Acta Cybernetica **12** (1996) 361-379.

Accepting Multi-Agent Systems II

Henning FERNAU * † ‡ Markus HOLZER * ‡

Abstract

We continue our previous research on cooperating distributed grammar systems (CDGS) and variants thereof as language acceptors [16]. Here, we classify the accepting capacity of CDGS working in the modes recently introduced by the authors together with Freund [14]. Moreover, we study (prescribed) teams as language accepting mechanisms.

In this way, we solve an open problem from the area of accepting grammars: there exists a grammar family such that its generating capacity is strictly more powerful that its accepting capacity, see [6] for a recent survey.

1 Introduction

In Artificial Intelligence (AI), a common methodology in order to achieve a goal, which can hardly be done by a single expert or agent, is to create a so-called multi-agent system which solves the task using distributed and cooperating agents, see [29] for a survey on this area. Blackboard architecture models [10] can be seen as an approach to model the communication aspects of the agents.

On the other hand, one general concept in problem solving methods in AI are production systems [23], which arise from a computational formalism proposed by Post [27] that was based on string replacement. Many generalizations of production systems are proposed in AI, e.g., rule-based systems, blackboard systems, or pattern-directed-inference systems. Production systems are closely connected to string rewriting, one of the backbones of formal language theory. In order to understand the nature of production systems, it is therefore natural to study them on well-known and traditional formal language theoretical devices. Based on blackboard systems, Csuhaj-Varjú and Dassow [7] introduced cooperating distributed grammar systems (CDGS), where each component (grammar) corresponds to the particular knowledge source of the system (this is an expert or agent), and the global database—the blackboard—is modelled by a common sentential form, where the

*Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, D-72076 Tübingen, Germany. E-mail:fernau/holzer@informatik.uni-tuebingen.de

†Supported by Deutsche Forschungsgemeinschaft grant DFG La 618/3-1.

‡Supported by 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

components perform their rewritings. Independently and sometimes with different motivation, also other authors introduced similar computational models, see, e.g., [1, 20, 21].

In the systems considered by Csuhaj-Varjú and Dassow and in a series of subsequent papers, all components work according to the same strategy; more precisely, the rewriting by another component can be done, e.g., after a given number of steps. For an overview on this topic, refer to [8, 25]. This model of multi-agent systems is not very realistic, because usually such agents have different capabilities. Therefore, a generalization of CDGS called hybrid CDGS has been investigated [22, 24, 25]. Another idea is to allow the formation of teams of agents, as proposed in [18, 19, 26].

In this paper, we take the original above-sketched idea of CDGS, but contrary to the approach of Csuhaj-Varjú *et al.* [8], we take, instead of generating grammars, accepting grammars as agents. They try to derive a given goal word (axiom) in a cooperating and distributive manner. In this way, we pursue our studies on accepting grammars and systems [3, 4, 5, 6, 11]. Accepting CDGS have been investigated in the works of the authors together with Bordihn [16] and Freund [12, 13]. For an intermediate approach, combining generating and accepting grammars in CDGS, we refer to Fernau and Holzer [15].

Depending on the mode in which the grammars cooperate, we obtain, on the one hand, equivalences between the accepting and generating case, but also, on the other hand, results which are fundamentally different, this means that accepting devices are much more powerful than generating ones or vice versa, thereby solving an open problem in the theory of accepting grammars [3].

Observe that, when defining accepting counterparts of existing generating devices, we want to carry over the original idea and motivation of the generating mechanism in order to define the corresponding accepting mechanism. Formally, such accepting grammars look like their generating counterparts, just turning the core productions "around" and keeping the control mechanism ruling the application of these productions textually the same. In the case of CDGS, this procedure is very well motivated by the original ideas stemming from AI: while generating devices correspond to forward deduction system, accepting devices correspond to backward deduction systems.

The present paper extends our studies on accepting CDGS in three directions: (1) We consider other working modes of the components which have been introduced in [14] and (2) we consider (external) hybridizations of these new modes (with the new and old ones). Finally (3) we briefly consider accepting CDGS with (prescribed) teams.

This is reflected in the organization of our paper. In the next section, we introduce the necessary notions. Section 3 lists the easy cases (the interval mode and t combined with greater than or equal to k) where we can profit from our previous results [16] on the t-mode. Section 4 contains the more interesting case incorporating the t-mode combined with either equal than or less equal than k. Strange things can be observed here, e.g., no unary infinite language can be accepted by CDGS working in $(t \wedge \leq k)$-mode. This leads to the first examples of

grammar mechanisms where the generating power of such CDGS is greater than their accepting power. In Section 5, we we consider accepting hybrid CDGS, where the (external) hybridization incorporates also the modes $(t \wedge \leq k)$, $(t \wedge = k)$, and $(t \wedge \geq k)$. Furthermore, we consider the number of components in a CDGS as a natural measure of descriptional complexity for CDGS. Since each component corresponds to one expert according to the original AI motivation, we may paraphrase our results as follows: in backward deduction systems, two experts are enough. This contrasts to the situation found in forward deduction systems where it is only known that three or four experts are enough. Finally, we consider in Section 6 CDGS with (prescribed) teams as language acceptors. Observe that similar systems have been studied in the context of array grammars from a quite practical viewpoint, see [12, 13].

2 Definitions

We assume the reader to be familiar with some basic notions of formal language theory, as contained in Dassow and Păun [9]. Especially, we consider two languages L_1, L_2 to be equal if and only if $L_1 \setminus \{\lambda\} = L_2 \setminus \{\lambda\}$, where λ denotes the empty word. We use $\subseteq$ to denote inclusion, while $\subset$ denotes strict inclusion. The set of positive integers is denoted by $\mathbf{N}$. The end of a proof or of a proved statement is marked by $\square$. Let $\mathcal{L}(\mathrm{FIN})$ be the family of finite languages. The families of languages generated by regular, linear context-free, context-sensitive, type-0 Chomsky grammars, ET0L systems, context-free ordered, context-free programmed, and context-free programmed grammars with appearance checking are denoted by $\mathcal{L}^{gen}(\mathrm{REG})$, $\mathcal{L}^{gen}(\mathrm{LIN})$, $\mathcal{L}^{gen}(\mathrm{CF})$, $\mathcal{L}^{gen}(\mathrm{CS})$, $\mathcal{L}^{gen}(\mathrm{RE})$, $\mathcal{L}^{gen}(\mathrm{ET0L})$, $\mathcal{L}^{gen}(\mathrm{O}, \mathrm{CF})$, $\mathcal{L}^{gen}(\mathrm{P}, \mathrm{CF})$, and $\mathcal{L}^{gen}(\mathrm{P}, \mathrm{CF}, \mathrm{ac})$, respectively. A subscript *fin* (k, or 1, respectively) denotes the family of languages generated by the appropriate device restricting the derivation to be of finite index (finite index k, finite index 1, respectively). For a definition of the finite index property we refer to [9].

A superscript *acc* instead of *gen* is used to denote the family of languages accepted by the appropriate device. If we want to exclude λ-rules, we add $-\lambda$ in our notations.

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

For the convenience of the reader, we repeat the basic definitions of CDGS, hybrid CDGS, respectively, adapted from Păun [24], in a way suitable for the interpretation both as generating and accepting systems.

A *cooperating distributed grammar system* (CDGS for short) of degree n, with $n \geq 1$, is a $(n+3)$-tuple $G = (N, T, S, P_1, \ldots, P_n)$, where N, T are disjoint alphabets of nonterminal and terminal symbols, respectively, $S \in N$ is the axiom, and $P_1, \ldots, P_n$ are finite sets of rewriting rules over $N \cup T$.

