alphabets. This paper investigates the binary context-free grammars as *mutually controlled grammars*: two context-free grammars generate strings imposing restrictions on selecting production rules to be applied in derivations. The paper shows that binary context-free grammars can generate matrix languages whereas binary regular and linear grammars have the same power as Chomskyan regular and linear grammars.

Keywords: formal language; binary grammar; context-free grammar; matrix grammar; relational grammar; computation power; chomsky hierarchy

1. Introduction

A “traditional” phrase-structure grammar (also known as a *Chomskyan grammar*) is a generative computational mechanism that produces strings (words) over some alphabet starting from the initial symbol and sequentially applying production rules that rewrite sequences of symbols [1–3]. According to the forms of production rules, phrase-structure grammars and their languages are divided into four families: regular, context-free, context-sensitive, and recursively enumerable [4,5].

Regular and context-free grammars, which have good computational and algorithmic properties, are widely used in modeling and studying of phenomena appearing in linguistics, computer science, artificial intelligence, biology, etc. [6,7]. However, many complex structures such as *duplication* (ww), *multiple agreements* ($a^n b^n c^n$) and *crossed agreements* ($a^n b^m c^n d^m$) found in natural languages, programming languages, molecular biology, and many other areas cannot be represented by context-free grammars [6]. Context-sensitive grammars that can model these and other “non-context-free” structures are too powerful in order to be used in applications [5,8]. In addition many computational problems related to context-sensitive grammars are undecidable, and known algorithms for decidable problems concerning to these grammars have exponential complexities [4–6,9].

An approach to overcome this problem is to define “in-between” grammars that are powerful than context-free grammars but have similar computational properties. *Regulated* grammars are such types of grammars that are defined by adding control mechanisms to underlying context-free grammars in order to select specific strings for their languages [5]. Several variants of regulated

grammars, such as matrix, programmed, ordered, context-conditional, random-context, tree-controlled grammars, have been defined according to control mechanisms used with the grammars [5,6,8]. Regulated grammars are classified into two main categories: rule-based regulated grammars that generate their languages under various production-related restrictions, and context-based regulated grammars that produce their languages under different context-related restrictions [10].

Though this type of the classification of regulated grammars allows us to understand the nature of restrictions imposed on the grammars, it does not clarify the role of control mechanisms from the aspect of the computational power. If we observe the control mechanisms used in both categories of regulated grammars, we can see that they consist of two parts [6]: (1) a “regular part”, which is represented by a regular language of the labels of production rules (in rule-based case) or a regular language over the nonterminal and/or terminal alphabets (in context-based case) and (2) an “irregular part”, which is represented by appearance checking (in rule-based case) and forbidding context (in context-based case), provides an additional power to the regular part.

If a regular part is solely used, the regulated grammars can generate small subsets of context-sensitive languages. On the other hand, if the both of regulations are used, then most regulated grammars generate all context-sensitive languages [6,10]. At this point, we again return to the same computational problems related to context-sensitive grammars, which we discussed early. Thus, we need to consider such regulation mechanisms that can grant to extend the family of context-free languages only to required ranges that cover necessary aspects of modeled phenomena. One of the possibilities to realize this idea can be the combination of several regular control mechanisms. One can consider a matrix of matrices or a matrix conditional context as combined regulation mechanisms. For instance, paper [11] studied simple-semi-conditional versions of matrix grammars with this approach. The problem in this case is that the combinations of such control mechanisms are, firstly, not natural and, secondly, they are too complex.

We propose, as a solution, an idea of imposing multiple regulations on a context-free grammar without combining them. We realize this idea by using *relational grammars*. A relational grammar is an n -ary grammar, i.e., a system of n terminal alphabets, n nonterminal alphabets, a set of n -tuples of productions and the initial n -tuple of nonterminals that generates the language of relations, i.e., tuples of strings over the terminal alphabets. On the other hand, a relational grammar can be considered of a system of n grammars in which each grammar generates own language using the corresponding productions enclosed in n -tuples. Thus, we can redefine a relational grammar as a system of *mutually controlled n grammars* where the grammars in relation generate their languages imposing restrictions on the applications of productions of other grammars. If we specify one grammar as the main, the other grammars in the system can be considered to be the regulation mechanisms controlling the generative processes of the main grammar.

This work is a preliminary work in studying mutually controlled grammars. In this paper, we define *binary context-free grammars* and study their generative power. We show that even mutually controlled two grammars can be as powerful as matrix grammars or other regulated grammars without appearance checking or forbidding context.

The paper is organized as follows. Section 2 surveys on regulated, parallel and relational grammars that are related to the introduced grammars. Section 3 contains necessary notions and notations used throughout the paper. Section 4 defines binary strings, languages and grammars. Section 5 introduces synchronized normal forms for binary grammars and shows that for any binary context-free (regular, linear) grammar there exists an equivalent binary grammar in synchronized form. Section 6 investigate the generative powers of binary regular, linear and context-free grammars. Section 7 discusses the results of the paper and the power of mutually controlled grammars, and indicates to possible topics for future research.

2. Regulated, Parallel and Relational Grammars

In this section, we briefly survey some variants of regulated grammars with respect to the control mechanisms associated with them, parallel grammars as well as relational grammars, which are related to the introduced mutually controlled grammars.

The purpose of regulation is to restrict the use of the productions in a context-free grammar to select only specific terminal derivations successful hence to obtain a subset of the context-free language generated in usual way. Various regulation mechanisms used in regulated grammars can be classified into general types by their common features.

Control by prescribed sequences of production rules where the sequence of productions applied in a derivation belong to a regular language associated with the grammar:

- *matrix grammars* [12]—the set of production rules is divided into matrices and if the application of a matrix is started, a second matrix can be started after finishing the application of the first one, as well as the rules have to be applied in the order given a matrix;
- *vector grammars* [13]—in which a new matrix can be started before finishing those which have been started earlier;
- *regularly controlled grammars* [14]—the sequence of production rules applied in a derivation belong to a given regular language associated with the grammar.

Control by computed sequences of production rules where a derivation is accompanied by a computation, which selects the allowed derivations:

- *programmed grammars* [15]—after applying a production rule, the next production rule has to be chosen from its success field, and if the left hand side of the rule does not occur in the sentential form, a rule from its failure field has to be chosen;
- *valence grammars* [16]—where with each sentential form an element of a monoid is associated, which is computed during the derivation and derivations where the element associated with the terminal word is the neutral element of the monoid are accepted.

Control by context conditions where the applicability of a rule depends on the current sentential form and with any rule some restrictions are associated for sentential forms which have to be satisfied in order to apply the rule:

- *random context grammars* [17]—the restriction is the belonging to a regular language associated with the rule;
- *conditional grammars* [18]—the restriction to special regular sets;
- *semi-conditional grammars* [19]—the restriction to words of length one in the permitting and forbidden contexts;
- *ordered grammars* [18]—a production rule can be applied if there is no greater applicable production rule.

Control by memory where with any nonterminal in a sentential form, its derivation is associated:

- *indexed grammars* [20]—the application of production rules gives sentential forms where the nonterminal symbols are followed by sequences of indexes (*stack* of special symbols), and indexes can be erased only by rules contained in these indexes but erasing of the indexes is done in reverse order of their appearance.

Control by external mechanism where a mechanism used to select derivations does not belong to the grammar:

- *graph-controlled grammars* [21,22]—the sequence of productions applied in a derivation to obtain a string corresponds to a path, whose nodes represent the production rules, in an associated bicolored digraph;

- *Petri net controlled grammars* [23,24]—the sequence of productions used to obtain a string of the language of a grammar corresponds to a firing sequence of transitions, which are labeled by the productions, from the initial marking to a final marking.

Parallelism is another nontraditional approach used with grammars where, instead of rewriting a single symbol in each derivation step, several symbols can be rewritten simultaneously. There are two main variants of parallel mechanisms associated with grammars. The first is **total parallelism**, which is used in the broad varieties of (Deterministic Extended Tabled Zero-Sided) Lindenmayer systems [5,25] where all symbols of strings including terminals are in each step rewritten by productions. The second is **partial parallelism** where all or some nonterminal symbols (not terminal symbols) are written in each step of the derivations:

- *absolutely parallel grammars* [26]—all nonterminals of the sentential form are rewritten in one derivation step;
- *Indian parallel grammars* [27]—all occurrences of one letter are replaced (according to one rule);
- *Russian parallel grammars* [28]—which combines the context-free and Indian parallel feature;
- *scattered context grammars* [29]—in which only a fixed number of symbols can be replaced in a step but the symbols can be different;
- *concurrently controlled grammars* [30]—the control over a parallel application of the productions is realized by a Petri net with different parallel firing strategies.

