On Competence in CD Grammar Systems*

Maurice H. ter Beek[1,**], Erzsébet Csuhaj-Varjú[2], Markus Holzer[3], and György Vaszil[2]

[1] Istituto di Scienza e Tecnologie dell'Informazione, Pisa, Italy
maurice.terbeek@isti.cnr.it
[2] Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest, Hungary
{csuhaj, vaszil}@sztaki.hu
[3] Institut für Informatik, Technische Universität München München, Germany
holzer@in.tum.de

Abstract. We investigate the generative power of cooperating distributed grammar systems (CDGSs), if the cooperation protocol is based on the level of competence on the underlying sentential form. A component is said to be $=k$-competent ($\leq k$-, $\geq k$-competent, resp.) on a sentential form if it is able to rewrite exactly k (at most k, at least k, resp.) different nonterminals appearing in that string. In most cases CDGSs working according to the above described cooperation strategy turn out to give new characterizations of the language families based on random context conditions, namely random context (context-free) languages and the biologically motivated family of languages generated by ET0L systems with random context. Thus, the results presented in this paper can shed new light on some longstanding open problems in the theory of regulated rewriting.

1 Introduction

A grammar system is a set of grammars that under a specific cooperation protocol generates one language. The idea to consider—contrary to the "one grammar generating one language" paradigm of classical formal language theory—a set of cooperating grammars generating one language first appeared in [9]. An intensive exploration of the potential of grammar systems was not undertaken until [3] established a link between cooperating distributed grammar systems (CDGSs) and blackboard systems as known from artificial intelligence. A blackboard system consists of several autonomous agents, a blackboard, and a control

* The first author was supported by an ERCIM postdoctoral fellowship and the third author was supported by project Centre of Excellence in Information Technology, Computer Science and Control, ICA1-CT-2000-70025, HUN-TING project, WP 5.

** This author's work for this paper was fully carried out during his stay at the Computer and Automation Research Institute of the Hungarian Academy of Sciences.

C.S. Calude, E. Calude and M.J. Dinneen (Eds.): DLT 2004, LNCS 3340, pp. 76–88, 2004.

mechanism. The control mechanism dictates some rules which the agents must respect during their joint effort to solve a problem stated on the blackboard. The only way in which the agents may communicate is via the blackboard, which represents the current state of the problem solving. If the problem solving is successful, the solution appears on the blackboard. CDGSs form a language-theoretic framework for modelling blackboard systems. Agents are represented by grammars, the blackboard is represented by the sentential form, control is regulated by a cooperation protocol of the grammars, and the solution is represented by a terminal word. By now, grammar systems form a well-established and well-recognized area within the theory of formal languages. The interested reader is referred to [6] for more information.

In this paper we examine some variants of cooperation protocols for CDGSs based on the level of competence that a component has on a sentential form. Competence-based cooperation protocols have already been studied in the literature, e.g., [1, 3–5, 9] We consider cooperation protocols that allow a component to start rewriting when such a competence condition is satisfied, and that require it to do so as long as the grammar satisfies this condition. Intuitively, a component is $=k$-competent ($\leq k$-competent, $\geq k$-competent, resp.) on a sentential form if it is able to rewrite exactly k (at most k, at least k, resp.) different nonterminals appearing in the sentential form. In the sequel we will call these cooperation protocols the $=k$-comp.-mode ($\leq k$-comp.-mode, $\geq k$-comp.-mode, resp.) of derivation. Hence the more different nonterminals of a sentential form a component is able to rewrite, the higher its (level of) competence on that string. By restricting the rewriting of the sentential form to components having a certain (level of) competence, we provide a formal interpretation of the requirement that agents must be competent enough before being able to participate in the problem solving taking place on the blackboard.

We demonstrate that these competence-based cooperation protocols are very powerful and closely related to rewriting mechanisms based on random context conditions. To be more precise, it is shown that CDGSs working in the $=1$-comp.- or ≤ 1-comp.-mode of derivation are at least as powerful as the family of languages generated by forbidding random context grammars, while CDGSs working according to the ≥ 1-comp.-mode of derivation characterize the family of ET0L languages. A slight increase in the level of competence gives a significant increase in generative power, namely already CDGSs working in the $=2$-comp.- or ≤ 2-comp.-mode of derivation characterize the family of random context languages or, equivalently, that of the recursively enumerable languages, while the ≥ 2-comp.-mode leads to the biologically motivated family of languages generated by ET0L systems with random context [11]. This is yet another characterization of the family of random context ET0L languages, which recently appeared several times in relation with CDGSs and non-standard derivation modes—see, e.g., [2]. The family of random context ET0L languages is of interest because it coincides with the family of recurrent programmed context-free languages and forms an intermediate class between the families of context-free random context languages and programmed context-free languages generated by grammars

without appearance checking [8]. In fact we show that rather simple component grammars suffice to simulate random context grammars or ET0L systems with random context, thus showing that it is indeed the cooperation protocol that is very powerful. So we hope that one can gain a deeper insight into the nature of (recurrent) programmed versus random context grammars without appearance checking such that new light is shed on some longstanding open questions.

