Bounded Parallelism in Array Grammars Used for Character Recognition

Henning Fernau[1]* and Rudolf Freund[2]

[1] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen
Sand 13, D-72076 Tübingen, Germany
email: fernau@informatik.uni-tuebingen.de
[2] Institut für Computersprachen, Technische Universität Wien
Resselgasse 3, A-1040 Wien, Austria
email: freund@csdec1.tuwien.ac.at

Abstract. The aim of this paper is to elaborate the power of cooperation in generating and analysing (handwritten) characters by array grammars. We present various non-context-free sets of arrays that can be generated in a simple way by cooperating distributed array grammar systems with prescribed teams working in different modes and show the power of the mechanism of cooperation for picture description and analysis as well as the efficiency of these models where several sets of productions work in parallel on the given sentential form.

1 Introduction

Cooperation of agents is a usual strategy for approaching complex problems. This strategy is supposed to increase the total competence of the individual agents working together for solving a common task. The recognition of specific patterns like (handwritten) characters can be seen as such a complex task that might be attacked by several agents working in parallel on the underlying pattern. Moreover, the forming of different teams of specialized agents working on the pattern in different modes during subsequent stages of the recognition procedure can improve the overall efficiency.

Cooperating array systems turned out to be quite useful for picture representation, "simple" systems being able to describe "complicated" sets of pictures [7]. A similar conclusion was obtained for picture description by using programmed array grammars [12], which, in fact, are grammar systems provided with a control on sequencing the work of components. Also the matrix array grammars introduced in [23] can be considered as a particular case of grammar systems; they consist of two components, the horizontal and the vertical one, working first in the horizontal and then in the vertical one, until producing an array that cannot be processed any more (which resembles the t-mode of derivation in [5], [6]).

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

Cooperating string grammar systems were introduced in [17] and further developed in [5]. In cooperating distributed (array) grammar systems, a finite number of components, i.e., sets of (array) productions, cooperates guided by a specific strategy, e.g., an activated component can perform an arbitrary number of derivation steps, exactly k derivation steps, at least k derivation steps, or at most k derivation steps; in the maximal derivation mode (t-mode), the activated component has to work as long as possible. The generative power of cooperating distributed grammar systems with several variants of cooperation strategies has been studied in many papers (for details the reader is referred to [6]); cooperating distributed array grammar systems were investigated in [7]. The formation of *teams* of productions as another method of cooperation was considered in [14] (where all possible teams of a constant size were considered) and in [20] (where the more flexible formation of *prescribed* teams was introduced).

In this paper, we restrict ourselves to two-dimensional array grammar systems with prescribed teams in order to obtain concise but depictive representations of the pictures in our examples; yet we think that already these results we elaborate in this paper demonstrate the power which evolves from cooperation when using prescribed teams in the case of array grammars. In particular, we sketch how these grammar systems can actually be employed for character recognition purposes. On the other hand, one of the advantages of array grammars is given by the simplicity to cover also higher dimensions (e.g., see [3]); in the same way, the mechanism of cooperation in array grammar systems can be extended to higher dimensions in an obvious and easy way, e.g., three-dimensional array grammar systems with prescribed teams promise to be an interesting tool for the generation and the analysis of three-dimensional objects.

In the next section, (two-dimensional) arrays and array grammars are defined, whereas in the third section array grammar systems with prescribed teams of array productions and the different derivation modes are introduced; we present some examples for languages of rather complicated pictures which can easily be described by array grammar systems with prescribed teams of array productions. In the fourth section, we describe how (handwritten) characters can be analysed by using suitable array grammar systems with prescribed teams of array productions; a short discussion of the results exhibited in this paper and an outlook to future research topics conclude the paper.

2 Arrays and Array Grammars

The reader is assumed to be familiar with the basic notions and results of formal language theory (e.g., see [8], [22]). Hence, in this section we only introduce the definitions and notations for arrays and array grammars ([4], [7], [11], [12], [21]).

