ELSEVIER

Theoretical Computer Science 259 (2001) 405–426

Theoretical Computer Science

www.elsevier.com/locate/tcs

Hybrid modes in cooperating distributed grammar systems: internal versus external hybridization

H. Fernau[a, 1], M. Holzer[a], R. Freund[b, *]

[a] *Wilhelm-Schickard Institut für Informatik, Universität Tübingen, Sand 13, D-72076 Tübingen, Germany*

[b] *Institut für Computersprachen, Technische Universität Wien, Karlsplatz 13, A-1040 Wien, Austria*

Received November 1998; revised August 1999
Communicated by A. Salomaa

Abstract

We introduce several internally hybrid derivation modes of cooperating distributed (CD) grammar systems. External hybridizations were investigated by Mitrana and Pă un: for example, some components of a CD grammar system, when enabled, have to work as long as possible – they work in the so-called t-mode –, and some others, when enabled, perform at least k derivation steps – this is the so-called $\geqslant k$-mode. On the other hand, in an internally hybrid grammar system combining the t- and $\geqslant k$-mode – we denote this combination by $(t \wedge \geqslant k)$ – each component, when enabled, has to work as long as possible, yet performing at least k derivation steps. In this paper, among other things, we show that such externally hybrid CD grammar systems with components working in the t-mode and the $\geqslant k$-mode, can be characterized by CD grammar systems with all components working in the $(t \wedge \geqslant k)$-mode, and these can be characterized by recurrent programmed grammars with appearance checking, or, as well, by ET0L systems with permitting random context.

Keywords: Grammar systems; Hybrid modes; Recurrent programmed grammers

1. Introduction

Cooperating distributed (CD) grammar systems first were introduced in [10] with motivations related to two-level grammars. Later, the investigation of CD grammar systems

* Corresponding author.
E-mail addresses: fernau@informatik.uni-tuebingen.de (H. Fernau), holzer@informatik.uni-tuebingen.de (M. Holzer), rudi@logic.at (R. Freund).

[1] Supported by Deutsche Forschungsgemeinschaft grant DFG La 618/3-1/2 "Komplexitätstheoretische Methoden für die adäquate Modellierung paralleler Berechnungen".

PII: S0304-3975(00)00022-0

became a vivid area of research after relating CD grammar systems with artificial intelligence (AI) notions [1], such as multi-agent systems or blackboard models for problem solving [12]. From this point of view, motivations for CD grammar systems can be summarized as follows: several grammars (agents or experts in the framework of AI), mainly consisting of rule sets (corresponding to scripts the agents have to obey to) are cooperating in order to work on a sentential form (representing their common work), finally generating terminal words (in this way solving the problem). The picture one has in mind is that of several grammars (mostly, these are simply classical context-free grammars called "components" in the theory of CD grammar systems) "sitting" around a table where there is lying the common workpiece, a sentential form. Some component takes this sentential form, works on it, i.e., it performs some derivation steps, and then returns it onto the table such that another component may continue the work.

Of course, there are several ways to formalize this collaboration. In particular, "how long" is a component allowed to work on a sentential form until maybe another component can contribute to this work? In other words, how is the agent reading its script? The following modes have thoroughly been investigated in the literature:

- $\Rightarrow^{\leqslant k}$: when enabled, the component has to perform at most k derivation steps.
- $\Rightarrow^{=k}$: when enabled, the component has to perform exactly k derivation steps.
- $\Rightarrow^{\geqslant k}$: when enabled, the component has to perform at least k derivation steps.
- $\Rightarrow^{*}$: when enabled, the component has to perform arbitrarily many derivation steps.
- $\Rightarrow^{t}$: when enabled, the component has to perform as many derivation steps as possible.

In CD grammar systems all components work according to the same mode. It is of course natural to alleviate this requirement, because it simply refers to different capabilities and working regulations of different experts in the original CD motivation. This leads to the notion of so-called hybrid CD grammar systems introduced by Mitrana and Păun [11, 13]. We introduce hybrid derivation modes which (partly) nicely characterize the external hybridizations explained above. This paper belongs to a series of papers on hybrid modes in CD grammar systems: [4] introduces hybrid modes in CD array grammar systems as a natural specification tool for array languages, [6] investigates accepting CD grammar systems with hybrid modes, while [7] stresses descriptional complexity issues.

We are now going to explain how we obtain these hybrid modes. The classical modes are defined in such a way that a derivation has to fulfill only *one* property, e.g., in the $\leqslant k$-mode at most k steps have to be performed. Skipping this restriction for the derivation allows us to build arbitrary boolean combinations of classical modes. Here we stick to logical AND combinations of two modes, hence a derivation has to fulfill two properties in common – an appropriate formalization is given in the next section.

In this way, we obtain the following modes:

- $\Rightarrow^{(\geqslant k_1 \wedge \leqslant k_2)}$: when enabled, the component has to perform at least k_1 and at most k_2 derivation steps.
- $\Rightarrow^{(t \wedge \geqslant k)}$: when enabled, the component has to perform as many derivation steps as possible, and at least k steps.
- $\Rightarrow^{(t \wedge = k)}$: when enabled, the component has to perform as many derivation steps as possible, and exactly k steps.
- $\Rightarrow^{(t \wedge \leqslant k)}$: when enabled, the component has to perform as many derivation steps as possible, and at most k steps.

For $f \in \{*, t\} \cup \{\leqslant k, = k, \geqslant k, \mid k \in \mathbf{N}\}$, combinations $(* \wedge f)$ are only an alternative notation for the original mode f; therefore, we have not listed them. In this paper, we focus on the two modes $(\geqslant k_1 \wedge \leqslant k_2)$ and $(t \wedge \geqslant k)$, and we do not include results concerning the combinations of the t-mode with the $= k$-mode or the $\leqslant k$-mode; these results are contained in [5] and will appear in another paper.

We compare external and internal hybridization in CD grammar systems (combining the same classical basic modes), regarding their generative capacities. It is interesting on its own right to relate the basic modes with the hybrid modes they comprise as well as to consider external hybridizations involving our new internally hybrid modes. Finally, we compare CD language families with certain variants of programmed languages, thereby linking CD language families with families well known from regulated rewriting. This also raises new interest in old open questions in the latter area.

The paper is organized as follows. In Section 2, we introduce the necessary notions. In Section 3, we show some closure properties of recurrent programmed languages, which will play a central rôle in the main result of this paper. Then, we present general results and techniques for reasoning about hybrid modes and we briefly discuss the combination of the modes $\leqslant k$, $= k$, $\geqslant k$, and $*$, which turns out to be a simple case. In Section 5, we study the combination of the modes t and $\geqslant k$. Such hybrid systems (be they externally hybrid or internally hybrid) characterize the class of recurrent programmed languages with appearance checking. Apart from the introduction of internally hybrid modes, this characterization (Theorem 20) can be seen as the main contribution of this paper to the theory of CD grammar systems, since it allows us to link the question of Păun [13] whether it is possible to characterize the recursively enumerable languages by using externally hybrid CD grammar systems working with arbitrary combinations of the basic modes with the old open question whether recurrent programmed languages with appearance checking, or, equivalently, ET0L languages with permitting random context characterize the recursively enumerable languages or not. In the last section, we review our results again and give a prospect on (possible) future work.

2. Definitions

We assume the reader to be familiar with some basic notions of formal language theory, as contained in [3]. In general, we have the following conventions: $\subseteq$ denotes inclusion, while $\subset$ denotes strict inclusion; the set of positive integers is denoted by $\mathbf{N}$.

The empty word is denoted by λ; $|\alpha|_A$ denotes the number of occurrences of the symbol A in α. We consider two languages L_1, L_2 to be equal if and only if $L_1 \setminus \{\lambda\} = L_2 \setminus \{\lambda\}$, and we simply write $L_1 = L_2$ in this case.

The families of languages generated by regular, linear context-free, context-free, context-sensitive, arbitrary type-0 Chomsky grammars, and ET0L systems are denoted by $\mathscr{L}(\mathrm{REG})$, $\mathscr{L}(\mathrm{LIN})$, $\mathscr{L}(\mathrm{CF})$, $\mathscr{L}(\mathrm{CS})$, $\mathscr{L}(\mathrm{RE})$, and $\mathscr{L}(\mathrm{ET0L})$, respectively. We attach $-\lambda$ in our notations if erasing rules are not permitted.