2 Definitions

We assume the reader to be familiar with the basic notions of formal languages as, e.g., contained in [7]. In general, we have the following conventions. Set difference is denoted by $\setminus$, set inclusion by $\subseteq$, and strict set inclusion by $\subset$. The cardinality of a set M is denoted by $|M|$. The empty word is denoted by λ.

A *random context grammar* is a quadruple $G = (N, T, P, S)$, where N, T, and $S \in N$ are the set of nonterminals, the set of terminals, and the start symbol, respectively. Moreover, P is a finite set of random context rules, i.e., triples of the form $(\alpha \to \beta, Q, R)$, where $\alpha \to \beta$ is a context-free production and $Q, R \subseteq N$ are its permitting and forbidding context, respectively. For $x, y \in (N \cup T)^*$ we write $x \Rightarrow y$ if and only if $x = x_1 \alpha x_2$, $y = x_1 \beta x_2$, all symbols of Q appear in $x_1 x_2$, and no symbol of R appears in $x_1 x_2$. If either Q and/or R is empty, then the corresponding context check is omitted. The language generated by G is defined as $L(G) = \{ w \in T^* \mid S \stackrel{*}{\Rightarrow} w \}$, where $\stackrel{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$. The family of languages generated by random context grammars is denoted by $\mathcal{L}(\mathrm{RC}, \mathrm{CF})$. It is known—see, e.g., [7]—that $\mathcal{L}(\mathrm{RC}, \mathrm{CF}) = \mathcal{L}(\mathrm{RE})$, where $\mathcal{L}(\mathrm{RE})$ denotes the class of recursively enumerable languages.

Random context grammars where all permitting contexts are empty are called *forbidding random context grammars*. In this case we are led to the family $\mathcal{L}(\mathrm{fRC}, \mathrm{CF})$ of forbidding random context languages. It is known—see, e.g., [7]—that $\mathcal{L}(\mathrm{ET0L}) \subset \mathcal{L}(\mathrm{fRC}, \mathrm{CF}) \subseteq \mathcal{L}(\mathrm{RC}, \mathrm{CF})$, where $\mathcal{L}(\mathrm{ET0L})$ denotes the family of languages generated by ET0L systems.

A *random context ET0L system* is a sixtuple $G = (\Sigma, H, \omega, \Delta, oc, noc)$, where the four tuple $(\Sigma, H, \omega, \Delta)$ is an ordinary ET0L system, with Σ as its total alphabet, $\Delta \subseteq \Sigma$ as its terminal alphabet, H as its set of tables (finite substitutions from Σ into Σ^*), $\omega \in \Sigma^+$ as its axiom, and oc, noc as functions from H to the subsets of Σ. For two strings $x, y \in \Sigma^*$, the relation $x \Rightarrow y$ holds if and only if there is an $h \in H$, such that all letters in $oc(h)$ occur in x, no letter of $noc(h)$ occurs in x, and $y \in h(x)$. Let $\stackrel{*}{\Rightarrow}$ denote the reflexive and transitive closure of $\Rightarrow$. The language generated by G is defined as $L(G) = \{ w \in \Delta^* \mid \omega \stackrel{*}{\Rightarrow} w \}$. The family of languages generated by random context ET0L systems is denoted by $\mathcal{L}(\mathrm{RC}, \mathrm{ET0L})$. It is known—see, e.g., [7]—that $\mathcal{L}(\mathrm{ET0L}) \subset \mathcal{L}(\mathrm{RC}, \mathrm{ET0L}) \subseteq \mathcal{L}(\mathrm{RE})$, but it is an open problem whether the latter inclusion is strict.

3 Competence in CD Grammar Systems

A *cooperating distributed grammar system* (CDGS) of degree n, with $n \geq 1$, is an $(n+3)$-tuple $G = (N, T, \alpha, P_1, \ldots, P_n)$, in which N and T are its disjoint alphabets of nonterminals and terminals, respectively, $\alpha \in (N \cup T)^*$ is its axiom, and $P_1, \ldots, P_n$ are finite sets of context-free productions over $N \times (N \cup T)^*$ that are called its components. The given definition of CDGSs differs from the usual one since arbitrary words from $(N \cup T)^*$ may serve as its axioms. For $x, y \in (N \cup T)^*$ and $1 \leq i \leq n$, we define a single rewriting step as $x \Rightarrow_i y$ if and only if $x = x_1 A x_2$ and $y = x_1 z x_2$, for some $A \to z \in P_i$. The subscript i thus refers to the component being used.

Next we recall from [4] the notion of competence that components of a CDGS have on a particular sentential form. First we define the domain of a component as $\mathrm{dom}(P_i) = \{ A \in N \mid A \to z \in P_i \}$. Consequently, component P_i, with $1 \leq i \leq n$, is said to be *k-competent* on a sentential form x in $(N \cup T)^*$ if and only if $|\mathrm{alph}_N(x) \cap \mathrm{dom}(P_i)| = k$, where $\mathrm{alph}_N(x) = \{ A \in N \mid x \in (N \cup T)^* A (N \cup T)^* \}$, i.e., it denotes the set of all nonterminals occurring in x. We abbreviate the (level of) competence of component P_i on x by $\mathrm{clev}_i(x)$.