For an alphabet V, by V^{2+} we denote the set of two-dimensional non-empty finite and connected arrays of symbols in V (patterns obtained by marking with symbols in V a finite number of unit squares of the plane; as neither the origin nor the axes of the plane are fixed, each pattern is identified by its marked squares, without reference to its "position" in the plane). The elements of V^{2+}

are called *pictures (arrays)* over V and sets of pictures (arrays) are called *array languages.*

Given an array $x \in V^{2+}$ (for some alphabet V) and a finite pattern α of symbols in $V \cup \{\#\}$, we can say that α is a *sub-pattern* of x, if we can place α on x such that all squares of α marked by symbols in V coincide with the corresponding symbols in x and each blank symbol # in α corresponds to a blank symbol # in x.

An *(isometric) array grammar* is a construct $G = (V_N, \#, V_T, S, P)$, where V_N, V_T are disjoint alphabets, # is a special (blank) symbol, $S \in N$, and P is a finite set of rewriting rules of the form $\alpha \to \beta$, where α, β are finite patterns over $V_N \cup V_T \cup \{\#\}$ satisfying the condition that the shapes of α and β are identical (we say that they are *isometric*); for a more precise definition of array grammars, the reader is referred to [4], [12], [18], or [21].

Thus, for an array grammar $G = (V_N, \#, V_T, S, P)$ we can define the relation $x \Longrightarrow y$, for $x, y \in (V_N \cup V_T)^{2+}$, if there is a rule $\alpha \to \beta \in P$ such that α is a sub-pattern of x and y is obtained by replacing α in x by β (remember that α and β are isometric). The reflexive and transitive closure of $\Longrightarrow$ is denoted by $\Longrightarrow^*$, and the array language generated by G is defined by $L(G) = \left\{x \in V_T^{2+} \mid S \Longrightarrow^* x\right\}$. An array production $\alpha \to \beta$ in an array grammar is said to be

1. *monotone* if the non-# symbols in α are not replaced by # in β,
2. *#-context-free* if α consists of exactly one nonterminal and some occurrences of blank symbols #,
3. *context-free* if it is #-context-free and β contains no symbol #;
4. *regular,* if it is of one of the following forms:

$$\# A \to B a,\ A \# \to a B,\ \begin{matrix}\# \\ A\end{matrix} \to \begin{matrix}B \\ a\end{matrix},\ \begin{matrix}A \\ \#\end{matrix} \to \begin{matrix}a \\ B\end{matrix},\ A \to a, \text{ where } \begin{cases} A, B \in V_N, \\ a \in V_T. \end{cases}$$

An array grammar is said to be of type $ENUMA$, $MONA$, $\#-CFA$, CFA, or $REGA$, respectively, if every array production in P is arbitrary, monotone, #-context-free, context-free, or regular, respectively. The same notation is used for the corresponding (families of) array languages. These families of array languages form a Chomsky-like hierarchy [4]: $REGA \subset CFA \subset MONA \subset ENUMA$.

3 Prescribed teams

The definition of the parallel application of a constant number of array productions to a given array is the crucial point in the definition of *array* grammars with prescribed teams of array productions. For the definition of (string) grammars with prescribed teams of context-free (string) productions the reader is referred to [20]. An *array grammar system with prescribed teams* is a construct $G = (V_N, \#, V_T, S, (R, T))$, where V_N and V_T are finite disjoint sets of nonterminal and terminal symbols, respectively, $\# \notin V_N \cup V_T$ is the blank symbol, $S \in V_N$ is the start symbol, R is a (non-empty) finite set of (non-empty) finite sets of array productions over $V_N \cup V_T$, and T is a (non-empty) finite set of *teams*, where each team is a (non-empty) subset of R, i.e.,

$R = \{R_h \mid 1 \leq h \leq n\}, n \geq 1,$

$R_h = \{p_{h,l} \mid 1 \leq l \leq n_h\}, 1 \leq h \leq n, n_h \geq 1,$

where the $p_{h,l}$ are array productions over $V_N \cup V_T$,

$T = \{Q_i \mid 1 \leq i \leq m\}, m \geq 1,$

$Q_i = \{P_{i,j} \mid 1 \leq j \leq m_i\}, m_i \geq 1,$

where $P_{i,j} \in R$ for $1 \leq j \leq m_i$ and $1 \leq i \leq m$.