A *one-input finite-state transducer with accepting states*, or a 1-*a-transducer* for short, is a sextuple $M = (Q, X, Y, \delta, q_0, Q_f)$, where Q is a finite set of states, X and Y are finite (input and output) alphabets, $q_0 \in Q$ is the initial state, $Q_f \subseteq Q$ is the set of accepting or final states, and δ is a finite subset of $Q \times X^* \times Q \times Y^*$. M is called *non-erasing,* if $\delta \subseteq Q \times X^* \times Q \times Y^+$; moreover, M is called *spelling,* if $\delta \subseteq Q \times (X \cup \{\lambda\}) \times Q \times (Y \cup \{\lambda\})$.

A word $h = h_1 \cdots h_n \in \delta^+$ is called a *computation* of the 1-a-transducer M if and only if

- $\mathrm{pr}_1(h_1) = q_0$, $\mathrm{pr}_3(h_n) \in Q_f$, and
- $\mathrm{pr}_1(h_{i+1}) = \mathrm{pr}_3(h_i)$ for all i with $1 \leqslant i \leqslant n-1$;

where pr_i are projections on δ^* defined by

$$\mathrm{pr}_i((x_1, x_2, x_3, x_4)) = x_i \quad \text{for } i \in \{1,2,3,4\} \quad \text{and} \quad (x_1, x_2, x_3, x_4) \in \delta.$$

The set of all computations of M is denoted by $C(M)$. The *1-a-transducer mapping* induced by M is defined by

$$M(L) = \mathrm{pr}_4(\mathrm{pr}_2^{-1}(L) \cap C(M))$$

for each language $L \subseteq X^*$.

A *k-restricted erasing* is a 1-a-transducer mapping τ which realizes the morphism $g_T : (T \cup \{\$\})^* \to T^*$ (where $\$ \notin T$), given by $a \mapsto a$ for $a \in T$ and $\$ \mapsto \lambda$, on the domain

$$\mathrm{dom}(\tau) = \left(\bigcup_{i=0}^{k} \{\$^i\} T \right)^+ .$$

Remark 1. It is easy to see that every 1-a-transducer can be realized by a spelling 1-a-transducer. Moreover, observe that every k -restricted erasing can be realized by a non-erasing 1-a-transducer.

A *programmed grammar* is a septuple $G = (N, T, P, S, \Lambda, \sigma, \phi)$, where N, T, and $S \in N$ are the set of nonterminals, the set of terminals, and the start symbol, respectively. In the following we use V_G to denote the set $N \cup T$. P is the finite set of context-free rules $A \to z$ with $A \in N$ and $z \in V_G^*$, and Λ is a finite set of labels (for the rules in P), such that Λ can also be interpreted as a function which outputs a rule when being given a label; σ and ϕ are functions from Λ into the set of subsets of Λ. For (x, r_1), (y, r_2) in $V_G^* \times \Lambda$ and $\Lambda(r_1) = (A \to z)$, we write $(x, r_1) \Rightarrow (y, r_2)$ if and only if either

(1) $x = x_1Ax_2$, $y = x_1zx_2$, and $r_2 \in \sigma(r_1)$, or
(2) $x = y$, the rule $A \to z$ is not applicable to x, and $r_2 \in \phi(r_1)$.
In the latter case, the derivation step is performed in the so-called *appearance checking mode*. The set $\sigma(r_1)$ is called success field and the set $\phi(r_1)$ is called failure field of r_1. As usual, the reflexive transitive closure of $\Rightarrow$ is denoted by $\overset{*}{\Rightarrow}$. The language generated by G is defined as $L(G) = \{w \in T^* \mid (S, r_1) \to (w, r_2)$ for some $r_1, r_2 \in \Lambda\}$. The family of languages generated by programmed grammars containing only context-free core rules is denoted by $\mathscr{L}(\mathrm{P}, \mathrm{CF}, \mathrm{ac})$. When no appearance checking features are involved, i.e., $\phi(r) = \emptyset$ for each label $r \in \Lambda$, we obtain the family $\mathscr{L}(\mathrm{P}, \mathrm{CF})$. Von Solms [14] considered *recurrent context-free programmed grammars*: A (context-free) programmed grammar G is called a *recurrent (context-free) programmed grammar* if, for every $p \in \Lambda$ of G, we have $p \in \sigma(p)$ and, moreover, $\phi(p) = \sigma(p)$ if $\phi(p) \neq \emptyset$. The corresponding language family is denoted by $\mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac})$ and, when no appearance checking features are involved, by $\mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda])$. It is known from [14] that $\mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac})$ equals the class of [propagating] ET0L languages with permitting random context, since the proof given there works also in the λ-free case. Note that we use bracket notations in order to express that the equation holds both in case of forbidding erasing rules and in the case of admitting erasing rules (consistently neglecting the contents between the brackets).

A *CD grammar system* of degree n, with $n \geqslant 1$, is an $(n+3)$ -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 start symbol, and $P_1, \ldots, P_n$ are finite sets of rewriting rules over $N \cup T$. Throughout this paper, we consider only regular, linear context-free, and context-free rewriting rules. For $x, y \in (N \cup T)^*$ and $1 \leqslant i \leqslant n$, we write $x \Rightarrow_i y$ if and only if $x = x_1Ax_2$, $y = x_1zx_2$ for some $A \to z \in P_i$. Hence, subscript i refers to the component to be used. Accordingly, $x \Rightarrow_i^m y$ denotes an m-step derivation using component number i, where $x \Rightarrow_i^0 y$ if and only if $x = y$.

Define the *classical basic modes* $B = \{*, t\} \cup \{\leqslant k, = k, \geqslant k \mid k \in \mathbf{N}\}$ and let $D = B \cup \{(\geqslant k \wedge \leqslant \ell) \mid k, \ell \in \mathbf{N}, k \leqslant \ell\} \cup \{(t \wedge \leqslant k), (t \wedge = k), (t \wedge \geqslant k) \mid k \in \mathbf{N}\}$. For $f \in D$ we define the relation $\Rightarrow_i^f$ by

$$x \Rightarrow_i^f y \Leftrightarrow \exists m \geqslant 0 : (x \Rightarrow_i^m y \wedge P(f, m, i, y)),$$

where P is a predicate defined as follows (let $k \in \mathbf{N}$ and $f_1, f_2 \in B$):

Predicate	Definition
$P(=k, m, i, y)$	$m = k$
$P(\leqslant k, m, i, y)$	$m \leqslant k$
$P(\geqslant k, m, i, y)$	$m \geqslant k$
$P(*, m, i, y)$	$m \geqslant 0$
$P(t, m, i, y)$	$\neg\exists z(y \Rightarrow_i z)$
$P((f_1 \wedge f_2), m, i, y)$	$P(f_1, m, i, y) \wedge P(f_2, m, i, y)$

Observe that not every combination of modes as introduced above introduces something really new. For example, the $(\geqslant k \wedge \leqslant k)$-mode is just another notation for the $=k$-mode. Especially, $*$ may be used as a "don't care" in our subsequent notations, since $P((* \wedge f_2), m, i, y)$ if and only if $P(f_2, m, i, y)$.

If each component of a CD grammar system may work in a different mode, then we get the notion of an (*externally*) *hybrid CD* (*HCD*) *grammar system* of degree n, with $n \geqslant 1$, which is an $(n+3)$-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 a CD grammar system, and $f_i \in D$, for $1 \leqslant i \leqslant n$. Thus, we can define the language *generated* by a HCD grammar system as:

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

If $F \subseteq D$ and $X \in \{\mathrm{REG}, \mathrm{LIN}, \mathrm{CF}\}$, then the family of languages generated by [λ-free] HCD grammar systems with degree at most n using rules of type X, each component working in one of the modes contained in F, is denoted by $\mathscr{L}(\mathrm{HCD}_n, X[-\lambda], F)$. In a similar way, we write $\mathscr{L}(\mathrm{HCD}_\infty, X[-\lambda], F)$ when the number of components is not restricted. If F is a singleton $\{f\}$, we simply write $\mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], f)$, where $n \in \mathbf{N} \cup \{\infty\}$; additionally, we write $L_f(G)$ instead of $L(G)$ to denote the language generated by the CD grammar system G in the mode f.

