Proceedings of Grammar Systems Week 2004, edited by E. Csuhaj-Varjú and Gy. Vaszil,
MTA SZTAKI, Budapest, pages 78 - 94, 2004.

CD Grammar Systems as Models of Distributed Problem Solving, Revisited

Henning Bordihn

Institut für Informatik, Universität Potsdam,
August-Bebel-Straße 89, D-14482 Potsdam, Germany
henning@cs.uni-potsdam.de

Markus Holzer

Institut für Informatik, Technische Universität München,
Boltzmannstraße 3, D-85748 Garching bei München, Germany
holzer@informatik.tu-muenchen.de

Abstract

Based on a derivation mode f for cooperating distributed (CD) grammar systems, we introduce a new form of cooperation protocol, the so called "cut-f-mode" of derivation. Intuitively, a cut-f-mode derivation, partitions (cuts) a sentential form into several subwords, where some of these subwords are distributed to the components, which derive words according to the original f-mode of derivation, and finally combines all these words together again. It is argued that these derivation modes are much closer to the original AI motivation of CD grammar systems. We investigate the cut-mode versions of the classical derivation modes $*$, $\leq k$, $= k$, and $\geq k$, of the competence based modes t and sf, as well as the cut-mode versions of the combined t-modes. It turns out that in most cases, the cut-f-mode turns out to be as most as powerful than the corresponding non-cut-mode, that is, the f-mode itself. Nevertheless, there are also some cases, where the power is even reduced to that of context-free grammars.

1 Introduction

The theory of cooperating distributed grammar systems—for an overview we refer to [7]—has become a vivid field in formal language theory since its origin in [4], with forerunner papers [10] and [1]. Cooperating distributed grammar systems have been introduced for describing, in terms of formal grammars and languages, communities of cooperating autonomous problem solving agents which use the blackboard model of problem solving. Furthermore, grammar systems are motivated in connection with the syntax of programming languages, since they can be seen as generalization of two-level substitution grammars to a multi-level concept [10]. Finally, in recent papers they have been considered as sequential counterparts of tabled Lindenmayer systems [2].

A cooperating distributed (CD, for short) grammar system consists of a finite set of (context-free) grammars, called components, performing derivation steps on a common sentential form in turns, according to some cooperation protocol. In terms of the blackboard model of problem solving, the components correspond to the independent problem solving agents, representing autonomous knowledge sources, the sentential form to the current state of the problem solving (the blackboard), where the knowledge sources can make changes, and the generated language represents the set of problem solutions (cf. [5]). Simple cooperation protocols are the so-called $*$-mode, $\leq k$-mode, $=k$-mode, or $\geq k$-mode, where a component, once started, has to perform an arbitrary number, at most k, exactly k, or at least k derivation steps, respectively. Moreover, there are two cooperation protocols which are based on the feature of competence of the problem solving agents on the current state of the problem solving: (1) In the t-mode, a component can start and has to remain deriving unless and until there is no nonterminal left in the sentential form to which one of its productions is applicable (that is, the component is not able to contribute to the problem solving any more), and (2) in the sf-mode, a component is allowed to become and has to remain active unless and until there is some nonterminal present in the sentential form which cannot be rewritten by this component (that is, the component does not possess the full competence on the current state of the problem solving).

We try to model the effect when agents contribute to the solution by solving subtasks of the whole problem. From the AI motivation this approach seems to be more adequate: In contrast to different rule-based architectures of problem solving systems, the blackboard model of problem solving emphasizes the highly opportunistic way in which the knowledge sources are applied during the problem solving process, cf. [11]. That is, there is no rigid prescription for the cooperation strategy of the knowledge sources. In terms of CD grammar systems, solving sub-tasks corresponds to working on a substring of the current sentential form. This led the authors to the concept of CD grammar systems working in the *cut-f-mode* of derivation, where f is one of the aforementioned cooperation protocols. In these *cut-f-modes*, in any derivation step the sentential form is partitioned (cut) into several substrings which can be associated to different components. In order to be as little rigid as possible, this association is done via a partial mapping, such that both some substrings of the sentential form and some components may be disregarded. Then, each component to which a substring has been associated works in (one and the same) derivation mode, more precisely, in the f-mode of derivation if the CD grammar system as a whole is driven in the *cut-f-mode*.

The overall picture that emerges from our investigates on this newly defined derivation mode is, that for some modes, the generative capacity is not affected at all, but there are also some cases, where the power is reduced to that of context-free grammars, if parts of the sentential forms can be distributed to several components in one and the same derivation step. In particular, the former is true for derivations like, e.g., $*$-, $=1$-, ≥ 1-, or ≤ 1-mode, while the latter is true for instance, for the t-mode of derivation, which describes the family of all ET0L languages in the non-cut variant. Moreover, we also find situations, where the cut-mode derivation does not become context-free. For instance, the sf-mode of derivation characterizes the

family of programmed context-free languages, while its cut-mode version describes an intermediate class between the families of recurrent programmed context-free and programmed context-free languages. The former language family equals the biologically motivated family of languages generated by ET0L systems with random context [15]. This is yet another result, where recurrent programmed languages appear in relation with CD grammar systems and unconventional derivation modes—see, e.g., [3].

The paper is organized as follows: The next section contains preliminaries, and we provide the basic definition of cut-f-mode derivations. Then in Section 3 we investigate the generative power of CD grammar systems working in these newly defined derivations modes and finally, we summarize our results and highlight the remaining open questions in Section 4.

2 Definitions

We assume the reader to be familiar with the basic notions of formal languages, as contained in [6]. In general, we have the following conventions: $\subseteq$ denotes inclusion, while $\subset$ denotes strict inclusion. The set of positive integers is denoted by $\mathbb{N}$ and the cardinality of a set M by $|M|$. Concerning our set notation, we abbreviately write $\{ X(k), Y(k) \mid P(k) \}$ instead of $\{ X(k) \mid P(k) \} \cup \{ Y(k) \mid P(k) \}$, where $X(k)$ and $Y(k)$ are objects with parameter k and $P(k)$ is some predicate on k. This notation is also extended to more than two kinds of objects having the same parameter. The empty word is denoted by λ. For $x \in V^*$, where V is some alphabet, and for $W \subseteq V$, let $|x|_W$ denote the number of occurrences of letters from W in x. If W is a singleton set $\{a\}$, we simply write $|x|_a$ instead of $|x|_{\{a\}}$. We consider two languages L_1 and L_2 to be equal if and only if $L_1 \setminus \{\lambda\} = L_2 \setminus \{\lambda\}$, and we simply write $L_1 = L_2$ in this case.

The families of languages generated by regular, context-free, context-sensitive, general type-0 Chomsky grammars, and ET0L systems are denoted by $\mathcal{L}(\mathrm{REG})$, $\mathcal{L}(\mathrm{CF})$, $\mathcal{L}(\mathrm{CS})$, $\mathcal{L}(\mathrm{RE})$, and $\mathcal{L}(\mathrm{ET0L})$, respectively. We attach $-\lambda$ in our notations if erasing rules are not permitted. Details about these families can be found in [6]. The class of finite languages is denoted by $\mathcal{L}(\mathrm{FIN})$.

