Revised submission to the Journal of Automata, Languages and Combinatorics
Revised on November 7, 2019

GENERAL CD GRAMMAR SYSTEMS AND THEIR SIMPLIFICATION

Radim Kocman Zbyněk Křivka Alexander Meduna

Centre of Excellence IT4Innovations
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno 612 66, Czech Republic
{ikocman,krivka,meduna}@fit.vutbr.cz

ABSTRACT

The present paper studies general CD grammar systems, whose components are general grammars, so they are computationally complete, and it investigates them working under the $*$ mode and t mode. Most importantly, the paper presents two types of transformations that turn arbitrary general grammars into equivalent two-component general CD grammar systems with a context-free component and a non-context-free component. From the first type of transformations, the non-context-free component results with two rules of the form $11 \rightarrow 00$ and $0000 \rightarrow \varepsilon$, while the other type of transformations produces the non-context-free component with two rules of the form $11 \rightarrow 00$ and $0000 \rightarrow 2222$. Apart from this significant reduction and simplification, the paper describes several other useful properties concerning these systems and the way they work. A formulation of several remarks and open problems closes this paper.

Keywords: general grammars, CD grammar systems, simulated non-context-free rules, homogeneous rules, evenly homogeneous rules

1. Introduction

The present paper, which assumes a familiarity with formal language theory (see [8, 15]), concerns grammar systems (see [1]). It concentrates its attention on two-component CD grammar systems working under the $*$ and t modes. Recall that under the former mode the context-free versions of these systems obviously generate only the family of context-free languages. More surprisingly, under the latter mode they are no more powerful than ordinary context-free grammars either. To increase their power, the present paper uses general CD grammar systems, whose components are general grammars, that are computationally complete—that is, they characterize the family of recursively enumerable languages. Most importantly, however, the paper explains how to turn arbitrary general grammars into equivalent two-component general CD grammar systems of very reduced and simplified forms.

To give an insight into this study in a greater detail, take any general grammar G. This paper demonstrates two types of transformations that turn G into a two-component general CD grammar system with one context-free component and one non-context-free component. For brevity, in this introductory section, Γ_1 and Γ_2 denote the systems resulting from the first type of transformations and the second type of transformations, respectively. Γ_1 has its non-context-free component containing the rules $11 \to 00$ and $0000 \to \varepsilon$, while Γ_2 has its non-context-free component containing the rules $11 \to 00$ and $0000 \to 2222$, where 0, 1, and 2 are new nonterminals. The paper proves that working under the $*$ and t modes, Γ_1 and Γ_2 are equivalent to G. Thus, more generally speaking, general CD grammar systems of these two forms are computationally complete—that is, they characterize the family of recursively enumerable languages. Apart from the computational completeness, it is worth mentioning the following other useful properties, (i) through (v), which make Γ_1 and Γ_2 simple and easy to apply in theory as well as in practice.

(i) Most importantly, observe that Γ_1 and Γ_2 utilize a very reduced number of non-context-free rules. One of their components is always purely context-free, and the other has only two non-context-free rules. Of course, computational completeness resulting from such strongly reduced versions of general CD grammar systems is more than highly appreciated from both a theoretical and practical standpoint.

(ii) Consider Γ_1. The paper demonstrates that working under the t mode, during every generation of a sentence, Γ_1 changes its components no more than once. Furthermore, if the system simulates the use of at least one non-context-free rule from the original grammar, it changes its components precisely once.

(iii) From a general viewpoint, taking a closer look at language-generating rewriting systems, we intuitively see that some of them generate sentences of the same language in a more similar way than others. Formal language theory has formalized this generative phenomenon in terms of close derivation simulations (see Chapter 6 in [10] and [11]). To give an insight into this formalization, consider grammatical models X and Y. Let $\Rightarrow$ denote a derivation step, and let $\Rightarrow^m$ denote m consecutive derivation steps. If there is a constant k such that for every derivation of the form

$$x_0 \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_n$$

in X, where x_0 is its start symbol, there is a derivation of the form

$$x_0 \Rightarrow^{k_1} x_1 \Rightarrow^{k_2} \cdots \Rightarrow^{k_n} x_n$$

in Y, where $k_i \leq k$ for each $1 \leq i \leq n$, we say that Y closely simulates X. In this sense, the paper demonstrates that under the $*$ mode Γ_1 and Γ_2 in many cases closely simulate G. This also makes these transformations distinct from some well-known transformations generating grammatical models with a reduced number of non-context-free rules (e.g., Geffert normal forms [4, 5]), since they usually require a very strict derivation flow for the resulting model.

(iv) The specific form of the rules in Γ_1 and Γ_2 makes it possible to utilize parallelization through the whole sentence generation process. In essence, it is possible to use multi-derivations that, during a derivation step, rewrite the sentential form at several positions at once, not just at a single position. This property also holds for uniform derivations that always rewrite at all possible positions at once.

(v) Before sketching the final property, we recall that a grammatical rule of the form $x \to y$, where x and y are strings, is homogeneous if x is formed by a string of identical symbols (see [9]). A homogeneous rule $x \to y$ is evenly homogeneous if y is also formed by a string of identical symbols and $|x| = |y|$. In a CD grammar system, a component is homogeneous if all its rules are homogeneous, and it is evenly homogeneous if all its rules are evenly homogeneous. A CD grammar system is rule-homogeneous if all its components are homogeneous. As obvious, any CD grammar system with context-free components is rule-homogeneous. Observe that Γ_1 and Γ_2 are both rule-homogeneous CD grammar systems with one context-free component and one homogeneous component. In fact, in Γ_2, the second component is evenly homogeneous.

The rest of this paper is organized as follows. Section 2 recalls all the basic terminology needed in this paper and formally presents the definition of general CD grammar systems. Section 3 introduces all fundamental techniques of the transformations on input grammars that satisfy Kuroda normal form. Sections 4 generalizes the previous techniques for arbitrary general grammars and presents the remaining results. Section 5 closes the study by pointing out some remarks and suggestions for further investigation.

2. Preliminaries and Definitions

This paper assumes that the reader is familiar with the theory of automata and formal languages (see [8, 15]). This section recalls only the crucial notions used in this paper.

We denote by $\mathrm{card}(X)$ the cardinality of a set X. For an alphabet (finite nonempty set), V, V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. Members of V^* are called *strings*. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation. For $x \in V^*$, $|x|$ denotes the length of x, $\mathrm{rev}(x)$ denotes the reversal of x, and $\mathrm{alph}(x)$ denotes the set of all symbols occurring in x; for instance, $\mathrm{alph}(0010) = \{0, 1\}$. Let **CF**, **CS**, and **RE** denote the families of context-free, context-sensitive, and recursively enumerable languages, respectively.

