Acta Cybernetica **12** (1996) 347-360.

On Competence and Completeness in CD Grammar Systems*

Henning BORDIHN † Erzsébet CSUHAJ-VARJÚ ‡ §

Abstract

In this paper, different concepts of t-mode derivations in CD grammar systems which can be encountered in the literature and generalizations thereof are considered both in generating and in accepting case. Moreover, the influence of completeness of the components as an additional requirement to the derivational capacity of CD grammar systems is investigated.

1 Introduction

The theory of grammar systems is a recent vivid field of formal language theory describing multi-agent symbol systems by tools of formal grammars and languages ([4]). Cooperating/distributed grammar systems, (CD grammar systems, for short) is one the important subfields of the area, launched for syntactic modelling distributed problem solving systems based on blackboard architectures ([3]). We note, however, that the term "cooperating grammars" was introduced first in [9], as a generalization of two-level substitution grammars to a multi-level concept.

A CD grammar system consists of a finite set of grammars that cooperate in deriving words of a common language. At any moment in time there is exactly one sentential form in derivation and the grammars work on this string in turns, according to some cooperation protocol. In this model, the cooperating grammars correspond to the cooperating independent problem solving agents, the sentential form in derivation represents information on the current state of the problem solving stored in a global database, the blackboard, and the obtained language describes the set of problem solutions.

*Research supported by the German-Hungarian Research Project "Formal Languages, Automata and Petri-Nets" (1995-1997), No. D/102 (formerly: No. OMFB-NPI-102) of the TÉT Foundation, Budapest, Hungary, and No. 233.6. of Forschungszentrum Karlsruhe, Germany

†Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, P.O.Box 4120, D-39016 Magdeburg, Germany. E-mail:bordihn@irb.cs.uni-magdeburg.de

‡Research supported by Hungarian Scientific Research Fund OTKA T 017105

§Computer and Automation Research Institute, Hungarian Academy of Sciences, Kende u. 13-17, H-1111 Budapest, Hungary. E-mail: csuhaj@sztaki.hu

Turning to the original motivation, it is a sensible question whether some important properties and features of agents that influence the behaviour of blackboard-type problem solving systems can be formalized and interpreted in the syntactic framework provided by CD grammar systems. In this paper we deal with two of these properties: competence and completeness of components, moreover, we study them both in the case of generating CD grammar systems and in the case of accepting ones.

The idea of accepting grammars and systems ([11]) is the following: Starting from a "terminal" word, the system tries to derive a given goal word (the axiom) where, according to [1, 2, 6], the yield relation is defined by textually the same words as in generating case. Possible restrictions to production sets are turned "around", e.g., coming to productions of the form $v \to a$ in the context-free case, where v is a (possibly empty) word and a is a symbol.

CD grammar systems as accepting devices (CD grammar systems consisting of accepting grammars), corresponding to backward deduction systems in contrast to generating CD grammar systems which correspond to forward deduction systems, were considered in [8].

Both competence and completeness can form a basis of the cooperation protocol. According to our approach, an agent is competent in a current state of the problem solving if it is able to contribute to the problem solution. In grammatical terms, the component grammar is competent in the derivation of the current sentential form if it is able to apply at least one of its productions to it.

So far there have been two kinds of cooperation protocols (t-modes of derivation) based on competence/incompetence of grammars introduced and examined: in the first case (hard t-mode), a grammar can start with the derivation if it is competent in the sentential form and stops with the derivation if it is no longer competent in the actual string (it has no production to apply; the agent is not able to contribute to the problem solving). In the second case, the start condition is the same but the stop condition differs: the grammar finishes the derivation if it is not able to derive a word different from the actual one (the competence of the grammar is not enough to change the state of the problem solving). We generalize the latter concept to a cooperation protocol called stagnation derivation mode (s-mode), where a component has to continue its work until and unless a word is derived from which no new word can be rewritten, i.e., it is impossible to derive a word which does not appear in the derivation before. Thus, the competence of the agent (of the component grammar) is not enough to leave a stagnating phase of the problem solving (the derivation).

In this paper we compare the power of context-free CD grammar systems working on the basis of the above cooperation protocols. We show that the three variants are equally powerful. In the case of generating CD grammar systems they identify the class of ET0L languages and in the case of accepting CD grammar systems they provide a description of the class of context-sensitive languages (supposing that λ-free productions are taken into account). These results, in the case of weak t-mode and stagnation mode of derivation do not change if we incorporate some requirement concerning completeness of the components.

The notion of completeness is well-known from Lindenmayer systems: For a set of productions of a usual L system it is required that, for any symbol a of the alphabet, there is at least one production rule replacing a. In [6], accepting L systems were investigated, where this concept of completeness does not apply any more.

Since a production set of an accepting (ET)0L system is seen as an inverse finite substitution it is required that, for any symbol a, there is at least one rule of the form $v \to a$. One might wish to replace this "right-completeness" condition by a "left-completeness" condition as one is used to have in generating case, but such a condition must be in accord with the finiteness of the set of productions.

One sensible approach to define completeness can be inherited from [9], where the derivation strategy of the cooperating generative context-free grammars is defined as follows: every component grammar can start with the derivation if it is full competent in the generation of the current sentential form (if it has a production for any nonterminal symbol appearing in the actual string) and stops with it if it is no longer satisfies this criteria (there is at least one nonterminal in the string for which the grammar has no rewriting rule).