A *programmed grammar* (see, for instance, [6, 13]) is a septuple

$$G = (N, T, P, S, \Lambda, \sigma, \phi),$$

where N, T, and $S \in N$ are the set of nonterminals, the set of terminals, and the start symbol, respectively. Here P is the finite set of productions of the form $\alpha \to \beta$, $\alpha, \beta \in (N \cup T)^*$, $|\alpha|_N > 0$, and Λ is a finite set of labels (for the productions in P), such that Λ can be also interpreted as a function which outputs a production when being given a label; σ and ϕ are functions from Λ into the set of subsets of Λ. Usually, the productions are written in the form

$$(r : \ \alpha \to \beta, \, \sigma(r), \, \phi(r)) \, ,$$

where r is the label of $\alpha \to \beta$. For (x, r_1) and (y, r_2) in $(N \cup T)^* \times \Lambda$ and $\Lambda(r_1) = (\alpha \to \beta)$, we write $(x, r_1) \Rightarrow (y, r_2)$ if and only if either $x = x_1 \alpha x_2$, $y = x_1 \beta x_2$ and

$r_2 \in \sigma(r_1)$, or $x = y$ and rule $\alpha \to \beta$ is not applicable to x, and $r_2 \in \phi(r_1)$. In the latter case, the derivation step is done in *appearance checking mode*. The set $\sigma(r_1)$ is called success field and the set $\phi(r_1)$ failure field of r_1. As usual, the reflexive transitive closure of $\Rightarrow$ is denoted by $\overset{*}{\Rightarrow}$. The language generated by G is defined as

$$L(G) = \{\, w \in T^* \mid (S, r_1) \overset{*}{\Rightarrow} (w, r_2) \text{ for some } r_1, r_2 \in \Lambda \,\}.$$

The family of languages generated by programmed grammars containing only context-free core rules is denoted by $\mathcal{L}(\mathrm{P}, \mathrm{CF}, \mathrm{ac})$. We replace CF by $\mathrm{CF} - \lambda$ in that notation if erasing rules are forbidden. When no appearance checking features are involved, i.e., $\phi(r) = \emptyset$ for each label $r \in \Lambda$, we are led to the families $\mathcal{L}(\mathrm{P}, \mathrm{CF})$ and $\mathcal{L}(\mathrm{P}, \mathrm{CF} - \lambda)$. A special variant of programmed grammars are recurrent programmed grammars introduced in [15]. A programmed context-free grammar G is a *recurrent programmed context-free grammar* if for every $p \in \Lambda$ of G, if $\phi(p) = \emptyset$, then $p \in \sigma(p)$, and if $\phi(p) \neq \emptyset$, then $p \in \sigma(p) = \phi(p)$. The corresponding language families are denoted by $\mathcal{L}(\mathrm{RP}, \mathrm{CF}, \mathrm{ac})$ and $\mathcal{L}(\mathrm{RP}, \mathrm{CF} - \lambda, \mathrm{ac})$. When no appearance checking features are involved, i.e., $\phi(r) = \emptyset$ for each label $r \in \Lambda$, we omit ac in that notation, again.

We use bracket notations like $\mathcal{L}(\mathrm{P}, \mathrm{CF}[-\lambda]) \subset \mathcal{L}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac})$ in order to say that the statement holds both in case of forbidding erasing productions and in the case of admitting erasing productions (neglecting the bracket contents).

Moreover, we need some more notation on grammar, namely we have to define the *finite index* restriction. Loosely speaking, the index of a grammar is the maximal number of nonterminals simultaneously appearing in a sentential form during a terminating derivation, considering the most economical derivation for each string. The finite index property is defined as follows: Let G be an arbitrary grammar type (e.g., context-free grammars, programmed grammars with or without appearance checking, etc.), and let N, T, and $S \in N$ be its nonterminal alphabet, terminal alphabet, and axiom, respectively. For a derivation

$$D : S = w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n = w$$

for w in G, with $w \in T^*$, we set

$$ind(D, G) = \max\{\, |w_i|_N \mid 1 \leq i \leq n \,\},$$

and, for $w \in T^*$, we define

$$ind(w, G) = \min\{\, ind(D, G) \mid D \text{ is a derivation for } w \text{ in } G \,\}.$$

The *index of grammar* G is defined as

$$ind(G) = \sup\{ ind(w, G) \mid w \in L(G) \,\}.$$

For a language L in the family $\mathcal{L}(X)$ of languages generated by grammars of type X we define

$$ind_X(L) = \inf\{\, ind(G) \mid L(G) = L \text{ and } G \text{ is of type } X \,\}.$$

For a family $\mathcal{L}(X)$, we set

$$\mathcal{L}_n(X) = \{ L \mid L \in L(X) \text{ and } ind_X(L) \leq n \}$$

for $n \in \mathbb{N}$, and $\mathcal{L}_{fin}(X) = \cup_{n \geq 1} \mathcal{L}_n(X)$.

A *cooperating distributed (CD) grammar system* of degree n, with $n \geq 1$, is an $(n+3)$-tuple $G = (N, T, P_1, P_2, \ldots, P_n, S)$, where N, T are disjoint alphabets of nonterminals and terminals, respectively, $S \in N$, and $P_1, P_2, \ldots, P_n$ are context-free rule sets called components. For $1 \leq i \leq n$, let

$$\mathrm{dom}(P_i) = \{ A \in N \mid \text{there is a word } v \text{ such that } A \to v \in P_i \}$$

denote the set of all nonterminals which can be rewritten by the component P_i. For $x, y \in (N \cup T)^*$ and $1 \leq i \leq n$, we write $x \Rightarrow_i y$ if and only if $x = x_1 A x_2$ and $y = x_1 z x_2$ for some $A \to z \in P_i$. Hence, subscript i refers to the component to be used. By $\Rightarrow_i^{\leq k}$, $\Rightarrow_i^{=k}$, $\Rightarrow_i^{\geq k}$, $\Rightarrow_i^*$, for $k \geq 1$, we denote a derivation consisting of at most k steps, exactly k steps, at least k steps, an arbitrary number of steps, respectively, executed by component P_i. Furthermore, we write $x \Rightarrow_i^t y$ if and only if $x \Rightarrow_i^* y$ and there is no z such that $y \Rightarrow_i z$. Moreover, $x \Rightarrow_i^{sf} y$ if and only if $x \Rightarrow_i^* x'$, $x' \Rightarrow_i y$, and P_i is sf-competent on x' but is *not* sf-competent on y, where a component P_i is said to be *sf-competent* on a word x if and only if (1) $x = u_0 A_1 u_1 A_2 u_2 \ldots u_{m-1} A_m u_m$ with $m \geq 0$, $u_j \in T^*$, for $0 \leq j \leq m$, and $A_j \in N$, for $1 \leq j \leq m$, and (2) for each j, for $1 \leq j \leq m$, there is a production $A_j \to w_j$ in P_i. Note that the definition of the derivation relation implies that component P_i is sf-competent on x and on all intermediate sentential forms in the derivation $x \Rightarrow_i^* x'$, too.