A *general grammar* or, more simply, a *grammar* is a quadruple, $G = (N, T, P, S)$, whose components are defined as follows. N and T are alphabets such that $N \cap T = \emptyset$. Symbols in N are referred to as *nonterminals*, while symbols in T are referred to as *terminals*. $S \in N$ is the start symbol of G. P is a finite set of *(general) rules* of the form $x \to y$, where $x, y \in (N \cup T)^*$ and $\mathrm{alph}(x) \cap N \neq \emptyset$. For brevity, we sometimes denote a rule $x \to y$ with a unique label p as $p\colon x \to y$, and instead of $p\colon x \to y \in P$, we simply write $p \in P$. The left-hand side x and the right-hand side y of p are denoted by lhs(p) and rhs(p), respectively. If $p \in P$ and $|\operatorname{rhs}(p)| = 0$, it is an ε-rule. The rule $p \in P$ is considered context-free if $|\operatorname{lhs}(p)| = 1$; otherwise, it is a non-context-free rule. If $x \to y \in P$ and $u, w \in (N \cup T)^*$, then $uxw \Rightarrow uyw$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$. Let $\Rightarrow^+$ and $\Rightarrow^*$ denote the transitive closure and transitive-reflexive closure of $\Rightarrow$, respectively. The *language generated by* G, $L(G)$, is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$.

Let $G = (N, T, P, S)$ be a grammar. G is in *Kuroda normal form* (see Section 8.3.3. in [8]) if every rule $p \in P$ has one of these three forms: (1) $AB \to CD$, (2) $A \to BC$, or (3) $A \to a$, where $A, B, C, D \in N$ and $a \in (T \cup \{\varepsilon\})$. If $x \to y \in P$ and $x \in \{A\}^+$ for some $A \in N$, then $x \to y$ is a *homogeneous rule* (see [9]). Furthermore, if also $y \in \{B\}^+$ for some $B \in (N \cup T)$ and $|x| = |y|$, then $x \to y$ is an *evenly homogeneous rule*. G is a *homogeneous grammar* if every $p \in P$ is homogeneous. Lastly, set ContextFree$(P) = \{p \in P : |\operatorname{lhs}(p)| = 1\}$ and NonContextFree$(P) = \{p \in P : |\operatorname{lhs}(p)| \geq 2\}$.

A *general cooperating distributed grammar system* (a *general CD grammar system* for short) is a construct $\Gamma = (N, T, P_1, P_2, \dots, P_n, S)$, $n \geq 1$, where N is the alphabet of nonterminals, T is the alphabet of terminals, $N \cap T = \emptyset$, $S \in N$ is the start symbol, and, for $1 \leq i \leq n$, each component P_i is a finite set of general rules. (For the original context-free definition see [1].) For $u, v \in V^*$, $V = N \cup T$, and $1 \leq k \leq n$, let $u \Rightarrow_{P_k} v$ denote a derivation step performed by the application of a rule from P_k. As usual, extend the relation $\Rightarrow_{P_k}$ to $\Rightarrow^m_{P_k}$ (the m-step derivation), $m \geq 0$, $\Rightarrow^+_{P_k}$, and $\Rightarrow^*_{P_k}$. In addition, we define the relation $u \Rightarrow^t_{P_k} v$ so that $u \Rightarrow^*_{P_k} v$ and there is no $w \in V^*$ such that $v \Rightarrow_{P_k} w$. The language generated by Γ working in the f mode, $f \in \{*, t\}$, denoted by $L_f(\Gamma)$, is defined as $L_f(\Gamma) = \{w \in T^* : S \Rightarrow^f_{P_{k_1}} w_1 \Rightarrow^f_{P_{k_2}} \cdots \Rightarrow^f_{P_{k_l}} w_l = w,\ l \geq 1,\ 1 \leq k_i \leq n,\ 1 \leq i \leq l\}$. Γ is referred to as *rule-homogeneous*, *evenly rule-homogeneous*, or *context-free* (instead of general) if all its rules are homogeneous, evenly homogeneous, or context-free, respectively.

Language families generated by context-free CD grammar systems with n components working in the f mode and allowing ε-rules are denoted by $CD^\varepsilon_n(f)$. When the number of components is not limited, we replace n by ∞. The following results are well-known (see Theorem 3.1 in [13]):

(i) $CD^\varepsilon_\infty(*) = \mathbf{CF}$,

(ii) $\mathbf{CF} = CD^\varepsilon_1(t) = CD^\varepsilon_2(t) \subset CD^\varepsilon_3(t) = CD^\varepsilon_\infty(t) = \mathbf{ET0L}$,

where **ET0L** denotes the family of languages generated by extended tabled interactionless Lindenmayer systems (see [12]).

The definition of general CD grammar systems can be easily modified so that the components are sets of rules of any type. Recall that for CD grammar systems having regular, linear, context-sensitive, or general components, their generative power does not change with the number of components (see [1, 13]), i.e., they always generate the families of regular, linear, context-sensitive, or recursively enumerable languages, respectively. Nonetheless, different results have been obtained by studying some other non-classical components—e.g., permitting, left-forbidding, and random context components (see [2, 6, 7])—where the number of components affects the resulting generative power.

It is clear that if we require computational completeness, we need components that use stronger mechanisms than basic context-free rules. In general, components with homogeneous rules have a similar effect as general components—a single homogeneous component can define **RE** by itself (see [9]). The same, however, does not hold for components with evenly homogeneous rules, which can define only sets of single symbols on their own. Therefore, one may wonder, what properties we can get if we combine together several relatively simple components of different types.

The rest of this paper studies two-component general CD grammar systems where the first component is always purely context-free and the second component contains either homogeneous or evenly homogeneous rules. Furthermore, we limit the non-context-free component so it contains only two rules.

3. Transformations from Kuroda Normal Form

In order to simplify the reasoning for underlying proofs, this section assumes that all input grammars satisfy Kuroda normal form. Nonetheless, in the following section, we show that Kuroda normal form is not necessary and that we can use similar techniques to convert any general grammar into a two-component general CD grammar system that also satisfies similar properties.

First, let us start with the most straightforward variant of a two-component general CD grammar system that works in the $*$ mode and has the second component homogeneous. The following proof will also serve as a framework for later proofs since the majority of the reasoning can be shared throughout the variants.

Theorem 1. *Let $G = (N, T, P, S)$ be a grammar in Kuroda normal form. Then, there exists a two-component general CD grammar system $\Gamma = (N', T, H, I, S)$ such that H is context-free, $I = \{11 \to 00,\ 0000 \to \varepsilon\}$, and $L_*(\Gamma) = L(G)$.*

Proof.
Construction.
Let $G = (N, T, P, S)$. Without any loss of generality, assume that $(N \cup T) \cap \{0, 1\} = \emptyset$. For $m = 2 + \mathrm{card}(\mathrm{NonContextFree}(P))$, define an injection g from $\mathrm{NonContextFree}(P)$ to $(\{01\}^+\{00\}\{01\}^+ \cap \{01, 00\}^m)$. From G, we construct the two-component general CD grammar system $\Gamma = (N', T, H, I, S)$, where $N' = N \cup \{0, 1\}$, $I = \{11 \to 00,\ 0000 \to \varepsilon\}$, and H is defined as follows:

(I) For every $AB \to CD \in P$ where $A, B, C, D \in N$,
add $A \to CDg(AB \to CD)$ and $B \to \mathrm{rev}(g(AB \to CD))$ to H.