Another perspective in using the notion of parallelism with grammars is a **grammar system**, which is a system of several phrase-structure grammars with own axioms, symbols and rewriting productions that can work simultaneously and generate own strings. One of such grammar systems is a *parallel communicating grammar system* [31,32], where the grammars start from separate axioms, work parallelly rewriting their own sentential forms, and also communicate with each other by request. The language of one distinguished grammar in the system is considered the language of the system.

A *relational grammar* (an *n-ary grammar*) can be considered to be another type of grammar systems consisting of several grammars that work by applying productions synchronously or asynchronously [33,34]. More precisely, an *n-ary grammar* (where *n* is a positive integer) is a system of *n* terminal alphabets, *n* nonterminal alphabets, a set of productions and an initial *n*-tuple of nonterminals. Each production is an *n*-tuple of common productions or empty places. An *n-ary grammar* generates the language of relations, i.e., *n*-tuples of strings over the terminal alphabets. Work [33] showed that classes of languages generated by relational grammars forms a hierarchy between the family of context-free languages and the family of context-sensitive languages. Paper [34] studied closure, projective and other properties of relational grammars, and generalized the Chomsky's classification for *n-ary grammars*. Several other papers [35–40] also investigated the properties of relational grammars and applied in solving problems appeared in natural and visual language processing.

3. Notions and Notations

Throughout the paper, we assume that the reader is familiar with the basic concepts and results of the theory of formal languages, Petri nets and relations; for details we refer to [4,9,41] (formal languages, automata, computation), [6,10] (regulated rewriting systems), [42,43] (Petri nets), [23,24,44] (Petri net controlled grammars), and [33,34,45–48] (finitary relations). Though, in this section, we recall all necessary notions and notations that are important for understanding this paper.

Basic conventions: the inclusion is denoted by $\subseteq$ and the strict (proper) inclusion is denoted by $\subset$. The symbol $\emptyset$ denotes the empty set. The powerset of a set X is denoted by $\mathcal{P}(X)$, while its cardinality is denoted by $|X|$. An ordered sequence of elements a, b is called a *pair* and denoted by (a, b) . Two pairs (a_1, a_2) and (b_1, b_2) are *equal* iff $a_1 = b_1$ and $a_2 = b_2$. Let X, Y be sets. The set of all pairs (a, b) , where $a \in X$ and $b \in Y$, is called the *Cartesian product* of X and Y , and denoted by $X \times Y$. Then, $X \times X = X^2$. A *binary relation* on sets X, Y is a subset of the Cartesian product $X \times Y$.

Strings, Languages and Grammars

We first recall the fundamental concepts of the formal language theory such as an *alphabet*, a *string* and a *language* from [41]:

Definition 1. An *alphabet* is a nonempty set of abstract symbols.

Definition 2. A *string* (or a *word*) over an alphabet Σ is a finite sequence of symbols from Σ . The sequence of zero symbols is called the *empty string*, and denoted by λ . The set of all strings over Σ is denoted by Σ^* . The set $\Sigma^* - \{\lambda\}$ is denoted by Σ^+ .

Definition 3. A subset of Σ^* is called a *language*.

Definition 4. The number of the occurrences of symbols in $w \in \Sigma^*$ is called its *length* and denoted by $|w|$. The number of occurrences of a symbol x in a string w is denoted by $|w|_x$.

Example 1. Let $\Sigma = \{a, b, c\}$ be an alphabet. Then, $w = aaabbbcc$ is a string over Σ where $|w| = 8$ and $|w|_a = |w|_b = 3, |w|_c = 2$. We can notice that w belongs to $L = \{a^n b^n c^m \mid n \geq 1, m \geq 1\}$, which is a language over Σ .

Next, we cite the definitions of context-free, matrix grammars and related notations which are more detailed given in [4,6].

Definition 5. A *context-free grammar* is a quadruple $G = (V, \Sigma, S, R)$ where V and Σ are disjoint alphabets of nonterminal and terminal symbols, respectively, $S \in V$ is the start symbol and $R \subseteq V \times (V \cup \Sigma)^*$ is a finite set of (production) rules. Usually, a rule (A, x) is written as $A \rightarrow x$. A rule of the form $A \rightarrow \lambda$ is called an *erasing rule*, and a rule of the form $A \rightarrow x$, where $x \in \Sigma^*$, is called *terminal*.

Definition 6. Let $G = (V, \Sigma, S, R)$ be a context-free grammar. If $R \subseteq V \times \Sigma^*(V \cup \{\lambda\})$, then G is called *regular*, and if $R \subseteq V \times \Sigma^*(V \cup \{\lambda\})\Sigma^*$, then it is called *linear*.

The families of regular, linear and context-free languages are denoted by $\mathcal{L}(\text{REG})$, $\mathcal{L}(\text{LIN})$ and $\mathcal{L}(\text{CF})$, respectively.

Definition 7. Let $G = (V, \Sigma, S, R)$ be a context-free grammar.

- The string $x \in (V \cup \Sigma)^+$ directly derives $y \in (V \cup \Sigma)^*$, written as $x \rightarrow y$, if and only if there is a rule $r = A \rightarrow \alpha \in R$ such that $x = x_1 A x_2$ and $y = x_1 \alpha x_2$.
- The reflexive and transitive closure of the relation $\Rightarrow$ is denoted by $\Rightarrow^*$.
- A derivation using the sequence of rules $\tau = r_1 r_2 \cdots r_n$ is denoted by $\xRightarrow{\tau}$ or $\xRightarrow{r_1 r_2 \cdots r_n}$.
- The language generated by a grammar G is defined by $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$.

Example 2. $G_1 = (\{S, A, B\}, \{a, b, c\}, S, R)$ where R contains the productions:

$$r_0 : S \rightarrow AB, r_1 : A \rightarrow aAb, r_2 : A \rightarrow ab, r_3 : B \rightarrow cB, r_4 : B \rightarrow c$$

is a context-free grammar, and it generates the language L in Example 1.

Definition 8. A *matrix grammar* is a quadruple $G = (V, \Sigma, S, M)$ where V, Σ, S are defined as for a context-free grammar, M is a finite set of matrices which are finite strings over a set of context-free rules (or finite sequences of context-free rules). The language generated by a matrix grammar G is defined by $L(G) = \{w \in \Sigma^* \mid S \xRightarrow{\pi} w \text{ and } \pi \in M^*\}$.

Example 3. $G' = (\{S, A, B\}, \{a, b, c\}, S, M)$ where M contains the matrices:

$$m_0 : (S \rightarrow AB), m_1 : (A \rightarrow aAb, B \rightarrow cB), m_2 : (A \rightarrow ab, B \rightarrow c)$$

is a matrix grammar, and it generates the language $\{a^n b^n c^n \mid n \geq 1\}$.

The family of languages generated by matrix grammars is denoted by $\mathcal{L}(\text{MAT})$.

Lastly, we retrieve the notions of a Petri net, a context-free Petri net and a Petri net controlled grammar from [23,24,44].

Definition 9. A Petri net is a construct $N = (P, T, F, \phi)$ where P and T are disjoint finite sets of places and transitions, respectively, $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, $\phi : (P \times T) \cup (T \times P) \rightarrow \{0, 1, 2, \dots\}$ is a weight function, where $\phi(x, y) = 0$ for all $(x, y) \in ((P \times T) \cup (T \times P)) - F$.

A Petri net can be represented by a bipartite directed graph with the node set $P \cup T$ where places are drawn as circles, transitions as boxes and arcs as arrows with labels $\phi(p, t)$ or $\phi(t, p)$. If $\phi(p, t) = 1$ or $\phi(t, p) = 1$, the label is omitted. A mapping $\mu : P \rightarrow \{0, 1, 2, \dots\}$ is called a marking. For each place $p \in P$, $\mu(p)$ gives the number of tokens in p .