Combining the former three modes with the t-mode requirement we obtain the modes $(t \wedge \leq k)$, $(t \wedge = k)$, and $(t \wedge \geq k)$ which are defined as follows—see, e.g., [8, 9]: There exists a derivation which satisfies both properties in common, e.g., $x \Rightarrow_i^{(t \wedge \leq k)} y$ if and only if there exists an m-step derivation from x to y using P_i such that $m \leq k$ and there is no z such that $y \Rightarrow_i z$.

Applying our idea on distributed problem solving to one of the aforementioned modes

$$f \in \{*, t, sf\} \cup \{\leq k, = k, \geq k \mid k \in \mathbb{N}\} \cup \{ (t \wedge \leq k), (t \wedge = k), (t \wedge \geq k) \mid k \in \mathbb{N} \}$$

leads us to the *cut-f-mode*, f_c-mode for short, which is defined as follows: $x \Rightarrow^{f_c} y$ if and only if

1. $x = x_0 x_1 \ldots x_m$ with $m \geq 0$, $x_i \in (N \cup T)^*$, for $0 \leq i \leq m$,
2. there is a partial injective mapping $\rho : \{0, 1, \ldots m\} \hookrightarrow \{1, 2, \ldots, n\}$ such that $y_i = x_i$, if $i \notin \mathrm{dom}(\rho)$, and $y_i = z_i$, if $x_i \Rightarrow_{\rho(i)}^f z_i$, and
3. $y = y_0 y_1 \ldots y_m$.

Here $\mathrm{dom}(\rho) = \{ i \mid \rho(i) \text{ is defined} \}$ denotes the set of indices in the decomposition $x = x_0 x_1 \ldots x_m$ to which some component is associated by ρ. Let

$$D = \{*, t, sf\} \cup \{\leq k, = k, \geq k \mid k \in \mathbb{N}\} \cup \{ (t \wedge \leq k), (t \wedge = k), (t \wedge \geq k) \mid k \in \mathbb{N} \}$$

and $D_c = \{ f_c \mid f \in D \}$. The language generated by the CD grammar system

$$G = (N, T, P_1, P_2, \ldots, P_n, S)$$

in the *non-cut* f-mode with $f \in D$ is defined as

$$L_f(G) = \{ w \in T^* \mid S \Rightarrow_{i_1}^{f} w_1 \Rightarrow_{i_2}^{f} \cdots \Rightarrow_{i_{m-1}}^{f} w_{m-1} \Rightarrow_{i_m}^{f} w_m = w \text{ with } m \geq 1,\ 1 \leq i_j \leq n, \text{ for } 1 \leq j \leq m \}$$

whereas the the language generated in the cut-f-mode with $f_c \in D_c$ is defined as

$$L_{f_c}(G) = \{ w \in T^* \mid S \Rightarrow^{f_c} w_1 \Rightarrow^{f_c} \cdots \Rightarrow^{f_c} w_{m-1} \Rightarrow^{f_c} w_m = w \text{ with } m \geq 1 \}.$$

If $f \in D \cup D_c$, then the family of languages generated by [λ-free] context-free CD grammar systems working in f-mode, is denoted by $\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f)$.

In order to clarify our definitions, we give two examples.

Example 1. *Let G be the CD grammar system $G = (N, T, P_1, P_2, P_3, S)$ with nonterminals $N = \{S, S', A, A', B, B'\}$, terminals $T = \{a, b, c\}$, and the production sets*

$$\begin{aligned} P_1 &= \{S \to S', S' \to AB\} \\ P_2 &= \{A \to aA'b, B \to B'c\} \\ P_3 &= \{A' \to A, B' \to B\} \\ P_4 &= \{A \to ab, B \to c\}. \end{aligned}$$

It is easy to see that running G in the $=2$-mode results in the non-context-free language $L_1 = \{ a^n b^n c^n \mid n \geq 1 \}$, since the only way to start the derivation is to use production set P_1 leading to the sentential form AB within two steps. Then, for all natural numbers $n \geq 0$, we find

$$a^n A b^n B c^n \Rightarrow_2^{=2} a^{n+1} A' b^{n+1} B' c^{n+1} \Rightarrow_3^{=2} a^{n+1} A b^{n+1} B c^{n+1}$$

or the terminating derivation

$$a^n A b^n B c^n \Rightarrow_4^{=2} a^{n+1} b^{n+1} c^{n+1}.$$

This shows the stated claim. Observe, that except for the axiom and the terminal word, all intermediate sentential forms contain exactly two *nonterminals. When considering the $(=2)_c$-mode a similar reasoning applies, since no production set is successfully applicable to a sentential form containing* one *nonterminal only, except from the application of P_1 to the axiom S. Thus, a production set P_i with $2 \leq i \leq 4$ will be successful only, if two appropriate nonterminals in the sentential form are present. Therefore, the only possible derivations in $(=2)_c$-mode are the derivations shown above.*

When comparing the t-mode and its cut-version we find that in the first mode the CD grammar system G generates the language $\{ a^n b^n c^n \mid n \geq 1 \}$ while in t_c-mode only the context-free language $\{ a^n b^n c^m \mid n, m \geq 1 \}$ will be obtained. The latter fact is obvious, since due to the cutting of the sentential form when running G in the t_c-mode the derivation of the two nonterminals will be decoupled. Therefore, the number of a's and b's are independently from the number of c's in the sentential form. Thus, we obtain the context-free language mentioned above.

Let us come to our second example.

Example 2. *The language* $L_1 = \{ a^n b^n c^n \mid n \geq 1 \}$ *is also generated by the CD grammar system* $G = (\{S, A, A', B, B'\}, \{a, b, c\}, P_1, P_2, P_3, P_4, S)$ *with the production sets*

$$\begin{aligned} P_1 &= \{S \to AB\} \\ P_2 &= \{A \to aA'b, A' \to A', B \to B'c\} \\ P_3 &= \{A' \to A, B' \to B, B \to B\} \\ P_4 &= \{A \to ab, B \to c\}, \end{aligned}$$

if it is driven in either sf- or sf_c*-mode. For the sf-mode, this is seen as follows. Every derivation starts with* P_1 *yielding* AB*. Starting with a sentential form* $a^n Ab^n Bc^n$*,* $n \geq 0$*, the derivation can terminate to* $a^{n+1}b^{n+1}c^{n+1}$ *with the help of* P_4 *or* P_2 *can be applied. Then, we have to distinguish two cases: (1)* $a^n Ab^n Bc^n \Rightarrow_2^{sf} a^{n+1}A'b^{n+1}B'c^{n+1}$ *by replacing* A *first and (2)* $a^n Ab^n Bc^n \Rightarrow_2^{sf} a^n Ab^n B'c^{n+1}$ *by applying* $B \to B'c$*, first and only. In the second case, the derivation is blocked, since no of the components is competent on the obtained sentential form. In the first case,* P_3 *is the only applicable component. If* A' *is replaced by* A *first, then the derivation is blocking again, since no component is competent both on* A *and on* B'*. Hence, first* B' *and then* A' *must be rewritten, yielding* $a^{n+1}Ab^{n+1}Bc^{n+1}$*. In conclusion,* $L_{sf}(G) = L_1$*.*