Based on the (level of) competence that the components have on a sentential form, we define the following cooperation protocols for CDGSs:

1. $x \Rightarrow_i^{\leq k\text{-comp.}} y$ if and only if there is a derivation $x = x_0 \Rightarrow_i x_1 \Rightarrow_i \cdots \Rightarrow_i x_{m-1} \Rightarrow_i x_m = y$ and it satisfies
   (a) $\mathrm{clev}_i(x_j) \leq k$ for $0 \leq j < m$ and (i) $\mathrm{clev}_i(x_m) = 0$ or (ii) $y \in T^*$, or
   (b) $\mathrm{clev}_i(x_j) \leq k$ for $0 \leq j < m$ and $\mathrm{clev}_i(x_m) > k$,
2. $x \Rightarrow_i^{=k\text{-comp.}} y$ if and only if there is a derivation $x = x_0 \Rightarrow_i x_1 \Rightarrow_i \cdots \Rightarrow_i x_{m-1} \Rightarrow_i x_m = y$ and it satisfies
   (a) $\mathrm{clev}_i(x_j) = k$ for $0 \leq j < m$ and $\mathrm{clev}_i(x_m) \neq k$, or
   (b) $\mathrm{clev}_i(x_0) = k$, $\mathrm{clev}_i(x_j) \leq k$ for $1 \leq j \leq m$, and $y \in T^*$.
3. $x \Rightarrow_i^{\geq k\text{-comp.}} y$ if and only if there is a derivation $x = x_0 \Rightarrow_i x_1 \Rightarrow_i \cdots \Rightarrow_i x_{m-1} \Rightarrow_i x_m = y$ and it satisfies
   (a) $\mathrm{clev}_i(x_j) \geq k$ for $0 \leq j < m$ and $\mathrm{clev}_i(x_m) < k$, or
   (b) $\mathrm{clev}_i(x_0) \geq k$ and $y \in T^*$.

Let $D = \{ \leq k\text{-comp.}, =k\text{-comp.}, \geq k\text{-comp.} \mid k \geq 1 \}$ and let $\Rightarrow^f$ denote $\Rightarrow_i^f$ for some i, with $1 \leq i \leq n$, and $f \in D$. The reflexive transitive closure of $\Rightarrow^f$ is denoted by $\overset{*}{\Rightarrow}^f$. The language generated by G in the f-mode of derivation, with $f \in D$, is $L_f(G) = \{ w \in T^* \mid \alpha \overset{*}{\Rightarrow}^f w \}$. The family of languages generated by CDGSs working in the f-mode of derivation is denoted by $\mathcal{L}(\mathrm{CD}, \mathrm{CF}, f)$.

Example 1. Let $G = (N, T, \alpha, P_1, \ldots, P_8)$ be a CDGS with set of nonterminals $N = \{A, A', B, B', C, D\}$, terminals $T = \{a, b, c\}$, axiom AB, and components

$$P_1 = \{A \to aA'b, B' \to B', C \to C\}, \qquad P_5 = \{A' \to C, B \to B\},$$
$$P_2 = \{A \to A, B \to B'c, C \to C\}, \qquad P_6 = \{A \to A, A' \to A', B' \to D\},$$
$$P_3 = \{A' \to A, B \to B, C \to C\}, \qquad P_7 = \{B' \to B', C \to \lambda\}, \text{ and}$$
$$P_4 = \{A' \to A', B' \to B, C \to C\}, \qquad P_8 = \{D \to \lambda\}.$$

When working in the $\leq$1-comp.-mode or =1-comp.-mode of derivation, G generates the language $L(G) = \{ a^n b^n c^n \mid n \geq 1 \}$. This can be seen as follows.

Starting from the axiom, the components P_1, P_3, P_5, and P_6 are all 1-competent. However, except for P_1, their application does not alter the axiom and hence these components remain 1-competent forever. In those cases the derivation thus enters a loop. From the axiom, the only two-step derivation that does not loop is thus $AB \Rightarrow_1^{=1\text{-comp.}} aA'bB \Rightarrow_2^{=1\text{-comp.}} aA'bB'c$. Consequently, a choice must be made. First we can apply P_5 to obtain the derivation $aA'bB'c \Rightarrow_5^{=1\text{-comp.}} aCbB'c \Rightarrow_6^{=1\text{-comp.}} aCbDc$, after which the derivation can be finished by $aCbDc \Rightarrow_7^{=1\text{-comp.}} abDc \Rightarrow_8^{=1\text{-comp.}} abc$ or instead by applying P_8 before P_7. Secondly, we can apply P_3 to obtain $aA'bB'c \Rightarrow_3^{=1\text{-comp.}} aAbB'c \Rightarrow_4^{=1\text{-comp.}} aAbBc$, after which this sequence of applications of P_1, P_2, P_3, and P_4 can be repeated $n-1$ times, for some $n \geq 1$, to obtain $a^n Ab^n Bc^n$. Subsequently, the derivation can be finished by $a^n Ab^n Bc^n \Rightarrow_1^{=1\text{-comp.}} a^n A'b^n Bc^n \Rightarrow_2^{=1\text{-comp.}} a^n A'b^n B'c^n \Rightarrow_5^{=1\text{-comp.}} a^n Cb^n B'c^n \Rightarrow_6^{=1\text{-comp.}} a^n Cb^n Dc^n \Rightarrow_7^{=1\text{-comp.}} a^n b^n Dc^n \Rightarrow_8^{=1\text{-comp.}} a^n b^n c^n$ or, instead, by interchanging the application of P_7 and P_8. Clearly, indeed the language $L_f(G) = \{ a^n b^n c^n \mid n \geq 1 \}$, with $f \in \{\leq 1\text{-comp.}, =1\text{-comp.}\}$, is generated.