Definition 10. A context-free Petri net (in short, a cf Petri net) with respect to a context-free grammar $G = (V, \Sigma, S, R)$ is a tuple $N = (P, T, F, \phi, \beta, \gamma, \iota)$ where

- (1) (P, T, F, ϕ) is a Petri net;
- (2) labeling functions $\beta : P \rightarrow V$ and $\gamma : T \rightarrow R$ are bijections;
- (3) there is an arc from place p to transition t if and only if $\gamma(t) = A \rightarrow \alpha$ and $\beta(p) = A$. The weight of the arc (p, t) is 1;
- (4) there is an arc from transition t to place p if and only if $\gamma(t) = A \rightarrow \alpha$ and $\beta(p) = x$ where $x \in V$ and $|\alpha|_x > 0$. The weight of the arc (t, p) is $|\alpha|_x$;
- (5) the initial marking ι is defined by $\iota(\beta^{-1}(S)) = 1$ and $\iota(p) = 0$ for all $p \in P - \{\beta^{-1}(S)\}$.

The following example ([23]) explains the construction of a cf Petri net.

Example 4. Let G_1 be a context-free grammar defined in Example 2. Figure 1 illustrates a cf Petri net N with respect to the grammar G_1 .

Definition 11. A Petri net controlled grammar is a tuple $G = (V, \Sigma, S, R, N, \gamma, M)$ where V, Σ, S, R are defined as for a context-free grammar and the construct $N = (P, T, F, \phi, \iota)$ is a Petri net, $\gamma : T \rightarrow R \cup \{\lambda\}$ is a labeling function and M is a set of final markings. The language generated by a Petri net controlled grammar G , denoted by $L(G)$, consists of all strings $w \in \Sigma^*$ such that there is a derivation $S \xrightarrow{r_1 r_2 \dots r_k} w$ and an occurrence sequence $v = t_1 t_2 \dots t_s$ which is successful for M such that $r_1 r_2 \dots r_k = \gamma(t_1 t_2 \dots t_s)$.

The family of languages generated by Petri net controlled grammars is denoted by $\mathcal{L}(\text{PN})$.

The hierarchical relationships of the language families defined above are summarized as follows.

Theorem 1. $\mathcal{L}(\text{REG}) \subset \mathcal{L}(\text{LIN}) \subset \mathcal{L}(\text{CF}) \subset \mathcal{L}(\text{MAT}) = \mathcal{L}(\text{PN})$.

The correctness of inclusions $\mathcal{L}(\text{REG}) \subset \mathcal{L}(\text{LIN}) \subset \mathcal{L}(\text{CF})$ was first shown in [1,3]. The proof of the strict inclusion $\mathcal{L}(\text{CF}) \subset \mathcal{L}(\text{MAT})$ can be found in [41]. The equality $\mathcal{L}(\text{MAT}) = \mathcal{L}(\text{PN})$ was established in [23].

Figure 1. A cf Petri net N associated with the grammar G_1 , where the places are labeled with the nonterminals and the transitions are labeled with the productions in one-to-one manner. Moreover, the input place of each transition corresponds to the left-hand side of the associated production and its output places correspond to the nonterminals in the right-hand side of the production. If a transition does not have output places, then the associated production is terminal.

4. Binary Strings, Languages and Grammars

In this section, we define binary strings, languages and grammars by modifying the n -ary counterparts initially studied in [33,34].

Definition 12. A pair of strings u_1, u_2 , is called an binary string over V and denoted by $u = (u_1, u_2)$. The binary empty string is denoted by (λ, λ) .

Definition 13. A subset L of $(V^*)^2$ is called a binary language.

Definition 14. The concatenation of binary strings $u = (u_1, u_2) \in (V^*)^2$ and $v = (v_1, v_2) \in (V^*)^2$ is defined as $uv = (u_1v_1, u_2v_2)$.

Definition 15. For two binary languages $L_1, L_2 \in (V^*)^2$, their

- union is defined as $L_1 \cup L_2 = \{w \mid w \in L_1 \text{ or } w \in L_2\}$,
- concatenation is defined as $L_1 L_2 = \{uv \mid u \in L_1 \text{ and } v \in L_2\}$.

Definition 16. For $L \in (V^*)^2$, its Kleene star is defined as

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

where $L^0 = \{(\lambda, \lambda)\}$ and $L^i = L^{i-1}L, i \geq 1$.

Definition 17. A binary context-free grammar is a quadruple

$$G = (V_1 \times V_2, \Sigma_1 \times \Sigma_2, (S_1, S_2), R)$$

where

- (1) $V_i, i = 1, 2$, are sets of nonterminal symbols,
- (2) Σ_i , with $V_i \cap \Sigma_i = \emptyset, i = 1, 2$, are sets of terminal symbols,
- (3) $(S_1, S_2) \in V_1 \times V_2$ is the start (initial) pair, and

(4) R is a finite set nonempty set of binary productions (rules).

A binary production is a pair (r_1, r_2) where $r_i, i = 1, 2$, is either empty or it is a context-free production, i.e., $r_i \in V_i \times (V_i \cup \Sigma_i)^*$.

Remark 1. A binary production (r_1, r_2) can be written as $u \rightarrow v$ where $u = (u_1, u_2)$, $v = (v_1, v_2)$ and $u_i \rightarrow v_i$ if $r_i \neq \emptyset$, $u_i = v_i = \lambda$, if $r_i = \emptyset$. For production $r_i = u_i \rightarrow v_i$, to indicate its left-hand side and right-hand side, we also use notations $lhs(r_i)$ and $rhs(r_i)$, i.e., $lhs(r_i) = u_i$ and $rhs(r_i) = v_i$.

Remark 2. In Definition 17, if each nonempty r_i is regular production, then the grammar G is called regular and if each nonempty r_i is linear production, then it is called linear.

Remark 3. We say $G_l = (V_1, \Sigma_1, S_1, R_1)$ and $G_r = (V_2, \Sigma_2, S_2, R_2)$ are left and right grammars with respect to the binary grammar G , respectively, where

$$R_1 = \{r_1 \mid (r_1, r_2) \in R \text{ and } r_1 \neq \emptyset\} \text{ and } R_2 = \{r_2 \mid (r_1, r_2) \in R \text{ and } r_2 \neq \emptyset\}.$$

Definition 18. Let G be a binary context-free grammar. Let u, v be pairs in $(V_1 \cup \Sigma_1)^* \times (V_2 \cup \Sigma_2)^*$. We say that u directly derives v , written as $u \rightarrow v$, if there exist pairs $x, z \in (V_1 \cup \Sigma_1)^* \times (V_2 \cup \Sigma_2)^*$ and a production $y \rightarrow w \in R$ such that $u = xyz$ and $v = xwz$. The reflexive and transitive closure of $\rightarrow$ is denoted by $\rightarrow^*$.

Definition 19. The binary language generated by a binary context-free grammar G is defined as

$$L(G) = \{(w_1, w_2) \in (\Sigma_1^* \times \Sigma_2^*) \mid (S_1, S_2) \rightarrow^* (w_1, w_2)\}.$$

Definition 20. The left and right languages are defined as

$$L_l(G) = \{w_1 \mid (w_1, w_2) \in L(G)\} \text{ and } L_r(G) = \{w_2 \mid (w_1, w_2) \in L(G)\},$$

i.e., the sets of left and right strings in all binary strings of $L(G)$, respectively.

Example 5. Consider the binary grammar

$$G_2 = (\{S_1\} \times \{S_2\}, \{a, b\} \times \{a, b\}, (S_1, S_2), R)$$

where R consists of the following productions

$$(S_1, S_2) \rightarrow (S_1 S_1, a S_2), (S_1, S_2) \rightarrow (\lambda, b S_2), (S_1, S_2) \rightarrow (\lambda, \lambda).$$

It is not difficult to see that, after n steps, the first production produces the pair $(S_1^{n+1}, a^n S_1)$. Then in order to eliminate all S_1 's, we apply the second production n times, and terminate derivation with applying the third production, which generates the pair string $(\lambda, a^n b^n)$. Thus, $L(G_2) = \{(\lambda, a^n b^n) \mid n > 0\}$, and $L_r(G_2) = \{a^n b^n \mid n > 0\} \in \mathcal{L}(\text{CF})$.