The fact $L_{sf_c}(G) = L_1$ *can similarly be seen if one takes into consideration the following observations:*

1. *P_2 cannot work on a string containing an A but no B since it would not terminate.*
2. *If P_2 deals with a string containing a B and no A, then the B' contained in the resulting string cannot be replaced, since P_3 can only terminate if it starts off with a string containing an A', but this A' cannot be produced by P_2 after B has been replaced before (see item 1.).*
3. *Analogously, P_3 can successfully work neither on a string containing a B' but no A' nor on a string containing an A' but no B'.*

3 The Power of Cut-Derivations in CD Grammar Systems

In this section we focus on the power of the newly defined CD grammar systems variants with the well known "classical" language families as introduced in the preceding section. First, we recall some known facts about the generative capacity of CD grammar systems working in the non-cut modes, see, for example, [7]. For $f \in \{*\} \cup \{=1, \geq 1\} \cup \{\leq k \mid k \geq 2\}$, CD grammar systems with context-free components characterize exactly the context-free languages, that is, $\mathcal{L}(\mathrm{CF}) = \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f)$, while those CD grammar systems working in f-mode with $f \in \{ = k, \geq k \mid k \geq 2 \}$

induce a proper superset of the family of context-free languages which is contained in the family of all context-free programmed languages without appearance checking. In other words, $\mathcal{L}(\mathrm{CF}) \subset \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f) \subseteq \mathcal{L}(\mathrm{P}, \mathrm{CF}[-\lambda])$, if $f \in \{ =k, \geq k \mid k \geq 2 \}$. It is unknown whether the second inclusion is strict. The next theorem shows that for the aforementioned derivation modes, there is no difference to its cut version.

Theorem 3. *Let* $f \in \{*\} \cup \{\leq k, =k, \geq k \mid k \in \mathbb{N}\}$. *Then*

$$\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f) = \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f_c).$$

Proof. Let $G = (N, T, P_1, P_2, \ldots, P_n, S)$ be a CD grammar system. We show that $L_f(G) = L_{f_c}(G)$. For the inclusion from left to right it is sufficient to argue that any f-mode derivation $x \Rightarrow_f y$ can be simulated by $x \Rightarrow^{f_c} y$ choosing $x = x_0$ and $\rho : \{0\} \hookrightarrow \{1, 2, \ldots, n\}$ to be the constant function $\rho(0) = i$. Conversely we argue that a f_c-derivation step can be sequentialized. This is seen as follows: Let $x \Rightarrow^{f_c} y$ with (1) $x = x_0 x_1 \ldots x_m$, (2) the injective partial mapping $\rho : \{0, 1, \ldots, m\} \hookrightarrow \{1, 2, \ldots, n\}$, and (3) $y = y_0 y_1 \ldots y_m$ fulfilling $y_i = x_i$, if $i \notin \mathrm{dom}(\rho)$, and $y_i = z_i$, if $x_i \Rightarrow^f_{\rho(i)} z_i$. To simplify presentation, let $\mathrm{dom}(\rho) = \{i_1, i_2, \ldots, i_r\}$ with $0 \leq i_1 \leq i_2 \leq \ldots \leq i_r \leq m$. Observe, that $y_i = x_i$ if $i \in \{0, 1, \ldots, m\} \setminus \mathrm{dom}(\rho)$. Then we find an f-mode derivation

$$\begin{aligned}
x &= y_0 y_1 \ldots y_{i_1 - 1} x_{i_1} y_{i_1 + 1} \ldots y_{i_2 - 1} x_{i_2} y_{i_2 + 1} \ldots y_m \\
&\Rightarrow^f_{\rho(i_1)} \quad y_0 y_1 \ldots y_{i_1 - 1} y_{i_1} y_{i_1 + 1} \ldots y_{i_2 - 1} x_{i_2} y_{i_2 + 1} \ldots y_m \\
&\Rightarrow^f_{\rho(i_2)} \quad y_0 y_1 \ldots y_{i_2 - 1} y_{i_2} y_{i_2 + 1} \ldots y_m \\
&\Rightarrow^f_{\rho(i_3)} \quad \ldots \\
&\Rightarrow^f_{\rho(i_r)} \quad y_0 y_1 \ldots y_m = y,
\end{aligned}$$

which simulates the original f_c-mode derivation. This proves $L_f(G) = L_{f_c}(G)$. □

An immediate consequence of the above given theorem is the following corollary.

Corollary 4. *1. If* $f \in \{*\} \cup \{=1, \geq 1\} \cup \{\leq k \mid k \geq 2\}$, *then*

$$\mathcal{L}(\mathrm{CF}) = \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f_c).$$

2. If $f \in \{ =k, \geq k \mid k \geq 2 \}$, *then*

$$\mathcal{L}(\mathrm{CF}) \subset \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f_c) \subseteq \mathcal{L}(\mathrm{P}, \mathrm{CF}[-\lambda]).$$

Next we turn our attention to the t-mode. For CD grammar systems working in the t-mode it was shown (see [5]) that

$$\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], t) = \mathcal{L}(\mathrm{ET0L})$$

and thus they build a strict superset of the family of context-free languages. In contrast to this ET0L characterization, we show that CD grammar systems running in the cut-t-mode, characterize only the family of context-free languages.

Theorem 5. $\mathcal{L}(\mathrm{CF}) = \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], t_c)$.

Proof. The inclusion from left to right holds since any context-free grammar can be interpreted as a CD grammar system with one component, generating the same language in t_c-mode. For the converse inclusion, let a CD grammar system $G = (N, T, P_1, P_2, \ldots, P_n, S)$ be given. For $1 \leq i \leq n$, set $N^{(i)} = \{ A^{(i)} \mid A \in \mathrm{dom}(P_i) \}$ and let $g_i : N \cup T \to N \cup T \cup N^{(i)}$ be the morphism defined by $g_i(X) = X^{(i)}$ if $X \in \mathrm{dom}(P_i)$ and $g_i(X) = X$ otherwise. Consider the CD grammar system

$$G' = (N', T, P_1', P_2', \ldots P_n', S)\,,$$

where $N' = N \cup \bigcup_{i=1}^{n} N^{(i)}$ and, for $1 \leq i \leq n$,

$$P_i' = \{ A \to g_i(v),\, A^{(i)} \to g_i(v) \mid A \to v \in P_i \}\,.$$

Observe that the axiom of G' is contained in N and that, for any string $x \in (N')^*$ and each $1 \leq i \leq n$, if $x \Rightarrow_i^t y$ according to G', then $y \in (N \cup T)^*$. Therefore, we find $L_{t_c}(G') = L_{t_c}(G)$.