4 The Power of $\leq$ k- and = k-Competence in CDGSs

It turns out that CDGSs working in the $\leq$1-comp.-mode or in the =1-comp.-mode are at least as powerful as forbidding random context grammars, but it remains an open problem to establish their exact computational power. Due to the lack of space the proof of the following theorem is left to the reader.

Theorem 1. *For* $f \in \{\leq 1, =1\}$, $\mathcal{L}(\text{fRC}, \text{CF}) \subseteq \mathcal{L}(\text{CD}, \text{CF}, f\text{-comp.})$. □

Next we consider CDGSs working in the $=k$-comp.-mode, with $k \geq 2$. It turns out that already for $k = 2$, such CDGSs characterize the class of random context languages (and thus the class of recursively enumerable languages).

Theorem 2. *For* $f \in \{\leq k, =k\}$ *and* $k \geq 2$, $\mathcal{L}(\text{CD}, \text{CF}, f\text{-comp.}) = \mathcal{L}(\text{RC}, \text{CF})$.

Proof. The inclusions from left to right are rather obvious, since CDGSs working in any of the above competence modes can be simulated by a random context grammar. As a formal proof would be quite tedious, we leave the technical details to the reader. We now prove the inclusion from right to left for the $=k$ case, for $k \geq 2$. The $\leq k$ case can be proved in a similar way, but is left to the reader.

We first prove the case that $k = 2$, and then sketch the necessary modifications for the cases with $k > 2$. Let $G = (N, T, P, S)$ be a random context grammar in normal form[1] with rules $p : (A \to z, Q, R) \in P$, where Q is the per-

[1] A (forbidding) random context grammar $G = (N, T, P, S)$ is in *normal form* if for every (forbidding) random context rule $(A \to z, Q, R) \in P$, we have $A \notin Q \cup R$. It is easy to show that for every (forbidding) random context grammar G, there exists a (forbidding) random context grammar G' in normal form that generates the same language, i.e., $L(G') = L(G)$.

mitting context and R is the forbidding context of p. Note that the fact that G is in normal form implies that $A \notin Q \cup R$. Obviously, we can assume that $Q \cap R = \emptyset$.

To simulate G we construct a CDGS G' with nonterminals $N' = M \cup N \cup \{F, X, Y, Z\}$, where $M = \{\, [p],\ [p, B],\ [p, \neg C] \mid p : (A \to z, Q, R) \in P,\ B \in Q,\ C \in R \,\}$, such that the unions above are disjoint, the set of terminals T is disjoint from N', the axiom is SZ, and the components are as defined below.

For each random context rule $p : (A \to z, Q, R)$, we construct the components $\{P_{p,start}\} \cup \{\, P^{check}_{p,B} \mid B \in Q \,\} \cup \{\, P^{check}_{p,\neg C} \mid C \in R \,\} \cup \{P_{p,apply}\}$ described below. At any moment, we can see from the subscripts of these components which step of the simulation is performed: After we *start* simulating the application of the rule $p \in P$, we check the (non-)presence of the permitting (forbidding) symbols from the permitting context, Q (forbidding context, R) of this rule, or we *apply* the context-free production $A \to z$ of this rule. Next to these components we introduce below two more components, P_X and P_{YZ}. For now we assume that both Q and R are nonempty. The other cases will later be dealt with separately.

The idea of the simulation is the following. Before the simulation of the application of a random context rule, a marker from the set M is introduced in the sentential form. With the aid of this marker, we check the presence of the permitting and the non-presence of the forbidding symbols. First we check the presence of the permitting symbols starting with the first such symbol. The only component that is able to rewrite the leading marker is 1-competent whenever this permitting symbol is not present in the sentential form, and 2-competent whenever this symbol is present. The moment in which this component is applied it introduces the symbol X and by doing so becomes $\geq$3-competent. This procedure is repeated for all the remaining permitting symbols. Secondly, we check that no forbidding symbol is present. Again, we start with the first such forbidding symbol. This time the only component that is able to rewrite the leading marker is 3-competent whenever this forbidding symbol is present, in which case no successful derivation exists.

We now present more details of the construction. The simulation starts with the application of the component

$$P_{p,start} = \{A \to [p]XY\} \cup \{X \to F,\ Y \to F,\ Z \to Z\} \cup \{\, L \to F \mid L \in M \,\},$$

for some rule $p : (A \to z, Q, R) \in P$. This component introduces the leading marker $[p]$ indicating that we start to simulate p, which thus requires the presence of A. The moment in which one occurrence of A is rewritten, this component moreover becomes $\geq$3-competent, which thus guarantees that only one such an occurrence is rewritten. Since during the whole simulation the leading marker as well as the symbols Y and Z are present in the sentential form, no simulation of a rule different from p can be started once the simulation of p was started.

Before testing the presence of the permitting context Q of p, we have to remove X (or the derivation eventually is blocked) with the component

$$P_X = \{X \to \lambda,\ Z \to Z\}.$$

This component is 2-competent whenever X is present in the sentential form, and becomes 1-competent after it has erased X.