Throughout this paper, we consider only context-free rewriting rules. Since we are interested in generating and accepting systems, we further distinguish between

so-called *generating rules*, which have the form $A \to w$, with $A \in N$ and $w \in (N \cup T)^*$, and *accepting rules*, which are of the form $w \to A$, with $A \in N$ and $w \in (N \cup T)^*$.

Let G be a CDGS with only generating rules. For $x, y \in (N \cup T)^*$ and $1 \leq i \leq n$, we write $x \Rightarrow_i y$ if and only if $x = x_1 A x_2$, $y = x_1 z x_2$ for some $A \to z \in P_i$. Hence, subscript i refers to the component to be used. By $\Rightarrow_i^{\leq k}$, $\Rightarrow_i^{=k}$, $\Rightarrow_i^{\geq k}$, $\Rightarrow_i^*$ we denote a derivation consisting of at most k steps, exactly k steps, at least k steps, an arbitrary number of steps, respectively. We also write $x \Rightarrow_i^t y$ if and only if $x \Rightarrow_i^* y$ and there is no z such that $y \Rightarrow_i z$. 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: there exists a derivation which satisfies both properties 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 $y \Rightarrow_i z$.

Let $D := \{*, t\} \cup \{\leq k, = k, \geq k \mid k \in \mathbf{N}\} \cup \{(\geq k_1 \wedge \leq k_2), (t \wedge \leq k), (t \wedge = k), (t \wedge \geq k) \mid k_1, k_2, k \in \mathbf{N} \text{ and } k_1 \leq k_2\}$.

The language *generated* in the f-mode, $f \in D$, by a CDGS G with only generating rules is defined as:

$$L_f^{gen}(G) := \{w \in T^* \mid S \Rightarrow_{i_1}^f \alpha_1 \Rightarrow_{i_2}^f \ldots \Rightarrow_{i_{m-1}}^f \alpha_{m-1} \Rightarrow_{i_m}^f \alpha_m = w \text{ with } m \geq 1, 1 \leq i_j \leq n, \text{ and } 1 \leq j \leq m\}.$$

If $f \in D$, the families of languages generated in f-mode by [λ-free] CDGS with at most n components are denoted by $\mathcal{L}^{gen}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], f)$. If the number of components is not restricted, we write $\mathcal{L}^{gen}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], f)$.

For CDGS with only accepting rules, we define the relations $x \Rightarrow_i y$ and $x \Rightarrow_i^f y$ accordingly. Hence, we define the language *accepted* in f-mode, $f \in D$, by a CDGS G with only accepting rules as follows:

$$L_f^{acc}(G) := \{w \in T^* \mid w \Rightarrow_{i_1}^f \alpha_1 \Rightarrow_{i_2}^f \ldots \Rightarrow_{i_{m-1}}^f \alpha_{m-1} \Rightarrow_{i_m}^f \alpha_m = S \text{ with } m \geq 1, 1 \leq i_j \leq n, \text{ and } 1 \leq j \leq m\}$$

If $f \in D$, the families of languages accepted in f-mode by [λ-free] CDGS with at most n components are denoted by $\mathcal{L}^{acc}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], f)$. If the number of components is not restricted, we write $\mathcal{L}^{acc}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], f)$.

If each component of a CDGS may work in a different mode, then we get the notion of *(externally) hybrid CDGS* of degree n, with $n \geq 1$, which is a $(n+3)$-tuple $G = (N, T, S, (P_1, f_1), \ldots, (P_n, f_n))$, where $N, T, S, P_1, \ldots, P_n$ are as in CDGS, and $f_i \in D$, for $1 \leq i \leq n$. Thus, we can define the language *generated* by a hybrid CDGS with only generating rules as:

$$L^{gen}(G) := \{w \in T^* \mid S \Rightarrow_{i_1}^{f_{i_1}} w_1 \Rightarrow_{i_2}^{f_{i_2}} \ldots \Rightarrow_{i_{m-1}}^{f_{i_{m-1}}} w_{m-1} \Rightarrow_{i_m}^{f_{i_m}} w_m = w \text{ with } m \geq 1, 1 \leq i_j \leq n, \text{ and } 1 \leq j \leq m\}$$

Accordingly, accepting hybrid CDGS can be defined. If $F \subseteq D$, the family of languages generated (accepted, respectively) by [λ-free] CDGS with at most n

components, each component working in one of the modes contained in F, are denoted by $\mathcal{L}^{gen}(\mathrm{HCD}_n, \mathrm{CF}[-\lambda], F)$ ($\mathcal{L}^{acc}(\mathrm{HCD}_n, \mathrm{CF}[-\lambda], F)$, respectively). Similarly, $\mathcal{L}^{gen}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F)$, and $\mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F)$, respectively, is written when the number of components is not restricted.

3 When generating is weaker than accepting

The easiest case from the new modes is the interval mode. Fortunately, the observations given in [16, page 128, Theorem 3.2] showing that in case of classical modes, except for the t-mode, equivalence between generating and accepting devices prevails, readily transfers to the $(\geq k_1 \wedge \leq k_2)$-mode. Hence, we get:

Theorem 3.1 *If* $N \in \mathbf{N} \cup \{\infty\}$ *and* $f \in \{(\geq k_1 \wedge \leq k_2) \mid k_1, k_2 \in \mathbf{N}, k_1 \leq k_2\}$, *then* $\mathcal{L}^{gen}(\mathrm{CD}_N, \mathrm{CF}[-\lambda], f) = \mathcal{L}^{acc}(\mathrm{CD}_N, \mathrm{CF}[-\lambda], f)$. □

We turn our attention to grammar systems working in $(t \wedge \geq k)$-mode. Note that $(t \wedge \geq 1)$- and the classical t-mode trivially coincide, which leads us to:

Theorem 3.2 *If* $N \in \mathbf{N} \cup \{\infty\}$, *then, for any* $N \geq 3$,

$$\begin{aligned}\mathcal{L}^{gen}(\mathrm{CF}) &= \mathcal{L}^{gen}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \geq 1)) &&= \mathcal{L}^{gen}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], (t \wedge \geq 1))\\ &\subset \mathcal{L}^{gen}(\mathrm{CD}_N, \mathrm{CF}[-\lambda], (t \wedge \geq 1)) &&= \mathcal{L}^{gen}(\mathrm{ET0L}).\end{aligned}$$

For k in general, the situation is a little bit different from the previous one.

Theorem 3.3 *If* $N \in \mathbf{N} \cup \{\infty\}$, *then, for any* $N \geq 3$ *and for each* $k \geq 2$,

$$\begin{aligned}\mathcal{L}^{gen}(\mathrm{CF}) &= \mathcal{L}^{gen}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \geq k)) &&\subset \mathcal{L}^{gen}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], (t \wedge \geq k))\\ &\subseteq \mathcal{L}^{gen}(\mathrm{CD}_N, \mathrm{CF}[-\lambda], (t \wedge \geq k)) &&= \mathcal{L}^{gen}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geq k)).\end{aligned}$$

The latter class coincides with the family of E[P]T0L languages with random context conditions [9, 28], as shown in [14], a special case of $\mathcal{L}^{gen}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac})$.

The case is totally different for accepting CDGS. Nevertheless, as in the case of t-mode, the admittance of λ-productions does not enhance the accepting power of CDGS working in $(t \wedge \geq k)$-mode.