Now, it is sufficient to prove that the context-free grammar

$$G'' = (N', T, \bigcup_{i=1}^{n} P_i', S)$$

generates the language $L(G')$. This is seen as follows. The inclusion $L(G') \subseteq L(G'')$ trivially holds, since one can mimic any derivation of G' by the context-free grammar G''. For $L(G'') \subseteq L(G')$ observe the following fact: consider an arbitrary derivation step $x \Rightarrow y$ according G'', where $x = z_1 A z_2$, $y = z_1 v z_2$, for some $z_1, z_2 \in (N \cup T)^*$, $A \in N$, and $A \to v$ is originally in P_i'. Note that x does not contain any symbol of the form $B^{(i)}$, $1 \leq i \leq n$. If y is a string over $N \cup T$, as well, then $x \Rightarrow^{t_c} y$ according to G' using the decomposition $x = x_0 x_1 x_2$ with $x_0 = z_1$, $x_1 = A$, $x_2 = z_2$, and the association ρ of components $\rho(1) = i$, keeping $\rho(0)$ and $\rho(2)$ undefined. If $|y|_{N^{(i)}} > 0$, then G'' has to continue the derivation earlier or later by replacing all the symbols from $N^{(i)}$ by some productions which originally come from P_i', too. Since G'' is a context-free grammar, we can assume without loss of generality that (1) these replacements are performed immediately by G'', (2) those replacements are continued until a string y' is obtained with $|y'|_{N^{(i)}} = 0$, and (3) no symbols $B \notin N^{(i)}$ are replaced during this derivation $y \overset{*}{\Rightarrow} y'$. Then $x \Rightarrow_i^{t_c} y'$ holds by the same arguments as given above. Now, one easily proofs by induction that any derivation of G'' can be simulated by G' in the t_c-mode, and since G'' and G' have the same axiom, we have $L(G'') \subseteq L(G')$. □

It is known that the other competence based derivation protocol, namely the *sf*-mode, is more powerful than the *t*-mode. In [10] the equalities

$$\mathcal{L}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac}) = \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], sf).$$

have been shown. Moreover, it is known that $\mathcal{L}(\mathrm{P}, \mathrm{CF}, \mathrm{ac}) = \mathcal{L}(\mathrm{RE})$ and that $\mathcal{L}(\mathrm{ET0L}) \subset \mathcal{L}(\mathrm{P}, \mathrm{CF} - \lambda, \mathrm{ac}) \subset \mathcal{L}(\mathrm{CS})$, see, e.g., [6].

Concerning the cut-*sf*-mode, it is already seen from Example 2 that it is more powerful than the cut-*t*-mode, as well, since also non-context-free languages can be described. In fact, we even obtain the following result.

Lemma 6. $\mathcal{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac}) \subseteq \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], sf_c)$.

Proof. First, let λ-productions be permitted. Let $G = (N, T, P, S, \Lambda, \sigma, \phi)$ be some recurrent programmed grammar with context-free core rules. From G we construct an CD grammar system $G' = (N', T, P_1, P_2, \ldots, P_n, S')$ such that $L_{sf_c}(G') = L(G)$ as follows. If $p \in \Lambda$ is a label to which $A \to v$ is associated in G, then let $N_p = \{p, p', p'', A_p, R_p\}$ be a set of new symbols and set $N' = \{S', R\} \cup N \cup \bigcup_{p \in \Lambda} N_p$, where R and S' are new symbols, again. Any derivation of G' is initiated by application of the component

$$P_{init} \quad = \quad \{\, S' \to pSR \mid p \in \Lambda \,\}\,.$$

With each production in P,

$$(p:\ A \to v,\ \{s_1, s_2, \ldots, s_k\},\ \{r_1, r_2, \ldots, r_m\})\,,$$

we associate the four new components

$$\begin{aligned}
P_{p,1} &= \{A \to A_p, p \to p', p' \to p'\} \cup \{\, B \to B \mid B \in N \,\} \\
P_{p,2} &= \{A_p \to v\} \cup \{\, p' \to s_i \mid 1 \le i \le k \,\} \cup \{\, B \to B \mid B \in N \,\} \\
P_{p,3} &= \{R \to R_p, R_p \to R_p, p \to p''\} \cup \{\, B \to B \mid B \in N \setminus \{A\} \,\} \\
P_{p,4} &= \{R_p \to R\} \cup \{\, p'' \to r_i, r_i \to r_i \mid 1 \le i \le m \,\} \cup \{\, B \to B \mid B \in N \setminus \{A\} \,\}
\end{aligned}$$

Finally, there is a terminating component

$$P_{term} \quad = \quad \{R \to \lambda\} \cup \{\, p \to \lambda \mid p \in \Lambda \,\}$$

The only component applicable to the axiom S' is P_{init}. Now, in every nonterminal sentential form, either R or R_p appears as right marker and one label symbol $p \in \Lambda$ or its primed or double primed version appears as left marker. Given a sentential form $p\alpha R$, $\alpha \in (N \cup T)^*$, the components $P_{p,1}$ and $P_{p,2}$ can be used in order to simulate the successful application of the production with label p of G, yielding a string $s_i \beta R$, $s_i \in \sigma(p)$ and $\beta \in (N \cup T)^*$, and $P_{p,3}$ and $P_{p,4}$ can simulate its application in appearance checking mode, yielding $r_i \alpha R$, $r_i \in \phi(p)$. For this, the complete sentential form is given to the components. When a string of the form pwR with $w \in T^*$ is obtained, the component P_{term} can be applied yielding w. This proves $L(G) \subseteq L_{sf_c}(G')$.

For the converse inclusion observe the following. A terminal word can only be obtained by applying P_{term} to a sentential form in $\Lambda T^* \{R\}$. Starting off with a sentential form $\alpha = p\beta R$, $p \in \Lambda$ and $\beta \in (N \cup T)^*$, neither the components $P_{q,2}$ nor $P_{q,4}$, $q \in \Lambda$, are applicable to any substring of α, since the presence of some symbol q or R_q is needed, respectively, in order to stop deriving. Therefore, we have to distinguish the following two cases:

1. Some $P_{q,3}$ becomes active first. If a component $P_{q,3}$ is applied to a substring α' of α, this component can become inactive only after the production $q \to q''$ has been used. Therefore, $q = p$ has to hold. Since p'' can be rewritten only with the help of $P_{p,4}$ and $P_{p,4}$ can stop deriving only by application of $R_p \to R$, the

symbol R_p must have been introduced when $P_{p,3}$ was active. Thus, $\alpha' = \alpha$ has to hold. In conclusion, $|\alpha|_A = 0$ since $P_{p,3}$ is not sf-competent on α otherwise. Furthermore, p'' must be replaced together with R_p during one and the same application of $P_{p,4}$. Therefore,

$$\alpha = p\beta R \Rightarrow^{sf_c}_{p,3} p''\beta R_p \Rightarrow^{sf_c}_{p,4} r_i\beta R\,,$$

$r_i \in \phi(p)$, is the only successful continuation of such derivation, simulating the application of the production with label p in appearance checking mode. Note that some $P_{q,1}$ may not be applied to some substring of $p''\beta R_p$ since it would introduce a symbol A_q such that $P_{p,4}$ cannot become active until A_q is rewritten again, but rewriting A_q can only be done with the help of $P_{q,2}$ which needs the presence of q' in order to finish its work in sf-mode of derivation.