Let the permitting context of p be $Q = \{B_1, \ldots, B_{\ell_p}\}$. The simulation continues by applying, for all $2 \leq j \leq \ell_p$, the components

$$P^{check}_{p,B_1} = \{[p] \to [p, B_1]\} \cup \{B_1 \to B_1X\} \cup \{X \to F\} \cup \{\, L \to F \mid L \in M \,\} \text{ and}$$

$$P^{check}_{p,B_j} = \{[p, B_{j-1}] \to [p, B_j]\} \cup \{B_j \to B_jX\} \cup \{X \to F\} \cup \{\, L \to F \mid L \in M \,\},$$

alternated with the component P_X erasing the X's inbetween.

The sequence of components thus applied is P^{check}_{p,B_1},P_X, $\ldots$, $P^{check}_{p,B_{\ell_p}}$, P_X. Each such a component P^{check}_{p,B_j}, $1 \leq j \leq |Q|$, is 2-competent once it is applied due to the presence of the symbol B_j, which is in accordance with the fact that rule p can only be applied when this symbol from its permitting context is present. If this symbol is not present, then such a component is 1-competent due to the presence of the leading marker $[p, B_{j-1}]$ and thus not applicable. Note, moreover, that if the X was not removed and the symbol B_j is not present, then such a component would be 2-competent. In that case it could either replace the X by an F or replace the leading marker $[p, B_{j-1}]$ by $[p, B_j]$, remain 2-competent, and replace either the X or $[p, B_j]$ by an F. Also all these cases are in accordance with the permitting context of p.

Now the moment in which the production $B_j \to B_jX$ is applied, this component becomes $\geq$3-competent, and it has successfully tested the presence of B_j. Note that no derivation can be successful in case the leading marker $[p, B_{j-1}]$ is not replaced by the application of $[p, B_{j-1}] \to [p, B_j]$ *before* the application of $B_j \to B_jX$. We also note that in case there were more occurrences of B_j in the sentential form, then still only one occurrence is rewritten.

When we arrive at this point, we have thus successfully tested the presence of all the symbols from the permitting context of rule p. Hence we are ready to test the non-presence of all the symbols from its forbidding context. Let $p : (A \to z, Q, R)$, and let $R = \{C_1, \ldots, C_{m_p}\}$. The simulation continues with the application of the component

$$P^{check}_{p,\neg C_1} = \{[p, B_{\ell_p}] \to [p, \neg C_1]\} \cup \{C_1 \to F\} \\ \cup \{X \to F,\ Z \to Z\} \cup \{\, L \to F \mid L \in M \setminus \{[p, \neg C_1]\} \,\}.$$

Given the current sentential form, this component is 2-competent or 3-competent, depending on the presence of C_1 in the sentential form. If C_1 is present, then no successful derivation exists. This is in accordance with the fact that in that case the rule p cannot be applied due to the fact that the symbol C_1 from its forbidding context is present in the sentential form.

Subsequently we apply, for all $2 \leq k \leq m_p$, the components

$$P^{check}_{p,\neg C_k} = \{[p, \neg C_{k-1}] \to [p, \neg C_k]\} \cup \{C_k \to F\} \\ \cup \{X \to F,\ Z \to Z\} \cup \{\, L \to F \mid L \in M \setminus \{[p, \neg C_k]\} \,\}.$$

The sequence of components thus applied is $P^{check}_{p,\neg C_2}, \ldots, P^{check}_{p,\neg C_{m_p}}$. If along the way a symbol C_k, for $2 \leq k \leq |R|$, from the forbidding context of the rule p is present, then no successful derivation exists.

When we arrive at this point without having introduced an F, we have thus successfully tested also the non-presence of all the symbols from the forbidding context of rule p. Hence we are ready to actually simulate the application of the context-free production $A \to z$ of the rule p. Obviously, we need to do so only once, but this is guaranteed by the fact that there is only one occurrence of the leading marker $[p, \neg C_{m_p}]$. To this aim, we apply the 2-competent component

$$P_{p,apply} = \{[p, \neg C_{m_p}] \to z\} \cup \{Z \to Z\} \cup \{\, L \to F \mid L \in M \,\},$$

which becomes 1-competent as soon as $[p, C_{m_p}] \to z$ is applied, in which case we have successfully simulated the application of the rule p.

All that remains is to bring the sentential form back to a form from which the simulation of another rule from G can be started or to finish the derivation. This is done by removing Y. To this aim we apply the 2-competent component

$$P_{YZ} = \{Y \to \lambda,\ Z \to \lambda\}.$$

At this point it is important to note that eventually it is this component P_{YZ} that can finish the derivation by removing not only Y, but also Z. However, it can be seen that no successful derivation exists if Z is removed rather than Y.

If component P_{YZ} is applied earlier on in the derivation, then such an application would remove either Y or Z, but not both. For the same reason as above, no successful derivation exists if Z is removed. Now assume that Y is removed. Since the only use of Y is to guarantee that component P_{YZ} is 2-competent when we want to finish the derivation by removing both Y and Z, an earlier application of component P_{YZ} is harmless as long as it occurs *before* the final application of a rule from G. If, on the contrary, Y is removed after the final application of a rule from G, then component P_{YZ} can never become 2-competent, thus Z can never be removed, and no successful derivation exists.