Clearly, this idea can be transferred to the accepting case (also applying to accepting Lindenmayer systems): a grammar component / set of productions is complete (thus full competent) for the current sentential form iff this sentential form can be partitioned into non-overlapping subwords each of which can be rewritten by a rule in the production set. We call this concept, that exhibits both completeness and competence, sentential-form-completeness (sf-completeness for short). Obviously, in generating case this concept coincides with the usual "left-completeness" condition known from L systems.

In [9] it is shown that context-free CD grammar systems with components working in sf-mode of derivation are equally powerful to the class of context-free programmed grammars with appearance checking. We show that in the case of accepting context-free CD grammar systems this protocol leads to the power of context-sensitive grammars, morevover, to reach this capacity at most five cooperating grammars are sufficient.

2 Basic definitions

We assume that the reader is familiar with the basic notions of formal language theory, for further details we refer to [11], and [5]. With our notations, we mostly follow [5]. Especially, we use $\subseteq$ to denote inclusion, $\subset$ to denote strict inclusion, and λ to denote the empty word. The length of a word w is denoted by $|w|$, $\mathbf{N}$ denotes the set of positive integers. Two languages L_1 and L_2 are considered to be equal iff $L_1 \setminus \{\lambda\} = L_2 \setminus \{\lambda\}$.

The family of languages generated by regular, context-free, context-sensitive, type-0 Chomsky grammars, ET0L systems, context-free programmed grammars, and context-free programmed grammars with appearance checking are denoted by $\mathcal{L}^{\text{gen}}$(REG), $\mathcal{L}^{\text{gen}}$(CF), $\mathcal{L}^{\text{gen}}$(CS), $\mathcal{L}^{\text{gen}}$(RE), $\mathcal{L}^{\text{gen}}$(ET0L), $\mathcal{L}^{\text{gen}}$(P,CF), and

$\mathcal{L}^{gen}$(P,CF,ac), respectively. In order to denote the family of languages accepted by a device of the corresponding type, we write the superscript *acc* instead of *gen*. If we want to exclude λ-productions, we add $-\lambda$ in our notations. Whenever we use bracket notations like $\mathcal{L}^{gen}(P,CF[-\lambda]) \subset \mathcal{L}^{gen}(P,CF[-\lambda],ac)$ we mean that the statement is true both in the case of neglecting the bracket contents and in the case of ignoring the brackets themselves.

We define CD grammar systems in a way suitable for the interpretation of both generating and accepting systems.

A *CD grammar system* of degree n, with $n \geq 1$, is an $(n+3)$-tuple

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

where N and T are two disjoint alphabets, the set of nonterminal and terminal symbols, respectively, $V = N \cup T$ is the total alphabet of G, $S \in N$ is the axiom, and $P_1, \ldots, P_n$ are finite sets of rewriting rules of the form $\alpha \to \beta$, $\alpha, \beta \in (N \cup T)^+$. In addition, we allow λ-rules of the form $\alpha \to \lambda$ in the generating case and $\lambda \to \beta$ in the accepting case. For $x, y \in (N \cup T)^*$, we write $x \underset{i}{\Longrightarrow} y$ iff $x = x_1 \alpha x_2$, $y = x_1 \beta x_2$ for some $\alpha \to \beta \in P_i$. Hence, subscript i refers to the production set (component) to be used. Furthermore, we denote by $\overset{f}{\underset{i}{\Longrightarrow}}$ a derivation executed by the i-th component according to some cooperation protocol f. For example, $\overset{\leq k}{\underset{i}{\Longrightarrow}}$ ($\overset{\geq k}{\underset{i}{\Longrightarrow}}$, $\overset{=k}{\underset{i}{\Longrightarrow}}$, or $\overset{*}{\underset{i}{\Longrightarrow}}$, respectively) denotes a derivation of at most k (at least k, exactly k, or an arbitrary number of) derivation steps as above.

For some cooperation protocol f, the language *generated* in f-mode (e.g., in $\leq k$-mode) by a CD grammar system G of degree n is

$$\mathcal{L}_f^{gen}(G) = \{w \in T^* \mid S = w_0 \overset{f}{\underset{i_1}{\Longrightarrow}} w_1 \overset{f}{\underset{i_2}{\Longrightarrow}} \cdots \overset{f}{\underset{i_{m-1}}{\Longrightarrow}} w_{m-1} \overset{f}{\underset{i_m}{\Longrightarrow}} w_m = w \text{ with } m \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq m\}.$$

The language *accepted* in f-mode by G is defined by

$$\mathcal{L}_f^{acc}(G) = \{w \in T^* \mid w = w_0 \overset{f}{\underset{i_1}{\Longrightarrow}} w_1 \overset{f}{\underset{i_2}{\Longrightarrow}} \cdots \overset{f}{\underset{i_{m-1}}{\Longrightarrow}} w_{m-1} \overset{f}{\underset{i_m}{\Longrightarrow}} w_m = S \text{ with } m \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq m\}.$$

The families of languages generated (accepted, respectively) in the f-mode by CD grammar systems with at most n [λ – free] context-free components are denoted by $\mathcal{L}^{gen}(CD_n, CF[-\lambda], f)$ (or $\mathcal{L}^{acc}(CD_n, CF[-\lambda], f)$). If the number of components is not restricted then we write $\mathcal{L}^{gen}(CD_\infty, CF[-\lambda], f)$ ($\mathcal{L}^{acc}(CD_\infty, CF[-\lambda], f)$).