Theorem 3.4 *If* $N \in \mathbf{N} \cup \{\infty\}$, *then, for any* $N \geq 2$ *and for each* $k \geq 1$,

$$\begin{aligned}\mathcal{L}^{gen}(\mathrm{CF}) &= \mathcal{L}^{acc}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \geq k)) &&\subset \mathcal{L}^{acc}(\mathrm{CD}_N, \mathrm{CF}[-\lambda], (t \wedge \geq k))\\ &= \mathcal{L}^{acc}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geq k)) &&= \mathcal{L}^{gen}(\mathrm{CS}).\end{aligned}$$

Proof. The first relation is obvious. By [16, Theorem 4.5], we know that, for each $N \geq 2$, $\mathcal{L}^{acc}(\mathrm{CD}_N, \mathrm{CF}[-\lambda], t) = \mathcal{L}^{acc}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], t) = \mathcal{L}^{gen}(\mathrm{CS})$. Since t- and $(t \wedge \geq 1)$-mode trivially coincide, the results carry over to the $(t \wedge \geq 1)$-mode. By introducing prolongating rules, we obtain the desired result for $(t \wedge \geq k)$-mode for k in general. □

Corollary 3.5 *Let $k \in \mathbf{N}$. Then, we have*

$$\mathcal{L}^{gen}(\mathrm{CF}) = \mathcal{L}^{gen}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \geq k)) = \mathcal{L}^{acc}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \geq k)).$$

Moreover, if $N \in \mathbf{N} \cup \{\infty\}$ with $N \geq 2$, we get

$$\mathcal{L}^{gen}(\mathrm{CD}_N, \mathrm{CF}[-\lambda], (t \wedge \geq k)) \subseteq \mathcal{L}^{acc}(\mathrm{CD}_N, \mathrm{CF}[-\lambda], (t \wedge \geq k)),$$

where the inclusion is known to be strict only in the absence of λ-rules. □

4 When accepting is weaker than generating

In the present section, we deal with CDGS working in $(t \wedge \leq k)$- and $(t \wedge = k)$-mode with context-free components. Again, at first we mention the known results in the generating case [14].

Theorem 4.1 *If $f \in \{(t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}\}$, then there exists a function $s_f : \mathbf{N} \to \mathbf{N}$ so that for each $n \in \mathbf{N}$,*

$$\begin{aligned}\mathcal{L}(\mathrm{FIN}) &= \mathcal{L}^{gen}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], f) \subseteq \mathcal{L}^{gen}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], f) \\ &\subset \mathcal{L}^{gen}(\mathrm{CD}_{s_f(n)}, \mathrm{CF}[-\lambda], f) \subset \mathcal{L}^{gen}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], f) = \mathcal{L}^{gen}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac}).\end{aligned}$$

The preceding theorem demonstrates that both the $(t \wedge \leq k)$- and the $(t \wedge = k)$-mode nicely fit into the known framework of formal language families. On the other hand, the accepting counterparts behave very strange in comparison to earlier results on accepting CDGS. This is shown in the next lemma.

Lemma 4.2 *For every $k \in \mathbf{N}$, no infinite one-letter language can be accepted by a CDGS working either in $(t \wedge \leq k)$ or $(t \wedge = k)$-mode.*

Proof. We only prove the statement for the $(t \wedge = k)$-mode. Assume that the CDGS $G = (N, T, S, P_1, \ldots, P_n)$ accepts an infinite one-letter language $L \subseteq \{a\}^*$ in $(t \wedge = k)$-mode. Set $M = \max\{ m \mid a^m \to A \in P_i \text{ for } 1 \leq i \leq n \}$. Since L is infinite, there exists a word a^m in L such that $m \geq 3k \cdot M$. On this word, there is no way to start the (accepting) derivation process, since every component is only able to handle at most $k \cdot M$ symbols a due to the $(t \wedge = k)$-mode. Thus, a^m does not belong L, which contradicts our assumption that L is infinite. □

Thus, $\{a\}^* \in \mathcal{L}^{gen}(\mathrm{REG}) \setminus \mathcal{L}^{acc}(\mathrm{CD}_N, \mathrm{CF}[-\lambda], f)$, if $N \in \mathbf{N} \cup \{\infty\}$ and $f \in \{(t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}\}$, but $\{\#\}\{a\}^* \in \mathcal{L}^{acc}(\mathrm{CD}_2, \mathrm{CF} - \lambda, (t \wedge = 1))$, which is shown in the following example.

Example. Let $G = (\{S, A, A', B\}, \{\#, a\}, S, P_1, P_2)$ be a CDGS with the sets $P_1 = \{\# \to B, Aa \to A', Aa \to S\}$ and $P_2 = \{B \to A, A' \to A\}$. It is easy to see that G working in $(t \wedge = 1)$-mode accepts $\{\#\}\{a\}^*$.

The idea of a special marking symbol generalizes to arbitrary regular languages. This shows that every marked regular language belongs to, e.g., the language family $\mathcal{L}^{acc}(\mathrm{CD}_2, \mathrm{CF}-\lambda, (t\wedge=1))$.

Before we consider CDGS with an arbitrary number of components, we study the case where two components are working in $(t\wedge\leq k)$- or $(t\wedge=k)$-mode together. Again, we recall what is known for these language families [14].

Theorem 4.3 *If* $f \in \{(t\wedge\leq k), (t\wedge=k) \mid k \in \mathbf{N}\}$, *then*

$$\mathcal{L}^{gen}(\mathrm{LIN}) \subseteq \mathcal{L}^{gen}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], f) \subseteq \mathcal{L}_k^{gen}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac}).$$

The latter inclusion is known to be strict only in case $k = 1$.

First, consider the trivially coinciding modes $(t\wedge\leq 1)$ and $(t\wedge=1)$ for accepting CDGS with two components. Surprisingly, we find:

Theorem 4.4 $\mathcal{L}^{acc}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], (t\wedge=1)) \subset \mathcal{L}^{gen}(\mathrm{LIN})$.

Before we prove this theorem, let us mention that the preceding theorem answers an open question stated in [3]: is there a grammar family such that the generating mode is strictly more powerful than the accepting mode? Combining the previous two theorems, we obtain:

Corollary 4.5 $\mathcal{L}^{acc}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], (t\wedge=1)) \subset \mathcal{L}^{gen}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], (t\wedge=1))$. □

For the proof of the theorem, we need a detailed analysis of the accepting $(t\wedge=1)$ derivation of the grammar system. Assume that we are given a grammar system $G = (N, T, S, P_1, P_2)$. On input w the system can mainly behave as follows. By the $(t\wedge=1)$-mode (like in the t-mode), the only way to accept a word is by an "interplay" of the two components, i.e., the sequence of production sets applied looks like $\ldots, P_1, P_2, P_1, P_2, \ldots$. W.l.o.g. assume that P_1 starts the derivation process, reducing a subword of w to some nonterminal, say A. The only way to continue is an application of a rule of P_2. At this point of derivation, we have to distinguish two cases, as illustrated in Figure 1.

1. The rule chosen from P_2 contains the previously introduced nonterminal A on the left-hand side. Hence, A with a left- and right-context is replaced by another nonterminal again. The only way to successively continue the derivation is to apply P_1 again, reducing a sub-word that contains the previously introduced nonterminal. Otherwise, the previous application of a rule of P_1 would have not been possible. Further analysis of the derivation process shows that in this case the derivation has a "fish-bone" structure like in a linear grammar.
2. The rule chosen from P_2 does not contain the previously introduced nonterminal A. Hence, after its application the derived sentential form contains exactly two nonterminals, say A and B. P_1 is not able to handle a derivation where A is involved, as long as P_2 has not changed the left- or right-context of A properly. Otherwise, the previous application of a rule of P_1 would have not been possible.