Let us now describe how to adapt the construction for the cases dealing with a random context rule $p : (A \to z, Q, R)$ in which Q and/or R is empty. We distinguish three cases and describe only the components that must be changed:

(1) In case $R \neq Q = \emptyset$, we construct no components of the form P^{check}_{p,B_j} and replace production $[p, B_{\ell_p}] \to [p, \neg C_1]$ by $[p] \to [p, \neg C_1]$ in component $P^{check}_{p,\neg C_1}$.
(2) In case $Q \neq R = \emptyset$, we remove all components of the form $P^{check}_{p,\neg C_k}$ and replace production $[p, \neg C_{m_p}] \to z$ by $[p, B_{\ell_p}] \to z$ in component $P_{p,apply}$.
(3) In case $Q = R = \emptyset$, we construct no components of the form P^{check}_{p,B_j} or $P^{check}_{p,\neg C_k}$ and replace production $[p, \neg C_{m_p}] \to z$ by $[p] \to z$ in component $P_{p,apply}$.

The CDGS G' constructed above correctly simulates the random context grammar G and generates the language $L(G)$, when working in the =2-comp.-mode. This proves the statement of this theorem for the case $k = 2$.

Let us now briefly discuss the proof of the more general case. Let $k > 2$. We now sketch how to adapt G' such that the resulting CDGS G'' correctly simulates the random context grammar G and generates the language $L(G)$, when working in the $=k$-comp.-mode. We do not specify all the resulting modifications, but rather take one such component and show how it is adapted for inclusion in G''.

Recall that the component $P_{p,start}$ becomes $\geq$3-competent (and can thus no longer be applied) the moment in which production $A \to [p]XY$ is applied. The reason for this is as follows. The application of this production introduces each of the symbols $[p]$, X, and Y to the sentential form. Since the component moreover contains a production for each of these symbols, this immediately makes the component $\geq$3-competent. To make the simulation work in the $=k$-comp. mode for $k \geq 3$, we replace the component $P_{p,start}$ in G' by the component $P''_{p,start} = \{A \to [p]XYZ_1 \cdots Z_{k-2}\} \cup \{X \to F,\ Y \to F,\ Z \to Z\} \cup \{Z_1 \to F,\ \ldots,\ Z_{k-2} \to F\} \cup \{\, L \to F \mid L \in M \,\}$, where $Z_1, \ldots, Z_{k-2}$ are new symbols different from $N' \cup T$. We leave the other modifications to the reader. □

5 The Power of $\geq$ k-Competence in CDGSs

We start our investigations with CDGSs working in the $\geq$1-comp.-mode. Since the $\geq$1-comp.-mode by definition equals the t-mode of derivation, as introduced in [3], we immediately obtain the following result, which is due to [3].

Theorem 3. $\mathcal{L}(\mathrm{CD}, \mathrm{CF}, \geq 1\text{-comp.}) = \mathcal{L}(\mathrm{ET0L})$. □

Next we consider CDGSs working in the $\geq k$-comp.-mode, with $k \geq 2$. It turns out that already for $k = 2$, such CDGSs characterize the class of random context ET0L languages.

Theorem 4. *For $k \geq 2$,* $\mathcal{L}(\mathrm{CD}, \mathrm{CF}, \geq k\text{-comp.}) = \mathcal{L}(\mathrm{RC}, \mathrm{ET0L})$.

Proof. Here we prove the inclusion from right to left. To prove the reverse inclusion, for any CGDS working in the $\geq k$-comp.-mode of derivation a recurrent programmed grammar can be constructed that simulates it. The quite tedious details are left to the reader.

The construction we use is strongly based on the one used in the proof of Theorem 2, except that the test for forbidding symbols is now incorporated in the component applying the simulated productions. Again, we first prove the case that $k = 2$, and then sketch the necessary modifications for the cases with $k > 2$. Let $G = (\Sigma, H, \omega, \Delta, oc, noc)$ be a random context ET0L system in normal form[2]. Without loss of generality we can assume $oc(h) \cap noc(h) = \emptyset$ for every table $h \in H$. To simulate G we construct a CDGS G' with nonterminals $N' = M \cup N \cup \{\, B' \mid B \in oc(h),\ h \in H \,\} \cup \{F, X, Y\}$, where $M = \{\, [h, B]_1,\ [h, B]_2,\ [h, B]_3 \mid h \in$

[2] A random context ET0L system $G = (\Sigma, H, \omega, \Delta, oc, noc)$ is in *normal form* if every table $h \in H$ is of the form $\{\, B \to B \mid B \in \Sigma \setminus \{A\} \,\} \cup h_A$, where $h_A = \{\, A \to z,\ A \to A \mid A \in \Sigma, z \in \Sigma^*, z \neq A \,\}$ or $h_A = \{\, A \to z \mid A \in \Sigma, z \in \Sigma^*, z \neq A \,\}$, and $A \notin oc(h) \cup noc(h)$. If table h is of the form $\{\, B \to B \mid B \in \Sigma \setminus \{A\} \,\} \cup h_A$ for some $A \in \Sigma$, then A is called the *active symbol* of h and $A \to z$ in h_A the *active production* of h. By standard constructions one can show that for every random context ET0L system G, there exists a random context ET0L system G' in normal form that generates the same language, i.e., $L(G') = L(G)$.