For $X \in \{ENUMA, MONA, \#\text{-}CFA, CFA\}$, G is called of type X if every production $p_{h,l}$, $1 \leq h \leq n$, $1 \leq l \leq n_h$, is of type X.

For a team Q_i, $1 \leq i \leq m$, $Q_i \in T$, $Q_i = \{P_{i,j} \mid 1 \leq j \leq m_i\}$, and two arrays $\mathcal{D}_1$ and $\mathcal{D}_2 \in (V_N \cup V_T)^{2+}$ a direct derivation step is defined by $\mathcal{D}_1 \vdash_{Q_i} \mathcal{D}_2$ if and only if there are array productions $p_j \in P_{i,j}$, $1 \leq j \leq m_i$, such that in $\mathcal{D}_1$ we can find m_i non-overlapping areas such that the sub-patterns of $\mathcal{D}_1$ located at these areas coincide with the left-hand sides of the array productions p_j and yield $\mathcal{D}_2$ by replacing them by the right-hand sides of the array productions p_j.

An application of the team Q_i to an array $\mathcal{D}_1$ therefore means the following: from each set $P_{i,j}$, one array production p_j is chosen such that $p_1, ..., p_{m_i}$ can be applied in a parallel manner to $\mathcal{D}_1$ without disturbing each other. Note that the array productions p_j need not all be different although coming from different sets within the team Q_i. Like in cooperating array grammar systems (see [7]) also for array grammar systems with prescribed teams we can define the derivation relations $\vdash_{Q_i}^{*}$, $\vdash_{Q_i}^{=k}$, $\vdash_{Q_i}^{\leq k}$, $\vdash_{Q_i}^{\geq k}$, and $\vdash_{Q_i}^{t}$, respectively, i.e., derivations with the team Q_i of arbitrary, of exactly k successive steps, of at most k steps, of at least k steps, and of as many steps as possible, respectively; this maximal derivation mode t (according to [20]) is defined more precisely by: $\mathcal{D}_1 \vdash_{Q_i}^{t} \mathcal{D}_2$ if and only if $\mathcal{D}_1 \vdash_{Q_i}^{*} \mathcal{D}_2$ and there is at least one component P_{i,j_0} in the team Q_i such that no array production in P_{i,j_0} can be applied to $\mathcal{D}_2$ anymore. In that way, we have defined different stop conditions for the teams, i.e., conditions when an active team must or can become inactive. Note that in the t-mode a derivation with a team Q_i can be blocked, although in every $P_{i,j}$ we can find an array production which is applicable to the underlying array. This can happen because no disjoint areas can be found such that from each $P_{i,j}$ an adequate array production can be applied at this area.

For each of these derivation modes $d \in F$, $F = \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$, defined above we can define an array language generated by the array grammar system with prescribed teams in the derivation mode d by

$$L(G,d) = \{\mathcal{D} \in V_T^{+2} \mid \{(v_S, S)\} \vdash_{Q_{i_1}}^{d} \mathcal{D}_1 \vdash_{Q_{i_2}}^{d} ... \vdash_{Q_{i_r}}^{d} \mathcal{D}_r = \mathcal{D},\ r \geq 1,\ 1 \leq i_j \leq m, \text{ for } 1 \leq j \leq r\}.$$

Let $X \in \{ENUMA, MONA, \#\text{-}CFA, CFA\}$. The family of array languages generated by array grammar systems with prescribed teams of array productions of type X in the derivation mode d is denoted by $PT(X, d)$. Some results on these families of array languges can be found in [13].

Example 1. The array language L_1 containing all right angles as depicted in the figure to the right can be generated by the array grammar system with prescribed teams of context-free array productions

$$G_1 = (\{S, U, R\}, \#, \{a\}, S, (R_1, T_1)),$$

where

$$\begin{array}{l} \overbrace{}^{n} \\ n\left\{\begin{array}{l} a\\ \vdots\\ a\\ a\,a\ldots a\end{array}\right. \end{array}$$

Right angle with arms of equal lengths.