Example 6. The grammar

$$G_3 = (\{S_1, A, B, C\} \times \{S_2, X, Y, Z\}, \{a, b, c\}^2, (S_1, S_2), R)$$

where R consists of the productions

$$\begin{aligned} &(S_1 \rightarrow ABC, S_2 \rightarrow X), \\ &(A \rightarrow aA, X \rightarrow Y), \quad (B \rightarrow bB, Y \rightarrow Z), \quad (C \rightarrow cC, Z \rightarrow X), \\ &(A \rightarrow a, X \rightarrow Z), \quad (B \rightarrow b, Z \rightarrow Y), \quad (C \rightarrow c, Y \rightarrow \lambda), \end{aligned}$$

generates the language $L(G_3) = \{(\lambda, a^n b^n c^n) \mid n > 0\}$ where the right language $L_r(G_3) = \{a^n b^n c^n \mid n > 0\} \notin \mathcal{L}(\text{CF})$.

Example 6 illustrates that binary context-free grammars can generate non-context-free languages, which implies that binary context-free grammars are more powerful than the Chomskyan context-free grammars. Thus, binary context-free grammars can be used in studying non-context-free structures such as cross-serial dependencies appearing in natural and programming languages using “context-free” tools such as parsing (derivation) trees.

We denote the families of binary languages generated by binary grammars with left and right grammars of type $X, Y \in \{\text{REG}, \text{LIN}, \text{CF}\}$ by $\mathcal{B}(X, Y)$. We also denote the families of the left and right languages generated by binary grammars by $\mathcal{B}(X)$, $X \in \{\text{REG}, \text{LIN}, \text{CF}\}$.

5. Synchronized Forms for Binary Grammars

In a binary grammar, the derivation of a string in the left or right grammar can *pause* or *stop* while the other still continues because of the rules of the forms $(\emptyset, r_2)$ and $(r_1, \emptyset)$. However, we show that the both first and second derivations by binary context-free grammars can be synchronized, i.e., in each derivation step, some pair of nonempty productions is applied and both derivations stop at the same time.

Definition 21. A binary context-free (regular, linear) grammar G is called *synchronized* if it does not have any production of the form $(\emptyset, r_2)$ or $(r_1, \emptyset)$.

Lemma 1. For every binary context-free grammar, there exist an equivalent synchronized binary context-free grammar.

Proof. Let $G = (V_1 \times V_2, \Sigma_1 \times \Sigma_2, (S_1, S_2), R)$ be a binary context-free grammar. Let

$$R_\emptyset = \{(r_1, r_2) \in R \mid r_1 = \emptyset \text{ or } r_2 = \emptyset\}.$$

We define the binary cf grammar $G' = (V'_1 \times V'_2, \Sigma_1 \times \Sigma_2, (S'_1, S'_2), R')$ where $V'_i = V_i \cup \{X, S'_1, S'_2\}$, $i = 1, 2$, where X, S'_1 and S'_2 are new nonterminals, and

$$\begin{aligned} R' = & (R - R_\emptyset) \\ & \cup \{(X \rightarrow X, r_2) \mid (\emptyset, r_2) \in R\} \\ & \cup \{(r_1, X \rightarrow X) \mid (r_1, \emptyset) \in R\} \\ & \cup \{(S'_1 \rightarrow S_1 X, S'_2 \rightarrow S_2 X), (X \rightarrow \lambda, X \rightarrow \lambda)\}. \end{aligned}$$

Then, the equality $L(G) = L(G')$ is obvious. $\square$

The proof of Lemma 1 cannot be used for showing that there is also a synchronized form for a binary regular or linear grammar. The next lemma illustrates the existence of an equivalent synchronized form for any binary regular grammar too.

Lemma 2. For every binary regular grammar, there exist an equivalent synchronized binary regular grammar.

Proof. Let $G = (V_1 \times V_2, \Sigma_1 \times \Sigma_2, (S_1, S_2), R)$ be a binary regular grammar. Let $R_\emptyset = \{(r_1, r_2) \in R \mid r_1 = \emptyset \text{ or } r_2 = \emptyset\}$ and

$$\begin{aligned} R_1 = & \{(r_1, r_2) \in R \mid \text{rhs}(r_1) \in T_1^*, \text{rhs}(r_2) \in T_2^*\}, \\ R_2 = & \{(r_1, r_2) \in R \mid \text{rhs}(r_1) \in T_1^* N_1, \text{rhs}(r_2) \in T_2^* N_2\}. \end{aligned}$$

The proof of the lemma consists of two parts. First, we replace all binary productions of the form $(A_1 \rightarrow x_1, r_2)$ and $(r_1, A_2 \rightarrow x_2)$ with the productions $(A_1 \rightarrow x_1 E, r_2)$ and $(r_1, A_2 \rightarrow x_2 E)$ where E is a new nonterminal, $x_i \in T_1^* \cup T_2^*$ and $\text{rhs}(r_i) \notin T_1^* \cup T_2^*$, $i = 1, 2$. By this change, the early stop of one of derivations by the left and right grammars is prevented. Let

$$R_3 = \{(r_1, r_2) \in R \mid \text{rhs}(r_1) \in T_1^*, \text{rhs}(r_2) \notin T_2^*\},$$

$$R_4 = \{(r_1, r_2) \in R \mid \text{rhs}(r_1) \notin T_1^*, \text{rhs}(r_2) \in T_2^*\}.$$

We set $V'_1 = V_1 \cup \{E\}$, $V'_2 = V_2 \cup \{E\}$ and

$$R'_3 = \{(A_1 \rightarrow x_1 E, r_2) \mid (A_1 \rightarrow x_1, r_2) \in R_3\},$$

$$R'_4 = \{(r_1, A_2 \rightarrow x_2 E) \mid (r_1, A_2 \rightarrow x_2) \in R_4\}$$

and $R' = (R - (R_3 \cup R_4)) \cup R'_3 \cup R'_4$.

Second, in order to eliminate empty productions, we replace all binary production rules of the form $(A_1 \rightarrow x_1 B_1, \emptyset)$ and $(\emptyset, A_2 \rightarrow x_2 B_2)$ with the pairs of production rules $(A_1 \rightarrow A_r, A_2 \rightarrow A_r)$, $(A_r \rightarrow x_1 B_1, A_r \rightarrow A_2)$ and $(A_1 \rightarrow A_r, A_2 \rightarrow A_r)$, $(A_r \rightarrow A_1, A_r \rightarrow x_2 B_2)$, respectively, where A_r s are new nonterminals. Thus, we define the following sets of new productions:

$$R_5 = \{(A_1 \rightarrow A_r, A_2 \rightarrow A_r), (A_r \rightarrow A_1, A_r \rightarrow x_2 B_2) \mid \\ r = (\emptyset, A_2 \rightarrow x_2 B_2) \in R' \text{ and } (A_1, A_2) \in V'_1 \times V'_2\},$$

$$R_6 = \{(A_1 \rightarrow A_r, A_2 \rightarrow A_r), (A_r \rightarrow x_1 B_1, A_r \rightarrow A_2) \mid \\ r = (A_1 \rightarrow x_1 B_1, \emptyset) \in R' \text{ and } (A_1, A_2) \in V'_1 \times V'_2\}$$

where A_r s are new nonterminal symbols introduced for each production $r = (A_1 \rightarrow \alpha_1, \emptyset)$ or $r = (\emptyset, A_2 \rightarrow \alpha_2)$ in R' . Let

$$V_{R,1} = \{A_r \mid r = (\emptyset, A_2 \rightarrow x_2 B_2) \in R'\}$$

and

$$V_{R,2} = \{A_r \mid r = (A_1 \rightarrow x_1 B_1, \emptyset) \in R'\}.$$

We define the binary regular grammar $G'' = (V''_1 \times V''_2, \Sigma_1 \times \Sigma_2, (S_1, S_2), R'')$ as follows:

$$V''_1 = V'_1 \cup V_{R,1},$$

$$V''_2 = V'_2 \cup V_{R,2},$$

$$R'' = R_1 \cup R_2 \cup R_5 \cup R_6 \cup \{(E \rightarrow \lambda, E \rightarrow \lambda)\}.$$

First, we show that $L(G) \subseteq L(G'')$. Consider a derivation

$$(S_1, S_2) = (w_{1,0}, w_{2,0}) \Rightarrow (w_{1,1}, w_{2,1}) \Rightarrow \cdots \Rightarrow (w_{1,n}, w_{2,n}) = (w_1, w_2)$$

in G where $(w_1, w_2) \in \Sigma_1^* \times \Sigma_2^*$.