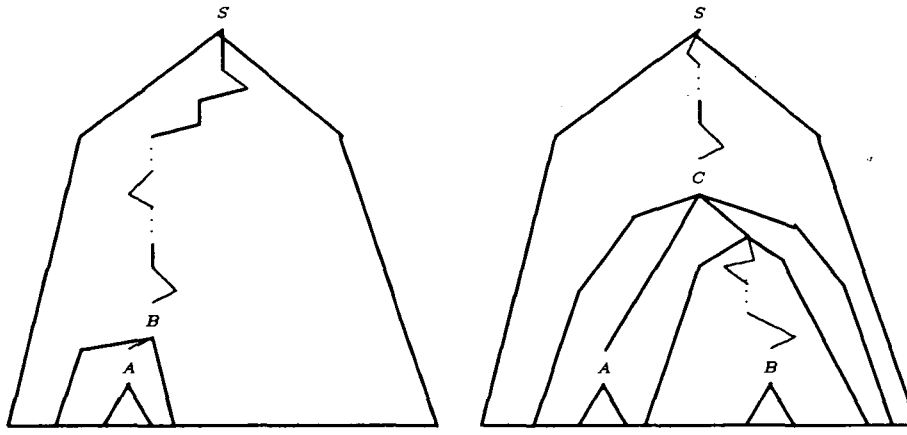

Figure 1: The cases 1. (left) and 2. (right; the case where A and B change their rôles are symmetric) of a ($t \wedge = 1$) derivation of a CDGS with two components.

Thus, the applicable rule from P_1 must contain nonterminal B on the left-hand side. This interplay between P_1 and P_2 goes on while successively reducing the word at the point where P_2 has made the first application of a rule at all. The derivation is nothing else than a fish-bone again. Then at some point P_i, for $1 \leq i \leq 2$, has prepared the context for the other to make an application of a rule where the nonterminal A is involved; this rule application breaks the fish-bone into two parts, when looking at the derivation as a whole. Afterwards, the interplay of the components continues, leading to a linear derivation structure.

We sketch the construction of a linear grammar that simulates the accepting derivation of G in a generative manner. Let $LS(P_i)$ denote the left-hand sides of the productions in P_i, i.e., $LS(P_i) = \{ w \mid w \to B \in P_i \}$.

The nonterminal of the linear grammar contains the following (finite amount of) information: (1) the actual nonterminal, (2) which component and rule starts the original derivation, (3) the control for the interplay, (4) the information which left-hand sides of the productions in both components are contained as sub-words in the sentential form derived so far, and (5) additional information to compute (4).

Thus, the generating linear grammar first guesses the component that starts and ends the original derivation process. Additionally, the first rule ever applied in the original derivation is guessed, say this is $\alpha \to A$. Further, also the form of the derivation (case 1. or 2.) is guessed. These cases are treated separately:

1. A linear derivation has to be performed, starting with the rule set that ends the original derivation. The interplay as well as the application of a rule is controlled by the information stored in the current nonterminal. In this situation, a rule $u \to B$ (a linear one!) from P_i, for $1 \leq i \leq 2$, is applicable if and only if the sentential form γ contains one occurrence of B, and no word from $LS(P_i)$ occurs in γ. The latter can

be tested with information (4). During the derivation simulation, the information (1), (3), (4), and (5) is updated. When terminating (simulating $\alpha \to A$) in addition it is checked (using information (2) and (3)) whether we actually simulate the component that starts the original derivation process.

2. The second case consists of two symmetric cases (see Figure 1). Assume $\beta \to B$ is the rule applied in the second step of the original derivation. In the first part of the derivation a linear derivation is done (see above). At some point, the grammar guesses to apply the rule (which is member of the component P_i, for $1 \leq i \leq 2$) that breaks the linear structure of the original derivation. This rule is *not* linear anymore, i.e., it looks like, e.g., $C \to uAvBw$ (the symmetric case $C \to uBvAw$ is similar). Then the derivation is continued as follows: apply the rule $C \to u\alpha vBw$ under the condition that no word of $LS(P_i)$ is stored as information (4), and then update information (4) and (5) according to $C \to uAvBw$. Thereafter, the derivation process is continued in a linear manner like in case 1. The only difference lies in the termination, because we already applied $\alpha \to A$. Therefore, we must test that $\beta \to B$ is member of the set P_j, for $1 \leq j \leq 2$, which does not start the original derivation and that no word from $\{\alpha\} \cup LS(P_j)$ is stored as information (4).

This completes our construction for $\mathcal{L}^{acc}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], (t \wedge = 1)) \subseteq \mathcal{L}^{gen}(\mathrm{LIN})$. The strictness of the inclusion follows from the lemma on one-letter languages. □

The question arises whether Theorem 4.4 generalizes to CDGS with more than two components and for derivation modes $(t \wedge \leq k)$ or $(t \wedge = k)$ in general. First, let us analyze the derivation trees obtained by such systems. We only discuss the $(t \wedge = k)$-mode, but it generalizes to the other mode as well.

Let G be a two component system working in $(t \wedge = 2)$-mode. Again, we must have an interplay between the two grammars to accept a word successfully. The number of possible tree structures, compared to the $(t \wedge = 1)$-case, increases significantly, but remains finite. Why this? The start component can introduce at most two nonterminals. Then, the application of the other grammar increases the number of occurrences of nonterminals again at most by two. Now, there are only three possibilities to continue the accepting derivation: either we reduce sub-words in a linear manner (sequential rule application), or we replace two nonterminals with some context in an application of a production set (parallel rule application), or we combine several nonterminals into one (union step). Moreover, the first and second step of the derivation can be only a sequential or a parallel one, but from then on, a sequence of parallel steps can only be followed by a sequence of sequential steps, and afterwards a union step. Finally, only a sequence of sequential steps mixed with a finite number of union steps can be performed until the axiom is reached. At this point, one observes that the number of nonterminals occurring in a sentential form is bounded. Obviously, with a similar construction as in the preceding $(t \wedge = 1)$ case, a programmed grammar with appearance checking can do the simulation job. Note that the whole derivation is of finite index, too.

With a much more detailed analysis also the case of CDGS with three grammars working in $(t \wedge = 1)$ mode can be done. In general, for arbitrary number n and k,

the number of possible structures for the derivation trees is bounded, so that a programmed grammar with appearance checking fulfilling the finite index restriction is able to simulate the original grammar system in a generating way. Thus, together with the lemma on one-letter languages and Theorem 4.1) we obtain:

Theorem 4.6 *If* $f \in \{ (t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N} \}$, *then, for any* $N \in \mathbf{N} \cup \{\infty\}$,

$$\mathcal{L}^{acc}(\mathrm{CD}_N, \mathrm{CF}[-\lambda], f) \subset \mathcal{L}^{gen}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], f) = \mathcal{L}^{gen}_{fin}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac}).$$

5 More on hybrid CDGS

As already observed in the previous section (see also [16]), accepting and generating modes coincide when only considering the $*$-, $\leq k$-, $= k$-, $\geq k$, and $(\geq k_1 \wedge \leq k_2)$-modes. We summarize these facts in the following theorem without proof.