3 Cooperation and Competence

In this section we compare the derivational capacity of CD grammar systems working under cooperation protocols based on competence/incompetence of the cooperating grammars. Since incompetence (disability of rewriting) realizes a terminating condition for the component grammar, these kind of cooperation protocols are called t-modes of derivations.

Let us have formal definitions.

A derivation

$$D: x = x_0 \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_{n-1} \Rightarrow x_n = y$$

from x to y is said to be

(i) of type *hard-t* (t_h) iff there is no z such that $y \Rightarrow z$,

(ii) of type *weak-t* (t_w) iff there is no $z \neq y$ such that $y \Rightarrow z$,

(iii) *stagnating* (s) iff there is no $z \notin \{x_i \mid 0 \leq i \leq n\}$ such that $y \Rightarrow^* z$.

Let $G = (N, T, S, P_1, \ldots, P_n)$ be a CD grammar system and let $f \in \{t_h, t_w, s\}$. For $x, y \in (N \cup T)^*$, we write $x \overset{f}{\underset{i}{\Longrightarrow}} y$ iff $x \overset{*}{\underset{i}{\Longrightarrow}} y$ and $x \overset{*}{\underset{i}{\Longrightarrow}} y$ is a derivation of type f.

Then, $\mathcal{L}^{\text{gen}}(\text{CD}_\infty, \text{CF}, t_w)$ is exactly the family of languages generated by CD grammar systems with an arbitrary number of context-free components working in t-mode as defined, e.g., in [4], whereas $\mathcal{L}^{\text{gen}}(\text{CD}_\infty, \text{CF}, t_h)$ equals the corresponding family as defined, e.g., in [8].

Observation 3.1 *By definition, if a derivation is of type t_h then it is of type t_w, and if it is of type t_w then it is stagnating.*
Hence, for a CD grammar system G, we have

(i) $L^{\text{gen}}_{t_h}(G) \subseteq L^{\text{gen}}_{t_w}(G) \subseteq L^{\text{gen}}_{s}(G)$ and

(ii) $L^{\text{acc}}_{t_h}(G) \subseteq L^{\text{acc}}_{t_w}(G) \subseteq L^{\text{acc}}_{s}(G)$. □

Note that productions of the form $A \to A$ block derivations in hard-t-mode whereas they can be neglected in CD grammar systems working in weak-t-mode. Nevertheless, we find the following lemma.

Lemma 3.2 *For $n \in \mathbf{N} \cup \{\infty\}$, we have*

$$\begin{aligned} \mathcal{L}^{\text{gen}}(CD_n, CF[-\lambda], t_h) &= \mathcal{L}^{\text{gen}}(CD_n, CF[-\lambda], t_w) \text{ and} \\ \mathcal{L}^{\text{acc}}(CD_n, CF[-\lambda], t_h) &= \mathcal{L}^{\text{acc}}(CD_n, CF[-\lambda], t_w) \end{aligned}$$

Proof. Suppose we have a CD grammar system working in t_w-mode. A simulating CD grammar system working in t_h-mode can be obtained by cancelling all rules of the form $A \to A$ from the productions sets of the original one. Conversely, if we replace all rules of the form $A \to A$ by $A \to F$, where F is a trap symbol, in a CD grammar system working in t_h-mode, we obtain a simulating grammar system working in the t_w-mode of derivations. □

In order to compare these families of languages with that one generated (accepted) by CD grammar systems working in s-mode, we take a better look at stagnating derivations.

Let

$$D: x = x_0 \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_n \Rightarrow \cdots$$

be a stagnating derivation which is not terminating. Then there is a finite language $L_{\text{stag}}(D)$ which consists of all words appearing in D such that, for any pair $(u, v) \in L_{\text{stag}}(D) \times L_{\text{stag}}(D)$, we have $u \overset{*}{\Rightarrow} v$ and $v \overset{*}{\Rightarrow} u$ in D.

Lemma 3.3 *Let G be a generating CD grammar system with context-free components. For any derivation D of G which is stagnating, all words in $L_{\text{stag}}(D)$ have one and the same length.*

Proof. Assume the contrary. Let u be a word of maximum length in $L_{\text{stag}}(D)$, i.e., if a rule is applicable to u then this rule has the form $A \to B$ or $A \to \lambda$, where A and B are symbols. Here, renaming rules $(A \to B)$ cannot introduce symbols B to which productions $B \to \beta$ with $|\beta| > 1$ can be applied, since otherwise u is not the word of maximum length in $L_{\text{stag}}(D)$. Thus, we can apply only rules of the form $A \to B$ and $A \to \lambda$ also to any word derived from u. In particular, this holds for any word $u' \in L_{\text{stag}}(D)$ with $|u'| < |u|$. This contradicts $u' \overset{*}{\Rightarrow} u$. □

In conclusion, if a derivation D is stagnating then there is a word in D such that only renaming rules can be applied in the sequel or it is a terminating derivation.

Lemma 3.4 $\mathcal{L}^{\text{gen}}(CD_\infty, CF[-\lambda], s) = \mathcal{L}^{\text{gen}}(ET0L)$.