For a derivation step $(w_{1,i-1}, w_{2,i-1}) \Rightarrow (w_{1,i}, w_{2,i})$, $1 \leq i \leq n$, the following cases are possible:

Case 1: the right-hand side is obtained by applying a production from $R_1 \cup R_2 \cup \{(E \rightarrow \emptyset, E \rightarrow \emptyset)\}$. Then, the same rule is also applied in the corresponding step in the simulating derivation of G'' .

Case 2: the right-hand side is obtained by applying a production of the form $r = (\emptyset, A_2 \rightarrow \alpha_2)$ or $r = (A_1 \rightarrow \alpha_1, \emptyset)$. Then, the production sequence $(A_1 \rightarrow A_r, A_2 \rightarrow A_r), (A_r \rightarrow A_1, A_r \rightarrow x_2 B_2) \in R_5$

or the production sequence $(A_1 \rightarrow A_r, A_2 \rightarrow A_r), (A_r \rightarrow xB_1, A_r \rightarrow A_2) \in R_6$ is applied in the corresponding step in the simulating derivation of G'' .

Case 3: the right-hand side is obtained by applying a rule of the form $(A_1 \rightarrow x_1, r_2), \text{rhs}(r_2) \neq \emptyset$ or $(r_1, A_2 \rightarrow x_2), \text{rhs}(r_1) \neq \emptyset$. Then, the rule $(A_1 \rightarrow x_1, r_2)$ or $(r_1, A_2 \rightarrow x_2)$ is applied in the corresponding step in the simulating derivation in G'' , and the derivation terminates with applying $(E \rightarrow \emptyset, E \rightarrow \emptyset)$.

The inclusion $L(G'') \subseteq L(G)$ is obvious:

(1) the application of a rule of the form $(A_1 \rightarrow x_1, r_2), \text{rhs}(r_2) \neq \emptyset$, or $(r_1, A_2 \rightarrow x_2), \text{rhs}(r_1) \neq \emptyset$, can be immediately replaced with the pair $(A_1 \rightarrow x_1, r_2)$ or $(r_1, A_2 \rightarrow x_2)$, respectively;

(2) if a rule of the form $(A_1 \rightarrow A_r, A_2 \rightarrow A_r)$ or $(A_1 \rightarrow A_r, A_2 \rightarrow A_r)$ is applied at some derivation step, the only applicable pair of productions then is $(A_r \rightarrow A_1, A_r \rightarrow xB_2)$ or $(A_r \rightarrow xB_1, A_r \rightarrow A_2)$, respectively, since A_r is the unique for each pair of productions. Thus, the sequence of pairs of productions $(A_1 \rightarrow A_r, A_2 \rightarrow A_r)$ and $(A_r \rightarrow A_1, A_r \rightarrow xB_2)$ or $(A_1 \rightarrow A_r, A_2 \rightarrow A_r)$ and $(A_r \rightarrow xB_1, A_r \rightarrow A_2)$ are replaced with $r = (\emptyset, A_2 \rightarrow \alpha_2)$ or $r = (A_1 \rightarrow \alpha_1, \emptyset)$, respectively. $\square$

Using the same arguments of the proof of Lemma 2, one can show that the similar fact also holds for binary linear grammars.

Lemma 3. *For every binary linear grammar, there exist an equivalent synchronized binary linear grammar.*

6. Generative Capacities of Binary Grammars

In this section, we discuss the generative capacities of binary regular, linear and context free grammars.

The following two lemmas immediately follows from the definitions of binary languages.

Lemma 4. $\mathcal{B}(X, Y) = \mathcal{B}(Y, X), X \in \{\text{REG}, \text{LIN}, \text{CF}\}.$

Lemma 5. $\mathcal{B}(\text{REG}) \subseteq \mathcal{B}(\text{LIN}) \subseteq \mathcal{B}(\text{CF}).$

Lemma 6. $\mathcal{L}(\text{REG}) \subseteq \mathcal{B}(\text{REG}), \mathcal{L}(\text{LIN}) \subseteq \mathcal{B}(\text{LIN}), \mathcal{L}(\text{CF}) \subseteq \mathcal{B}(\text{CF}).$

Proof. Let $G = (V, \Sigma, S, R)$ be a context-free (regular, linear) grammar. Then, we define the binary context-free (regular, linear) grammar $G' = (V \times V, \Sigma \times \Sigma, R', (S, S))$ by setting, for each production $r = A \rightarrow \alpha \in R$, the production $r' = (\emptyset, A \rightarrow \alpha)$ in R' . Then, it is not difficult to see that $L(G) = L_r(G')$. In the same way, we can also show that $L(G) = L_l(G')$. $\square$

Now we show that binary regular and linear grammars generate regular and linear languages, respectively.

Lemma 7. $\mathcal{B}(\text{LIN}) \subseteq \mathcal{L}(\text{LIN}).$

Proof. Let $G = (V_1 \times V_2, \Sigma_1 \times \Sigma_2, (S_1, S_2), R)$ be a binary linear grammar. Without loss of generality, we can assume that the grammar G is synchronized. We set $V = \{[A_1, A_2] \mid (A_1, A_2) \in V_1 \times V_2\}$ where $[A_1, A_2], (A_1, A_2) \in V_1 \times V_2$, are new nonterminals, and we define

$$R' = \{[A_1, A_2] \rightarrow x_2[B_1, B_2]y_2 \mid (A_1 \rightarrow x_1B_1y_1, A_2 \rightarrow x_2B_2y_2) \in R\} \\ \cup \{[A_1, A_2] \rightarrow x_2 \mid (A_1 \rightarrow x_1, A_2 \rightarrow x_2) \in R\}.$$

Then, $G' = (V, \Sigma_2, [S_1, S_2], R')$ is a linear grammar, and $L_r(G) = L(G')$ is obvious. Hence, $L_r(G)$ is linear, i.e., $\mathcal{B}(\text{LIN}) \subseteq \mathcal{L}(\text{LIN}).$ $\square$

Corollary 1. $\mathcal{B}(\text{REG}) \subseteq \mathcal{L}(\text{REG})$.

Next, we show that binary context-free grammars are more powerful than Chomskyan context-free grammars.

Lemma 8. $\mathcal{L}(\text{CF}) \subset \mathcal{B}(\text{CF})$.

Proof. By Lemma 6, $\mathcal{L}(\text{CF}) \subseteq \mathcal{B}(\text{CF})$. Let us consider a binary context-free grammar $G_4 = (\{S, C, D, E\} \times \{S, A, B\}, \{a, b, c\}^2, (S, S), R)$ where R consists of the following productions:

- (1) $(S \rightarrow S, S \rightarrow AB),$
- (2) $(S \rightarrow C, A \rightarrow aA), \quad (C \rightarrow S, B \rightarrow aB),$
- (3) $(S \rightarrow D, A \rightarrow bA), \quad (D \rightarrow S, B \rightarrow bB),$
- (4) $(S \rightarrow E, A \rightarrow \lambda), \quad (E \rightarrow a, B \rightarrow \lambda),$

Any successful derivation starts with production (1). For each production sequence $(i), 2 \leq i \leq 4$, if the first production is applied, then, the only applicable production in the derivation is the second one. If after (1), production sequence (4) is applied, then the derivation generates the binary string (a, λ) . Else, after some steps, the derivation results in $(S, wAwB), w \in \{a, b\}^+$, by applying production sequences (2) or/and (3), and then, it terminates by applying production sequence (4). Thus,

$$L(G_4) = \{(a, ww) \mid w \in \{a, b\}^*\}.$$

Since $L_r(G_4) \notin \mathcal{L}(\text{CF})$, we have the strict inclusion $\mathcal{L}(\text{CF}) \subset \mathcal{B}(\text{CF})$. $\square$

Next, we show that binary context-free grammars are at least as powerful as matrix grammars.

Lemma 9. $\mathcal{L}(\text{MAT}) \subseteq \mathcal{B}(\text{CF})$.