In order to clarify our definitions, we give a short example:

Example 2. Let $G = (N, T, S, P_1, P_2)$ be a CD grammar system with

$$N = \{S, A, B, A', B'\},$$
$$T = \{a, b, c\},$$
$$P_1 = \{S \to S, S \to AB, A' \to A, B' \to B\} \quad \text{and}$$
$$P_2 = \{A \to aA'b, B \to cB', A \to ab, B \to c\}.$$

The reader may verify that we have

$$L_{f_1}(G) = \{a^n b^n c^m \mid n, m \geqslant 1\},$$
$$L_{f_2}(G) = \{a^n b^n c^n \mid n \geqslant 1\},$$
$$L_{f_3}(G) = \emptyset,$$

where

$$f_1 \in \{*, t\} \cup \{=1, \geqslant 1\} \cup \{\leqslant k \mid k \geqslant 1\} \cup \{(\geqslant 1 \wedge \leqslant k) \mid k \geqslant 1\} \cup \{(t \wedge \geqslant 1)\} \cup \{(t \wedge \leqslant k) \mid k \geqslant 2\},$$
$$f_2 \in \{=2, \geqslant 2\} \cup \{(\geqslant 2 \wedge \leqslant k \mid k \geqslant 2\} \cup \{(t \wedge =2), (t \wedge \geqslant 2)\}$$

and

$$f_3 \in \{=k, \geqslant k \mid k \geqslant 3\} \cup \{(\geqslant k \wedge \leqslant \ell) \mid 3 \leqslant k \leqslant \ell\}$$
$$\cup \{(t \wedge =1), (t \wedge \leqslant 1)\} \cup \{(t \wedge =k), (t \wedge \geqslant k) \mid k \geqslant 3\}.$$

As regards externally hybrid CD grammar systems based on G – we now abbreviate $(N, T, S, (P_1, f), (P_2, g))$ by $G_{f,g}$ – for the cases not already covered by the considerations on G as a CD grammar system we obtain

$$L(G_{f_1,f_2}) = L(G_{f_2,f_1}) = \{a^n b^n c^n \mid n \geqslant 1\},$$
$$L(G_{f_3,f_1}) = L(G_{f_3,f_2}) = \{abc\},$$
$$L(G_{f_1,f_3}) = L(G_{f_2,f_3}) = \emptyset,$$

where f_1, f_2, f_3 have the same meaning as above.

Let us finally mention that regular and linear components working in one of the modes introduced above can only generate regular or linear languages, respectively, since all the necessary informations can be put into the only non-terminal symbol (an example for such a construction is worked out in [11, Theorem 2]). This observation would also be true if we required normal forms of regular grammars, e.g., using only rules of the forms $A \to aB$, $A \to a$, where A, B are non-terminal symbols and a is a terminal symbol, as it is commonly done in the Russian literature. The only difference would be that some of the following lemmas (formulated for context-free grammars, but also valid for arbitrary regular grammars) would not be true in the case of not admitting unit rules, i.e., rules of the form $A \to B$.

3. Closure properties of recurrent programmed languages

Closure properties of recurrent programmed languages have not yet been studied in the literature besides some variant in [8]. Since this language class plays a central rôle in this paper, we will supply such a study in the following.

Recall that a language family is called a *trio* if it is closed under non-erasing homomorphism, inverse homomorphism, and intersection with regular sets. By the theorem of Nivat, a family of languages is a trio if and only if it is closed under non-erasing 1-a-transducer mappings.

Moreover, a language family is called a *full trio* if it is closed under homomorphism, inverse homomorphism, and intersection with regular sets. By the theorem of Nivat and Remark 1, a family of languages is a full trio if and only if it is closed under spelling 1-a-transducer mappings.

In [8, Theorem 3.4], it was shown that $\mathscr{L}(\mathrm{RP}, \mathrm{CF})$ is closed under spelling 1-a-transducer mappings, hence it forms a full trio. This construction can readily be transferred to the case when allowing appearance checking, so that immediately we may infer the following result.

Corollary 3. *$\mathscr{L}(\mathrm{RP}, \mathrm{CF}[, \mathrm{ac}])$ forms a full trio, and $\mathscr{L}(\mathrm{RP}, \mathrm{CF} - \lambda[, \mathrm{ac}])$ is closed under non-erasing spelling 1-a-transducer mappings.*

The operations "non-erasing homomorphism" and "intersection with regular sets" can easily be realized by non-erasing spelling 1-a-transducer mappings. The inverse morphism h^{-1} given by $h : \{a\}^* \to \{a\}^*$, $a \mapsto a^2$ is a non-erasing 1-a-transducer which is not spelling, so that we cannot restrict ourselves to spelling transducers when having to prove that a language family is a (non-full) trio.

The following observation is very helpful here:

Proposition 4. *The relation $\tau \subseteq X^* \times Y^*$ is a non-erasing 1-a-transducer mapping if and only if τ can be represented as the composition $\tau_2\tau_1$ of a non-erasing spelling 1-a-transducer mapping τ_1 and a restricted erasing τ_2.*

Proof. By definition, both τ_1 and τ_2 are non-erasing 1-a-transducer mappings. Since non-erasing 1-a-transducer mappings are closed under composition, the "only if"-part follows.

On the other hand, let δ be the rule set defining some non-erasing 1-a-transducer realizing τ. Let

$$k = \max\{|u| \mid (q, u, q', v) \in \delta\}.$$

Define the rule set of a 1-a-transducer realizing τ_1 as follows:

$$\delta' := \{(q, u, q', v\$^k) \mid (q, u, q', v) \in \delta\}.$$

By introducing intermediate states, it is easy to find a spelling non-erasing 1-a-transducer realizing τ_1 from δ'. Since τ is non-erasing, an output word of τ_1 cannot have more than k symbols \$ in a sequence, hence, we may obtain a k-restricted erasing τ_2 which erases every symbol \$ in order to arrive at the desired representation $\tau = \tau_2\tau_1$. □

Corollary 5. *A family of languages is a trio if and only if it is closed under restricted erasing and non-erasing spelling 1-a-transducer mappings.*

Lemma 6. *$\mathscr{L}(\mathrm{RP}, \mathrm{CF} - \lambda[, \mathrm{ac}])$ is closed under restricted erasings.*

Proof. We only sketch the main idea of the proof in the following: first, if $G = (N, T, P, S, \Lambda, \sigma, \phi)$ is a λ-free recurrent programmed grammar and τ is a k-restricted erasing, we may assume that $L(G) \subseteq \mathrm{dom}(\tau)$, since $\mathrm{dom}(\tau)$ is a regular set and $\mathscr{L}(\mathrm{RP}, \mathrm{CF} - \lambda[, \mathrm{ac}])$ is closed under intersection with regular sets, and we want to construct a grammar for $\tau(L(G))$.

Then, we can construct a new λ-free recurrent programmed grammar G' from G with total alphabet $\{[w] \mid w \in (N \cup T)^+ \wedge 1 \leqslant |w| \leqslant k+1\}$ such that $w \in L(G)$ if and only if $[w_1] \ldots [w_m] \in L(G')$, where $w = w_1 \ldots w_m$ and $1 \leqslant |w_i| \leqslant k+1$ for all i with $1 \leqslant i \leqslant m$.

Finally, $\tau(L(G)) = \tau'(L(G'))$, where τ' is the non-erasing (partial) homomorphism mapping each $[w]$ with $w \in T^+$, $|w| \leqslant k+1$, and $\tau(w) \neq \lambda$ to $\tau(w)$, which completes the proof that $\tau(L(G))$ can be realized by a λ-free recurrent programmed grammar. □

The preceding results immediately imply the following (thereby repairing a small gap in [8]):

Corollary 7. $\mathscr{L}(\mathrm{RP}, \mathrm{CF} - \lambda[, \mathrm{ac}])$ *forms a trio.*

By using standard constructions, it is easy to prove the following closure properties:

Lemma 8. *The language families* $\mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda][, \mathrm{ac}])$ *are closed under union and mirror image.*

In recurrent programmed grammars, it is possible to check the non-occurrence of non-terminal symbols by introducing trap symbols in rules that can be skipped in the appearance checking mode, hence, we may use standard constructions to show closure under Kleene plus (respectively Kleene star) and catenation for $\mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac})$, so that finally we get the following result:

Theorem 9. $\mathscr{L}(\mathrm{RP}, \mathrm{CF} - \lambda, \mathrm{ac})$ *is an abstract family of languages* (*AFL*), *while* $\mathscr{L}(\mathrm{RP}, \mathrm{CF}, \mathrm{ac})$ *is even a full AFL.*

4. General observations

In this section, we will mainly collect two general techniques that are very useful when dealing with hybrid systems, namely the prolongation technique and the least common multiple (lcm) technique, which both were also used by Mitrana [11].