Proof. The inclusion $\mathcal{L}^{\text{gen}}(\text{ET0L}) \subseteq \mathcal{L}^{\text{gen}}(\text{CD}_\infty, \text{CF}[-\lambda], s)$ follows by the construction given in [4, pp. 40–42] for the t_h-mode case.

Thus, it is left to prove $\mathcal{L}^{\text{gen}}(\text{CD}_\infty, \text{CF}[-\lambda], s) \subseteq \mathcal{L}^{\text{gen}}(\text{ET0L})$. Let $\Gamma = (N, T, P_1, P_2, \ldots, P_n, S)$ be a generating CD grammar system. For any nonterminal A and $1 \leq i \leq n$, we set $L_A^i = SF(G_A^i)$ where G_A^i is the context-free grammar $G_A^i = (N, T, P_i, A)$, i.e., L_A^i is the set of all sentential forms which can be generated by the i-th component of Γ starting with A. Furthermore, let

$$C_A^i = \begin{cases} L_A^i & \text{iff } L_A^i \subseteq N \text{ and, for all } B \in L_A^i, \text{ we have } L_B^i = L_A^i \\ \emptyset & \text{otherwise} \end{cases},$$

$$M^i = \{B \in N \mid \text{there is no } \beta \text{ such that } B \to \beta \in P_i\},$$

and

$$N_{\text{stag}}^i = M^i \cup \bigcup_{A \in N} C_A^i.$$

Obviously, N_{stag}^i is the set of nonterminal symbols which induce stagnation of the i-th component, more precisely, if the i-th component is active, stagnation appears iff a sentential form $w \in (N_{\text{stag}}^i \cup T)^*$ has been derived. Now, construct the ET0L system $G = (V, T, \mathcal{P}, S)$, with $V = N \cup T \cup \{A_i \mid A \in N, 1 \leq i \leq n\} \cup \{F\}$ and, for $1 \leq i \leq n$, $\mathcal{P}$ contains the following tables:

$$\begin{aligned} P_{i,1} &= \{A \to A_i \mid A \in N\} \cup \{a \to a \mid a \in T\} \cup \{X \to F \mid X \notin N \cup T\} \\ P_{i,2} &= \{A_i \to h_i(w) \mid A \to w \in P_i\} \cup \{a \to a \mid a \in T\} \cup \{A_i \to A_i \mid A \in N\} \cup \\ &\quad \{A_j \to F \mid A \in N, j \neq i\} \cup \{A \to F \mid A \in N\} \cup \{F \to F\}, \end{aligned}$$

where h_i is the morphism defined by $h_i(A) = A_i$ for $A \in N$ and $h_i(a) = a$ for $a \in T$,

$$P_{i,3} = \{A_i \to A \mid A \in N^i_{\text{stag}}\} \cup \{a \to a \mid a \in T\} \cup \{X \to F \mid X \notin N^i_{\text{stag}} \cup T\}.$$

Here, table $P_{i,1}$ simulates the selection of component P_i of Γ by "colouring" the nonterminals in the sentential form (replacing them by their corresponding subscripted version), $P_{i,2}$ simulates the application of the chosen component, and $P_{i,3}$ allows the system to leave the i-th component iff stagnation appears. The rules with the trap symbol F on their right-hand sides forbid shortcuts. Thus, we have $L^{\text{gen}}(G) = L^{\text{gen}}(\Gamma)$. □

The proof for $\mathcal{L}^{\text{gen}}(\text{ET0L}) \subseteq \mathcal{L}^{\text{gen}}(\text{CD}_3, \text{CF}[-\lambda], t)$ in [4, pp. 40–42] is given by a construction of a CD grammar system working in hard-t-mode with three components. Together with Observation 3.1, we can summarize the results as follows.

Corollary 3.5 *For $f \in \{t_h, t_w, s\}$, we have*

$$\begin{aligned}\mathcal{L}^{\text{gen}}(ET0L) &= \mathcal{L}^{\text{gen}}(CD_\infty, CF[-\lambda], t_h)\\ &= \mathcal{L}^{\text{gen}}(CD_\infty, CF[-\lambda], t_w)\\ &= \mathcal{L}^{\text{gen}}(CD_\infty, CF[-\lambda], s)\\ &= \mathcal{L}^{\text{gen}}(CD_3, CF[-\lambda], f).\end{aligned}$$

Let us turn to the accepting case. Clearly, it is impossible to get any terminating or stagnating derivation of a component of an accepting CD grammar system if λ-rules, i.e., rules of the form $\lambda \to v$, are present. Hence, we restrict ourselves to the λ-free case. Moreover, it is a direct consequence that as in the case of generating CD grammar systems, also in the case of accepting CD grammar systems with context-free components it holds that all words in $L_{\text{stag}}(D)$ have one and the same length for any stagnating derivation D.

In [8] it is shown how to construct an accepting CD grammar system with two context-free λ-free components working in hard-t-mode in order to simulate a given context-senstive grammar. On the other hand, it is obvious that any accepting CD grammar system with non-erasing context-free components in t_h- or t_w-mode can be simulated by a linear-bounded automaton. But even such a system working in s-mode can be simulated by a linear-bounded automaton. This fact is obvious if we take into consideration that a derivation is stagnating only if it is terminating or after deriving a certain sentential form the rules that can be applied to the sentential form are only certain renaming rules, that is, productions of the form $A \to B$, with $A, B \in N$. Thus, we easily find the next theorem.