2. Some $P_{q,1}$ becomes active first. Consider a derivation

$$\alpha \Rightarrow^{sf_c}_{q_1,1} \alpha_1 \Rightarrow^{sf_c}_{q_2,1} \alpha_2 \ldots \Rightarrow^{sf_c}_{q_\ell,1} \alpha_\ell$$

for some $\ell \geq 1$. Then we have $|\alpha_\ell|_{A_{q_i}} > 0$ for all $1 \leq i \leq \ell$. Let $\alpha_\ell = q_\ell u_0 A_{q_{i_1}} u_1 A_{q_{i_2}} u_2 \ldots A_{q_{i_\ell}} u_\ell$, where $u_j \in (N \cup T)^*$ for $1 \leq j \leq \ell$. The only possibility to get rid of a symbol A_{q_i} is the application of $P_{q_i,2}$ which can stop deriving on a sentential form only if q_i' is present. Therefore and since the application of some $P_{q,3}$ or $P_{q,4}$ must kept excluded from such derivations (see the arguments given for the first case), a string in $\Lambda(N \cup T)^*\{R\}$ can be obtained only if the following derivation is possible (due to the order of the symbols in α_ℓ):

$$\alpha_\ell \Rightarrow^{sf_c}_{q_{i_1},2} \gamma_1 \Rightarrow^{sf_c}_{q_{i_2},2} \gamma_2 \Rightarrow^{sf_c}_{q_{i_3},2} \ldots \Rightarrow^{sf_c}_{q_{i_\ell},2} \gamma_\ell\,,$$

where $q_{i_1} = q_\ell$ and $q_{i_{j+1}} \in \sigma(q_{i_j})$, for $1 \leq j < \ell$, hold. This simulates the successful applications of the corresponding productions of G.

Except from the fact that the two phases can be merged when components $P_{q,1}$ and when components $P_{q,2}$ are applied, no further terminating derivations are possible. Hence, $L_{sf_c}(G') \subseteq L(G)$, and the proof is finished for the case that λ-productions are allowed.

Let us remark that, even if the CD grammar system were forced, by some appropriate colouring of the symbols, to apply the component $P_{p,2}$ immediately after $P_{p,1}$ has been used, one would not be able to control how many symbols A are marked as A_p and then replaced with v. Therefore, the feature $p \in \sigma(p)$ for all $p \in \Lambda$ is an evident constraint, here.

For the λ-free case we argue as follows: By standard arguments the family $\mathcal{L}(\mathrm{CD}, \mathrm{CF} - \lambda, sf_c)$ is closed under union and embraces the finite languages. Let $L \subseteq T^*$ be in $\mathcal{L}(\mathrm{RP}, \mathrm{CF} - \lambda, \mathrm{ac})$, then

$$L = \bigcup_{a,b \in T} (a \cdot \delta_{a,b}(L) \cdot b) \cup (L \cap T^2) \cup (L \cap T) \cup (L \cap \{\lambda\}),$$

where $\delta_{a,b}(L) = \{ w \in T^+ \mid awb \in L \}$. Since L is in $\mathcal{L}(\mathrm{RP}, \mathrm{CF} - \lambda, \mathrm{ac})$, the language $\delta_{a,b}(L)$ is in $\mathcal{L}(\mathrm{RP}, \mathrm{CF} - \lambda, \mathrm{ac})$ due to closure properties of that family under left and right derivatives. The closure under derivatives is obvious, since $\mathcal{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac})$ is a [full] AFL as shown in [8]. Thus, for the proof of the present assertion, it is sufficient to show that $\delta_{a,b,c}(L) \cdot abc$ is in $\mathcal{L}(\mathrm{CD}, \mathrm{CF} - \lambda, sf_c)$, provided that $\delta_{a,b,c}(L)$ is in $\mathcal{L}(\mathrm{RP}, \mathrm{CF} - \lambda, \mathrm{ac})$.

To this end, it is sufficient to exchange the terminating component P_{term} in the above construction with

$$Q_{term} \quad = \quad \{R \to b\} \cup \{ p \to a \mid p \in \Lambda \}$$

and the rest of the proof is given by the same arguments as in the first case. □

Any CD grammar system working in the sf_c-mode can be simulated by some Turing machine and, if λ-rules are forbidden, by some linear bounded automaton. Thus, together with the known facts about context-free CD grammar system working in the sf-mode, the following corollary is obtained.

Corollary 7. *1.* $\mathcal{L}(\mathrm{RP}, \mathrm{CF}, \mathrm{ac}) \subseteq \mathcal{L}(\mathrm{CD}, \mathrm{CF}, sf_c) \subseteq \mathcal{L}(\mathrm{P}, \mathrm{CF}, \mathrm{ac}) = \mathcal{L}(\mathrm{RE})$

2. $\mathcal{L}(\mathrm{RP}, \mathrm{CF} - \lambda, \mathrm{ac}) \subseteq \mathcal{L}(\mathrm{CD}, \mathrm{CF} - \lambda, sf_c) \subseteq \mathcal{L}(\mathrm{CS})$

It is left open which of the inclusions in this corollary are strict. Finally we turn our attention to the combined modes. First let us summarize what is known for CD grammar systems working in non-cut modes. In case of the $(t \wedge \geq k)$-mode it was shown in [8] that

$$\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], (t \wedge \geq 1)) \quad = \quad \mathcal{L}(\mathrm{ET0L})$$

and

$$\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], (t \wedge \geq k)) \quad = \quad \mathcal{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac}) \quad \text{if } k \geq 2,$$

and in [9] it was shown that

$$\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f) = \mathcal{L}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda]),$$

for each $f \in \{ (t \wedge = k), (t \wedge \leq k) \mid k \geq 1 \}$, where $\mathcal{L}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda])$ denotes the family of languages of finite index generated by programmed context-free grammars without appearance checking. Loosely speaking, a grammar has finite index if the number of nonterminals in a derivation is bounded by a constant. For the definition of the *finite index* property see, e.g., [6].