Lemma 10 (Prolongation technique). *Let* $\gamma \in \{*, t\}$, $\Delta \in \{\leqslant, =, \geqslant\}$, *and* $k, \ell \in \mathbf{N}$ *such that k divides* ℓ. *Every context-free component working in the* $(\gamma \wedge \Delta k)$*-mode can be replaced by a context-free component working in the* $(\gamma \wedge \Delta \ell)$*-mode. The simulating component has* λ*-rules only if the simulated component has* λ*-rules.*

Proof. The case $\ell = k$ is trivial. Let k divide ℓ properly, i.e., $l = kd$ for some $d \geqslant 2$. Let $\{A_1, \ldots, A_m\}$ be the set of nonterminals occurring as left-hand sides of rules of the component under consideration. We introduce a number of new non-terminals, namely $\{(A_i, j) \mid 1 \leqslant i \leqslant m \wedge 1 \leqslant j < d\}$. Instead of a rule $A_i \to w$ in our original component, we introduce rules $A_i \to (A_i, 1)$, $(A_i, j) \to (A_i, j+1)$ for $1 \leqslant j < d-1$, and $(A_i, d-1) \to w$ in the new simulating component. □

Observe that in the construction above, it is possible that there remain "coloured" symbols (A_i, j) in the sentential form when leaving the simulating component. When

considering grammar systems, this is not bad as long as in addition we colour each component individually, which guarantees that derivations in different simulating components cannot interfer. When considering externally hybrid system, components working in the t-mode and hybrid variants thereof should block in the presence of such coloured symbols (e.g., by unit rules of the form $(A_i, j) \to (A_i, j)$).

Theorem 11. *Let* $\gamma \in \{*, t\}$, $\Delta \in \{\leqslant, =, , \geqslant\}$, *and* $n, k, \ell \in \mathbf{N}$ (*or* $n = \infty$) *such that* k *divides* ℓ. *Then we have*

$$\mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (\gamma \wedge \Delta k)) \subseteq \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (\gamma \wedge \Delta \ell)).$$

Observe that this theorem implies a prime number lattice structure on the families of languages $\mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (\gamma \wedge \Delta k))$ where $n \in \mathbf{N} \cup \{\infty\}$ is fixed and $k \in \mathbf{N}$ is varying.

The least common multiple technique (lcm technique) has already been used in [11, Lemma 3].

Lemma 12 (Lcm technique). *Let* $\gamma \in \{*, t\}$, $\Delta \in \{\geqslant, =, \leqslant\}$. *The context-free components* $P_1, \ldots, P_m$ *working in the modes* $(\gamma \wedge \Delta k_1), \ldots, (\gamma \wedge \Delta k_m)$, *respectively*, *can be replaced by m context-free components each working in the* $(\gamma \wedge \Delta \ell)$*-mode*, *where* $\ell = \mathrm{lcm}\{\, k_i \mid 1 \leqslant i \leqslant m \,\}$. *The simulating components have* λ*-rules only if the simulated components have* λ*-rules.*

Proof. By the prolongation technique, the component P_i working in the mode $(\gamma \wedge \Delta k_i)$ can be replaced by one component working in the mode $(\gamma \wedge \Delta \ell)$. □

Thus, we directly obtain the next theorem which shows that, under certain circumstances, external hybridization may be replaced by internal hybridization.

Theorem 13. *Let* $\gamma \in \{*, t\}$, $\Delta \in \{\geqslant, =, \leqslant\}$, *and* $\{k_1, \ldots, k_m\} \subseteq \mathbf{N}$. *Then, for* $n \in \mathbf{N} \cup \{\infty\}$, *we have*

$$\mathscr{L}(\mathrm{HCD}_n, \mathrm{CF}[-\lambda], \{(\gamma \wedge \Delta k_i) \mid 1 \leqslant i \leqslant m\}) \subseteq \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (\gamma \wedge \Delta \ell)),$$

where $\ell = \mathrm{lcm}\{\, k_i \mid 1 \leqslant i \leqslant m \,\}$.

Note that due to the lcm technique, we pay off with large values of ℓ when avoiding external hybridization. This last result immediately allows us to state the following result, which basically means that in many situations external hybridization of internally hybrid modes does not add to the generative power compared with the corresponding internal hybridization alone.

Corollary 14. *Let* $\gamma \in \{*, t\}$, $\Delta \in \{\leqslant, =, \geqslant\}$, *and* $n \in \mathbf{N} \cup \{\infty\}$. *Then we have*

$$\mathscr{L}(\mathrm{HCD}_n, \mathrm{CF}[-\lambda], \{\, (\gamma \wedge \Delta k) \mid k \in \mathbf{N} \,\}) = \bigcup_{\ell \in \mathbf{N}} \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (\gamma \wedge \Delta \ell)).$$

As a first simple application of our general results of the present section, we briefly study the internally hybrid mode $(\geqslant k_1 \wedge \leqslant k_2)$, the so-called *interval mode*. In the next section, we will study the mode $(t \wedge \geqslant k)$ in more detail. As regards the modes $(t \wedge \leqslant k)$ and $(t \wedge = k)$, we refer the interested reader to our report [5], which is available from the authors on request. Finally, observe that the possible logical combination $(= k_1 \wedge \leqslant k_2)$ is only meaningful when $k_2 \geqslant k_1$, and then it coincides with the old $= k_1$-mode. A similar comment is valid for the $(\geqslant k_1 \wedge = k_2)$-mode.

Obviously, one context-free component working in the mode $\leqslant k$, $= k$, $\geqslant k$, or $*$ corresponds to the same component working in the mode $(\geqslant 1 \wedge \leqslant k)$, $(\geqslant k \wedge \leqslant k)$, $(\geqslant k \wedge \leqslant 2k-1)$, or $(\geqslant 1 \wedge \leqslant 1)$, respectively. On the other hand, one context-free component working in the $(\geqslant k_1 \wedge \leqslant k_2)$-mode can be simulated by $k_2 - k_1 + 1$ copies of this context-free component working in the $= k$-mode for some k with $k_1 \leqslant k \leqslant k_2$, each of which in turn can be simulated by a context-free component working in the $= \ell$-mode, where $\ell = \mathrm{lcm}\{\, k \mid k_1 \leqslant k \leqslant k_2 \,\}$ according to the lcm technique. Combining all these observations with [11, Lemma 3] we get:

Lemma 15.

$$\begin{aligned}
&\mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], \{(\geqslant k_1 \wedge \leqslant k_2) \mid k_1, k_2 \in \mathbf{N}, k_1 \leqslant k_2\}) \\
&\quad = \mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], \{\leqslant k, = k, \geqslant k \mid k \in \mathbf{N}\}) \\
&\quad = \bigcup_{\substack{k_1, k_2 \in \mathbf{N} \\ k_1 \leqslant k_2}} \mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (\geqslant k_1 \wedge \leqslant k_2)) \\
&\quad = \bigcup_{\ell \in \mathbf{N}} \mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], = \ell).
\end{aligned}$$

The reader may have noticed that we did not formulate our last results caring about the number of components. Indeed, we do not know whether such more precise statements hold or not.

Observe that via the interval mode certain properties of languages can be expressed quite naturally. Recently, this has been shown in the case of array grammars specifying the allowed lengths of lines in array patterns representing a character [4].

5. Combining t and $\geqslant k$

This section is structured as follows: first we show that external and internal hybridization also coincide when combining t-mode and $\geqslant k$-mode, employing more involved simulations. Then we prove that the corresponding language family coincides with the class of languages definable by ET0L systems with permitting random context or, equivalently, by recurrent programmed grammars (with appearance checking), see [3, 14]. It is an old question whether this language family coincides with that one definable by programmed grammars with appearance checking or not. Since the family $\mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], B)$ (using the classical modes B only) therefore includes the family

of ET0L languages with permitting random context and, at the same time, is a subfamily of $\mathscr{L}(\mathrm{P},\mathrm{CF}[-\lambda],\mathrm{ac})$, the question posed by Păun [13] whether $\mathscr{L}(\mathrm{HCD}_\infty,\mathrm{CF}[-\lambda],B)$ coincides with $\mathscr{L}(\mathrm{P},\mathrm{CF}[-\lambda],\mathrm{ac})$ or not is a tough one, indeed. Finally, we consider the number of components in some detail, also comparing the new internally hybrid mode $(t\wedge\geqslant k)$ with the "parent modes" t and $\geqslant k$.