(II) For every $A \to x \in P$ where $A \in N$ and $x \in (\{\varepsilon\} \cup T \cup N^2)$, add $A \to x$ to H.

The construction of Γ is completed.

Basic idea.

(a) The rules (I) and the second component I simulate the derivation steps made by $\mathrm{NonContextFree}(P)$ in G. That is, $xABy \Rightarrow xCDy$ according to $AB \to CD \in P$

in G, where $x, y \in (N \cup T)^*$, is simulated in Γ as

$$\begin{aligned} xABy &\Rightarrow_H xCDg(AB \to CD)By \\ &\Rightarrow_H xCDg(AB \to CD)\,\mathrm{rev}(g(AB \to CD))y \\ &\Rightarrow_I^{2m-1} xCDy. \end{aligned}$$

Γ makes the $(2m-1)$-step derivation $xCDg(AB \to CD)\,\mathrm{rev}(g(AB \to CD))y \Rightarrow_I^{2m-1} xCDy$ by using only the rules $11 \to 00$ and $0000 \to \varepsilon$. During this derivation, the string between D and y always contains exactly one occurrence of consecutive identical symbols that can be rewritten, so this derivation actually verifies that the simulation of $xABy \Rightarrow xCDy$ is made properly.

(b) The rules (II) simulate the use of ContextFree(P) in G.

The reader may notice that the simulation of non-context-free rules resembles similar techniques used in general grammars (see [14, 3, 4, 5, 9]). However, this is traditionally done either by using several types of matching parentheses (see [14, 3]), which is not a suitable form for homogeneous rules, or it requires a significant non-local change in the generation flow of the original grammar (see [4, 5, 9]), which denies the close derivation simulations.

Formal proof.
We prove $L_*(\Gamma) = L(G)$. It was already proven in part B of the proof of Theorem 1 in [3] and in the proof of Theorem 1 in [14] that the rules of the form $XY \to w$, where $X, Y \in N$, $w \in (N \cup T)^*$, can be replaced with $X \to wL_i$, $Y \to R_i$, $L_iR_i \to \varepsilon$, where L_i, R_i are new unique nonterminals for each rule. Thus, we only need to prove that our encoding with injection g simulates the same behavior and that it works in two-component general CD grammar systems.

First, we establish the terminology that we will use throughout this proof: Any consecutive sequence of nonterminals 0 and 1 is referred to as *verification code*. We say that a sequence is *rewritable* if there is a rule in Γ that can be used on the sequence. We recognize three distinct types of verification codes in the sentential form:

- *unconnected verification code* – This is the initial sequence from the rules (I). Considering some rule $AB \to CD \in P$, it is either $g(AB \to CD)$ or $\mathrm{rev}(g(AB \to CD))$. In more detail, we identify $g(AB \to CD)$ as a *left unconnected verification code* (the code of the form $\{01\}^+\{00\}\{01\}^+$) and $\mathrm{rev}(g(AB \to CD))$ as a *right unconnected verification code* (the code of the form $\{10\}^+\{00\}\{10\}^+$).
- *connected verification code* – This sequence is established when some left and right unconnected verification codes merge together in the sentential form, and we identify it as connected until it has the form $\{01\}\{0,1\}^*\{10\}$.
- *leftover* – The remaining sequence $\{0000\}$.

Now, we show that our encoding properly simulates $L_iR_i \to \varepsilon$. First, we establish a claim on how a connected verification code can be rewritten.

Claim 2. *Let $AB \rightarrow CD \in P$, where $A, B, C, D \in N$. A (standalone) connected verification code can be reduced to a leftover if and only if it was initially established as $g(AB \rightarrow CD)\,\mathrm{rev}(g(AB \rightarrow CD))$.*

Proof. Let $AB \rightarrow CD, EF \rightarrow UV \in P$, where $A, B, C, D, E, F, U, V \in N$, such that $g(AB \rightarrow CD) = (01)^k 00 (01)^l$ and $g(EF \rightarrow UV) = (01)^p 00 (01)^q$, where $k, l, p, q \geq 1$, $k + l + 1 = m$, $p + q + 1 = m$, $k \neq p$. There are two distinct cases how a connected verification code can be initially established: either $g(AB \rightarrow CD) rev(g(AB \rightarrow CD))$ or $g(AB \rightarrow CD) rev(g(EF \rightarrow UV))$. (There are four ways how to pair the unconnected verification codes but only two distinct cases since the rules can be swapped.)

The first case creates the sequence $(01)^k 00 (01)^l (10)^l 00 (10)^k$. Initially, the rules $11 \rightarrow 00$ and $0000 \rightarrow \varepsilon$ are used l times to erase $(01)^l (10)^l$. Then, the rule $0000 \rightarrow \varepsilon$ has to be used. Next, both rules are used $k - 1$ times again until only the sequence 0110 remains. And finally, the rule $11 \rightarrow 00$ creates the leftover.

In the second case, the sequence is $(01)^k 00 (01)^l (10)^q 00 (10)^p$. Assume that $l > q$; the result is analogical for the opposite situation. The rules $11 \rightarrow 00$ and $0000 \rightarrow \varepsilon$ can be used q times, and it leads to the sequence $(01)^k 00 (01)^{l-q} 00 (10)^p$. Nonetheless, this sequence cannot be rewritten any further.

In both cases, the derivation steps cannot be done in any other way. Thus, it is clear that a connected verification code can be reduced to a leftover if and only if it is established as $g(AB \rightarrow CD)\,\mathrm{rev}(g(AB \rightarrow CD))$ for some $AB \rightarrow CD \in P$. ■

Next, we demonstrate that there is no sentential form in which a verification code could be rewritten in some unintended way.

Claim 3. *In any reachable sentential form, a verification code can be rewritten only if it can be identified as either a connected verification code or a sequence of 0's containing a leftover as its substring.*

Proof. Verification codes can be rewritten only with the rules $11 \rightarrow 00$ and $0000 \rightarrow \varepsilon$. Any verification code generated into the sentential form is initially in the form $\{01\}^+ \{00\} \{01\}^+$ or $\{10\}^+ \{00\} \{10\}^+$. Clearly, no rule can rewrite this code as long as it remains alone. When the left and right unconnected verification codes are joined together, a connected verification code is established where the rewriting can occur. In contrast, observe that unconnected verification codes joined in different ways do not establish any rewritable sequence.

Considering the proof of Claim 2, a connected verification code is always in the form $\{01\}\{0, 1\}^* \{10\}$ until it is reduced to a leftover. Observe that if this form is joined together with other connected or unconnected verification codes, it does not establish any new rewritable sequence.

Lastly, the leftover 0000 is clearly rewritable on its own, but it can be also merged with other verification codes and that can create an even longer rewritable sequence of 0's. Nonetheless, observe that only the rule $0000 \rightarrow \varepsilon$ can be applied on this sequence, which erases precisely the leftover. Thus, this cannot affect the form of the other verification codes.