Theorem 8. *If* $f \in \{ (t \wedge \leq k), (t \wedge = k), (t \wedge \geq k) \mid k \geq 1 \}$, *then*

$$\mathcal{L}(\mathrm{CF}) \subseteq \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f_c).$$

Proof. Let $G = (N, T, P, S)$ be a context-free grammar. In the remainder we restrict ourselves to the case $k = 1$. The result generalizes to arbitrary k, by using the prolongation technique, as elaborated in [8].

We construct a CD grammar system G' with nonterminals

$$N' = N \cup \{ A' \mid A \in N \},$$

where the unions are disjoint, with terminals T and axiom S. Define the homomorphism $h : (N \cup T)^* \to (N' \cup T)^*$ as follows: Let $h(A) = A'$ for $A \in N$ and $h(a) = a$ otherwise. Then we construct two production sets

$$P_1 = \{ A \to h(w) \mid (A \to w) \in P \} \quad \text{and} \quad P_2 = \{ A' \to A \mid A \in N \}.$$

This completes the description of the CD grammar system G'. It is easy to see that $L_{f_c}(G') = L(G)$, for $f \in \{(t \wedge \leq 1), (t \wedge = 1), (t \wedge \geq 1)\}$. This proves the claim. □

Corollary 9. *If* $f \in \{ (t \wedge = k), (t \wedge \geq k) \mid k \geq 2 \}$, *then*

$$\mathcal{L}(\mathrm{CF}) \subset \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f_c).$$

Proof. The inclusion follows from Theorem 8. The strictness follows from the CD grammar system specified in Example 1 when running in cut-$(t\wedge=2)$ or cut-$(t\wedge\geq k)$, for $k \geq 2$, since the non-context-free language $\{ a^n b^n c^n \mid n \geq 1 \}$ is generated. In case of the cut-$(t\wedge=k)$-mode, for $k > 2$, the rules in the grammar of the above mentioned example have to be adapted accordingly with the prolongation technique in order to ensure that both nonterminals have to be presented during a $(t \wedge = k)$-derivation step. □

Since the $(t \wedge \geq 1)$-mode trivially coincides with the t-mode, we obtain the following corollary because of Theorem 5.

Corollary 10. $\mathcal{L}(\mathrm{CF}) = \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], (t \wedge \geq 1)_c)$.

For the cut-$(t \wedge \geq k)$- and cut-$(t \wedge = k)$-mode in general we find the following situation. The below given theorem shows that with some cut-modes one can do even programmed context-free language of finite index.

Theorem 11. *If* $f \in \{ (t \wedge = k), (t \wedge \geq k) \mid k \geq 2 \}$, *then*

$$\mathcal{L}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda]) \subset \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f_c).$$

Proof. We first show the inclusion. By a standard argument the involved CD grammar systems language family is closed under union and embraces the finite languages. Let $L \subseteq T^*$ be a language in $\mathcal{L}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda])$, then

$$L = \bigcup_{a \in T} (a \cdot \delta_a(L) \cap L) \cup (L \cap T) \cup (L \cap \{\lambda\}).$$

Since L is in $\mathcal{L}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda])$, language $\delta_a(L) = \{ w \in T^+ \mid aw \in L \}$ is in $\mathcal{L}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda])$ due to the closure properties of that family under derivatives. Thus, it is sufficient for the proof of the present assertion to show that $\{a\} \cdot \delta_a(L)$ is in $\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f_c)$, for $f \in \{ (t \wedge = k), (t \wedge \geq k) \mid k \geq 2 \}$, provided that $\delta_a(L)$ is a programmed context-free language of finite index. In the remainder we restrict ourselves to the case $k = 2$. The result generalizes to arbitrary k, by using the prolongation technique, as elaborated

Let $G = (N, T, P, S, \Lambda, \sigma, \phi)$ be a programmed grammar of finite index generating $\delta_a(L)$. Without loss of generality we assume that $N \cap \Lambda = \emptyset$, that for every rule $\Lambda(p) = (A \to w, \sigma, \emptyset)$ nonterminal A does not appear in w, and $p \notin \sigma(p)$. Moreover, due to the finite index restriction we may assume that every nonterminal appears at most once in any derivable sentential form.

We construct a CD grammar system G' with nonterminals

$$N' = N \cup \{ p, p' \mid p \in \Lambda \} \cup \{S', S''\},$$

where the unions are disjoint, with terminals T and axiom S'. To start the derivation, we use S' as axiom and the component

$$P_{init} = \{S' \to S''\} \cup \{ S'' \to pS \mid p \in \Lambda \}.$$

Then for each rule $\Lambda(p) = (A \to w, \sigma, \emptyset)$ we construct the component

$$P_p = \{ p \to q \mid q \in \sigma(p) \} \cup \{A \to w\}.$$

Note that the $(t \wedge = 2)$-mode enforces the application of at exactly two rules, which must be the rule for the label p and the rule for the nonterminal A, because of the requirement that every nonterminal appears at most once in any derivable sentential form. Therefore the $(t \wedge = 2)_c$-derivation cuts the sentential form in such a way that only *one* production set will be applicable.

Finally, to terminate the derivation, process the component

$$P_{term} = \{ p \to p', p' \to a \mid p \in \Lambda \}$$

is used. Observe, that after an application of P_{term} no other component, and in particular no component P_p, is successfully applicable to the sentential form because of the requirement on the appearance of nonterminals in any derivation. This completes the description of the CD grammar system G'.

Obviously, the constructed grammar system G' simulates the programmed context-free grammar G of finite index correctly and generates the language $\{a\} \cdot \delta_a(L)$. Moreover, note that the CD grammar system has λ-productions only if the programmed grammar has λ-productions.

The strictness immediately follows from Theorem 8 and the fact that the Dyck language (even over one pair of parentheses) cannot be generated by programmed context-free grammar of finite index as shown in [14]. □

Finally, we consider the cut-$(t \wedge = 1)$- and cut-$(t \wedge \leq k)$-mode, for $k \geq 1$. For all these modes we obtain a characterization of the context-free languages.

Theorem 12. *If* $f \in \{ (t \wedge \leq k) \mid k \geq 1 \} \cup \{(t \wedge = 1)\}$, *then*

$$\mathcal{L}(\mathrm{CF}) = \mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f_c).$$

Proof. The inclusion from left to right was already shown in Theorem 8. Thus, it remains to consider the converse inclusion. Thus, let $G = (N, T, P_1, P_2, \ldots, P_n, S)$ be a CD grammar system working in f_c-mode, for the aforementioned f. First we

concentrate on f equals the $(t\wedge=1)$-mode. First, observe, that a cut-mode derivation can be sequentialized, such that only one production set is applicable to a subsentential form, i.e., the mapping ρ has domain size one. Now consider a derivation step: Let $x = x_0x_1x_2$ be a sentential form with $x_i \in (N \cup T)^*$, for $0 \leq i \leq 2$, and let $\rho(1) = i$, for some $1 \leq i \leq n$. Since a successful $(t \wedge =1)$-derivation on the sentential form x_1 has to apply exactly one rule of P_i and afterwards no rule of P_i is applicable anymore, we can assume that x_1 is an element of N, without changing the original successful derivation. This allows us to simulate the cut-$(t \wedge =1)$-derivation by a context-free derivation, choosing the appropriate rule of P_i. Thus, define the following context-free grammar $G = (N, T, P, S)$ with production set P containing the rules

$$P = \{ A \to \alpha \mid A \Rightarrow_i^{(t\wedge=1)} \alpha \text{ for some } 1 \leq i \leq n \}.$$

By our previous investigation it is easy to see that the language generated by G is equivalent to the language generated by the CD grammar system working in cut-$(t \wedge =1)$-mode.