Theorem 5.1 *If* $F \subseteq \{*\} \cup \{ \leq k, = k, \geq k \mid k \in \mathbf{N} \} \cup \{ (\geq k_1 \wedge \leq k_2) \mid k_1, k_2 \in \mathbf{N}, k_1 \leq k_2 \}$, *then, for any* $N \in \mathbf{N} \cup \{\infty\}$,

$$\mathcal{L}^{gen}(\mathrm{HCD}_N, \mathrm{CF}[-\lambda], F) = \mathcal{L}^{acc}(\mathrm{HCD}_N, \mathrm{CF}[-\lambda], F).$$

To exhibit the relations between other mode combinations, we have to explore their generating and accepting power in more detail. From [14], we summarize:

Theorem 5.2 *1. If* $\emptyset \neq F \subseteq \{*, t\} \cup \{ \leq k \mid k \in \mathbf{N} \} \cup \{= 1, \geq 1\}$, *then*

$$\mathcal{L}^{gen}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F \cup \{(t \wedge = 1)\}) = \mathcal{L}^{gen}(\mathrm{O}, \mathrm{CF}[-\lambda]).$$

2. If $\emptyset \neq F \subseteq \{ = k, \geq k \mid k \geq 2 \} \cup \{ (t \wedge \geq k) \mid k \geq 2 \}$, *then*

$$\mathcal{L}^{gen}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F \cup \{(t \wedge = 1)\}) = \mathcal{L}^{gen}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac}).$$

3. If $\emptyset \neq F \subseteq \{*, t\} \cup \{ \leq k, = k, \geq k, (t \wedge \geq k) \mid k \in \mathbf{N} \} \cup \{(\geq k_1 \wedge \leq k_2) \mid k_1, k_2 \in \mathbf{N}$ *and* $k_1 \leq k_2 \}$, *then*

$$\mathcal{L}^{gen}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F \cup \{(t \wedge = 2)\}) = \mathcal{L}^{gen}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac}).$$

4. Let $\emptyset \neq F \subseteq \{*, t\} \cup \{ \leq k \mid k \in \mathbf{N} \}$. *For every* $k \in \mathbf{N}$, $k \geq 2$,

$$\mathcal{L}^{gen}(\mathrm{O}, \mathrm{CF}[-\lambda]) \subseteq \mathcal{L}^{gen}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F \cup \{(t \wedge \leq k)\}) \subseteq \mathcal{L}^{gen}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ut}).$$

Since ordered languages are strictly included in programmed languages with unconditional transfer, at least one of the inclusions is strict.

Theorem 5.3 *Let* $F \subseteq D$ *contain one mode from* $\{*\} \cup \{ \leq k, = k, \geq k \mid k \in \mathbf{N} \} \cup \{ (\geq k_1 \wedge \leq k_2) \mid k_1, k_2 \in \mathbf{N}$ *and* $k_1 \leq k_2 \}$ *and one from* $\{ (t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N} \}$. *Then, we have:*

1. $\mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}-\lambda, F) = \mathcal{L}^{gen}(\mathrm{CS})$, *and*
2. $\mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}, F) = \mathcal{L}^{gen}(\mathrm{RE})$.

Proof. Our proof is very similar to the t-mode case shown in [16, Theorem 4.2]. We give only technical details here. First, we consider the λ-free case. It is easy to construct a simulating linear bounded automaton accepting $L^{acc}(G)$ in case not admitting λ-rules. Therefore, the inclusion $\mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F) \subseteq \mathcal{L}^{gen}(\mathrm{CS})$ is clear. We have to show the other inclusion.

By a standard argument, it can be shown that (*) $\mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F)$ is closed under union and embraces the context-free languages. Let $L \in \mathcal{L}^{gen}(\mathrm{CS})$, $L \subseteq T^*$. Then, $L = \bigcup_{a,b,c \in T}(\{a\}T^+\{bc\} \cap L) \cup (L \cap T) \cup (L \cap T^2) \cup (L \cap T^3)$. Since L is context-sensitive, $L_{abc} = \{ w \in T^+ \mid awbc \in L \}$ is context-sensitive due to the closure of $\mathcal{L}^{gen}(\mathrm{CS})$ under derivatives. By (*), it is sufficient to show that $\{a\}M\{bc\} \in \mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}-\lambda, F)$ provided that $M \subseteq T^+$ is context-sensitive.

By simple prolongation arguments, it suffices to show that $\{a\}M\{bc\}$ belongs to $\mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}-\lambda, \{*\} \cup \{(t \wedge = 1)\})$ provided that $M \subseteq T^+$ is context-sensitive.

Let $G = (N, T, S, P)$ be a context-sensitive grammar without λ-productions in Kuroda normal form generating M. Let us assume a unique label r being attached to any genuine context-sensitive rule of the form $XU \to YZ$ with $X, U, Y, Z \in N$; the set of labels is denoted by $\mathrm{Lab}_{\mathrm{cs}} = \{r_1, \ldots, r_R\}$.

We construct a $(\mathrm{HCD}_\infty, \mathrm{CF}-\lambda, \{*\} \cup \{(t \wedge = 1)\})$ system

$$G' = (N', T, S', P_0, P_{1,1}, P_{1,2}, P_{1,3}, P_{1,4} \ldots, P_{R,1}, P_{R,2}, P_{R,3}, P_{R,4})$$

accepting $\{a\}M\{bc\}$. The common terminal alphabet of these grammars is T, and their nonterminal alphabet is

$$N' \;=\; N \cup \{C_1, \ldots C_R\} \cup \{ D', D'' \mid D \in N \cup T \} \cup \{ A, B, C, S', F \}$$

(the unions being disjoint).

The component P_0 working in $*$-mode equals $\{ a \to A, b \to B, c \to C \} \cup \{ASBC \to S'\} \cup \{ w \to D \mid D \in N \text{ and } D \to w \in P, w \in T \} \cup \{ D \to D', D \to D'' \mid D \in N \}$ and is used for four purposes: (1) It turns the left delimiter a into A and the right delimiters b and c into B and C (initialization). (2) The check of the correctness of this initialization application is postponed until the last applicable production $ASBC \to S'$ does its work (termination). (3) Context-free rules can be simulated here. (4) Colouring of nonterminals into (double-)primed counterparts prepares the simulation of a genuine context-sensitive rule.

Finally, we introduce four production sets working in $(t \wedge = 1)$-mode for the simulation of a genuine context-sensitive production $r_\rho : XU \to YZ \in P$:

$$\begin{aligned}
P_{\rho,1} &= \{C \to C_\rho\} \cup \{ Y'z \to F \mid z \neq Z'' \} \cup \{ yZ'' \to F \mid y \neq Y' \}, \\
P_{\rho,2} &= \{C \to F\} \cup \{ C_\sigma \to F \mid \sigma \neq \rho \} \cup \\
&\quad \{Y' \to X\} \cup \{ D' \to F, E'' \to F \mid D, E \in N \wedge E'' \neq Z'' \}, \\
P_{\rho,3} &= \{C \to F\} \cup \{ C_\sigma \to F \mid \sigma \neq \rho \} \cup
\end{aligned}$$

$$\{Z'' \to U\} \cup \{D' \to F, D'' \to F \mid D \in N\}, \quad \text{and}$$
$$P_{\rho,4} = \{C_\rho \to C\} \cup \{C_\sigma \to F \mid \sigma \neq \rho\} \cup \{D' \to F, D'' \to F \mid D \in N\}.$$

Observe that the second production set only serves for checking whether the first production set has correctly nondeterministically selected two adjacent marked occurrences Y and Z. These checks are always possible, since we introduced left and right end-markers A and B, respectively.