The above description covers all obtainable forms of verification codes, and only the connected verification code and the sequence of 0's containing a leftover as its substring can be rewritten. Thus, Claim 3 holds. ■

From Claims 2 and 3, it is obvious that the encoding successfully simulates the unique nonterminals L_i, R_i and the erasing rules $L_iR_i \to \varepsilon$.

Next, we prove $L(G) \subseteq L_*(\Gamma)$; more precisely, by induction on the number of derivation steps, we demonstrate Claim 4.

Claim 4. *For every $w \in (N \cup T)^*$ and $i \geq 0$, $S \Rightarrow^i w$ in G implies $S \Rightarrow^*_{k_1} w_1 \Rightarrow^*_{k_2} \cdots \Rightarrow^*_{k_l} w_l = w$, $l \geq 1$, $k_j \in \{H, I\}$, $1 \leq j \leq l$, in Γ.*

Proof. *Basis:* Let $i = 0$. Then, $w = S$. Clearly, $S \Rightarrow^*_H S$.
Induction hypothesis: Assume that the implication of Claim 4 holds for every $i \leq o$, where o is a non-negative integer.
Induction step: Consider any derivation of the form $S \Rightarrow^{o+1} \beta$ in G, where $\beta \in (N \cup T)^*$. Express $S \Rightarrow^{o+1} \beta$ as $S \Rightarrow^o \alpha \Rightarrow \beta$, where $\alpha \in (N \cup T)^*$. By the induction hypothesis, $S \Rightarrow^*_{k_1} w_1 \Rightarrow^*_{k_2} \cdots \Rightarrow^*_{k_l} w_l = \alpha$, $l \geq 1$, $k_j \in \{H, I\}$, $1 \leq j \leq l$, in Γ. There are the following two possibilities how G can make $\alpha \Rightarrow \beta$:

(1) Let $AB \to CD \in P$, $\alpha = xABy$, $\beta = xCDy$, $x, y \in (N \cup T)^*$, $A, B, C, D \in N$. According to (a) in the basic idea and from Claims 2 and 3,

$$\begin{aligned} xABy &\Rightarrow_H xCDg(AB \to CD)By \\ &\Rightarrow_H xCDg(AB \to CD)\,\mathrm{rev}(g(AB \to CD))y \\ &\Rightarrow^{2m-1}_I xCDy \end{aligned}$$

in Γ. Consequently, $S \Rightarrow^*_{k_1} w_1 \Rightarrow^*_{k_2} \cdots \Rightarrow^*_{k_{l'}} w_{l'} = xCDy = \beta$, $l' \geq 1$, $k_j \in \{H, I\}$, $1 \leq j \leq l'$, in Γ.

(2) Let $A \to z \in P$, $\alpha = xAy$, $\beta = xzy$, $x, y \in (N \cup T)^*$, $A \in N$, $z \in (\{\varepsilon\} \cup T \cup N^2)$. From (b) in the basic idea, $xAy \Rightarrow_H xzy$ in Γ. Consequently, $S \Rightarrow^*_{k_1} w_1 \Rightarrow^*_{k_2} \cdots \Rightarrow^*_{k_{l'}} w_{l'} = xzy = \beta$, $l' \geq 1$, $k_j \in \{H, I\}$, $1 \leq j \leq l'$, in Γ.

The induction step is completed, so Claim 4 holds. ■

Lastly, we prove $L_*(\Gamma) \subseteq L(G)$. We show that, for any $y \in L_*(\Gamma)$, there is a sequence of derivation steps in Γ that precisely follows the intended order from the basic idea, so $y \in L(G)$.

Claim 5. *Any successful derivation sequence generating $y \in L_*(\Gamma)$ in Γ can be reordered so it satisfies the form*

$$\begin{aligned} S = v_{0_3} &\Rightarrow^*_H v_{1_0} \Rightarrow_H v_{1_1} \Rightarrow_H v_{1_2} \Rightarrow^{2m-1}_I v_{1_3} \\ &\Rightarrow^*_H v_{2_0} \Rightarrow_H v_{2_1} \Rightarrow_H v_{2_2} \Rightarrow^{2m-1}_I v_{2_3} \\ &\ \vdots \\ &\Rightarrow^*_H v_{k_0} \Rightarrow_H v_{k_1} \Rightarrow_H v_{k_2} \Rightarrow^{2m-1}_I v_{k_3} \Rightarrow^*_H v_{(k+1)_0} = y, \end{aligned}$$

where for $i = 0, 1, \dots, k$ in $v_{i_3} \Rightarrow_H^* v_{(i+1)_0}$ every sentential form is over $(N \cup T)^*$; and for $j = 1, \dots, k$ the sentential forms in the derivation $v_{j_0} \Rightarrow_H v_{j_1} \Rightarrow_H v_{j_2} \Rightarrow_I^{2m-1} v_{j_3}$ have the structure:

$$\begin{aligned}
&v_{j_0} = u_j A_j B_j w_j,\\
&v_{j_1} = u_j C_j D_j g(A_j B_j \to C_j D_j) B_j w_j,\\
&v_{j_2} = u_j C_j D_j g(A_j B_j \to C_j D_j)\,\mathrm{rev}(g(A_j B_j \to C_j D_j)) w_j,\\
&v_{j_3} = u_j C_j D_j w_j,\\
&\textit{for some } A_j B_j \to C_j D_j \in P,\ u_j, w_j \in (N \cup T)^*,\ A_j, B_j, C_j, D_j \in N.
\end{aligned}$$

Proof. First, all nonterminals in N can be rewritten only with the context-free rules of the component H. This implies that it does not matter in which order we rewrite them in the sentential form. Second, consider rules $A \to CDg(AB \to CD), B \to \mathrm{rev}(g(AB \to CD)) \in H$, where $A, B, C, D \in N$, $AB \to CD \in P$. Claims 2 and 3 show that only the verification code $g(AB \to CD)\,\mathrm{rev}(g(AB \to CD))$ can be successfully erased. It follows that we can always establish some order of derivations in which the sequence $v_{j_0} \Rightarrow_H v_{j_1} \Rightarrow_H v_{j_2} \Rightarrow_I^{2m-1} v_{j_3}$ holds for each simulated non-context-free rule. ■

From the reordered derivations of Claim 5 in Γ and from (I) and (II), we see that $v_{0_3} \Rightarrow^* v_{(k+1)_0}$ in G. Therefore, $y \in L_*(\Gamma)$ implies $y \in L(G)$. Thus, $L_*(\Gamma) \subseteq L(G)$.

As $L(G) \subseteq L_*(\Gamma)$ and $L_*(\Gamma) \subseteq L(G)$, $L_*(\Gamma) = L(G)$. Thus, Theorem 1 holds. □

Corollary 6. *The resulting two-component general CD grammar system Γ from the proof of Theorem 1 closely simulates the original grammar G.*

Proof. For any resulting Γ, we can find a bounded constant k such that for every possible derivation $u \Rightarrow v$ in G there is a k'-step derivation in Γ that gives the same result and $k' \leq k$. Furthermore, for a given Γ, we can easily determine the minimal possible k.