$H,\ B \in oc(h)\}$, such that the unions are disjoint, terminals T disjoint from N', axiom XYS, and the components defined below.

For each table $h \in H$, with $oc(h) = \{B_1, \ldots, B_{\ell_h}\}$, we construct the components $\{\, P_{h,B_j,1},\ P_{h,B_j,2},\ P_{h,B_j,3} \mid 1 \le j \le \ell_h \,\} \cup \{P_{h,apply}\}$ described below. We also introduce one more component, P_{finish}. For now we assume that both $oc(h)$ and $noc(h)$ are nonempty. The other cases will later be dealt with separately.

The idea of the simulation is similar to that of the proof of Theorem 2. We now describe more details of the construction. Let $h \in H$ be a table of G and let $oc(h) = \{B_1, \ldots, B_{\ell_h}\}$. The simulation of h starts by applying the component

$$P_{h,B_1,1} = \{\, A \to [h, B_1]_1 \mid A \text{ is the active symbol of } h \,\} \cup \{[h, B_1]_1 \to [h, B_1]_1,\ B_1 \to B_1'\} \cup \{\, L \to F \mid L \in M \,\}.$$

This component can be applied if and only if both A, the active symbol of h, and $B_1 \in oc(h)$ are present, after which it remains $\ge$2-competent until all occurrences of B_1 have been primed. Moreover, we shall shortly see that no successful derivation exists unless all occurrences of A have been replaced by $[h, B_1]_1$.

Consequently, the component

$$P_{h,B_1,2} = \{[h, B_1]_1 \to [h, B_1]_2\} \cup \{B_1' \to B_1,\ B_1 \to B_1\} \cup \{\, A \to F \mid A \text{ is the active symbol of } h \,\} \cup \{\, L \to F \mid L \in M \setminus \{[h, B_1]_2\} \,\}$$

remains $\ge$2-competent until all occurrences of $[A_i, p_i, 1]_1$ have been replaced by $[h, B_1]_2$. Moreover, we shall shortly see that no successful derivation exists unless at least one occurrence of B_1' is unprimed. However, due to the presence of $B_1 \to B_1$ this means that this component remains $\ge$2-competent until all occurrences of B_1' are unprimed. Since this is the only component capable of unpriming B_1', it is this component that guarantees that no successful derivation exists if component $P_{h,B_1,1}$ has not replaced all occurrences of A by $[h, B_1]_1$.

The component which guarantees that no successful derivation exists if component $P_{h,B_1,2}$ has not unprimed all occurrences of B_1' is

$$P_{h,B_1,3} = \{[h, B_1]_2 \to [h, B_1]_3\} \cup \{B_1 \to B_1\} \cup \{\, L \to F \mid L \in M \setminus \{[h, B_1]_3\} \,\}.$$

This component is $\ge$2-competent if and only if B_1 is present. Since this is the only component replacing $[h, B_1]_2$ by $[h, B_1]_3$, no successful derivation exists if this component is not applied.

Now that we have successfully tested the presence of the first permitting symbol, the simulation continues by doing the same for the remaining permitting symbols, i.e., by applying, for all $2 \le j \le \ell_h$, the components

$$P_{h,B_j,1} = \{[h, B_{j-1}]_3 \to [h, B_j]_1,\ [h, B_j]_1 \to [h, B_j]_1\} \cup \{B_j \to B_j'\} \cup \{\, L \to F \mid L \in M \,\},$$

$$P_{h,B_j,2} = \{[h, B_j]_1 \to [h, B_j]_2\} \cup \{B_j' \to B_j,\ B_j \to B_j\} \cup \{\, A \to F \mid A \text{ is the active symbol of } h \,\} \cup \{\, L \to F \mid L \in M \setminus \{[h, B_j]_2\} \,\}, \text{ and}$$

$$P_{h,B_j,3} = \{[h, B_j]_2 \to [h, B_j]_3\} \cup \{B_j \to B_j\} \cup \{\, L \to F \mid L \in M \setminus \{[h, B_j]_3\} \,\}.$$

Hence, the sequence of components applied is $P_{h,B_2,1}$, $P_{h,B_2,2}$, $P_{h,B_2,3}$, $\ldots$, $P_{h,B_{\ell_h},1}$, $P_{h,B_{\ell_h},2}$, $P_{h,B_{\ell_h},3}$. If along the way a permitting symbol B_j, for $2 \leq j \leq \ell_h$, is not present, then the derivation is blocked due to the fact that in that case component $P_{h,B_j,1}$ is 1-competent and thus cannot be applied.