If we allow λ-productions, these can be put in the P_0-component, too. □

Corollary 5.4 *Let $F \subseteq \{t\} \cup \{(t \wedge \leq k), (t \wedge = k), (t \wedge \geq k) \mid k \in \mathbf{N}\}$. If F contain one of the modes $\{t\} \cup \{(t \wedge \geq k) \mid k \in \mathbf{N}\}$ and one of the modes $\{(t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}\}$, then we have $\mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F) = \mathcal{L}^{gen}(\mathrm{CS})$.*

Proof. By our result [16, Theorem 4.2] regarding t-components only, it is only to prove that additional components working in one of the $\{(t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}\}$-modes do not enhance the accepting power. Again, λ-rules do not add to the power, and an easy simulation by a linear bounded automaton shows the assertion. □

When contrasting generating and accepting grammars, we obtain:

Theorem 5.5 *1. Let $F \subseteq D$ contain one of the modes $\{*, t\} \cup \{\leq k, = k, \geq k, (t \wedge \geq k) \mid k \in \mathbf{N}\} \cup \{(\geq k_1 \wedge \leq k_2) \mid k_1, k_2 \in \mathbf{N} \text{ and } k_1 \leq k_2\}$ and one of the modes $\{(t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}\}$. Then, we have:*

$$\mathcal{L}^{gen}(\mathrm{HCD}_\infty, \mathrm{CF} - \lambda, F) \subset \mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF} - \lambda, F).$$

2. Assume that $F \subseteq D$ contains one of the modes $\{*\} \cup \{\leq k, = k, \geq k \mid k \in \mathbf{N}\} \cup \{(\geq k_1 \wedge \leq k_2) \mid k_1, k_2 \in \mathbf{N},\ k_1 \leq k_2\}$ and one of the modes $\{(t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}\}$. Then, we have:

$$\mathcal{L}^{gen}(\mathrm{HCD}_\infty, \mathrm{CF}, F) \subseteq \mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}, F).$$

(a) This inclusion is known to be strict in case F contains none of the modes $\{(t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}, k \geq 2\}$.

(b) This inclusion is known to be non-strict in case F contains one of the modes $\{(t \wedge = k) \mid k \in \mathbf{N}, k \geq 2\}$.

3. Let $F \subseteq \{t\} \cup \{(t \wedge \geq k), (t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}\}$ and let F contain one of the modes $\{t\} \cup \{(t \wedge \geq k) \mid k \in \mathbf{N}\}$ and one of the modes $\{(t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}\}$.

(a) If $F \cap (\{(t \wedge = k) \mid k \in \mathbf{N}, k \geq 2\} \cup \{(t \wedge \geq k) \mid k \in \mathbf{N}, k \geq 2\}) \neq \emptyset$, we have

$$\mathcal{L}^{gen}(\mathrm{CS}) = \mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}, F) \subset \mathcal{L}^{gen}(\mathrm{HCD}_\infty, \mathrm{CF}, F) = \mathcal{L}^{gen}(\mathrm{RE}).$$

(b) Otherwise, $\mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}, F)$ *is not contained in* $\mathcal{L}^{gen}(\mathrm{HCD}_\infty, \mathrm{CF}, F)$, *since* $\mathcal{L}^{gen}(\mathrm{CS})$ *is not contained in* $\mathcal{L}^{gen}(\mathrm{O}, \mathrm{CF})$. □

Theorem 5.6 *Let* $F \subseteq D$ *contain one of the modes* $\{*, t\} \cup \{\leq k, = k, \geq k \mid k \in \mathbf{N}\} \cup \{(\geq k_1 \wedge \leq k_2) \mid k_1, k_2 \in \mathbf{N} \text{ and } k_1 \leq k_2\} \cup \{(t \wedge \geq k) \mid k \in \mathbf{N}\}$ *and one of the modes* $\{(t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}\}$. *Then, we have:*

$$\mathcal{L}^{acc}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F) = \mathcal{L}^{acc}(\mathrm{HCD}_2, \mathrm{CF}[-\lambda], F).$$

Proof. We can distinguish two cases:

1. If $F \cap (\{t\} \cup \{(t \wedge \geq k) \mid k \in \mathbf{N}\}) \neq \emptyset$, then [16, Theorem 4.5] is applied.
2. If $F \cap (\{*\} \cup \{\leq k, = k, \geq k \mid k \in \mathbf{N}\} \cup \{(\geq k_1 \wedge \leq k_2) \mid k_1, k_2 \in \mathbf{N} \wedge k_1 \leq k_2\}) \neq \emptyset$, then following idea is helpful: First, due to our last theorem, it is sufficient to consider $*$- and $(t \wedge = 1)$-components, because every context-sensitive (or even recursively enumerable, if λ-rules are admitted) language can be accepted in such a way. Then, possibly a prolongation argument is applied. It is possible to colour each symbol of the originally given hybrid CDGS with a special colour indicating the $(t \wedge = 1)$-component we are going to apply next. Furthermore, we assume only coloured versions of terminals appearing in the sentential form. All original $(t \wedge = 1)$-components are put together, where each set of productions only works on its private alphabet. In addition, mixtures are excluded introducing productions of the form $XY \to F$ (where X and Y are from different colours or terminal symbols). □

Naturally, Theorem 5.6 cannot be improved, since (hybrid) CDGS with context-free rules having one component can accept at most the context-free languages. Just as an aside, we remark in this place:

Theorem 5.7 *For every* $k \in \mathbf{N}$,

$$\begin{aligned}\mathcal{L}(\mathrm{FIN}) &= \mathcal{L}^{acc}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge = k)) = \mathcal{L}^{gen}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge = k)) \\ &= \mathcal{L}^{acc}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \leq k)) = \mathcal{L}^{gen}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \leq k)).\end{aligned}$$ □

Theorem 5.6 contrasts sharply with our results in the generating case. Unfortunately, many points are still open here. Therefore, we only quote two preliminary results which prove that accepting hybrid systems are much more powerful than their generating counterparts in many cases.

Theorem 5.8 *1. If* $f \in \{(t \wedge \leq 1), (t \wedge = 1)\}$, *then, for* $n \in \{1, 2\}$, *we have*

$$\mathcal{L}^{gen}(\mathrm{HCD}_n, \mathrm{CF}[-\lambda], \{*, t\} \cup \{\leq k \mid k \in \mathbf{N}\} \cup \{f\}) = \mathcal{L}^{gen}(\mathrm{CF}).$$

2. Let $\emptyset \neq F \subseteq \{t\} \cup \{(t \wedge \geq k) \mid k \in \mathbf{N}\}$. *For every* $f \in \{(t \wedge \leq k), (t \wedge = k) \mid k \in \mathbf{N}\}$, *we have*

$$\mathcal{L}^{gen}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], F \cup \{f\}) = \mathcal{L}^{gen}(\mathrm{HCD}_4, \mathrm{CF}[-\lambda], F \cup \{f\}).$$

Especially, observe that

$$\begin{aligned}\mathcal{L}^{gen}(\mathrm{CF}) &= \mathcal{L}^{gen}(\mathrm{HCD}_2, \mathrm{CF}[-\lambda], \{*\} \cup \{(t \wedge = 1)\}) \\ &\subset \mathcal{L}^{acc}(\mathrm{HCD}_2, \mathrm{CF}[-\lambda], \{*\} \cup \{(t \wedge = 1)\}) = \mathcal{L}^{gen}(\mathrm{RE})\end{aligned}$$

is an amazing jump from the generating to the accepting power.