$R_1 = \{P_1, P_2, P_3, P_4, P_5\}, T_1 = \{Q_1, Q_2, Q_3\},$
$Q_1 = \{P_1\}, Q_2 = \{P_2, P_3\}, Q_3 = \{P_4, P_5\},$
$P_1 = \left\{ \begin{matrix} \# & \\ S & \# \end{matrix} \rightarrow \begin{matrix} U & \\ a & R \end{matrix} \right\}, P_2 = \left\{ \begin{matrix} \# \\ U \end{matrix} \rightarrow \begin{matrix} U \\ a \end{matrix} \right\}, P_3 = \{R\# \rightarrow aR\},$
$P_4 = \{U \rightarrow a\}, P_5 = \{R \rightarrow a\},$

in the derivation modes $*$, $= 1$, ≥ 1, and $\leq k$ for $k \geq 1$, i.e., $L(G_1, d) = L_1$ for $d \in \{*, = 1, \geq 1\} \cup \{\leq k \mid k \geq 1\}$, whereas $L(G_1, d')$ is empty for the other derivation modes $d' \in \{= k, \geq k \mid k \geq 2\} \cup \{t\}$.

It is easy to see that $L(G_1, d) = L_1$: After once applying the singleton team Q_1, the team Q_2 is applied $n - 2$ times $(n \geq 2)$ in such a way that from P_2 the array production $\begin{matrix} \# \\ U \end{matrix} \rightarrow \begin{matrix} U \\ a \end{matrix}$ is applied, whereas from P_3 the array production $R\# \rightarrow aR$ is taken. Finally, the team Q_3 is applied in the last step, i.e., from P_4 the array production $U \rightarrow a$ is taken, whereas from P_5 the array production $R \rightarrow a$ is applied, which yields the terminal array as depicted above with arms of equal lengths. Observe that except for the array production in P_1 all other array productions appearing in G_1 are even regular. □

In addition to the single derivation modes in F, we can also consider complex modes of the form $\{f_1, ..., f_d\}$, $f_i \in F$, $F = \{*, t\} \cup \{= k, \leq k, \geq k\}$, and assign different modes of that form to each team. In that way we obtain internally hybrid array grammar systems with prescribed teams (theoretical results about internally hybrid string grammar systems can be found in [9]):

An *internally hybrid array grammar system with prescribed teams* is a construct $G = (V_N, \#, V_T, S, (R, T))$, where V_N, V_T, $\#$, S, R are as in array grammars with prescribed teams, and T is a (non-empty) finite set of teams and complex derivation modes, i.e.,

$R = \{R_h \mid 1 \leq h \leq n\}, n \geq 1,$
$R_h = \{p_{h,l} \mid 1 \leq l \leq n_h\}, 1 \leq h \leq n, n_h \geq 1,$
where the $p_{h,l}$ are array productions over $V_N \cup V_T$,
$T = \{(Q_i, F_i) \mid 1 \leq i \leq m\}, m \geq 1,$
F_i is a non-empty finite subset of F, $1 \leq i \leq m$,
$Q_i = \{P_{i,j} \mid 1 \leq j \leq m_i\}, m_i \geq 1,$
where $P_{i,j} \in R$ for $1 \leq j \leq m_i$ and $1 \leq i \leq m$.

For $X \in \{ENUMA, MONA, \#\text{-}CFA, CFA\}$, G is called to be of type X if every production $p_{h,l}$, $1 \leq h \leq n$, $1 \leq l \leq n_h$, is of type X.

For a pair (Q_i, F_i) consisting of a team and its assigned set of derivation modes and two arrays $\mathcal{D}_1$ and $\mathcal{D}_2 \in (V_N \cup V_T)^{+2}$ a direct derivation step is defined by $\mathcal{D}_1 \vdash_{(Q_i,F_i)} \mathcal{D}_2$ if and only if $\mathcal{D}_1 \vdash^{f}_{Q_i} \mathcal{D}_2$ for every $f \in F_i$, i.e., for each derivation mode f in F_i the array $\mathcal{D}_2$ must be derivable from the array $\mathcal{D}_1$ by the team Q_i in the mode f.