Consider the proof of Claim 4 and the mentioned possibilities how G can make $\alpha \Rightarrow \beta$. Any context-free rule is simulated in one derivation step. The non-context-free rules require two initial derivation steps and the rewriting of the verification code. The length of the rewriting depends on the size of m, and it takes $2m - 1$ steps to complete. The minimal possible k for a given Γ is therefore $2m + 1$. □

Next, we consider a two-component general CD grammar system with the same structure but working in the t mode.

Theorem 7. *Let $G = (N, T, P, S)$ be a grammar in Kuroda normal form. Then, there exists a two-component general CD grammar system $\Gamma = (N', T, H, I, S)$ such that H is context-free, $I = \{11 \to 00,\ 0000 \to \varepsilon\}$, and $L_t(\Gamma) = L(G)$.*

Proof.
Construction. The process of construction remains identical to Theorem 1. For a grammar $G = (N, T, P, S)$, $m = 2 + \mathrm{card}(\mathrm{NonContextFree}(P))$, and injection g, we

construct the two-component general CD grammar system $\Gamma = (N', T, H, I, S)$, where $N' = N \cup \{0, 1\}$, and H and I contain the rules as described in Theorem 1.

Basic idea.
Recall that, during the generation of a sentence, a CD grammar system working in the t mode switches its components only if the process is not finished and there are no possible derivations with the previous component. Consider the general behavior of Γ. It starts the generation with S. For the first derivation, applicable rules can be found only in H, so this component has to be used. However, H also contains all rules simulating the original rules of G. Consequently, the first derivation in the t mode has to simulate all rules in G and cannot rewrite any generated verification codes. Nonetheless, we prove that the verification codes can be successfully erased afterwards for all simulated non-context-free rules at once.

Formal proof.
We prove $L_t(\Gamma) = L(G)$. First, let us prove the statement introduced above. For convenience, consider the homomorphism $\varphi : (N' \cup T)^* \to (N \cup T)^*$ where $\varphi(a) = a$ and $\varphi(b) = \varepsilon$, for all $a \in (N \cup T)$ and $b \in \{0, 1\}$.

Claim 8. *For every $u \in (N \cup T)^*$ and $i \geq 0$, $S \Rightarrow^i u$ in G implies $S \Rightarrow_H^* w \Rightarrow_I^t u$ in Γ, where $w \in (N' \cup T)^*$ and $\varphi(w) = u$. Furthermore, w satisfies the form $w = p_1q_1 \cdots p_nq_n$, where $n \geq 1$, $p_j \in (N \cup T)^*$, $q_j \in \{0, 1\}^*$, $1 \leq j \leq n$, and every q_j represents a verification code that can be successfully erased on its own.*

Proof. *Basis:* Let $i = 0$. Then, $u = S$. Clearly, $S \Rightarrow_H^0 S \Rightarrow_I^t S$, and the required form also holds.
Induction hypothesis: Assume that Claim 8 holds for every $i \leq o$, where o is a non-negative integer.
Induction step: Consider any derivation of the form $S \Rightarrow^{o+1} \beta$ in G, where $\beta \in (N \cup T)^*$. Express $S \Rightarrow^{o+1} \beta$ as $S \Rightarrow^o \alpha \Rightarrow \beta$, where $\alpha \in (N \cup T)^*$. By the induction hypothesis, $S \Rightarrow_H^* w \Rightarrow_I^t \alpha$, where $\varphi(w) = \alpha$, in Γ. There are the following two possibilities how G can make $\alpha \Rightarrow \beta$:

(1) Let $AB \to CD \in P$, $\alpha = xABy$, $\beta = xCDy$, $x, y \in (N \cup T)^*$, $A, B, C, D \in N$. Consider w in the required form. Let $w = p_1q_1 \cdots p_kAq_kBp_{k+1}q_{k+1} \cdots p_nq_n$, where $n \geq 1$, $1 \leq k \leq n$, $p_j \in (N \cup T)^*$, $q_j \in \{0, 1\}^*$, $1 \leq j \leq n$, and also $p_1 \cdots p_k = x$ and $p_{k+1} \cdots p_n = y$. Then,

$$\begin{aligned}
& w = p_1q_1 \cdots p_kAq_kBp_{k+1}q_{k+1} \cdots p_nq_n \\
& \Rightarrow_H p_1q_1 \cdots p_kCDg(AB \to CD)q_kBp_{k+1}q_{k+1} \cdots p_nq_n \\
& \Rightarrow_H p_1q_1 \cdots p_kCDg(AB \to CD)q_k \operatorname{rev}(g(AB \to CD))p_{k+1}q_{k+1} \cdots p_nq_n \\
& = w'
\end{aligned}$$

in Γ, and there are two possible situations regarding these steps:

(A) If $q_k = \varepsilon$, the steps add a new connected verification code. By Claims 2 and 3, such a code can be successfully erased on its own, so the required form holds. Consequently, $S \Rightarrow_H^* w' \Rightarrow_I^t \beta$ in Γ.

(B) If $q_k \neq \varepsilon$, the steps prolong some existing verification code. However, since q_k has to be erasable on its own, observe that this creates a properly nested structure that is also erasable on its own, so the required form holds. Consequently, $S \Rightarrow_H^* w' \Rightarrow_I^t \beta$ in Γ.

(2) Let $A \to z \in P$, $\alpha = xAy$, $\beta = xzy$, $x, y \in (N \cup T)^*$, $A \in N$, $z \in (\{\varepsilon\} \cup T \cup N^2)$. Consider w in the required form. Let $w = p_1q_1 \cdots p_kAq_kp_{k+1}q_{k+1} \cdots p_nq_n$, where $n \geq 1$, $1 \leq k \leq n$, $p_j \in (N \cup T)^*$, $q_j \in \{0,1\}^*$, $1 \leq j \leq n$, and also $p_1 \cdots p_k = x$ and $p_{k+1} \cdots p_n = y$. Then,

$$\begin{aligned} w &= p_1q_1 \cdots p_kAq_kp_{k+1}q_{k+1} \cdots p_nq_n \\ &\Rightarrow_H p_1q_1 \cdots p_kzq_kp_{k+1}q_{k+1} \cdots p_nq_n = w' \end{aligned}$$

in Γ. The required form clearly holds, and thus $S \Rightarrow_H^* w' \Rightarrow_I^t \beta$ in Γ.

The induction step is completed, so Claim 8 holds. ■

Consider $S \Rightarrow^* y$, where $y \in T^*$, in G. By Claim 8, this implies $S \Rightarrow_H^* w \Rightarrow_I^t y$, where $w \in (T \cup \{0,1\})^*$, in Γ. It is obvious that, in such a case, $\Rightarrow_H^*$ behaves exactly the same as $\Rightarrow_H^t$. Thus, $L(G) \subseteq L_t(\Gamma)$. Nonetheless, it is clear that Γ working in the t mode can no longer closely simulate G.

Since the t mode is a restricted case of the $*$ mode, it must hold that $L_t(\Gamma) \subseteq L_*(\Gamma)$. From the proof of Theorem 1, $L_*(\Gamma) = L(G)$. Therefore, $L_t(\Gamma) \subseteq L(G)$.