Theorem 3.6 *For $f \in \{t_h, t_w, s\}$, we have*

$$\begin{aligned}\mathcal{L}^{\text{gen}}(CF) = \mathcal{L}^{\text{acc}}(CF) &= \mathcal{L}^{\text{acc}}(CD_1, CF - \lambda, f)\\ &\subset \mathcal{L}^{\text{acc}}(CD_2, CF - \lambda, f)\\ &= \mathcal{L}^{\text{acc}}(CD_\infty, CF - \lambda, f)\\ &= \mathcal{L}^{\text{gen}}(CS) = \mathcal{L}^{\text{acc}}(CS).\end{aligned}$$

4 Cooperation and Completeness

In this section we investigate how the derivational capacity of CD grammar systems changes when the components satisfy some completeness criteria. We show that these additional conditions do not necessarily alter the derivational power, even in some t-mode cases, where, actually, the communication protocol is determined by the incompleteness of the components.

First, let us discuss the concept of completeness.

From the theory of (generating) (ET)0L systems we know a condition called *left-completeness*: For each symbol a of the alphabet V of the system there is at least one production $a \to v$, $v \in V^*$. Analogously, *right-completeness* (originally for accepting (ET)0L systems) is defined: For each symbol a of the alphabet V there is at least one production $v \to a$, $v \in V^*$. These conditions can be modified for CD grammar systems as follows: A set P of *generating* context-free productions (a component of the CD grammar system) is said to be left-complete (right-complete) iff there is at least one production of the form $A \to \beta$ for each nonterminal A (at least one production of the form $A \to x$ for each symbol x of the total alphabet, respectively) in P. A set P of *accepting* context-free productions is called left-complete (right-complete) iff there is at least one production of the form $x \to A$ for each symbol x of the total alphabet (at least one production of the form $\alpha \to A$ for each nonterminal A, respectively) in P.

We concede that the above concept of right-completeness for the generating case and that of left-completeness for the case of accepting sets of productions are not very satisfying. For, e.g., the accepting case, one might take into account the possibility to require the following: If $w \to a$, with $|w| = k$, is in the set of productions P then $P \subseteq V^* \times V$ is surjective from $\bigcup_{i=1}^{k} V^i$ into V. This requirement is no restriction, since we can simply add rules of the form $v \to F$ for such words $v \in \bigcup_{i=1}^{k} V^i$ which do not appear on left-hand sides in the given production set, and those "dummy rules" are out of any influence to the rewriting process. Moreover, we get no genuine completeness at all.

Another concept of completeness which is appropriate both for generating and for accepting devices can be defined, based on the cooperation protocol used in [9], as follows:

Let V be an alphabet and let $L \subseteq V^*$ be a language over V. A set of production rules P is said to be *sentential-form-complete* (*sf-complete*, for short) *with respect to* L iff every word $w \in L \setminus \{\lambda\}$ has a factorization $w = x_1 x_2 \cdots x_n$ such that, for each i, $1 \leq i \leq n$, there is a rule $x_i \to \beta \in P$, with $\beta \in V^*$.

Then, e.g., in a (generating) E0L system $G = (V, \Sigma, P, \omega)$, P has to be *sf-complete* with respect to V^*.

Observation 4.1 *If a set of production rules P is left-complete then it is sf-complete with respect to $(domP)^*$, where $domP$ denotes the set of all symbols ap-*

pearing on left-hand sides of the rules in P.

For a generating context-free set P, we even find equivalence between these two properties. □

A CD grammar system $G = (N, T, S, P_1, \ldots, P_n)$ is called left-complete or right-complete iff each P_i, $1 \leq i \leq n$, is left-complete or right-complete, respectively. G is said to be sf-complete iff each P_i, $1 \leq i \leq n$, is sf-complete with respect to N^* if G has generating context-free components, and with respect to $(N \cup T)^*$ if G has accepting context-free components.

Note that it is sensible to differentiate between generating and accepting mode in this definition since in generating mode only nonterminal symbols can be rewritten whereas in accepting case there must also be rules for rewriting terminals.

Theorem 4.2 *The requirement of left-completeness, right-completeness, and / or sf-completeness does not alter the derivational power in the case of the following grammars and systems:*

(i) generating and accepting [E][T]0L systems,

(ii) generating and accepting context-free grammars,

(iii) generating and accepting CD grammar systems with [λ-free] context-free components working in $*$-, in weak-t-, or in stagnating mode,

(iv) generating and accepting CD grammar systems with [λ-free] context-free components working in $\leq k$-, $\geq k$-, or $= k$-mode, with $k \geq 1$.

Proof. Let N be the set of nonterminals and T be the set of terminals of the system under consideration.

(i) Add rules $x \to x$ for any $x \in V$ to the set of productions (or to each production table, respectively) both in the generating and in the accepting case.

(ii)-(iii) In the generating case, to the set of productions (or to each component, respectively) add rules $A \to A$ for any $A \in N \cup \{F\}$, where F is a new nonterminal symbol and add $F \to a$ for any $a \in T$ (in order to get right-completeness). In the accepting case, add $T' = \{a' \mid a \in T\}$ to the set of nonterminal symbols. Then, to the set of productions (or to each component, respectively), add rules $A \to A$ for each $A \in N \cup T'$, and $a \to a'$ for each $a \in T$. Moreover, in the case of CD grammar systems working in weak-t or stagnating mode, we have to add the rules in $\{v \to \beta \mid v \in s(\alpha), \alpha \to \beta \in P_i\}$ to component P_i, where s is the finite substitution defined by $s(A) = \{A\}$, for $A \in N$, and $s(a) = \{a, a'\}$, for $a \in T$.