Moreover, it is always interesting to see for what hybridization is really good. In this respect, we want to contrast $\mathcal{L}^{gen}(\mathrm{CF}) = \mathcal{L}^{acc}(\mathrm{CD}_2, \mathrm{CF}, *)$ and $\mathcal{L}^{gen}(\mathrm{LIN}) = \mathcal{L}^{acc}(\mathrm{CD}_2, \mathrm{CF}, (t \wedge = 1))$ with $\mathcal{L}^{gen}(\mathrm{RE}) = \mathcal{L}^{acc}(\mathrm{HCD}_2, \mathrm{CF}, \{*\} \cup \{(t \wedge = 1)\})$.

6 (Prescribed) Teams as acceptors

A *cooperating distributed grammar system with prescribed teams* (PTCDGS for short), confer [17, 18, 19, 26], is a construct $G = (N, T, S, P_1, \ldots, P_n, Q_1, \ldots, Q_m)$, with $n, m \in \mathbf{N}$, where $(N, T, S, P_1, \ldots, P_n)$ is a usual CDGS and $Q_1, \ldots, Q_m$ are *teams*, i.e., subsets of $\{P_1, \ldots, P_n\}$. If each subset of $\{P_1, \ldots, P_n\}$ can be a team, then we say that G has *free teams*, G is called a *cooperating distributed grammar system with teams* (TCDGS for short).

For $x, y \in (N \cup T)^*$, we write $x \Rightarrow_{Q_i} y$ for some team $Q_i = \{P_{j_1}, \ldots, P_{j_s}\}$ if and only if $x = x_1 A_1 x_2 A_2 \ldots x_s A_s x_{s+1}$, $y = x_1 y_1 x_2 y_2 \ldots x_s y_s x_{s+1}$, where $x_\ell \in (N \cup T)^*$, $1 \leq \ell \leq s+1$, $A_r \to y_r \in P_{j_r}$, $1 \leq r \leq s$. Having defined the one-step derivation, we can easily define derivations in Q_i of k steps, at most k steps or at least k steps, and of any number of steps, denoted again by $\Rightarrow_{Q_i}^{=k}$, $\Rightarrow_{Q_i}^{\leq k}$, $\Rightarrow_{Q_i}^{\geq k}$, $\Rightarrow_{Q_i}^{*}$, respectively. For maximal derivations in a team Q_i, we can consider three variants:

1. $x \Rightarrow_{Q_i}^{t_0} y$ if and only if $x \Rightarrow_{Q_i}^{*} y$ and there is no z such that $y \Rightarrow_{Q_i} z$ [18].

2. $x \Rightarrow_{Q_i}^{t_1} y$ if and only if $x \Rightarrow_{Q_i}^{*} y$ and for no component $P_{j_r} \in Q_i$ and no z there is a derivation $y \Rightarrow_{P_{j_r}} z$ [19].

3. $x \Rightarrow_{Q_i}^{t_2} y$ if and only if $x \Rightarrow_{Q_i}^{*} y$ and there is a component $P_{j_r} \in Q_i$ such that for no z there is a derivation $y \Rightarrow_{P_{j_r}} z$ [26].

Given a (P)TCDGS G working in mode $f \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \in \mathbf{N}\}$, we define the derivation relation $x \Rightarrow y$ if and only if there exists a team Q_i such that $x \Rightarrow_{Q_i}^{f} y$. As usual, the language generated in f-mode by G is defined as

$$\begin{aligned}L_f^{gen}(G) := \{ w \in T^* \mid S \Rightarrow_{Q_{i_1}}^{f} \alpha_1 \Rightarrow_{Q_{i_2}}^{f} \ldots \Rightarrow_{Q_{i_{\ell-1}}}^{f} \alpha_{\ell-1} \Rightarrow_{Q_{i_\ell}}^{f} \alpha_\ell = w \text{ with} \\ \ell \geq 1, 1 \leq i_j \leq m, \text{ and } 1 \leq j \leq \ell \}.\end{aligned}$$

We denote by $\mathcal{L}^{gen}((\mathrm{P})\mathrm{TCD}, \mathrm{CF}[-\lambda], f)$ the family of languages generated by [λ-free] (P)TCDGS. Correspondingly, accepted languages and language classes are defined. We summarize the known results on generating (P)TCDGS.

Theorem 6.1 *1. For all $f \in \{*\} \cup \{\leq k, = k, \geq k \mid k \in \mathbf{N}\}$,*

$$\mathcal{L}^{gen}(\mathrm{PTCD}, \mathrm{CF}[-\lambda], f) = \mathcal{L}^{gen}(\mathrm{P}, \mathrm{CF}[-\lambda]).$$

2. *For all* $f \in \{t_0, t_1, t_2\}$,

$$\mathcal{L}^{gen}(\mathrm{TCD}, \mathrm{CF}[-\lambda], f) = \mathcal{L}^{gen}(\mathrm{PTCD}, \mathrm{CF}[-\lambda], f) = \mathcal{L}^{gen}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac}).$$

The strictness of the inclusion $\mathcal{L}^{gen}(\mathrm{TCD}, \mathrm{CF}[-\lambda], f) \subseteq \mathcal{L}^{gen}(\mathrm{P}, \mathrm{CF}[-\lambda])$, for $f \in \{*\} \cup \{\leq k, = k, \geq k \mid k \in \mathbf{N}\}$ is open. However, it is quite clear that the generating and accepting capacity of (P)TCDGS working in one of the modes $f \in \{*\} \cup \{\leq k, = k, \geq k \mid k \in \mathbf{N}\}$ coincides.

Theorem 6.2 *For all* $f \in \{*\} \cup \{\leq k, = k, \geq k \mid k \in \mathbf{N}\}$, *we have:*

1. $\mathcal{L}^{acc}(\mathrm{PTCD}, \mathrm{CF}[-\lambda], f) = \mathcal{L}^{gen}(\mathrm{PTCD}, \mathrm{CF}[-\lambda], f) = \mathcal{L}^{gen}(\mathrm{P}, \mathrm{CF}[-\lambda])$,
2. $\mathcal{L}^{acc}(\mathrm{TCD}, \mathrm{CF}[-\lambda], f) = \mathcal{L}^{gen}(\mathrm{TCD}, \mathrm{CF}[-\lambda], f)$. □

As regards the different t-modes, the observations sketched in the following allow us to carry over our result contained in [16, Theorems 4.2, and 4.5]:

1. If we have only one-element teams, all t-modes introduced for PTCDGS coincide (with the classical t-mode). Hence, such PTCDGS accept all context-sensitive languages.

2. The simulation also works when permitting larger (arbitrary) teams, since it is possible to use different colours in the simulation of different genuine context-sensitive rules (simulating a context-sensitive grammar in Kuroda normal form). "Wrong" colours are alway sent to the failure symbol. Possibly, if we choose teams containing more than one component, two or more rules of the originally given context-sensitive grammar are simulated in parallel, but this does no harm, since a sequentialization choosing singleton teams is always possible. Further observe that a simulation of a genuine context-sensitive rule by a t-mode component as given in [16, Theorems 4.2, and 4.5] always takes the same number of steps, so that no garbage can be derived employing the t_2-mode. Hence, such TCDGS can also accept all context-sensitive languages.