Since the modes $(t\wedge\geqslant 1)$ and t are trivially equivalent, as a simple application of the prolongation technique (cf. Lemma 10), we obtain:

Lemma 16. *For each $k\in\mathbf{N}$, every context-free component working in the t-mode can be simulated by a context-free component working in the $(t\wedge\geqslant k)$-mode. The simulating component has λ-rules only if the simulated component has λ-rules.*

The simulation of the second parent mode via the combined (hybrid) mode is more involved:

Lemma 17. *For each $k\in\mathbf{N}$, every context-free component working in the $\geqslant k$-mode can be simulated by four context-free components working in the $(t\wedge\geqslant k)$-mode. The simulating components have λ-rules only if the simulated component has λ-rules.*

Proof. Let P be a context-free component working in the $\geqslant k$-mode with terminal alphabet T and non-terminal alphabet N. We construct four context-free components $P^{(i)}$ with non-terminal alphabet N' working in the $(t\wedge\geqslant k)$-mode which serve for the same task. Let $V=N\cup T$ and set

$$N':=N\cup((V\cup\{L\})\times\{-2,-1,0,1,\ldots,k+1\})\cup\{F\},$$

where F is a trap symbol and L is a non-terminal representing λ. Define $h:(V\times\{-1,0,1,\ldots,k\})\cup T\to V\cup T$ to be the morphism given by $(A,j)\mapsto A$ for $A\in N$ and $a\mapsto a$ for $a\in T$, and consider the following sets of rules:

In order to initialize the simulation of P, all non-terminals A are converted to their variants $(A,0)$ or $(A,-1)$, using unit rules as a very simple prolongation technique:

$$P^{(1)}:=\{A\to A,A\to(A,0),A\to(A,-1)\mid A\in N\}\cup\{X\to F\mid X\in N'\backslash(N\times\{-1,0\})\}.$$

The next component

$$\begin{aligned}P^{(2)}:=\{&(A,0)\to w'\mid A\Rightarrow^i w \text{ in } P,\ 1\leqslant i\leqslant k,\ h(w')=w,\\ &\text{and } w'\in W^*(V\times\{i\})W^*,\\ &\text{where } W=(N\times\{-1,0\})\cup T\}\\ \cup\{&(A,0)\to(L,i)\mid A\Rightarrow^i\lambda \text{ in } P \text{ and } 1\leqslant i\leqslant k\}\\ \cup\{&(A,0)\to(A,0)\mid A\in N\}\end{aligned}$$

$$\cup \{ (A,k) \to w' \mid A \to w \in P,\ h(w') = w, \text{ and } w' \in (V \times \{k\})^+ \}$$
$$\cup \{ (A,k) \to (L,k) \mid A \to \lambda \in P \}$$
$$\cup \{ (A,k) \to (A,k+1) \mid A \in N \cup \{L\} \}$$

hopefully simulates at least k derivation steps of P, which then is tested by the third component. The number of simulated derivation steps is stored in the second components i of the non-terminals of the form (B,i) with $B \in V \cup \{L\}$. The variant $(B,-1)$ of B means that we choose not to apply any rule of P to B, while the variants (B,k) and $(B,k+1)$ signal that (at least) k derivation steps have already been simulated. All the variants $(C,-1)$, $(C,k+1)$, and (C,i) with $C \in N \cup \{L\}$ and $1 \leqslant i < k$, as well as the non-terminals (a,j) with $a \in T$ and $1 \leqslant j \leqslant k$, serve as a way out of this component, since there are no rules with such non-terminals as left-hand sides.

Production set

$$P^{(3)} := \{ (a,i) \to (a,i-1) \mid a \in T \text{ and } 1 < i \leqslant k \}$$
$$\cup \{ (A,i) \to (A,i-1) \mid A \in N \cup \{L\} \text{ and } 1 < i < k \}$$
$$\cup \{ (A,k+1) \to (A,j) \mid A \in N \cup \{L\}, \text{ and } j = -2 \text{ if } k = 1 \text{ and } j = k-1 \text{ if } k > 1 \}$$
$$\cup \{ (B,1) \to (B,-2) \mid B \in V \cup \{L\} \}$$

can perform (at least) k derivation steps only if a sufficiently large number of steps has been simulated (i.e., at least k steps) by $P^{(2)}$, thus yielding a sentential form containing (besides terminals) only non-terminals of the forms $(A,-1)$ with $A \in N$ and $(B,-2)$ with $B \in V \cup \{L\}$. In case of an insufficient number of simulated steps, $P^{(3)}$ cannot perform (at least) k derivation steps, and the simulation stops at this stage.

$$P^{(4)} := \{ (A,-2) \to A, (A,-1) \to A, (A,-1) \to (A,-1) \mid A \in V \}$$
$$\cup \{ (A,-2) \to (A,-2) \mid A \in V \cup \{L\} \} \cup \{ (L,-2) \to \lambda \}$$
$$\cup \{ (A,i) \to F \mid A \in V \cup \{L\} \text{ and } 0 \leqslant i \leqslant k+1 \}.$$

The colouring component $P^{(4)}$ has the additional task to simulate the erasing rules of P. Note that in case P has no erasing rules, none of the simulating components has erasing rules, too. □

Whereas in the preceding two lemmas we have shown that the basic parent modes t and $\geqslant k$ can be simulated by the hybrid mode $(t \wedge \geqslant k)$, we now prove the opposite direction.

Lemma 18. *For each $k \in \mathbf{N}$, every context-free component which works in the $(t \wedge \geqslant k)$-mode can be simulated by three context-free components working in the t-mode*

and one component working in the $\geqslant k$-mode. The simulating components have λ-rules only if the simulated component has λ-rules.

Proof. The proof is quite similar to the one of the preceding lemma. Therefore, we only indicate the necessary changes. Now, let P be a context-free component working in the $(t \wedge \geqslant k)$-mode and let V_P be the set of all nonterminals appearing on the left-hand sides of the productions in P.

We consider the four components $P^{(i)}$, $1 \leqslant i \leqslant 4$, as defined in the proof of the previous lemma, yet we not only can omit all unit rules, but we also have to take care that the rules in $P^{(1)}$ and $P^{(2)}$ fulfill the following conditions:

- the variant $(A,0)$ of a nonterminal $A \in N$ is generated if and only if $A \in V_P$;
- all the variants (A,i) with $i \in \{-1, k+1\}$ or $1 \leqslant i < k$ of a nonterminal $A \in N$ are generated if and only if $A \notin V_P$.

$P^{(3)}$ is the only component working in the $\geqslant k$-mode, whereas the other components are working in the t-mode. □

Again we should like to mention that when using these constructions given in the preceding two lemmas for grammar systems, we have to add "individual component colours" to each nonterminal of a simulating component and, moreover, in the components working in the $(t \wedge \geqslant k)$-mode or in the t-mode, rules (e.g., unit rules $A \to A$ or trap rules $A \to F$ introducing the trap symbol F) prohibiting the interference of different components have to be added.

Using the lemmas proved above and Corollary 14, we obtain the following:

Theorem 19. *For each $k \geqslant 1$, we have*

$$\mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], \{t, \geqslant k\}) = \mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)).$$

Moreover,

$$\begin{aligned}
&\mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], \{t\} \cup \{ \geqslant k \mid k \in \mathbf{N} \}) \\
&\quad = \mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], \{ (t \wedge \geqslant k) \mid k \in \mathbf{N} \}) \\
&\quad = \bigcup_{k \in \mathbf{N}} \mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)).
\end{aligned}$$

Observe that we can again trade off internal versus external hybridization.