As $L(G) \subseteq L_t(\Gamma)$ and $L_t(\Gamma) \subseteq L(G)$, $L_t(\Gamma) = L(G)$. Thus, Theorem 7 holds. □

Corollary 9. *The resulting two-component general CD grammar system Γ from the proof of Theorem 7 changes its components, during every generation of a sentence, no more than once.*

Proof. This proof directly follows the basic idea of Theorem 7 and Claim 8. Γ always starts the process with the symbol S and component H, since H is the only component that can generate something from S. If the first derivation does not use any simulated non-context-free rules, then Γ never switches components, because the result of such a derivation is already a final sentence. If the result contains verification codes, then Γ switches to the component I that finishes the generation. Since I cannot introduce any new nonterminals of the original grammar, Γ is not able to switch again. □

For the remaining results, we change the second component of the two-component general CD grammar system so it is evenly homogeneous. We show that such a system also works correctly in both the $*$ mode and the t mode.

Theorem 10. *Let $G = (N, T, P, S)$ be a grammar in Kuroda normal form. Then, there exists a two-component general CD grammar system $\Gamma = (N', T, H, I, S)$ such that H is context-free, $I = \{11 \to 00,\ 0000 \to 2222\}$, and $L_*(\Gamma) = L_t(\Gamma) = L(G)$.*

Proof.
Construction.
Let $G = (N, T, P, S)$. Without any loss of generality, assume that $(N \cup T) \cap$

$\{0,1,2\} = \emptyset$. For $m = 2 + \mathrm{card}(\mathrm{NonContextFree}(P))$, define an injection g from $\mathrm{NonContextFree}(P)$ to $(\{01\}^{+}\{00\}\{01\}^{+} \cap \{01,00\}^{m})$. From G, we construct the two-component general CD grammar system $\Gamma = (N', T, H, I, S)$, where $N' = N \cup \{0,1,2\}$, $I = \{11 \rightarrow 00,\ 0000 \rightarrow 2222\}$, and H is defined as follows:

(I) For every $AB \rightarrow CD \in P$ where $A, B, C, D \in N$,
add $A \rightarrow CDg(AB \rightarrow CD)$ and $B \rightarrow \mathrm{rev}(g(AB \rightarrow CD))$ to H.

(II) For every $A \rightarrow x \in P$ where $A \in N$ and $x \in (\{\varepsilon\} \cup T \cup N^2)$, add $A \rightarrow x$ to H.

(III) Add $2 \rightarrow \varepsilon$ to H.

The construction of Γ is completed.

Note that this resembles the construction from Theorem 1. We only added one new nonterminal and a rule that can erase it. Also the basic idea for the simulation process remains almost the same.

Formal proof (sketch).
First, consider the verification codes. We adjust our terminology and say that the verification code can also contain occurrences of nonterminal 2. Furthermore, the connected verification code now always holds the form $\{01\}\{0,1,2\}^*\{10\}$. Note that only the rule $0000 \rightarrow 2222$ can generate 2's and that only the rule $2 \rightarrow \varepsilon$ can rewrite these nonterminals further. It follows that the proof of Claim 2 can be trivially adapted for the modified structure, and thus Claim 2 also holds in this system. The following claim introduces a slightly modified version of Claim 3.

Claim 11. *In any reachable sentential form, a verification code can be rewritten only if it can be identified as a connected verification code, sequence of* 0*'s containing a leftover as its substring, or nonterminal* 2*.*

Proof. The proof is analogical to Claim 3. ■

From Claim 2 and 11, it is clear that the purpose of verification codes holds.

Next, consider the $*$ mode. It is obvious that Γ has to switch its components several times if some connected verification code needs to be erased, since the rules of I rewrite only 0's and 1's and the rule from H rewrites 2's. For brevity, let $u \Longrightarrow^{l} v$ denote the sequence $u \Rightarrow_{k_1} v_1 \Rightarrow_{k_2} \cdots \Rightarrow_{k_l} v_l = v$, $k_j \in \{H, I\}$, $1 \leq j \leq l$. Considering the basic idea in Theorem 1, we can clearly replace the original derivation sequence $\Rightarrow_{I}^{2m-1}$ with a new derivation sequence $\Longrightarrow^{6m-1}$. This change can be also straightforwardly applied on Claims 4 and 5 and their proofs. Consequently, $L_*(\Gamma) = L(G)$.

Lastly, consider the t mode. We introduce a modified version of Claim 8. For convenience and brevity, consider the homomorphism $\varphi : (N' \cup T)^* \rightarrow (N \cup T)^*$ where $\varphi(a) = a$ and $\varphi(b) = \varepsilon$, for all $a \in (N \cup T)$ and $b \in \{0,1,2\}$; and let $u \Longrightarrow^{t} v$ denote the sequence $u \Rightarrow_{k_1}^{t} v_1 \Rightarrow_{k_2}^{t} \cdots \Rightarrow_{k_l}^{t} v_l = v$, $l \geq 1$, $k_j \in \{H, I\}$, $1 \leq j \leq l$.

Claim 12. *For every $u \in (N \cup T)^*$ and $i \geq 0$, $S \Rightarrow^{i} u$ in G implies $S \Rightarrow_{H}^{*} w \Longrightarrow^{t} u$ in Γ, where $w \in (N' \cup T)^*$ and $\varphi(w) = u$. Furthermore, we consider w to be generally in the form $w = p_1 q_1 \cdots p_n q_n$, where $n \geq 1$, $p_j \in (N \cup T)^*$, $q_j \in \{0,1\}^*$, $1 \leq j \leq n$, and every q_j represents a verification code that can be successfully erased on its own.*

Proof. The proof by induction is analogical to Claim 8. ■

Consider $S \Rightarrow^* y$, where $y \in T^*$, in G. By Claim 12, this implies $S \Rightarrow_H^* w \Longrightarrow^t y$, where $w \in (T \cup \{0,1\})^*$, in Γ. It is again obvious that, in such a case, $\Rightarrow_H^*$ behaves exactly the same as $\Rightarrow_H^t$. Thus, $L(G) \subseteq L_t(\Gamma)$. It is clear that Γ working in the t mode cannot closely simulate G. Furthermore, it is not even possible to bound the number how many times Γ changes its components during the generation of a sentence, since verification codes can be arbitrarily nested and the erasing process needs to constantly switch the components.

As $L_*(\Gamma) = L(G)$, $L(G) \subseteq L_t(\Gamma)$, and $L_t(\Gamma) \subseteq L_*(\Gamma)$, $L_*(\Gamma) = L_t(\Gamma) = L(G)$. Thus, Theorem 10 holds. □

Corollary 13. *If the two-component general CD grammar system* Γ *from the proof of Theorem 10 works in the* $*$ *mode, it can closely simulate the original grammar* G.

Proof. The reasoning is the same as for Corollary 6. For any resulting Γ, we can find a bounded constant k such that for every possible derivation $u \Rightarrow v$ in G there is a k'-step derivation in Γ that gives the same result and $k' \leq k$. Furthermore, for a given Γ, we can easily determine the minimal possible k.