3. As regards λ-rules, they are simply useless in case of classical CDGS working in t-mode, since such a rule is always applicable. A similar argument is applied to (P)TCDGS working in t_1-mode. The situation is different for CDGS working in t_0-mode or in t_2-mode. Why? First, we can assume that the type-0-grammar we are going to simulate has only one production of the form $E \rightarrow \lambda$, where E is a special nonterminal symbol serving as a place-holder for the empty word. Moreover, due to the closure properties of $\mathcal{L}^{gen}(\mathrm{RE})$, we can assume an additional left-marker symbol $\#$. Now, λ-productions can be simulated by three components, $P_{\lambda,1} = \{\# \rightarrow \#, \# \rightarrow \#'\}$, $P_{\lambda,2} = \{\#' \rightarrow \#\}$, and $P_{\lambda,3} = \{\lambda \rightarrow E\}$. When combining $P_{\lambda,1}$ and $P_{\lambda,3}$ into one team, arbitrarily many E's can be introduced. When using free teams, other combinations are now possible which may block a t-mode derivation prematurely. Therefore, this case remains as an open question.

We collect our observations in the following.

Theorem 6.3 *For each $f \in \{t_0, t_1, t_2\}$, we find in general*

1. $\mathcal{L}^{acc}(\text{PTCD}, \text{CF} - \lambda, f) = \mathcal{L}^{acc}(\text{TCD}, \text{CF} - \lambda, f) = \mathcal{L}^{gen}(\text{CS})$, *and*
2. $\mathcal{L}^{gen}(\text{CS}) \subseteq \mathcal{L}^{acc}(\text{TCD}, \text{CF}, f) \subseteq \mathcal{L}^{acc}(\text{PTCD}, \text{CF}, f) \subseteq \mathcal{L}^{gen}(\text{RE})$.

More specifically, we obtain by our third observation:

3. $\mathcal{L}^{acc}(\text{PTCD}, \text{CF}, t_1) = \mathcal{L}^{acc}(\text{TCD}, \text{CF}, t_1) = \mathcal{L}^{gen}(\text{CS})$, *and*
4. $\mathcal{L}^{acc}(\text{PTCD}, \text{CF}, t_0) = \mathcal{L}^{acc}(\text{PTCD}, \text{CF}, t_2) = \mathcal{L}^{gen}(\text{RE})$. □

Comparing the generating versus the accepting capacity, we get:

Corollary 6.4 *For each $f \in \{t_0, t_1, t_2\}$, we find*

1. $\mathcal{L}^{gen}((\text{P})\text{TCD}, \text{CF} - \lambda, f) \subset \mathcal{L}^{acc}((\text{P})\text{TCD}, \text{CF} - \lambda, f)$;
2. $\mathcal{L}^{acc}((\text{P})\text{TCD}, \text{CF}, f) \subseteq \mathcal{L}^{gen}((\text{P})\text{TCD}, \text{CF}, f)$;
3. $\mathcal{L}^{acc}((\text{P})\text{TCD}, \text{CF}, t_1) \subset \mathcal{L}^{gen}((\text{P})\text{TCD}, \text{CF}, t_1)$;
4. $\mathcal{L}^{acc}(\text{PTCD}, \text{CF}, g) = \mathcal{L}^{gen}(\text{PTCD}, \text{CF}, g)$, *for $g \in \{t_0, t_2\}$*. □

7 Conclusions

We continued our studies on accepting systems of grammars, paying special attention towards internally hybrid modes and teams. In this way, we also found first examples of grammar mechanisms whose generating power is greater than its accepting power.

In [2], two variants of the t-mode, namely weak t and stagnation, have been introduced: In weak t-mode a component P_j works on a string up to the point a sentential form w is obtained with $w \Rightarrow_{P_j} v$ implies $w = v$. This corresponds to the adult mechanism known from the theory of Lindenmayer systems. Now, another component may start its work with w.

The stagnation-mode is defined as follows: a component P_j works on a string deriving subsequently $w_1 \Rightarrow_{P_j} w_2 \Rightarrow_{P_j} w_3 \Rightarrow_{P_j} \cdots \Rightarrow_{P_j} w_n$, and $w_n \Rightarrow_{P_j} v$ implies $w_i = v$ for some $1 \leq i \leq n$. Now, another component may start its work with w_n.

Analyzing the proof of [16, Theorem 4.2], we see that both variants also characterize the context-sensitive languages when seen as language acceptors. These results on the accepting capacity of these modes have been independently obtained from [2].

References

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

[2] H. Bordihn and E. Csuhaj-Varjú. On Competence and Completeness in CD Grammar Systems. In this volume.

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

[4] H. Bordihn and H. Fernau. Accepting programmed grammars without nonterminals. In *5. GI Theorietag "Automaten und Formale Sprachen", Technical Report 9503, Universität Gießen, Arbeitsgruppe Informatik*, pages 4–16, 1995.

[5] H. Bordihn and H. Fernau. Accepting grammars and systems: an overview. In J. Dassow, G. Rozenberg, and A. Salomaa, editors, *Developments in Language Theory II; at the crossroads of mathematics, computer science and biology*, pages 199–208. Singapore: World Scientific, 1996.

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

[7] 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.

[8] E. Csuhaj-Varjú et al. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation.* London: Gordon and Breach, 1994.

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

[10] R. Engelmore and T. Morgan. *Blackboard Systems.* Addison-Wesley, 1988.

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

[12] H. Fernau and R. Freund. Accepting array grammars with control mechanisms. Unpublished manuscript, 1996.

[13] H. Fernau and R. Freund. Bounded parallelism in array grammars used for character recognition. In P. Perner, P. Wang, and A. Rosenfeld, editors, *Advances in Structural and Syntactical Pattern Recognition (Proceedings of the SSPR'96)*, volume 1121 of *LNCS*, pages 40–49. Berlin: Springer, 1996.

[14] H. Fernau, R. Freund, and M. Holzer. External versus internal hybridization for cooperating distributed grammar systems. Technical Report TR 185-2/FR-1/96, Technische Universität Wien (Austria), 1996.

[15] H. Fernau and M. Holzer. Bidirectional cooperating distributed grammar systems. Technical Report WSI-96-1, Universität Tübingen (Germany), Wilhelm-Schickard-Institut für Informatik, 1996.

[16] 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.

[17] R. Freund. Array grammars with prescribed teams of array productions. In *Developments in Language Theory II; at the crossroads of mathematics, computer science and biology*, pages 220–229. London: Gordon and Breach, 1996.

[18] R. Freund and Gh. Păun. A variant of team cooperation in grammar systems. *Journal of Universal Computer Science*, 1(2):105–130, 1995.

[19] L. Kari, A. Mateescu, Gh. Păun, and A. Salomaa. Teams in cooperating distributed grammar systems. *Journal of Experimental and Theoretical AI*, 7:347–359, 1995.

[20] 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. Berlin: Springer, 1978.

[21] R. Meersman, G. Rozenberg, and D. Vermeir. Persistent ET0L systems. *Information Sciences*, 18:189–212, 1979.

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

[23] N. J. Nilsson. *Principles of Artificial Intelligence*. Berlin: Springer, 1982.

[24] 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.

[25] Gh. Păun. Grammar systems: a grammatical approach to distribution and cooperation. In *Automata, Languages and Programming; 22nd International Colloquium, ICALP'95, Szeged, Hungary*, volume 944 of *LNCS*, pages 429–443. Berlin: Springer, 1995.

[26] Gh. Păun and G. Rozenberg. Prescribed teams of grammars. *Acta Informatica*, 31:525–537, 1994.

[27] E. L. Post. Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics*, 65:197–215, 1943.

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

[29] M. J. Wooldridge and N. R Jennings. Agent theories, architectures and languages: a survey. In M. J. Wooldridge and N. R. Jennings, editors, *Intelligent Agents; ECAI-94 Workshop on Agent Theories, Architectures, and Languages (Amsterdam 1994)*, volume 890 of *LNCS (LNAI)*, pages 1–39. Berlin: Springer, 1994.