Are there other characterizations of the family of languages encountered in the preceding theorem? Indeed, it is one of the main results of this paper that these language families coincide with recurrent programmed languages with appearance checking, which we show in the following theorem:

Theorem 20. *If $k > 1$, then*

$$\mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)) = \mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac}).$$

Proof. First, we show how to simulate a CD grammar system $G=(N,T,S,P_1,\ldots,P_n)$ with $P_i=\{p_{i,1},\ldots,p_{i,n_i}\}$ and $p_{i,j}=A_{i,j}\to w_{i,j}$. For $1\leqslant i\leqslant n$ and $1\leqslant j\leqslant n_i$, $p_{i,j}$ is simulated by rules $\Lambda(r_{i,j,\ell})=A_{i,j}\to w_{i,j}$. For $1\leqslant\ell<k$, let $\sigma(r_{i,j,\ell})=\{r_{i,j',\ell+1}\mid 1\leqslant j'\leqslant n_i\}\cup\{r_{i,j,\ell}\}$ and $\phi(r_{i,j,\ell})=\emptyset$. Define $\sigma(r_{i,j,k})=\{r_{i,j',k}\mid 1\leqslant j'\leqslant n_i\}\cup\{t_{i,1}\}$ and $\phi(r_{i,j,k})=\emptyset$. After (at least) k such rule applications (the $r_{i,j,\ell}$-rules test the $\geqslant k$-property of derivation), it has to be checked whether the t-mode condition is met. This is done by rules labelled $t_{i,j}$ with $\Lambda(t_{i,j})=A_{i,j}\to F$ with $1\leqslant i\leqslant n$ and $1\leqslant j\leqslant n_i$. For these rules, let $\sigma(t_{i,j})=\phi(t_{i,j})=\{t_{i,j},t_{i,j+1}\}$, for $1\leqslant i\leqslant n$ and $1\leqslant j<n_i$, and let $\sigma(t_{i,n_i})=\phi(t_{i,n_i})=\{t_{i,n_i}\}\cup\{r_{i',j,1}\mid 1\leqslant i'\leqslant n\wedge 1\leqslant j\leqslant n_{i'}\}$) for $1\leqslant i\leqslant n$.

The initialization rule is defined by $\Lambda(\text{init})=S'\to S$ with $\sigma(\text{init})=\{\text{init}\}\cup\{r_{i,j,1}\mid 1\leqslant i\leqslant n\wedge 1\leqslant j\leqslant n_i\}$ and $\phi(\text{init})=\emptyset$. In total, we have the simulating grammar $G'=(N\cup\{S',F\},T,P,S',\Lambda,\sigma,\phi)$, where the rule set P is implicitly defined above.

For the other inclusion, note that every language $L\subseteq T^*$ can be written as (disjoint) union

$$L=\bigcup_{a\in T}\{a\}\tau_a(L)\cup L',$$

where L' is a finite set and $\tau_a(L)=\{w\in T^+\mid aw\in L\}$, i.e., τ_a is a sort of left quotient of L by a. Since τ_a is a non-erasing 1-a-transducer mapping, $\tau_a(L)\in\mathscr{L}(\mathrm{RP},\mathrm{CF}[-\lambda],\mathrm{ac})$ if $L\in\mathscr{L}(\mathrm{RP},\mathrm{CF}[-\lambda],\mathrm{ac})$ by Theorem 9. As the families $\mathscr{L}(\mathrm{CD}_\infty,\mathrm{CF}[-\lambda],(t\wedge\geqslant k))$ are easily seen to be closed under union and non-erasing (renaming) morphisms, it suffices to show that $\{\$\}L\in\mathscr{L}(\mathrm{CD}_\infty,\mathrm{CF}[-\lambda],(t\wedge\geqslant k))$ for every $L\in\mathscr{L}(\mathrm{RP},\mathrm{CF}[-\lambda],\mathrm{ac})$, where \$ is a special symbol with $\$\notin T$.

Now, we give the simulation of a recurrent programmed grammar $G=(N,T,P,S,\Lambda,\sigma,\phi)$. We can assume $N\cap\Lambda=\emptyset$. Moreover, let $\$\notin T$ be a new terminal symbol. We sketch the proof of the simulation only for the case $k=2$. The other modes (with $k>2$) can be obtained by using prolongation techniques at the "external label symbol", which represents the current label within the sentential form and finally goes to \$.

G is simulated by the CD grammar system $G'=(N\cup N'\cup\bar{N}\cup\Lambda\cup\Lambda'\cup\Lambda''\cup\{\tilde{S},F\},T\cup\{\$\},\tilde{S},\text{init},\text{exit},c_1,c_2,\text{succ}(p_1),\ldots,\text{succ}(p_L),\text{fail}(p_1),\ldots,\text{fail}(p_\ell))$, where N' and $\bar{N}$ contain primed and barred versions of the non-terminals of G; Λ' and Λ'' contain primed and double-primed versions of the labels of G. $\tilde{S}$ is the new start symbol, and F is a trap symbol. The label set Λ of G equals $\{p_1,\ldots,p_L\}$, where we assume that the labels in $\{p_1,\ldots,p_\ell\}$ label all the rules with non-empty failure field, i.e., for all i with $1\leqslant i\leqslant L$ we require $i\leqslant\ell$ if and only if $\phi(p_i)\neq\emptyset$.

Consider the simulation of a rule $\Lambda(p)=A\to w$ with $p\in\sigma(p)=\phi(p)$. First we guess whether the success or the failure case is entered. For each of these cases, a separate simulating component is introduced whose rules are described next.

(1) Failure case: Component fail(p) contains rules $p\to p'$ and $p'\to q''$ with $q\in\phi(p)$. Furthermore, the rules $A\to F$, $\bar{A}\to F$, and $A'\to F$ as well as $B\to F$ and $B'\to F$ for all $B\in N\backslash\{A\}$ belong to fail(p).

(2) Success case: Component succ(p) has rules $p\to q'$ for $q\in\sigma(p)$ and $A'\to\bar{w}$, where $\bar{w}$ is obtained from w by barring every occurrence of non-terminals in w. Further-

more, it contains $r \to F$ for all labels $r \in \Lambda \setminus \{p\}$, and $A \to F$ as well as $B \to F$ and $B' \to F$ for all $B \in N \setminus \{A\}$. Observe that the $(t \wedge \geqslant 2)$-mode will force at least one rule $A' \to \bar{w}$ to be applied in a successful application.

In case of empty failure field, the first case is omitted.

We have two colouring components:

(1) Component c_1 contains $B \to \bar{B}$ and $B \to B'$ for every non-terminal $B \in N$. An application of c_1 has to precede a successful application of a component succ(p) with $\Lambda(p) = A \to w$. By priming them, c_1 serves to select some occurrences of the non-terminal A to be replaced in a following application of the component succ(p). On the other hand, all other non-terminals have to be barred in order not to be sent to the trap symbol F by the rules in succ(p). In the same way, an application of c_1 has to precede a successful application of a component fail(p); all non-terminals $B \neq A$ have to be barred in order to protect them from being sent to the trap symbol F by the rules in fail(p).

(2) Component c_2 contains $p'' \to p'$, $p' \to p'$, and $p' \to p$ for every label p and $\bar{B} \to B$, $B' \to F$ for every non-terminal $B \in N$. Component c_2 is used to regain a sentential form of shape pw with $w \in V_G^*$, $p \in \Lambda$.

The exit point is given by the component called exit containing $p \to p$, $p \to \$$, $p' \to F$ and $p'' \to F$ for every label $p \in \Lambda$ and $B \to F$, $\bar{B} \to F$, $B' \to F$ for every symbol $B \in N$. The entry point is given by the component called init containing $\tilde{S} \to \tilde{S}$, $\tilde{S} \to pS$ for every label $p \in \Lambda$.

By induction, we can show that $\{\$\}L(G) = L(G')$. We only indicate some details of the induction steps.

First, we consider the inclusion $\{\$\}L(G) \subseteq L(G')$. A successful derivation step $(x, p) \Rightarrow (y, q)$ in G (with $q \in \sigma(p)$) can be simulated by applying the components c_1, succ(p), and c_2 consecutively, so that the sentential form px of G' is transformed into qy. If p is applied in the appearance checking manner, then $(x, p) \Rightarrow (x, q)$ in G (with $q \in \phi(p)$) can be simulated by applying the components c_1, fail(p), and c_2 consecutively.

On the other hand, we have to show the inclusion $\{\$\}L(G) \supseteq L(G')$. First, observe that our inductive argument especially shows that every successful derivation $\tilde{S} \overset{*}{\Rightarrow} w \in \{\$\}T^*$ in G' can be decomposed into

$$\tilde{S} \Rightarrow pS \Rightarrow^+ q_1x_1 \Rightarrow^+ q_2x_2 \Rightarrow^+ \cdots \Rightarrow^+ q_mx_m \Rightarrow^+ x_m,$$

where $p, q_i \in \Lambda$ and $x_i \in V_G^*$ for $1 \leqslant i \leqslant m$, $x_m \in \{\$\}T^*$, and moreover, every other sentential form occurring in the observed derivation does not lie in ΛV_G^*.