Let $X \in \{ENUMA, MONA, \#\text{-}CFA, CFA\}$. The family of array languages generated by internally hybrid array grammar systems with prescribed teams of array productions of type X is denoted by $PTIH(X)$.

Remark 1. Observe that obviously not every combination of single derivation modes yields a meaningful complex derivation mode, e.g., $\{\leq k_1, \geq k_2\}$ yields a contradiction for $k_1 < k_2$; on the other hand, for $k_1 \geq k_2$ this set describes an interval of derivation steps between k_1 and k_2; for $k_1 = k_2$ the set $\{\leq k_1, \geq k_2\}$ has the same effect as $\{= k_2\}$. □

Example 2. The array language L_1 containing all right angles with arms of equal lengths as already described in Example 1 in a similar way can be generated by the internally hybrid array grammar system with prescribed teams of context-free array productions

$G_2 = (\{S, U, R\}, \#, \{a\}, S, (R_2, T_2))$,

$R_2 = \{P_1, P_2, P_3, P_4, P_5\}$, $T_2 = \{(Q_1, \{f\}), (Q_2, \{g\}), (Q_3, \{h\})\}$,

$Q_1 = \{P_1\}$, $Q_2 = \{P_2, P_3\}$, $Q_3 = \{P_4, P_5\}$,

$$P_1 = \left\{ \begin{array}{ll} \# & \\ S & \# \end{array} \rightarrow \begin{array}{ll} U & \\ a & R \end{array} \right\},\ P_2 = \left\{ \begin{array}{l} \# \\ U \end{array} \rightarrow \begin{array}{l} U \\ a \end{array} \right\},\ P_3 = \{R\# \rightarrow aR\},$$

$P_4 = \{U \rightarrow a\}$, $P_5 = \{R \rightarrow a\}$,

with f and h being any of the derivation modes t, $*$, $= 1$, ≥ 1, or $\leq k$ for $k \geq 1$, and g being any of the derivation modes $*$, $= 1$, ≥ 1, or $\leq k$ for $k \geq 1$. By taking $\geq k$ (with $k \geq 2$) for g we obtain the sublanguage of L_1 containing all right angles the arms of which have a length of at least $k + 1$. □

Example 3. We now construct the array language L_H of H shapes with the horizontal line at the middle of the vertical ones. (This language is used in [24] as an example of a set of pictures that can be generated by an indexed right-linear matrix array grammar, but not by a context-free matrix array grammar.) This array language L_H can be generated by the internally hybrid array grammar system with prescribed teams of context-free array productions

$G_3 = (\{S_1, S_2, A_1, A_2, A_3, A_4\}, \#, \{a\}, S, (R_3, T_3))$,

$R_3 = \{P_j \mid 1 \leq j \leq 8\}$, $T_3 = \{(Q_1, \{t\}), (Q_2, \{t\}), (Q_3, \{t\})\}$,

$Q_1 = \{P_1\}$, $Q_2 = \{P_2, P_3, P_4\}$, $Q_3 = \{P_5, P_6, P_7, P_8\}$,

$$P_1 = \left\{ \begin{array}{ll} \# & \\ S_1 & \# \\ \# & \end{array} \rightarrow \begin{array}{ll} A_1 & \\ a & S_2 \\ A_2 & \end{array} \right\},\ P_2 = \left\{ S_2\# \rightarrow a S_2,\ \begin{array}{ll} & \# \\ S_2 & \# \\ & \# \end{array} \rightarrow \begin{array}{ll} & A_3 \\ a & a \\ & A_4 \end{array} \right\},$$