Again, any context-free rule is simulated in one derivation step. The non-context-free rules require two initial derivation steps and the rewriting of the verification code. The length of the rewriting depends on the size of m, and in this case it takes $6m - 1$ steps to complete. The minimal possible k for a given Γ is therefore $6m + 1$. □

4. Transformations from General Grammars

This section considers transformations that turn arbitrary general grammars into equivalent two-component general CD grammar systems. Since the previous section already established several transformations from Kuroda normal form, the most straightforward approach would be to convert any general grammar into Kuroda normal form and then use the previous transformations; however, considering the resulting properties of the system, this approach may be undesirable. First, if we want to keep the system close to the original grammar, the transformation into the normal form already considerably impacts the grammar. Second, the system may generate unnecessarily nested verification codes that can be inconvenient for parallelization. Therefore, we introduce transformations that directly work with general grammars. We say that a transformation from general grammars into two-component general CD grammar systems is *direct* if it keeps the original context-free rules intact and splits the non-context-free rules proportionally to the number of symbols on their left-hand sides.

Theorem 14. *Let* $G = (N, T, P, S)$ *be a general grammar such that* $\mathrm{alph}(\mathrm{lhs}(p)) \cap T = \emptyset$ *for all* $p \in P$. *Then, there exists its direct transformation into a two-component general CD grammar system* $\Gamma = (N', T, H, I, S)$ *such that* H *is context-free,* $I = \{11 \rightarrow 00,\ 0000 \rightarrow \varepsilon\}$, *and* $L_*(\Gamma) = L_t(\Gamma) = L(G)$.

Proof.
Construction.
Let $G = (N, T, P, S)$. Without any loss of generality, assume that $(N \cup T) \cap \{0, 1\} = \emptyset$. For $n = \max(\{|\operatorname{lhs}(p)| : p \in \text{NonContextFree}(P)\})$ and some $m \geq 3$, define an injection g from $\text{NonContextFree}(P) \times \{1, \dots, n{-}1\}$ to $(\{01\}^+\{00\}\{01\}^+ \cap \{01, 00\}^m)$. From G, we construct the two-component general CD grammar system $\Gamma = (N', T, H, I, S)$, where $N' = N \cup \{0, 1\}$, $I = \{11 \rightarrow 00,\ 0000 \rightarrow \varepsilon\}$, and H is defined as follows:

(I) For every $r : X_1 \cdots X_m \rightarrow x \in P$ where $m \geq 2$, $X_1, \dots, X_m \in N$, and $x \in (N \cup T)^*$, add the following rules to H:
$X_1 \rightarrow x g(r, 1)$,
$X_2 \rightarrow \operatorname{rev}(g(r, 1)) g(r, 2)$,
$\vdots$
$X_{m-1} \rightarrow \operatorname{rev}(g(r, m{-}2)) g(r, m{-}1)$,
$X_m \rightarrow \operatorname{rev}(g(r, m{-}1))$.

(II) For every $X \rightarrow x \in P$ where $X \in N$ and $x \in (N \cup T)^*$, add $X \rightarrow x$ to H.

The construction of Γ is completed.

Formal proof (sketch).
To prove the correctness of the above construction, we utilize the previous construction from Theorem 1, and we go backwards through the transformation of general grammars into Kuroda normal form.

From Theorem 8.3.3.1 in [8], we use the following transformation of a general grammar, $G = (N, T, P, S)$, into an equivalent Kuroda normal form grammar, $G_{\text{KNF}} = (N_{\text{KNF}}, T, P_{\text{KNF}}, S)$, which has five distinct steps that modify original rules and add new auxiliary nonterminals. We outline only the necessary basics since details are rather lengthy. All capital letters in the description represent some nonterminals from N_{KNF}. At start, $N_{\text{KNF}} = N$. The five steps follow:

(1) Each occurrence of a terminal, $a \in T$, is replaced with a new nonterminal a', and we add a new rule $a' \rightarrow a$.
(2) Every $A_1 \cdots A_m \rightarrow B_1 \cdots B_n$, where n and m satisfy $0 \leq n < m$, is replaced with $A_1 \cdots A_m \rightarrow B_1 \cdots B_n C_{n+1} \cdots C_m$, where C_{n+1} through C_m denote occurrences of a new nonterminal C. We also add a new rule $C \rightarrow \varepsilon$.
(3) Every $A \rightarrow B$ is replaced with $A \rightarrow BC$ and $C \rightarrow \varepsilon$. C is a new nonterminal.
(4) Every $A_1 \cdots A_m \rightarrow B_1 \cdots B_n$, where $2 \leq m$ and $3 \leq n$, is repeatedly replaced with $A_1 A_2 \rightarrow B_1 C$ and $C A_3 \cdots A_m \rightarrow B_2 \cdots B_n$. C is a new nonterminal.
(5) Every $A \rightarrow B_1 \cdots B_n$, where $3 \leq n$, is replaced with the standard chain of rules:
$A \rightarrow B_1 \langle B_2 \cdots B_n \rangle$, $\langle B_2 \cdots B_n \rangle \rightarrow B_2 \langle B_3 \cdots B_n \rangle$, $\dots$,
$\langle B_{n-2} \cdots B_n \rangle \rightarrow B_{n-2} \langle B_{n-1} B_n \rangle$, $\langle B_{n-1} B_n \rangle \rightarrow B_{n-1} B_n$.
$\langle B_2 \cdots B_n \rangle, \dots, \langle B_{n-1} B_n \rangle$ are new nonterminals.

Let G be a general grammar such that $\operatorname{alph}(\operatorname{lhs}(p)) \cap T = \emptyset$ for all $p \in P$. Let G_{KNF} be a grammar in Kuroda normal form that was created from G according to the above algorithm. And lastly, let Γ_{KNF} be a two-component general CD grammar system that was created from G_{KNF} according to the construction from Theorem 1. The proofs of Theorems 1 and 7 have already shown that we can rewrite nonterminals of G_{KNF} in the sentential form of Γ_{KNF} in any order and that the t mode expands all nonterminals of the original input grammar in one derivation. Therefore, we can, in a backward way, recreate the desired form of rules in Γ from the rules of Γ_{KNF}.

First, consider the original context-free rules of G. They are affected only by the transformation into Kuroda normal form in steps (1), (3), and (5). Therefore, we can easily recreate their original form so that it corresponds with (II). This is possible because each time the transformation into Kuroda normal form splits a rule, it defines some new nonterminal for which only one specific rule is applicable. Therefore, this next rule has to be also eventually applied in Γ_{KNF}, and it causes no issue if both rules are applied together as one in Γ.