Proof. Let $G = (V, \Sigma, S, M)$ be a matrix grammar where M consists of matrices $m_1, m_2, \dots, m_n$ with $m_i : (r_{i,1}, r_{i,2}, \dots, r_{i,k(i)}), 1 \leq i \leq n$. We set the following sets of new nonterminals

$$\begin{aligned} V_1 &= \{Y_i \mid 1 \leq i \leq n\} \cup \{Z_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq k(i)\} \cup \{S_1, X\}, \\ V_2 &= V \cup \{S_2\}, \end{aligned}$$

We construct the following binary productions

- (1) the *start* production: $(S_1 \rightarrow X, S_2 \rightarrow S),$
- (2) the *matrix entry* productions: $(X \rightarrow Y_i, \emptyset), 1 \leq i \leq n,$
- (3) the *matrix processing* productions:

$$(Y_i \rightarrow Z_{i,1}, r_{i,1}), (Z_{i,1} \rightarrow Z_{i,2}, r_{i,2}), \dots, (Z_{i,k(i)-1} \rightarrow Z_{i,k(i)}, r_{i,k(i)})$$

where $1 \leq i \leq n$,

- (4) the *matrix exit* productions: $(Z_{i,k(i)} \rightarrow X, \emptyset), 1 \leq i \leq n.$
- (5) the *terminating* production: $(X \rightarrow \lambda, \emptyset).$

We define the binary context-free grammar $G' = (V_1 \times V_2, \Sigma \times \Sigma, (S_1, S_2), R)$ where R consists of all productions (1)–(5) constructed above.

Claim 1: $L(G) \subseteq L(G')$. Let

$$D : S = w_0 \xRightarrow{m_{i_1}} w_1 \xRightarrow{m_{i_2}} w_2 \cdots \xRightarrow{m_{i_k}} w_k = w \in \Sigma^*$$

be a derivation in G . We construct the derivation D' in G' that simulates D . The derivation starts with the step:

$$(S_1, S_2) \Rightarrow (X, S).$$

Since, m_{i_1} is the first matrix applied in derivation D , the next step in D' is

$$(X, S) \Rightarrow (Y_{i_1}, S).$$

Furthermore,

$$(Y_{i_1}, S) \xrightarrow{(Y_{i_1} \rightarrow Z_{i_1}, r_{i_1})} \dots \xrightarrow{(Z_{i_1, k(i_1)-1} \rightarrow Z_{i_1, k(i_1)}, r_{i_1, k(i_1)})} w_1.$$

By applying $(Z_{i_1, k(i_1)} \rightarrow X, \emptyset)$,

$$w_1 \xrightarrow{(Z_{i_1, k(i_1)} \rightarrow X, \emptyset)} w_1,$$

we return X to the derivation, and then, we can continue the simulation of the application of the matrix m_{i_2} in the same manner. Thus, D' simulates D , and $L(G) \subseteq L(G')$.

Claim 2: $L(G) \subseteq L(G')$. Any successfully terminating derivation in G' starts by applying production $(S_1 \rightarrow X, S_2 \rightarrow S)$, followed by applying $(X \rightarrow Y_i, \emptyset)$ for some $1 \leq i \leq n$. Then only possible productions to be applied are matrix processing productions of the form (3). When the application of the productions in the currently active sequence starts, the productions of another sequence of the form (3) cannot be applied. In order to switch to another sequence of productions of the form (3), the corresponding production of the form (4) must be applied after finishing the application of all productions in the current sequence in the given order. To successfully terminate the derivation, the productions of the forms (4) and (5) must be applied. By construction, each sequence of productions of the form (3) simulates some matrix from G , any successful derivation in G' can be simulated by a successful derivation in G . Thus, $L(G') \subseteq L(G)$. $\square$

The lemma above shows that any matrix language can be generated by a binary grammar where one of its grammars is regular and the other is context-free. Here, the natural question arises whether there is a binary grammar with both grammars are context-free that generates a non-matrix language or not. Next lemma shows that binary context-free grammars can only generate matrix languages even if their both grammars are context-free.

Lemma 10. $\mathcal{B}(\text{CF}) \subseteq \mathcal{L}(\text{MAT})$.

Proof. Let $G = (V_1 \times V_2, \Sigma_1 \times \Sigma_2, (S_1, S_2), R)$ be a binary context-free grammar. Without loss of generality, we assume that G is in a synchronized form. Let $R_2 = \{r_2 \mid (r_1, r_2) \in R\}$. The proof idea is as follows: First, we will construct a context-free Petri net N with respect to G . Second, we define a Petri net controlled grammar G' where the underlying right grammar $G_r = (V_2, \Sigma_2, S_2, R_2)$ is controlled by the Petri net N . Then we show that $L_r(G) = L(G')$, i.e., $\mathcal{B}(\text{CF}) \subseteq \mathcal{L}(\text{PN})$.

Part 1: We construct the cf Petri net $N = (P, T, F, \phi, \beta, \gamma, \iota)$ with respect to the nonterminals of the left grammar and productions of the grammar G by setting its components in the following way:

- (P, T, F, ϕ) is a Petri net;
- the labeling functions $\beta : P \rightarrow V_1$ and $\gamma : T \rightarrow R$ are bijections;
- there is an arc from place p to transition t if and only if $\gamma(t) = (r_1, r_2)$ and $\beta(p) = \text{lhs}(r_1)$. The weight of the arc (p, t) is 1;
- there is an arc from transition t to place p if and only if $\gamma(t) = (r_1, r_2)$ and $\beta(p) = X$ where $X \in V_1$ and $|\text{rhs}(r_1)|_X > 0$. The weight of the arc (t, p) is $|\text{rhs}(r_1)|_X$;
- the initial marking ι is defined by $\iota(\beta^{-1}(S_1)) = 1$ and $\iota(p) = 0$ for all $p \in P - \{\beta^{-1}(S_1)\}$.

Part 2: Using the right grammar $(V_2, \Sigma_2, R_2, S_2)$, we define the PN controlled grammar $G' = (V_2, \Sigma_2, R_2, S_2, N, \eta, M)$ where $N = (P, T, F, \phi, \beta, \gamma, \iota)$ is the cf Petri net defined above, $\eta : T \rightarrow R_2$

is a labeling function and M is a set of final markings. We set $M = \emptyset$ and $\eta(t) = r_2$ if and only if $\gamma(t) = (r_1, r_2)$.

Part 3: Now we show that $L_r(G) = L(G')$. Let

$$D : (S_1, S_2) \xrightarrow{(r_{1,1}, r_{2,1})} (w_{1,1}, w_{2,1}) \xrightarrow{(r_{1,2}, r_{2,2})} \dots \xrightarrow{(r_{1,k}, r_{2,k})} (w_{1,k}, w_{2,k})$$

be a derivation in G where $(w_{1,k}, w_{2,k}) \in \Sigma_1^* \times \Sigma_2^*$. We show that the derivation D can be simulated by the derivation D' in the grammar G' constructed as follows. D' starts with S_2 , and by definition of N , $\iota(\beta^{-1}(S_1)) = 1$, thus, transition $t_1 = \gamma^{-1}((r_{1,1}, r_{2,1}))$ is enabled. In the first step, we obtain

$$D' : S_2 \xrightarrow{r_{2,1}} w_{2,1} \text{ where } \eta(t_1) = r_{2,1}.$$

When transition t_1 occurs the place in N corresponding to each nonterminal in $\text{rhs}(r_{1,1}) = w_{1,1}$ receive the tokens whose number is equal to the number of the occurrence of the nonterminal in $\text{rhs}(r_{1,1})$.