$P_3 = \{A_1 \rightarrow A_1\}$, $P_4 = \{A_2 \rightarrow A_2\}$,

$$P_5 = \left\{ \begin{array}{l} \# \\ A_1 \end{array} \rightarrow \begin{array}{l} A_1 \\ a \end{array},\ A_1 \rightarrow a \right\},\ P_6 = \left\{ \begin{array}{l} A_2 \\ \# \end{array} \rightarrow \begin{array}{l} a \\ A_2 \end{array},\ A_2 \rightarrow a \right\},$$

$$P_7 = \left\{ \begin{array}{l} \# \\ A_3 \end{array} \rightarrow \begin{array}{l} A_3 \\ a \end{array},\ A_3 \rightarrow a \right\},\ P_8 = \left\{ \begin{array}{l} A_4 \\ \# \end{array} \rightarrow \begin{array}{l} a \\ A_4 \end{array},\ A_4 \rightarrow a \right\}.$$

We obtain $L(G_3) = L_H$. Here is an example of a derivation in G_3:

$$S_1 \Longrightarrow^{\{t\}}_{Q_1} \begin{array}{c} A_1 \\ a \\ A_2 \end{array} S_2 \Longrightarrow^{\{t\}}_{Q_2} \begin{array}{cccc} A_1 & & & A_3 \\ a & a & a & a \\ A_2 & & & A_4 \end{array} \Longrightarrow^{\{t\}}_{Q_3} \begin{array}{cccc} a & & & a \\ a & & & a \\ a & a & a & a \\ a & & & a \\ a & & & a \end{array}$$

If the final array productions $A_i \to a$ are not applied synchronously taken from P_{4+i}, $1 \leq i \leq 4$, then the derivation in G_3 is blocked without any possibility to yield a terminal array any more.

As all the derivation modes equal the maximal derivation mode, the previous construction not only shows that $L_H \in PTIH(CFA) \setminus CFA$, but also that $L_H \in PT(CFA,t) \setminus CFA$, because $L(G_3', t) = L_H$ for the array grammar system with prescribed teams $G_3' = (\{S_1, S_2, A_1, A_2, A_3, A_4\}, \#, \{a\}, S, (R_3, T_3'))$, $T_3' = \{Q_1, Q_2, Q_3\}$. By taking $T_3 = \{(Q_1, \{t\}), (Q_2, \{t, \geq k\}), (Q_3, \{t, \geq m\})\}$ in G_3 we can generate only those arrays of H shapes where the length of the horizontal line is *at least* $k+3$ and the length of the vertical lines is *at least* $2m+3$. With $(Q_2, \{t, \leq k\})$ instead, the length of the horizontal line is *restricted to be at most* $k+3$. By taking $(Q_2, \{t, \geq k, \leq l\})$, $k \leq l$, the length of the vertical lines is determined to be an uneven number between $2k+3$ and $2l+3$. □

4 Analysing systems

In pattern recognition, we are more interested in analysing devices than in generating mechanisms as they were introduced in the preceding section. In order to avoid the inherent non-determinism that usually arises when turning from generating grammars to analysing or accepting grammars ([2]), we propose another interpretation of the generating array grammars described above in order to obtain analysing mechanisms, which, for example, can be used for pattern recognition, e.g., for the recognition of (handwritten) characters in a similar way as described for array grammars in [25] and for programmed array grammars in [10]:

Given G, an internally hybrid array grammar system with prescribed teams of type CFA, we only allow one further derivation step with a selected team, if the following conditions hold:

1. The number of non-terminal symbols in the current sentential form equals the number of components of the team, i.e., by applying the team all the non-terminal symbols appearing in the current array are derived in parallel.
2. The shape of the sentential form after this derivation step is part of the shape of the originally given array and, moreover, at each position where we already find a terminal symbol in this sentential form, this symbol must coincide with the corresponding symbol at this position in the originally given array.

A given array is said to be *accepted by G* if there is a derivation sequence in G obeying the rules above and finally leading to a terminal array coinciding with the originally given array.