A similar reasoning applies for the cut-$(t \wedge \leq k)$-mode, since a $(t \wedge \leq k)$-mode derivation on a sentential form can be split or cut into several $(t \wedge \leq \ell)$-mode derivations, for $1 \leq \ell \leq k$, on nonterminals only. Thus, the production set of the context-free grammar has to be modified in the following way: Define

$$P = \{ A \to \alpha \mid A \Rightarrow_i^{(t\wedge\leq\ell)} \alpha \text{ for some } 1 \leq \ell \leq k \text{ and } 1 \leq i \leq n \}.$$

Then the constructed context-free grammar is equivalent to the original CD grammar system working in cut-$(t \wedge \leq k)$-mode. This proves the stated claim. □

4 Conclusions

We examined the generative power of CD grammar systems when working in a new form of the derivation mode, the so called cut-mode. We summarize our results, comparing the classes $\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f_c)$ with $\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f)$, in Table 1. In most cases, the cut-f-mode turned out to be as most as powerful than the corresponding non-cut-mode, that is, the f-mode itself. For some modes, the generative capacity is not affected at all, but there are also some cases, where the power is reduced to that of context-free grammars, if parts of the sentential forms can be distributed to several components in one and the same derivation step. In particular, this is true for the t-mode of derivation, which describes the family of all ET0L languages in the non-cut variant. But exactly for this t-mode, the results are rather contesting the motivation of CD grammar systems from the AI point of view: Although CD grammar systems are a very natural and handy tool for describing non-context-free languages with context-free productions and they serve as sequential counterparts of tabled Lindenmayer systems, the re-interpretation of their properties in the framework of distributed problem solving is fairly problematic. In our approach, the cut-t-mode turns out to be useless, since it does not add to the power of context-free grammars. On the other hand, there seems to be something said for the $\geq k$- and $=k$-modes, $k \geq 2$, the generative capacity of which remains the same when they are driven in the cut variant. Also the sf- and some of the combined t-modes seem to

Derivation Mode f	$\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f)$	$\mathcal{L}(\mathrm{CD}, \mathrm{CF}[-\lambda], f_c)$
$*,\ =1,\ \geq 1,\ \leq k,$ for $k \geq 1$	$\mathcal{L}(\mathrm{CF})$	$\mathcal{L}(\mathrm{CF})$
$=k,\ \geq k,$ for $k \geq 2$	$\mathcal{L}(\mathrm{CF}) \subset \cdot \subseteq \mathcal{L}(\mathrm{P}, \mathrm{CF}[-\lambda])$	
t	$\mathcal{L}(\mathrm{ET0L})$	$\mathcal{L}(\mathrm{CF})$
sf	$\mathcal{L}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac})$	$\mathcal{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac}) \subseteq \cdot \subseteq \mathcal{L}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac})$
$(t \wedge \geq 1)$	$\mathcal{L}(\mathrm{ET0L})$	$\mathcal{L}(\mathrm{CF})$
$(t \wedge \geq k),$ for $k \geq 2$	$\mathcal{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac})$	$\mathcal{L}(\mathrm{CF}) \subset \cdot$ $\mathcal{L}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda]) \subset \cdot$
$(t \wedge =1),\ (t \wedge \leq k),$ for $k \geq 2$	$\mathcal{L}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda])$	$\mathcal{L}(\mathrm{CF})$
$(t \wedge =k),$ for $k \geq 2$	$\mathcal{L}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda])$	$\mathcal{L}(\mathrm{CF}) \subset \cdot$ $\mathcal{L}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda]) \subset \cdot$

Table 1: Generative capacity of CD grammars systems working in non-cut- and cut-mode derivations compared.

be of interest in that respect. Besides the combined derivation modes considered in this paper, there is another, external variant of hybrid CD grammar systems, e.g., see [12]. Those systems can also be considered in our setting. This is left for future research work.

References

[1] A. Atanasiu and V. Mitrana. The modular grammars. *International Journal of Computer Mathematics*, 30:101–122, 1989.

[2] H. Bordihn, E. Csuhaj-Varjú, and J. Dassow. CD grammar systems versus L systems. In Gh. Păun and A. Salomaa, editors, *Grammatical Models of Multi Agent Systems*, pages 18–32. Gordon and Breach, 1999.

[3] H. Bordihn and M. Holzer. Grammar systems with negated conditions in their cooperation protocols. *Journal of Universal Computer Science* 6 (2000), 1165–1184.

[4] E. Csuhaj-Varjú and J. Dassow. On cooperating/distributed grammar systems. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 26(1/2):49–63, 1990.

[5] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, and Gh. Păun. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation.* Gordon and Breach, 1994.

[6] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory*, volume 18 of *EATCS Monographs in Theoretical Computer Science.* Berlin: Springer, 1989.

[7] J. Dassow, Gh. Păun, and G. Rozenberg. Grammar systems. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 2, pages 155–213. Springer, 1997.

[8] H. Fernau, R. Freund, and M. Holzer. Hybrid modes in cooperating distributed grammar systems: internal versus external hybridization. *Theoretical Computer Science*, 259(1–2):405–426.

[9] H. Fernau, M. Holzer, and R. Freund. Bounding resources in cooperating distributed grammar systems. In S. Bozapalidis, editor, *Proc. of the 3rd Int. Conf. Developments in Language Theory*, pages 261–272. Aristotle University of Thessaloniki, 1997.

[10] R. Meersman and G. Rozenberg. Cooperating grammar systems. In *Proceedings of Mathematical Foundations of Computer Science MFCS'78*, volume 64 of *LNCS*, pages 364–374. Springer, 1978.

[11] P. H. Nii. Blackboard systems. In *The Handbook of Artificial Intelligence*, volume 4, pages 1–82, Addison-Wesley, 1989.

[12] V. Mitrana. Hybrid cooperating/distributed grammar systems. *Computers and Artificial Intelligence*, 12(1):83–88, 1993.

[13] D. J. Rosenkrantz. Programmed grammars and classes of formal languages. *Journal of the ACM*, 16(1):107–131, 1969.

[14] B. Rozoy. The Dyck language $D_1'^*$ is not generated by any matrix grammar of finite index. *Information and Computation (formerly Information and Control)*, 74:64–89, 1987.

[15] S. H. von Solms. Some notes on ET0L-languages. *International Journal of Computer Mathematics*, 5(A):285–296, 1976.