(iv) Let F_1 and F_2 be two new nonterminal symbols. Add $F_1 \to F_1$ and $F_2 \to F_2$ to each component. Moreover, add rules $x \to F_1$ for $x \in N$ in the generating case and for $x \in N \cup T$ in the accepting case, respectively. This guarantees left- and sf-completeness. Furthermore, add rules $F_2 \to x$ for $x \in N \cup T$ in the generating case

and for $x \in$ in the accepting case, respectively, in order to get right-completeness, too. □

The next theorem deals with the hard-t-mode of derivation where the situation is different.

Theorem 4.3 *(i) Right-completeness is no restriction for generating as well as for accepting CD grammar systems with [λ-free] context-free components working in hard-t-mode.*

(ii) Both the family of languages generated by left-complete CD grammar systems with [λ-free] context-free components working in hard-t-mode and the family of languages generated by sf-complete CD grammmar systems with [λ-free] context-free components working in hard-t-mode is equal to $\mathcal{L}(CF)$.
The corresponding families of languages in the accepting case are empty.

Proof. (i) Add rules $F \to x$ for all symbols x of the total alphabet in the generating case and for all nonterminal symbols in the accepting case, respectively, and the rule $F \to F$, where F is a new nonterminal symbol, again. Statement (ii) is obvious. □

5 Cooperation and sf-completeness

In [9], cooperating (generating) grammar systems were defined in such a way that the concept of sf-completeness was used as the basis of the cooperation protocol. In this section we present results about the derivational capacity and size complexity of accepting CD grammar systems with components cooperating in the above manner.

Using our notation, we first give the following definition which is appropriate both for generating and accepting CD grammar systems.

Let $G = (N, T, S, P_1, \ldots, P_n)$ be a context-free CD grammar system and let h be a morphism defined by $h(A) = A$, for $A \in N$ and $h(a) = \lambda$, for $a \in T$. In *sf-mode of derivation* the rewriting has to be performed by one and the same component P_i until and unless it is *disabled*, i.e., a sentential form w has been derived such that P_i is not sf-complete any more with respect to $h(w)$ in the generating case and with respect to w in the accepting case.

The family of languages generated by CD grammar systems with at most n [λ-free] context-free components working in the sf-mode is denoted by $\mathcal{L}^{\text{gen}}(\text{CD}_n, \text{CF}[-\lambda], sf)$. If there is no limit for the number of components then we write the subscript ∞ instead of n. The language families defined by the corresponding accepting devices are denoted analogously.

In contrast to the results about generating CD grammar systems with context-free components working in (weak-)t-mode, for the systems defined by Meersman and Rozenberg the following result is shown ([9]):

Theorem 5.1 $\mathcal{L}^{\text{gen}}(CD_\infty, CF[-\lambda], sf) = \mathcal{L}^{\text{gen}}(P, CF[-\lambda], ac)$. □

As it concerns the λ-free accepting case, we find the same hierarchical relationship as for (hard-)t-mode proved in [8], but here, we have also nonempty families of languages accepted by devices containing λ-rules.

Theorem 5.2 (i) $\mathcal{L}^{\mathrm{acc}}(\mathrm{CD}_\infty, \mathrm{CF}-\lambda, sf) = \mathcal{L}^{\mathrm{gen}}(\mathrm{CS}) = \mathcal{L}^{\mathrm{acc}}(\mathrm{CS})$

(ii) $\mathcal{L}^{\mathrm{acc}}(\mathrm{CD}_\infty, \mathrm{CF}, sf) = \mathcal{L}^{\mathrm{gen}}(\mathrm{RE}) = \mathcal{L}^{\mathrm{acc}}(\mathrm{RE})$

Proof. (i) Since any CD grammar system from $(\mathrm{CD}_\infty, \mathrm{CF}-\lambda, sf)$ can be simulated by a linear-bounded automaton (that is, by a context-sensitive grammar), we only show that the reverse inclusion holds. For, our proof is based on the underlying idea of the proof of [5, Theorem 3.3]. Let us have a generating context-sensitive grammar $G = (N, T, P, S)$ in Kuroda normal form, without λ productions. Assume that a unique label r is attached to any context-sensitive rule, of the form $XU \to YZ$ with $X, U, Y, Z \in N$, in P. Let us denote the set of labels by $\mathrm{Lab}(P) = \{r_1, r_2, \ldots, r_R\}$. Let $\bar{T} = \{\bar{a} \mid a \in T\}$, $\{\bar{\bar{T}} = \{\bar{\bar{a}} \mid a \in T\}$ and let h be a morphism defined by $h(a) = A$ for $A \in N$ and $h(a) = \bar{a}$ for $a \in T$. For a string $w \in (N \cup T)^*$ let us denote by $\bar{w} = h(w)$.