When we arrive at this point, we have thus successfully tested the presence of all the symbols from the permitting context of table h. Hence we are ready to test the non-presence of all the symbols from its forbidding context and to subsequently simulate the application of its active production $A \to z$ by replacing some of the occurrences of $[h, B_{\ell_h}]_3$ by z (and the remaining occurrences by A). The simulation continues with the application of the component

$$P_{h,apply} = \{[h, B_{\ell_h}]_3 \to z,\ [h, B_{\ell_h}]_3 \to A \mid A \to z \text{ is the active production of } h\} \cup \{X \to X\} \cup \{\, C \to F \mid C \in noc(h)\,\} \cup \{\, L \to F \mid L \in M\,\}.$$

In this component we use the extra marker X to guarantee its $\geq$2-competence whenever all the permitting symbols are present. In case any forbidding symbol $C \in noc(h)$ is present, then a failure symbol F must be introduced or else $P_{h,apply}$ remains $\geq$2-competent. This is in accordance with the fact that in that case no active production from table h can be applied due to the fact that a symbol from its forbidding context is present in the sentential form. Hence we have successfully applied table $h \in H$ and the sentential form is in a form from which the simulation of another table from G can be started.

It remains to erase the symbols X and Y from the sentential form as soon as a successful derivation of a terminal word in G has been simulated, i.e., when the sentential form is XYw, for some $w \in T^*$. This is achieved by the component

$$P_{finish} = \{X \to \lambda,\ Y \to \lambda\}.$$

Note that both X and Y are erased by this component if and only if it is applied to a sentential form XYw, for some $w \in T^*$. In all other cases, only one of these symbols is erased because this component becomes 1-competent the moment this happens. Since neither of these symbols can be rewritten by any component other than P_{finish}, no successful derivation exists if this component is applied to a sentential form that is not of the form XYw with $w \in T^*$.

Similar to the way we did this in the proof of Theorem 2, our construction can easily be adapted for tables $h \in H$, where $oc(h)$ and/or $noc(h)$ is empty.

This completes the description of the CDGS G'. It is left to the reader to verify that whenever components are applied in an order different from the one prescribed above by the leading marker, then no successful derivation exists. This is achieved by the inclusion of productions in components which guarantee—where necessary—that a failure symbol F (which can never be rewritten) is introduced, or that the derivation is blocked because no more component can be applied. Both of these cases clearly block the derivation. The CDGS G' constructed above correctly simulates the random context ET0L system G and generates the language $L(G)$, when working in the $\geq$2-comp.-mode. This proves the statement of

this theorem for the case $k = 2$. The proof of the more general case is rather straightforward and it is thus left to the reader. □

6 Conclusion

In this paper we have introduced the $\leq k$-comp.-, $=k$-comp.-, and $\geq k$-comp.-mode of derivation, with $k \geq 1$, as cooperation protocols for CDGSs. They enable a component of a CDGS to rewrite a sentential form only if it is at most, exactly, or at least k-competent, resp., on that string. CDGSs working in the $\leq$2-comp.- or =2-comp.-mode of derivation characterize the class of recursively enumerable languages, while those working in the $\geq$2-comp.-mode of derivation characterize the class of random context ET0L languages, which in turn equals the class of recurrent programmed languages with appearance checking [11], that is obviously included in $\mathcal{L}$(RE), but it is not known whether it is strictly included or not. In Theorem 4 we provide yet another alternative characterization of this language class, which thus might shed new light on this longstanding open problem.

Finally, the components of the CDGSs used in the proofs in this paper are very simple grammars, with only a limited number of productions. The results of this paper thus demonstrate that cooperating agents with a rather restricted level of competence are able to solve arbitrarily complicated problems. Furthermore, if these agents are represented by context-free grammars, then there is no difference between the cases in which each agent has an exact level of competence and those in which it has a bounded level of competence—compare with [9].

References

1. H. Bordihn and E. Csuhaj-Varjú, On competence and completeness in CD grammar systems. *Acta Cybernetica* 12, 4 (1996), 347–361.
2. H. Bordihn and M. Holzer. Grammar systems with negated conditions in their cooperation protocols. *JUCS* 6 (2000), 1165–1184.
3. E. Csuhaj-Varjú and J. Dassow, On cooperating distributed grammar systems. *EIK* 26 (1990), 49–63.
4. E. Csuhaj-Varjú, J. Dassow, and M. Holzer, On a Competence-based Cooperation Strategy in CD Grammar Systems. Technical Report 2004/3, Theoretical Computer Science Research Group, Computer and Automation Research Institute, Hungarian Academy of Sciences, 2004.
5. E. Csuhaj-Varjú, J. Dassow, and M. Holzer. CD Grammar Systems with Competence Based Entry Conditions in Their Cooperation Protocols. Technical Report 2004/4, Theoretical Computer Science Research Group, Computer and Automation Research Institute, Hungarian Academy of Sciences, 2004.
6. E. Csuhaj-Varjú, J. Dassow, J. Kelemen, and Gh. Păun, *Grammar Systems—A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, 1994.

7. J. Dassow and Gh. Păun, *Regulated Rewriting in Formal Language Theory*, *EATCS Monographs on Theoretical Computer Science* 18, Springer-Verlag, Berlin, 1989.
8. H. Fernau and D. Wätjen, Remarks on regulated limited ET0L systems and regulated context-free grammars. *TCS* 194 (1998), 35–55.
9. R. Meersman and G. Rozenberg, Cooperating grammar systems. In *Proceedings MFCS'78*, *LNCS* 64, Springer-Verlag, Berlin, 1978, 364–374.
10. Gh. Păun and G. Rozenberg, Prescribed teams of grammars. *Acta Informatica* 31 (1994), 525–537.
11. S.H. von Solms. Some notes on ET0L-languages. *IJCM* 5 (1976), 285–296.