Second, consider the context-sensitive rules $A_1 \cdots A_m \to B_1 \cdots B_n$ of G, where $A_1, \ldots, A_m \in N$ and $B_1, \ldots, B_n \in (N \cup T)^*$. First, their right-hand side is affected by (1). Next, they are rewritten with steps (2) and (4) so they have the form: $r_1 \colon A_1A_2 \to B_1C_1$, $r_2 \colon C_1A_3 \to B_2C_2$, and so on. These remaining context-sensitive rules are then transformed into Γ_{KNF} as: $A_1 \to B_1C_1g(r_1)$, $A_2 \to \mathrm{rev}(g(r_1))$, $C_1 \to B_2C_2g(r_2)$, $A_3 \to \mathrm{rev}(g(r_2))$, etc. Working backwards, we get the rules of the form: $A_1 \to B_1 \cdots B_n g(r_{m-1}) \cdots g(r_1)$, $A_2 \to \mathrm{rev}(g(r_1))$, $A_3 \to \mathrm{rev}(g(r_2))$, etc. Observe that this creates a nested structure of verification codes. (The same situation inevitably happens in the t mode of Γ_{KNF}.) However, since these are the only rules with the verification codes $g(r_1), \ldots, g(r_{m-1})$, we can safely rearrange the codes in the rules as: $A_1 \to B_1 \cdots B_n g(r_1)$, $A_2 \to \mathrm{rev}(g(r_1))g(r_2)$, ..., $A_{m-1} \to \mathrm{rev}(g(r_{m-2}))g(r_{m-1})$, $A_m \to \mathrm{rev}(g(r_{m-1}))$. This form then directly corresponds with (I).

Since $L_*(\Gamma_{\mathrm{KNF}}) = L_t(\Gamma_{\mathrm{KNF}}) = L(G)$, it must hold that $L_*(\Gamma) = L_t(\Gamma) = L(G)$. □

The similar result can be easily achieved for the two-component general CD grammar system where $I = \{11 \to 00,\ 0000 \to 2222\}$. Furthermore, it should be also obvious that the other properties from the previous section (close simulation and switching of components) still hold in this more general transformation.

Lastly, we introduce a modification of the above transformation that works with all general grammars. However, it is not possible to directly use our previous approach for grammars that have rules with terminals on their left-hand sides. Consequently, the resulting system may not be able to closely simulate the original general grammar. We say that a transformation is *semi-direct* if it separates terminals from the left-hand side of the rules but otherwise behaves as a direct transformation.

Theorem 15. *Let $G = (N, T, P, S)$ be a general grammar. Then, there exists its semi-direct transformation into a two-component general CD grammar system $\Gamma = (N', T, H, I, S)$ such that H is context-free, $I = \{11 \to 00,\ 0000 \to \varepsilon\}$, and $L_*(\Gamma) = L_T(\Gamma) = L(G)$.*

Proof. The proof by construction is simple. We use the core idea of step (1) from the transformation of general grammars into Kuroda normal form. First, let $T_{\mathrm{rep}} = \{a : a \in T, a \in \mathrm{alph}(\mathrm{lhs}(r)), r \in P\}$. We replace each occurrence of $a \in T_{\mathrm{rep}}$ in the rules with a new nonterminal a', and we add $a' \to a$ to P. Now, the grammar satisfies the condition for the construction from Theorem 14. Thus, the construction of Γ is completed. $\square$

Again, the same holds for the system with the evenly homogeneous component.

5. Concluding Remarks

We close this paper by formulating some remarks and open problems. First, take a closer look at the properties of all presented transformations.

- As already stated in Section 1, multi-derivations are performed so that during a derivation step, the current sentential form may be simultaneously rewritten at several positions, not just at a single position. More formally, let $\Gamma = (N, T, P_1, P_2, \ldots, P_n, S)$ be a general CD grammar system, n be a positive integer, and $u_i \Rightarrow_{P_k} v_i$, $u_i, v_i \in (N \cup T)^*$, $1 \leq i \leq n$. Then, Γ makes a direct multi-derivation step from $u_1 u_2 \cdots u_n$ to $v_1 v_2 \cdots v_n$, symbolically written as $u_1 u_2 \cdots u_n \, {}_{multi}{\Rightarrow}_{P_k} v_1 v_2 \cdots v_n$. Based on ${}_{multi}{\Rightarrow}_{P_k}$, define $L_f(\Gamma)$ by analogy with the definition of $L_f(\Gamma)$ in Section 2. Consider the systems constructed in the proofs of Theorems 1, 7, 10, 14, and 15 in Sections 3 and 4. Observe that both of their components H and I always allow the free use of multi-derivations since this cannot disturb the generation process in any way.

Finally, we propose two challenging problems.

- Consider the computationally complete general CD grammar systems presented in this paper. Can we find a different combination of even more restricted components so that the resulting systems are still computationally complete?
- Introduce new restricted transformations of general CD grammar systems so they characterize some other well-known language families, such as the families of matrix and context-sensitive languages.

Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science – LQ1602; the TAČR grant TE01020415; and the BUT grant FIT-S-17-3964. The authors thank the anonymous referees for their useful comments and suggestions.

References

[1] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, G. Paun, *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach Science Publishers, Inc., 1994.

[2] E. Csuhaj-Varjú, T. Masopust, G. Vaszil, Cooperating distributed grammar systems with permitting grammars as components. *Romanian Journal of Information Science and Technology* **12** (2009) 2, 175–189.

[3] V. Geffert, Grammars with context dependency restricted to synchronization. In: *Mathematical Foundations of Computer Science 1986*. LNCS 233, 1986, 370–378.

[4] V. Geffert, Context-free-like forms for the phrase-structure grammars. In: *Mathematical Foundations of Computer Science 1988*. LNCS 324, 1988, 309–317.

[5] V. Geffert, Normal forms for phrase-structure grammars. *RAIRO – Theoretical Informatics and Applications – Informatique Théorique et Applications* **25** (1991) 5, 473–496.

[6] F. Goldefus, T. Masopust, A. Meduna, Left-forbidding cooperating distributed grammar systems. *Theoretical Computer Science* **411** (2010) 40–42, 3661–3667.

[7] Z. Křivka, T. Masopust, Cooperating distributed grammar systems with random context grammars as components. *Acta Cybernetica* **20** (2011), 269–283.

[8] A. Meduna, *Automata and Languages: Theory and Applications*. Springer, London, 2000.

[9] A. Meduna, D. Kolář, Homogenous grammars with a reduced number of non-context-free productions. *Information Processing Letters* **81** (2002) 5, 253–257.

[10] A. Meduna, M. Švec, *Grammars with Context Conditions and Their Applications*. Wiley, 2005.

[11] A. Meduna, M. Švec, T. Kopeček, Equivalent language models that closely simulate one another and their illustration in terms of l systems. *International Journal of Computer Mathematics* **84** (2007) 11, 1555–1566.

[12] G. Rozenberg, A. Salomaa, *Handbook of Formal Languages, Vol. 1: Word, Language, Grammar*. Springer-Verlag, 1997.

[13] G. Rozenberg, A. Salomaa, *Handbook of Formal Languages, Vol. 2: Linear Modeling: Background and Application*. Springer-Verlag, 1997.

[14] W. J. Savitch, How to make arbitrary grammars look like context-free grammars. *SIAM Journal on Computing* **2** (1973) 3, 174–182.

[15] D. Wood, *Theory of Computation: A Primer*. Addison-Wesley, Boston, 1987.