We construct an accepting CD grammar system

$$\Gamma = (N', T, S', P_{\mathrm{init}}, P_{\mathrm{CF}}, P_{1,1}, P_{1,2}, P_{1,3}, P'_{1,1}, P'_{1,2}, P'_{1,3} \cdots, P'_{R,1}, P'_{R,2}, P'_{R,3})$$

such that $L^{acc}_{sf}(\Gamma) = L(G)$ holds. Let Γ be defined with

$$\begin{aligned} N' = N \cup \bar{T} \cup \bar{\bar{T}} \cup \{S', F\} \cup \{[A, r], (A, r) \mid A \in N \text{ and } r \in \mathrm{Lab}(P)\} \\ \cup \{x' \mid x \in N \cup \bar{T}\} \cup \{< x, r > \mid x \in N \cup \bar{T} \text{ and } r \in \mathrm{Lab}(P)\} \end{aligned}$$

(the unions being disjoint). The components of Γ are constructed as follows:

$$\begin{aligned} P_{\mathrm{init}} &= \{a \to \bar{a}, \bar{a} \to \bar{a}, \bar{a} \to \bar{\bar{a}} \mid a \in T\}, \\ P_{\mathrm{CF}} &= \{S \to S'\} \cup \{x \to x \mid x \in N \cup \bar{T}\} \cup \{\bar{w} \to C \mid C \to w \in P\} \cup \\ &\quad \{Y \to [Y, r] \mid r : XU \to YZ \in P\} \cup \{x' \to x \mid x \in N \cup \bar{T}\} \cup \\ &\quad \{\bar{\bar{a}} \to \bar{a} \mid a \in T\}, \end{aligned}$$

and, for $1 \le r \le R$, $r : XU \to YZ$:

$$\begin{aligned} P_{r,1} &= \{[Y, r] \to [Y, r], Z \to (Z, r)\} \cup \{x \to x \mid x \in N \cup \bar{T}\} \cup \{x' \to x \mid x \in N \cup \bar{T}\} \\ P_{r,2} &= \{[Y, r](Z, r) \to F\} \cup \{x \to x \mid x \in N \cup \bar{T}\} \cup \{x \to < x, r > \mid x \in N \cup \bar{T}\} \\ P_{r,3} &= \{[Y, r] \to X, (Z, r) \to U\} \cup \{x \to x \mid x \in N \cup \bar{T}\} \cup \\ &\quad \{< x, r > \to x' \mid x \in N \cup \bar{T}\} \\ P'_{r,1} &= \{[Y, r] \to [Y, r], Z \to Z'\} \\ P'_{r,2} &= \{[Y, r]Z' \to F, [Y, r] \to Y'\} \\ P'_{r,3} &= \{Y' \to X, Z' \to U, X \to X\} \end{aligned}$$

Production set P_{init} is for starting the derivation process. Obviously, by P_{CF} context-free derivation steps of G are simulated whereas the components

$P_{r,1}, P_{r,2}, P_{r,3}$ and $P'_{r,1}, P'_{r,2}, P'_{r,3}$ simulate applications done by the rule with label r after replacing exactly one appearance of symbol Y in the sentential form by $[Y,r]$. The first group of production sets handles the situation when the sentential form is of the form $u[Y,r]Zv$, with $uv \in (N \cup \bar{T})^+$, and the second is for the situation when the sentential form is $[Y,r]Z$. In the first case it is necessary to replace a symbol $< x, r >$ in order to leave $P_{r,3}$ which can only be introduced by application of $P_{r,2}$. But $P_{r,2}$ can be active only if the symbols $[Y,r]$ and (Z,r) are neighbouring in the "correct" manner or they do not appear at all. In the latter case, the application of $P_{r,2}$ and $P_{r,3}$ remain without any effect. Shortcuts are impossible since a component must be fully competent when applied. Similarly, it is easy to see that production sets $P'_{r,1}, P'_{r,2}, P'_{r,3}$ can be successfully applied only if the sentential form is of the form $[Y,r]Z$. Hence, $L^{\mathrm{acc}}(\Gamma) = L^{\mathrm{gen}}(G)$.

(ii) Without loss of generality we can assume the given type-0 grammar to have only rules as a grammar in Kuroda normal form only having rules of the form $A \to \lambda$, with $A \in N$, in addition. Thus, we can use the same construction as in (i) only giving additional rules $\lambda \to A$ to component P_{CF} if needed. The other direction of the proof follows by the Church theses or it can be shown by construction of a Turing machine. □

Finally, we investigate the question if the number of components can be restricted for devices working in sf-mode. Indeed, we find that 5 components are sufficient in order to describe the whole language family.

Theorem 5.3 (i) $\mathcal{L}^{\mathrm{gen}}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], sf) = \mathcal{L}^{\mathrm{gen}}(\mathrm{CD}_5, \mathrm{CF}[-\lambda], sf)$

(ii) $\mathcal{L}^{\mathrm{acc}}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], sf) = \mathcal{L}^{\mathrm{acc}}(\mathrm{CD}_5, \mathrm{CF}[-\lambda], sf)$

Proof. Let $G = (N, T, S, P_1, \ldots, P_n)$ be a CD grammar system of degree $n > 5$ working in sf-mode. Construct a CD grammar system working in sf-mode of the same type with 5 components according to the following basic idea:
Component 1 contains all rules of the given system, but the symbols in the rules carry the number of the component of G, where they originally belong to, as subscript. For simulating work of component P_i, the symbols in the sentential form must carry subscript i, too. Then, this component becomes disabled iff there is no rule stemming from P_i which is applicable to the current sentential form, and one can continue with component 2 or 3.
In *component 2* to even subscripts i of the symbols of the sentential form, one is added (modulo n) in order to change the production set of G which shall be simulated.
Component 2' is the only component which can get active after applying component 2. Here, it is checked whether all symbols have changed their subscript. If yes, a continuation with component 1 or 3 is possible, otherwise the derivation is blocked.
Components 3 and 3' do the analogous job as components 2 and 2' for odd subscripts.