So let us consider a sentential form px of G', where $x \in V_G^*$ contains at least one non-terminal. Then only c_1 can be applied without introducing the trap symbol F. We now have a sentential form px', where x' is obtained from x by either priming or barring the occurring non-terminals. At this point, three continuations are possible which do not necessarily introduce the trap symbol:

(a) We can use c_2 to obtain px again (provided that at least two non-terminals occur in x); obviously, this does not give us any advance in the derivation.

(b) We can use fail(p) without introducing the trap symbol if no primed or barred version of the left-hand side of rule $\Lambda(p)$ is present in x', which is the case if and only if x does not contain the left-hand side of rule $\Lambda(p)$, so that we get a sentential form $q''x'$ with $q \in \phi(p)$ and $x' \in (\bar{N} \cup T)^*$. Here, we now must continue using c_2 in order to get a sentential form not containing the trap symbol. Hence, we regain a sentential form of the kind with which we started our argument.

(c) We can use succ(p) without introducing the trap symbol if at least one of the occurrences of the left-hand side A of $\Lambda(p)$ was previously primed. When using succ(p), all (say n) primed occurrences of A will be replaced by $\bar{w}$, corresponding to the right-hand side w of $\Lambda(p) = A \to w$. Note that n repetitive applications of rule p are possible, because in recurrent programmed grammars we require $p \in \sigma(p)$. We leave component succ(p) with a sentential form $q'\bar{y}$, where $\bar{y}$ contains, besides terminals, possibly barred symbols from $\bar{N}$ (all other non-terminals from $N \cup N'$ – except A' – would have been sent to the trap symbol F). Hence, we now are forced to apply c_2 in order to avoid the introduction of F. Then, we get a sentential form of type qy, $q \in \Lambda$, $y \in V_G^*$. Altogether, this derivation $px \overset{*}{\Rightarrow} qy$ in G' can be simulated by n derivation steps $(x, p) \Rightarrow^n (y, q)$ in G.

Finally, a sentential form px with $x \in T^*$ can be transformed into $\$x$ by applying component exit. □

Hence, we have linked certain classes of CD languages with recurrent programmed languages with appearance checking. Observe that in this way the problem whether arbitrary hybrid CD grammar systems (using the basic modes) characterize $\mathscr{L}(\mathrm{RE})$ or not is connected with the old open question whether the trivial inclusion

$$\mathscr{L}(\mathrm{RP}, \mathrm{CF}, \mathrm{ac}) \subseteq \mathscr{L}(\mathrm{P}, \mathrm{CF}, \mathrm{ac}) = \mathscr{L}(\mathrm{RE})$$

is strict or not. Notice that it is even unknown whether $\mathscr{L}(\mathrm{RP}, \mathrm{CF}, \mathrm{ac})$ contains non-recursive languages.

Remark 21. The inclusion

$$\mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], \geqslant k) \subseteq \mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda]).$$

is obvious for $k = 1$ and can be proved for each $k > 1$ by using the same construction as in the preceding theorem.

In [8, Theorem 3.2] it was shown that

$$\mathscr{L}(\mathrm{RC}, \mathrm{CF}[-\lambda]) \subseteq \mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda]) \subseteq \mathscr{L}(\mathrm{P}, \mathrm{CF}[-\lambda]),$$

where $\mathscr{L}(\mathrm{RC}, \mathrm{CF}[-\lambda])$ is the family of languages that can be generated by permitting random context grammars, see [3]. Hence, we have a link to the old open problem whether permitting random context grammars are as powerful as programmed grammars without appearance checking or not.

As regards the hierarchical structure of $\mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geqslant k))$ for different k, we have the prime number lattice structure implied by the prolongation technique (cf. Lemma 10), but we do not know whether there are strict inclusions besides the one contained in the following corollary.

Corollary 22. *For $k > 1$, we have*

$$\mathscr{L}(\mathrm{ET0L}) = \mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geqslant 1)) \subset \mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)).$$

Proof. Trivially, $\mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geqslant 1)) = \mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], t)$, and the latter class equals $\mathscr{L}(\mathrm{ET0L})$, see [2, Theorem 3.10]. $\mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac})$, which coincides with $\mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geqslant k))$ by Theorem 20, strictly contains $\mathscr{L}(\mathrm{ET0L})$ in [3, Theorem 8.3]. □

Observe that the reasoning of the last corollary also gives an alternative proof of [13, Theorem 3.5].

In the remainder of this section, we turn our attention to the number of components working together in a grammar system, because it is a natural measure of descriptional complexity. It is known that an arbitrary number of components working in the t-mode can be simulated by (at most) three components working in the t-mode, see [1, 11, Lemma 2]; the proof of the quoted lemma is basically valid for $(t \wedge \geqslant k)$-mode components, too; only the colouring components have to be prolongated in order to turn them into $(t \wedge \geqslant k)$-mode components, see Lemma 16 above.

As the $(t \wedge \geqslant 1)$-mode and the simple t-mode coincide, the following result is obvious by the characterization of $\mathscr{L}(\mathrm{CF})$ and $\mathscr{L}(\mathrm{ET0L})$ given in [2, Theorem 3.10].

Corollary 23. *Let $n \in \mathbf{N} \cup \{\infty\}$, $n \geqslant 3$. Then we have*

$$\begin{aligned}\mathscr{L}(\mathrm{CF}) = \mathscr{L}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \geqslant 1)) &= \mathscr{L}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], (t \wedge \geqslant 1)) \\ &\subset \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (t \wedge \geqslant 1)) \\ &= \mathscr{L}(\mathrm{ET0L}).\end{aligned}$$

For arbitrary $k \geqslant 2$ the situation is a little bit different from the previous case.

Theorem 24. *Let $n \in \mathbf{N} \cup \{\infty\}$, $n \geqslant 3$. For each $k \geqslant 2$,*

$$\begin{aligned}\mathscr{L}(\mathrm{CF}) = \mathscr{L}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)) &\subset \mathscr{L}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)) \\ &\subseteq \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)) \\ &= \mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac}) \\ &\subseteq \mathscr{L}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac}).\end{aligned}$$

Proof. The inclusions themselves are trivial or follow by Pawn [13, Theorem 3.6] together with the well-known equivalence of matrix and programmed grammars. Using

individually coloured non-terminals for each component, any CD grammar system with an arbitrary number of $(t \wedge \geqslant k)$-mode components can be simulated by a CD grammar system with three components, where two components serve as switches between the non-terminal colours, and one component does the actual simulation.

In order to prove the strictness of the first inclusion, we show how the non-context-free language $L = \{a_1^n a_2^n \dots a_{k+1}^n \mid n \geqslant 1\}$ can be generated by a CD grammar system with two $(t \wedge \geqslant k)$-mode components: consider the grammar system $G = (N, T, S_1, P_1, P_2)$, where P_1, P_2 work in the $(t \wedge \geqslant k)$-mode. For both components, we take $N = \{S_i, A_i, A_i', \mid 1 \leqslant i \leqslant k\}$ as non-terminal alphabet and $T = \{a_1, \dots, a_{k+1}\}$ as terminal alphabet. The components P_1 and P_2 are defined as follows:

$$P_1 = \{S_i \to S_{i+1} \mid 1 \leqslant i < k\} \cup \{S_k \to A_1 \cdots A_k\} \cup \{A_i' \to A_i \mid 1 \leqslant i \leqslant k\}$$

and

$$\begin{aligned} P_2 = {} & \{A_i \to a_i A_i' \mid 1 \leqslant i \leqslant k-1\} \cup \{A_k \to a_k A_k' a_{k+1}\} \\ & \cup \{A_i \to a_i \mid 1 \leqslant i \leqslant k-1\} \cup \{A_k \to a_k a_{k+1}\}. \end{aligned}$$

Then we have $L(G) = L$, since every derivation of G leading to a terminal word is of the form

$$S_1 \Rightarrow_1^{=k} A_1 \dots A_k \cdots \Rightarrow_2^{=k} a_1^n \dots a_k^n a_{k+1}^n,$$

where the intermediate steps are of the form

$$a_1^i A_1 \dots a_k^i A_k a_{k+1}^i \Rightarrow_2^{=k} a_1^{i+1} A_1' \dots a_k^{i+1} A_k' a_{k+1}^{i+1} \Rightarrow_1^{=k} a_1^{i+1} A_1 \dots a_k^{i+1} A_k a_{k+1}^{i+1};$$

if a non-vanishing number of occurrences of A_i' less than k is obtained by using P_2 then neither P_1 nor P_2 can perform k derivation steps any more. Hence, G generates L, which is a non-context-free language. □

As we have shown in Theorem 19, we can trade off internal versus external hybridization in the case of mixing t-mode and $\geqslant k$-mode. It is interesting to compare the language classes obtained by CD grammar systems with n collaborating components in each of these modes with the corresponding class obtained by CD grammar systems with n components working in the $(t \wedge \geqslant k)$-mode.