The first condition guarantees that during a derivation the number of non-terminal symbols in an underlying sentential form is bounded by the maximal number of components of a team. According to these definitions, all the examples of internally hybrid array grammar systems with prescribed teams of type CFA constructed in the preceding section can also be interpreted as analysing devices in the sense defined above, thus accepting the same array languages as they generate. If we allow #-context-free array productions instead of context-free ones only just in order to be able to deal with possible gaps in the pixel images of patterns to be recognized, we have to use a weaker condition

2'. At each position where we already find a terminal symbol in the underlying sentential form, this symbol must coincide with the corresponding symbol at this position in the originally given array.

This weaker condition means that the non-terminal symbols of the current sentential form may also occupy positions that are only occupied by the blank symbol in the originally given array.

How can such grammars be implemented as character recognizers?

We sketch our approach by using Example 1:

As we have to start with the context-free start production in P_1 with the right-hand side $\begin{matrix} U & \\ a & R \end{matrix}$ we have to search for a sub-pattern $\begin{matrix} a & \\ a & a \end{matrix}$. In case an element of L_1 is given, the starting point is unique, and it can be found using, e.g., the 2-dimensional Knuth-Morris-Pratt algorithm presented by Bird [1], also refer to [15]. The nonterminals U and R correspond to heads of a parallel machine with common memory (or simply to pointers in case of a sequential implementation). There are obvious implementations of our algorithm on PRAMs and related machine models [16]. Because of condition 1, we only have a bounded parallelism in the recognition process, which is indeed a very realistic assumption. The recognition process proceeds with applying Q_2 until one of the heads sees a blank symbol "ahead". Finally Q_3 has to be applied once. If then there are no unread symbols left in the given pattern, the pattern belongs to L_1.

In a similar manner, it is possible to construct a deterministic parallel recognizer for H shapes from grammar G_3. Of course, in general nondeterministic parallel recognizers are obtained (as sketched in [19] for parallel array grammars). It would be interesting also from a theoretical point of view to investigate further the determinism restriction introduced in this section. In addition, several such parallel character recognizers may work in parallel on the same given pattern.

As the lines appearing in pixel images of patterns like handwritten characters may have deviations, we also have to allow additional rules in the component of a team describing such a line, e.g., we can replace the component

$$P_2 = \left\{ S_2\,\# \rightarrow a\,S_2,\ \ \begin{matrix} & \# \\ S_2 & \# \\ & \# \end{matrix} \rightarrow \begin{matrix} & A_3 \\ a & a \\ & A_4 \end{matrix} \right\}$$

in the internally hybrid array grammar system with prescribed teams G_3 from Example 3 by

$$P_2 = \left\{ S_2\#\rightarrow a\,S_2,\ \begin{matrix} S_2 & \\ & \# \end{matrix} \rightarrow \begin{matrix} a & \\ & S_2 \end{matrix},\ \begin{matrix} & \# \\ S_2 & \end{matrix} \rightarrow \begin{matrix} & S_2 \\ a & \end{matrix},\ \begin{matrix} & \# \\ S_2 & \# \\ & \# \end{matrix} \rightarrow \begin{matrix} & A_3 \\ a & a \\ & A_4 \end{matrix} \right\}$$

where the array productions $\begin{matrix} S_2 & \\ & \# \end{matrix} \rightarrow \begin{matrix} a & \\ & S_2 \end{matrix}$ and $\begin{matrix} & \# \\ S_2 & \end{matrix} \rightarrow \begin{matrix} & S_2 \\ a & \end{matrix}$ allow deviations of the horizontal line of the character H. In a similar way, array productions can be added to P_5, P_6, P_7, P_8, in order to allow deviations in the vertical lines forming the H. Analogously to [10], [25], these deviations can be summed up to an error or distance measure in an additional attribute vector assigned to the sentential forms, and finally remaining pixels not covered by a derivation can also be counted as errors increasing this distance measure that is a measure for the distance between a given pattern and an ideal one. For lack of space we cannot go into the details of these constructions, because in this paper our emphasis is lying on how generating devices can be used as analysing mechanisms in an efficient way using the advantages of bounded parallelism.