Now, we give the formal description of that system for the accepting case. The construction for the generating case can be given analogously.

Let $G' = (N', T, S', P_1, P_2, P_2', P_3, P_3')$, where

$$N' = N \cup \{S'\} \cup \{A_i, A_i', A_i'' \mid A \in N \cup T \text{ and } 1 \leq i \leq n\}$$

(the unions being disjoint). Furthermore, let h_i be the morphism defined by $h_i(A) = A_i$ for $A \in N \cup T$, $1 \leq i \leq n$. Then, the components of G' are constructed as follows:

$$\begin{aligned}
P_1 &= \{h_i(\alpha) \to h_i(\beta) \mid \alpha \to \beta \in P_i,\ 1 \leq i \leq n\} \cup \{A_i'' \to A_i \mid 1 \leq i \leq n\} \cup \\
&\quad \{S_i \to S' \mid 1 \leq i \leq n\}, \\
P_2 &= \{A_i \to A_{i+1} \mid i \equiv 0 \bmod 2\} \cup \{A_{i+1} \to A_{i+1} \mid i \equiv 0 \bmod 2\} \cup \\
&\quad \{A_i \to A_{i+1}' \mid i \equiv 0 \bmod 2\} \cup \{A_i'' \to A_i \mid i \equiv 0 \bmod 2\} \cup \\
&\quad \{A \to A_1 \mid A \in N \cup T\} \cup \{A \to A_1' \mid A \in N \cup T\} \cup Q_2, \\
P_2' &= \{A_{i+1} \to A_{i+1} \mid i \equiv 0 \bmod 2\} \cup \{A_{i+1}' \to A_{i+1}'' \mid i \equiv 0 \bmod 2\}, \\
P_3 &= \{A_i \to A_{i+1} \mid i \equiv 1 \bmod 2\} \cup \{A_{i+1} \to A_{i+1} \mid i \equiv 1 \bmod 2\} \cup \\
&\quad \{A_i \to A_{i+1}' \mid i \equiv 1 \bmod 2\} \cup \{A_i'' \to A_i \mid i \equiv 1 \bmod 2\} \cup Q_3, \\
P_3' &= \{A_{i+1} \to A_{i+1} \mid i \equiv 1 \bmod 2\} \cup \{A_{i+1}' \to A_{i+1}'' \mid i \equiv 1 \bmod 2\},
\end{aligned}$$

where

$$Q_2 = \begin{cases} \{A_n \to A_1, A_n \to A_1' \mid A \in N \cup T\} \cup & \\ \{A_1 \to A_1 \mid A \in N \cup T\} & \text{if } n \equiv 0 \bmod 2 \\ \emptyset & \text{if } n \equiv 1 \bmod 2 \end{cases} \quad \text{and}$$

$$Q_3 = \begin{cases} \{A_n \to A_1, A_n \to A_1' \mid A \in N \cup T\} \cup & \\ \{A_1 \to A_1 \mid A \in N \cup T\} & \text{if } n \equiv 1 \bmod 2 \\ \emptyset & \text{if } n \equiv 0 \bmod 2 \end{cases}.$$

Clearly, the system has to start with component P_2 by rewriting any terminal a by a_1. The axiom can be derived after yielding S_i for some i with P_1. Note that, by technical reasons, after applying the rules in Q_3 the system has to continue with component P_2' and then with P_3. Moreover, each component is a set of accepting context-free productions; in this connection the modifications are necessary for proving the statement in the generating case. □

Acknowledgements: The authors are grateful to Henning Fernau and Markus Holzer for discussions on the topic.

References

[1] H. Bordihn and H. Fernau. Accepting grammars with regulation. *International Journal of Computer Mathematics*, 53:1–18, 1994.

[2] H. Bordihn and H. Fernau. Accepting grammars and systems via context condition grammars. *Journal of Automata, Languages and Combinatorics*, 1(2):97–112, 1996.

[3] 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.

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

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

[6] H. Fernau and H. Bordihn. Remarks on accepting parallel systems. *International Journal of Computer Mathematics*, 56:51–67, 1995.

[7] H. Fernau and M. Holzer. Accepting multi-agent systems II. *Acta Cybernetica*, 1996. In this volume.

[8] H. Fernau, M. Holzer, and H. Bordihn. Accepting multi-agent systems: The case of cooperating distributed grammar systems. *Computers and Artificial Intelligence*, 15(2–3):123–139, 1996.

[9] R. Meersman and G. Rozenberg. Cooperating grammar systems. In *Proc. MFCS'78*, volume 64 of *LNCS*, pages 364–373. Berlin: Springer, 1978.

[10] Gh. Păun. On the generative capacity of hybrid CD grammar systems. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 30(4):231–244, 1994.

[11] A. Salomaa. *Formal Languages*. Academic Press, 1973.