Corollary 25. *In general, for each* $n, k \in \mathbf{N}$ (or $n = \infty$), *we have*

$$\begin{aligned} \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], t) &= \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (t \wedge \geqslant 1)) \\ &\subseteq \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)). \end{aligned}$$

More specifically, for $k \in \mathbf{N}$ *and* $k > 1$*:*
(1) $\mathscr{L}(\mathrm{CF}) = \mathscr{L}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], t) = \mathscr{L}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \geqslant k))$;
(2) $\mathscr{L}(\mathrm{CF}) = \mathscr{L}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], t) \subset \mathscr{L}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], (t \wedge \geqslant k))$;

(3) *for all* $n \geqslant 3$ *or* $n = \infty$,

$$\mathscr{L}(\mathrm{ET0L}) = \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], t) \subset \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)).$$

Proof. The strictness of the trivial inclusions follows from Corollary 22 and Theorem 24. □

Of course, it is also interesting to compare the parent mode $\geqslant k$ with the hybrid mode $(t \wedge \geqslant k)$ as regards their generative power.

Theorem 26. *For* $k, n \in \mathbf{N}$, *and* $n \geqslant 3$ *or* $n = \infty$, *we have*

$$\mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], \geqslant k) \subset \mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)).$$

Proof. As according to the preceding corollary we have

$$\mathscr{L}(\mathrm{ET0L}) \subseteq \mathscr{L}(\mathrm{CD}_3, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)),$$

we obtain $L = \{a^{2^m} \mid m \geqslant 0\} \in \mathscr{L}(\mathrm{CD}_3, \mathrm{CF} - \lambda, (t \wedge \geqslant k))$; according to [9], $L \notin \mathscr{L}(\mathrm{P,CF})$, which family includes $\mathscr{L}(\mathrm{CD}_n, \mathrm{CF}[-\lambda], \geqslant k)$ by Pawn [13, Diagram 1]. The inclusion itself follows from Lemma 17 combined with Theorem 24. □

Furthermore, we trivially know that, for each $k \in \mathbf{N}$,

$$\mathscr{L}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], \geqslant k) = \mathscr{L}(\mathrm{CD}_1, \mathrm{CF}[-\lambda], (t \wedge \geqslant k)),$$

since this language family equals $\mathscr{L}(\mathrm{CF})$. We do not know anything about the relation when we restrict our attention to at most two components. We suspect that $\mathscr{L}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], \geqslant k)$ and $\mathscr{L}(\mathrm{CD}_2, \mathrm{CF}[-\lambda], (t \wedge \geqslant k))$ are incomparable, if $k \geqslant 2$.

6. Summary and open problems

We have investigated an internal mode hybridization and contrasted this to the already examined external mode hybridization: internally hybrid modes may be used to characterize their externally hybrid counterparts. This immediately also yields results concerning the power of hybrid modes and their parent modes. In short, the hybrid modes considered in this paper, especially the combination of the t-mode and the $\geqslant k$-mode, are at least as powerful as their parent modes. This situation will change drastically when turning to the modes $(t \wedge \leqslant k)$ and $(t \wedge = k)$, see [5].

Moreover, we found several interesting links between families of languages defined by CD grammar systems working in particular with hybrid modes and certain classes of programmed languages (already in the very first paper on CD grammar systems [10], such links were observed). This is of special importance because it connects the field of CD grammar systems with the better explored area of regulated rewriting. On

the other hand, new light on old problems in regulated rewriting is shed, in particular in relation with recurrent programmed grammars.

In our opinion, the greatest still open problems in this area are:

(1) What are the exact relations between the following language families for varying $k \in \mathbf{N}$ (besides the prime lattice inclusion structure):
   (a) $\mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], \geqslant k)$,
   (b) $\mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], =k)$, and
   (c) $\mathscr{L}(\mathrm{CD}_\infty, \mathrm{CF}[-\lambda], (t \wedge \geqslant k))$?

(2) What are the exact relations between the following language families:
   (a) $\mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], \{\geqslant k \mid k \in \mathbf{N}\})$,
   (b) $\mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], \{=k \mid k \in \mathbf{N}\})$,
   (c) $\mathscr{L}(\mathrm{RC}, \mathrm{CF}[-\lambda])$, and
   (d) $\mathscr{L}(\mathrm{P}, \mathrm{CF}[-\lambda])$?

(3) Are the following inclusions strict:
   (a) $\mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], B\backslash\{t\}) \subseteq \mathscr{L}(\mathrm{P}, \mathrm{CF}[-\lambda])$;
   (b) $\mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda], \mathrm{ac}) \subseteq \mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], B) \subseteq \mathscr{L}(\mathrm{P}, \mathrm{CF}[-\lambda], \mathrm{ac})$;
   (c) $\mathscr{L}(\mathrm{HCD}_\infty, \mathrm{CF}[-\lambda], \{\geqslant k \mid k \in \mathbf{N}\}) \subseteq \mathscr{L}(\mathrm{RP}, \mathrm{CF}[-\lambda]) \subseteq \mathscr{L}(\mathrm{P}, \mathrm{CF}[-\lambda])$?

Let us finally mention that internally hybrid modes are also very interesting from a different point of view: when using (H)CD grammar systems as language acceptors, we were able to obtain the first examples where generating grammars are more powerful than accepting grammars, which solves an open problem in the theory of accepting grammars, see [6].

References

[1] E. Csuhaj-Varjú, J. Dassow, On cooperating/distributed grammar systems, J. Inform. Process. Cybernet. EIK 26 (1/2) (1990) 49–63.

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

[3] J. Dassow, Gh. Păun, Regulated Rewriting in Formal Language Theory, EATCS Monographs in Theoretical Computer Science, vol. 18, Springer, Berlin, 1989.

[4] H. Fernau, R. Freund, Bounded parallelism in array grammars used for character recognition, in: P. Perner, P. Wang, A. Rosenfeld (Eds.), Advances in Structural and Syntactical Pattern Recognition, Proc. SSPR'96, Lecture Notes in Computer Science, vol. 1121, Springer, Berlin, 1996,, pp. 40–49.

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

[6] H. Fernau, M. Holzer, Accepting multi-agent systems II, Acta Cybernet. 12 (1996) 361–379.

[7] H. Fernau, M. Holzer, R. Freund, Bounding resources in cooperating distributed grammar systems, in: S. Bozapalidis (Ed.), Proc. 3rd Internat. Conf. Developments in Language Theory, Aristotle University of Thessaloniki, 1997, pp. 261–272.

[8] H. Fernau, D. Wätjen, Remarks on regulated limited ET0L systems and regulated context-free grammars, Theoret. Comput. Sci. 194 (1–2) (1998) 35–55.

[9] D. Hauschildt, M. Jantzen, Petri net algorithms in the theory of matrix grammars, Acta Inform. 31 (1994) 719–728.

[10] R. Meersman, G. Rozenberg, Cooperating grammar systems, in: Proc. Math. Found. Comput. Sci. MFCS'78, Lecture Notes in Computer Science, vol. 64, Springer, Berlin, 1978, pp. 364–374.

[11] V. Mitrana, Hybrid cooperating/distributed grammar systems, Comput. Artif. Intell. 12 (1) (1993) 83–88.
[12] N.J. Nilsson, Principles of Artificial Intelligence, Springer, Berlin, 1982.
[13] Gh. Păun, On the generative capacity of hybrid CD grammar systems, J. Inform. Process. Cybernet. EIK 30 (4) (1994) 231–244.
[14] S.H. von Solms, Some notes on ET0L-languages, Internat. J. Comput. Math. 5(A) (1976) 285–296.