Suppose that for some $1 \leq i < k$, we constructed the first i steps of the derivation D' :

$$D' : S_2 \xrightarrow{r_{2,1}} w_{2,1} \xrightarrow{r_{2,2}} \dots \xrightarrow{r_{2,i}} w_{2,i}$$

with $r_{2,1}r_{2,2} \dots r_{2,i} = \eta(t_1t_2 \dots t_i)$ where $(r_{1,1}, r_{2,1})(r_{1,2}, r_{2,2}) \dots (r_{1,i}, r_{2,i}) = \gamma(t_1t_2 \dots t_i)$, which corresponds to the first i steps of D . By definition, $\gamma(t_i) = (r_{1,i}, r_{2,i})$ and $\eta(t_i) = r_{2,i}$. When transition t_i fires, a token moves from the input place of t_i , that is labeled by the left-hand side of $r_{1,i}$, to its output places, that are labeled by the nonterminals occurring in the right-hand side of $r_{1,i}$. Thus, these nonterminals are also occur in $w_{1,i}$. The next step in D occurs by applying the pair $(r_{1,i+1}, r_{2,i+1})$:

$$(w_{1,i}, w_{2,i}) \xrightarrow{(r_{1,i+1}, r_{2,i+1})} (w_{1,i+1}, w_{2,i+1}).$$

Since production $r_{1,i+1}$ is applicable in the current step, its left-hand side occurs in $w_{1,i}$. It follows that the transition t_{i+1} , $\gamma(t_{i+1}) = (r_{1,i+1}, r_{2,i+1})$, can fire. Consequently, we choose the production $r_{2,i+1}$ with $\eta(t_{i+1}) = r_{2,i+1}$, in D' , and obtain

$$w_{2,i} \xrightarrow{r_{2,i+1}} w_{2,i+1}.$$

The last step in D results in $(w_{1,k}, w_{2,k}) \in \Sigma_1^* \times \Sigma_2^*$ that is obtained by applying the pair $(r_{1,k}, r_{2,k})$. Then, we choose $r_{2,k}$ with $\eta(t_k) = r_{2,k}$ in D' . Since $w_{1,k} \in \Sigma_1^*$, i.e., it does not contain nonterminals, all places of N have no tokens. Thus, $M = \emptyset$. It shows that $L_r(G) \subseteq L(G')$.

Let

$$D' : S_2 \xrightarrow{r_{2,1}} w_{2,1} \xrightarrow{r_{2,2}} \dots \xrightarrow{r_{2,k}} w_{2,k}$$

with $r_{2,1}r_{2,2} \dots r_{2,k} = \eta(t_1t_2 \dots t_k)$. By definition, we immediately obtain $\gamma(t_1t_2 \dots t_k) = (r_{1,1}, r_{2,1})(r_{1,2}, r_{2,2}) \dots (r_{1,k}, r_{2,k})$ for some $(r_{1,i}, r_{2,i}) \in R$, $1 \leq i \leq k$. Then, we can construct the derivation D in the grammar G

$$D : (S_1, S_2) \xrightarrow{(r_{1,1}, r_{2,1})} (w_{1,1}, w_{2,1}) \xrightarrow{(r_{1,2}, r_{2,2})} \dots \xrightarrow{(r_{1,k}, r_{2,k})} (w_{1,k}, w_{2,k}),$$

which shows that $L(G') \subseteq L_r(G)$. Thus, $\mathcal{B}(\text{CF}) \subseteq \mathcal{L}(\text{PN})$. By Theorem 1, $\mathcal{B}(\text{CF}) \subseteq \mathcal{L}(\text{MAT})$. $\square$

We summarize the results obtained above in the following theorem.

Theorem 2.

$$\mathcal{B}(\text{REG}) = \mathcal{L}(\text{REG}) \subset \mathcal{B}(\text{LIN}) = \mathcal{L}(\text{LIN}) \subset \mathcal{L}(\text{CF}) \subset \mathcal{L}(\text{MAT}) = \mathcal{B}(\text{CF}).$$

7. Conclusions

In this paper, we redefined binary grammars as mutually controlled grammars where either grammar in a relation generates own language imposing restriction to the other.

Though binary grammars are asynchronous systems by their definitions, we showed that they can also work in synchronized mode (Lemmas 1–3), i.e., the both grammars in a binary relation generate strings with derivations where the grammars apply some productions in each step, and stop at the same time. This feature of binary grammars allows using one grammar in a relation as a regulation mechanism for the other.

We have studied the generative capacity of binary context-free grammars, and showed that binary regular and linear grammars have the same power as their Chomskyan alternatives, i.e., traditional regular and linear grammars, respectively (Lemmas 6 and 7 and Corollary 1). On the other hand, we have proved that binary context-free grammars are much more powerful than traditional context-free grammars (Lemma 8), that is, they generate all matrix languages even if binary grammars consist of regular and context-free pairs (Lemma 9). Moreover, we established that using context-free grammars as the components of relations does not increase the computational power of binary context-free grammars, i.e., they remain equivalent to matrix grammars (Lemma 10). Using the inclusion hierarchies in Theorem 1 and the results of the paper, we obtained the comparative hierarchy for binary regular, linear and context-free grammars (Theorem 2). We have also illustrated that binary grammars have practical significance: Example 6, and Lemma 8 show that *cross-serial dependencies* such as duplication and multiple agreements—non-context-free syntactical structures appearing in natural and programming languages—can be expressed with binary grammars.

Here, we emphasize that ternary or higher degree relational context-free grammars are more powerful than binary ones, and can be used in modeling “*nested*” *cross-serial dependencies*. Let us assume that a ternary context-free grammar is defined similarly to binary grammars. Then, the reader can convince himself that the following language

$$\{ (a_1^{n_1} b_1^{n_1} c_1^{n_1})^n (a_2^{n_2} b_2^{n_2} c_2^{n_2})^n (a_3^{n_3} b_3^{n_3} c_3^{n_3})^n \mid n, n_1, n_2, n_3 \geq 1 \},$$

the language of *nested mutual agreements*, can be generated by a ternary context-free grammar

$$G = (\{S_1, S_2, S_3\}, \{a_i, b_i, c_i \mid i = 1, 2, 3\}, (S_1, S_2, S_3), R)$$

where R consists of the following tuples of productions:

$$\begin{array}{ll} (S_1 \rightarrow X_1, S_2 \rightarrow X_2, S_3 \rightarrow A_1 A_2 B_1 B_2 C_1 C_2), & (X_1 \rightarrow X_1, X_2 \rightarrow X'_2, A_1 \rightarrow a_1 A_1 b_1), \\ (X_1 \rightarrow X_1, X'_2 \rightarrow X_2, A_2 \rightarrow c_1 A_2), & (X_1 \rightarrow Y_1, X'_2 \rightarrow Y_2, A_2 \rightarrow c_1 A_2), \\ (Y_1 \rightarrow Y_1, Y_2 \rightarrow Y'_2, B_1 \rightarrow a_2 B_1 b_2), & (Y_1 \rightarrow Y_1, Y'_2 \rightarrow Y_2, B_2 \rightarrow c_2 B_2), \\ (Y_1 \rightarrow Z_1, Y'_2 \rightarrow Z_2, B_2 \rightarrow c_2 B_2), & (Z_1 \rightarrow Z_1, Z_2 \rightarrow Z'_2, C_1 \rightarrow a_3 C_1 b_3), \\ (Z_1 \rightarrow Z_1, Z'_2 \rightarrow Z_2, C_2 \rightarrow c_3 C_2), & (Z_1 \rightarrow X_1, Z'_2 \rightarrow X_2, C_2 \rightarrow c_3 C_2), \\ (X_1 \rightarrow X_1, X_2 \rightarrow X'_2, A_1 \rightarrow a_1 b_1), & (X_1 \rightarrow Y_1, X'_2 \rightarrow Y_2, A_2 \rightarrow c_1), \\ (Y_1 \rightarrow Y_1, Y_2 \rightarrow Y'_2, B_1 \rightarrow a_2 b_2), & (Y_1 \rightarrow Z_1, Y'_2 \rightarrow Z_2, B_2 \rightarrow c_2), \\ (Z_1 \rightarrow Z_1, Z_2 \rightarrow Z'_2, C_1 \rightarrow a_3 b_3), & (Z_1 \rightarrow \lambda, Z'_2 \rightarrow \lambda, C_2 \rightarrow c_3). \end{array}$$

The detailed study of higher degree relational grammars as mutually controlled grammars will be the topic of our next investigation.

Author Contributions: Conceptualization, S.T. and A.A.A. (Ali Amer Alwan); methodology, S.T. and R.A.; validation, A.A.A. (Ali Abd Almisreb) and Y.G.; formal analysis, A.A.A. (Ali Abd Almisreb) and Y.G.; investigation, S.T. and R.A.; writing—original draft preparation, S.T. and R.A.; writing—review and editing, S.T. and A.A.A. (Ali Amer Alwan); supervision, S.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the United Arab Emirates University Start-Up Grant 31T137.

Acknowledgments: We would like to thank the anonymous reviewers for their valuable comments and useful remarks about this paper.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study and in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CF	context-free
REG	regular
LIN	linear
MAT	matrix

References

- Chomsky, N. Three models for the description of languages. *IRE Trans. Inf. Theory* **1956**, *2*, 113–124. [\[CrossRef\]](#)
- Chomsky, N. *Syntactic Structure*; Mouton: Gravenhage, The Netherland, 1957.
- Chomsky, N. On certain formal properties of grammars. *Inf. Control* **1959**, *2*, 137–167. [\[CrossRef\]](#)
- Hopcroft, J.; Motwani, R.; Ullman, J. *Introduction to Automata Theory, Languages, and Computation*; Pearson: London, UK, 2007.
- Rozenberg, G.; Salomaa, A. (Eds.) *Handbook of Formal Languages*; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1–3.
- Dassow, J.; Păun, G. *Regulated Rewriting in Formal Language Theory*; Springer: Berlin/Heidelberg, Germany, 1989.
- Păun, G.; Rozenberg, G.; Salomaa, A. *DNA Computing. New Computing Paradigms*; Springer: Berlin/Heidelberg, Germany, 1998.
- Meduna, A.; Soukup, O. *Modern Language Models and Computation. Theory with Applications*; Springer: Berlin/Heidelberg, Germany, 2017.
- Sipser, M. *Introduction to the Theory of Computation*; Cengage Learning: Boston, MA, USA, 2013.
- Meduna, A.; Zemek, P. *Regulated Grammars and Automata*; Springer: Berlin/Heidelberg, Germany, 2014.
- Meduna, A.; Kopeček, T. Simple-Semi-Conditional Versions of Matrix Grammars with a Reduced Regulated Mechanism. *Comput. Inform.* **2004**, *23*, 287–302.
- Abraham, A. Some questions of phrase-structure grammars. *Comput. Linguist.* **1965**, *4*, 61–70. [\[CrossRef\]](#)
- Cremers, A.; Mayer, O. On vector languages. *J. Comp. Syst. Sci.* **1974**, *8*, 158–166. [\[CrossRef\]](#)
- Ginsburg, S.; Spanier, E. Control sets on grammars. *Math. Syst. Theory* **1968**, *2*, 159–177. [\[CrossRef\]](#)
- Rozenkrantz, D. Programmed grammars and classes of formal languages. *J. ACM* **1969**, *16*, 107–131. [\[CrossRef\]](#)
- Păun, G. A new generative device: valence grammars. *Rev. Roum. Math. Pures Appl.* **1980**, *25*, 911–924.
- Cremers, A.; Maurer, H.; Mayer, O. A note on leftmost restricted random context grammars. *Inform. Proc. Lett.* **1973**, *2*, 31–33. [\[CrossRef\]](#)
- Fris, I. Grammars with partial ordering of the rules. *Inform. Control* **1968**, *12*, 415–425. [\[CrossRef\]](#)
- Kelemen, J. Conditional grammars: Motivations, definitions and some properties. In *Proc. Conf. Automata, Languages and Mathematical Sciences*; Peak, I., Szep, J., Eds.; Salgótarján, Hungary, 1984; pp. 110–123.
- Aho, A. Indexed grammars. An extension of context-free grammars. *J. ACM* **1968**, *15*, 647–671. [\[CrossRef\]](#)
- Wood, D. Bicolored Digraph Grammar Systems. *RAIRO Inform. Théorique et Appl. /Theor. Inform. Appl.* **1973**, *1*, 145–150. [\[CrossRef\]](#)
- Wood, D. A Note on Bicolored Digraph Grammar Systems. *IJCM* **1973**, *3*, 301–308.

23. Dassow, J.; Turaev, S. Petri net controlled grammars: the power of labeling and final markings. *Rom. J. Inf. Sci. Technol.* **2009**, *12*, 191–207.
24. Dassow, J.; Turaev, S. Petri net controlled grammars: the case of special Petri nets. *J. Univers. Comput. Sci.* **2009**, *15*, 2808–2835.
25. Prusinkiewicz, P.; Hanan, J. *Lindenmayer Systems, Fractals, and Plants*; Lecture Notes in Biomathematics; Springer: Berlin, Germany, 1980; Volume 79.
26. Rajlich, V. Absolutely parallel grammars and two-way deterministic finite state transducers. *J. Comput. Syst. Sci.* **1972**, *6*, 324–342. [[CrossRef](#)]
27. Siromoney, R.; Krithivasan, K. Parallel context-free languages. *Inform. Control* **1974**, *24*, 155–162. [[CrossRef](#)]
28. Levitina, M. On some grammars with global productions. *NTI Ser.* **1972**, *2*, 32–36.
29. Greibach, S.; Hopcroft, J. Scattered context grammars. *J. Comput. Syst. Sci.* **1969**, *3*, 232–247. [[CrossRef](#)]
30. Mavlankulov, G.; Othman, M.; Turaev, S.; Selamat, M.; Zhumabayeva, L.; Zhukabayeva, T. Concurrently Controlled Grammars. *Kybernetika* **2018**, *54*, 748–764. [[CrossRef](#)]
31. Păun, G.; Santean, L. Parallel communicating grammar systems: the regular case. *Ann. Univ. Buc. Ser. Mat.-Inform.* **1989**, *37*, 55–63.
32. Csuhaj-Varjú, E.; Dassow, J.; Kelemen, J.; Păun, G. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*; Gordon and Beach Science Publishers: New York, NY, USA, 1994.
33. Král, J. On Multiple Grammars. *Kybernetika* **1969**, *5*, 60–85.
34. Čulík II, K. n -ary Grammars and the Description of Mapping of Languages. *Kybernetika* **1970**, *6*, 99–117.
35. Bellert, I. Relational Phrase Structure Grammar and Its Tentative Applications. *Inf. Control* **1965**, *8*, 503–530. [[CrossRef](#)]
36. Bellert, I. Relational Phrase Structure Grammar Applied to Mohawk Constructions. *Kybernetika* **1966**, *3*, 264–273.
37. Crimi, C.; Guercio, A.; Nota, G.; Pacini, G.; Tortora, G.; Tucci, M. Relation Grammars and their Application to Multidimensional Languages. *J. Vis. Lang. Comput.* **1991**, *4*, 333–346. [[CrossRef](#)]
38. Wittenburg, K. Earley-Style Parsing for Relational Grammars. In Proceedings of the IEEE Workshop on Visual Languages, Seattle, WA, USA, 15–18 September 1992; pp. 192–199.
39. Wittenburg, K.; Weitzman, L. Relational Grammars: Theory and Practice in a Visual Language Interface for Process Modeling. In *Visual Language Theory*; Marriott, K., Meyer, B., Eds.; Springer Science & Business Media: New York, NY, USA, 1998; pp. 193–217.
40. Johnson, D. On Relational Constraints on Grammars. In *Grammatical Relations*; Cole, P., Sadock, J., Eds.; BRILL: Leiden, The Netherlands, 2020; pp. 151–178.
41. Martín-Vide, C.; Mitrana, V.; Păun, G. (Eds.) *Formal Languages and Applications*; Springer: Berlin/Heidelberg, Germany, 2004.
42. Baumgarten, B. *Petri-Netze. Grundlagen und Anwendungen*; Wissenschaftsverlag: Mannheim, Germany, 1990.
43. Reisig, W.; Rozenberg, G. (Eds.) *Lectures on Petri Nets I: Basic Models*; Springer: Berlin, Germany, 1997; Volume 1441.
44. Dassow, J.; Turaev, S. Petri net controlled grammars with a bounded number of additional places. *Acta Cybernetica* **2009**, *19*, 609–634.
45. Novák, V.; Novotný, M. Binary and Ternary Relations. *Math. Bohem.* **1992**, *117*, 283–292.
46. Novák, V.; Novotný, M. Pseudodimension of Relational Structures. *Czechoslov. Math. J.* **1999**, *49*, 541–560. [[CrossRef](#)]
47. Cristea, I.; Ștefănescu, M. Hypergroups and n -ary Relations. *Eur. J. Comb.* **2010**, *31*, 780–789. [[CrossRef](#)]
48. Chaisansuk, N.; Leeratanavalee, S. Some Properties on the Powers of n -ary Relational Systems. *Novi Sad J. Math.* **2013**, *43*, 191–199.