5 Summary

In this paper, we have introduced internally hybrid array grammar systems with prescribed teams not only as generating devices for various (non-context-free) sets of rather complex pictures, but also as syntactic analysing mechanisms for pixel images of patterns like (handwritten) characters. The special interpretation of the application of the components of a team in such an array grammar system forces the components to work in parallel on all the non-terminal symbols of the current sentential form which allows for efficient implementations avoiding the difficulties usually arising with the inherent non-determinism in analysing grammars. Thorough practical investigations of all the variants of array grammar systems with prescribed teams introduced in this paper as well as a comparison of these variants with other methods (e.g., see [3], [10], [26]) remain for future research.

Acknowledgements. We are very grateful to Gheorghe Păun and Markus Holzer for many interesting and fruitful discussions on different variants of team grammar systems as well as on many specific topics of array grammars.

References

1. R. S. Bird, Two dimensional pattern matching, *IPL* **6** (1977) pp. 168 – 170.
2. H. Bordihn, H. Fernau, Accepting grammars and systems via context condition grammars, *Journal of Automata, Languages and Combinatorics* **1** (1996).
3. C. H. Chen, L. F. Pau, P. S.-P. Wang (eds.), *Handbook of Pattern Recognition & Computer Vision*, World Scientific Publ., Singapore, 1993.

4. C. R. Cook, P. S.-P. Wang, A Chomsky hierarchy of isotonic array grammars and languages, *Computer Graphics and Image Processing* **8** (1978) pp. 144 – 152.
5. E. Csuhaj-Varjú, J. Dassow, On cooperating distributed grammar systems, *EIK* **26** (1990) pp. 49 – 63.
6. E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems.* (Gordon and Breach, 1994).
7. J. Dassow, R. Freund, Gh. Păun, Cooperating array grammar systems, *Int. Journ. of Pattern Recognition and Artificial Intelligence* **9**, 6 (1995) pp. 1 – 25.
8. J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory.* (Springer, Berlin, 1989).
9. H. Fernau, R. Freund, M. Holzer, "External versus internal hybridization for cooperating distributed grammar systems", Technical Report 185-2/FR-1/96, Technische Universität Wien, 1996.
10. R. Freund, "Syntactic recognition of handwritten characters by programmed array grammars with attribute vectors", Seventh International Conference on Image Analysis and Processing, Bari, Italy, in: *Progress in Image Analysis and Processing III* (ed. S. Impedovo, World Scientific Publ., Singapore, 1993) pp. 357 – 364.
11. R. Freund, Gh. Păun, One-dimensional matrix array grammars, *EIK* **29** (1993) pp. 1 – 18.
12. R. Freund, "Control mechanisms on #-context-free array grammars", in: *Mathematical Aspects of Natural and Formal Languages.* (ed. Gh. Păun, World Sci. Publ., Singapore, 1994) pp. 97 – 136.
13. R. Freund, "Array grammars with prescribed teams of array productions", Developments in Language Theory'95, Magdeburg, 1995.
14. L. Kari, A. Mateescu, Gh. Păun, A. Salomaa, "Teams in cooperating grammar systems", *Journal of Experimental and Theoretical AI* 7 (1995) pp. 347–359.
15. R. M. Karp, R. E. Miller, A. L. Rosenberg, "Rapid identification of repeated patterns in strings, trees and arrays", in: *Proc. 4th Annual ACM Symposium on Theory of Computing* (1972) pp. 125 – 136.
16. J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science; Volume A: Algorithms and Complexity.* (The MIT Press/Elsevier, 1990).
17. R. Meersman, G. Rozenberg, "Cooperating grammar systems", in: *Proc. MFCS '78*, LNCS **64** (Springer, 1978) pp. 364 – 374.
18. D. L. Milgram, A. Rosenfeld, "Array automata and array grammars", in: *Inform. Processing '71*, (North-Holland, 1972) pp. 69 – 74.
19. A. Nakamura, K. Aizawa, Acceptors for isometric parallel context-free array languages, *IPL* **13** (1981) pp. 182 – 186.
20. Gh. Păun, G. Rozenberg, Prescribed teams of grammars, *Acta Informatica* **31**, 6 (1994) pp. 525 – 537.
21. A. Rosenfeld, *Picture Languages.* (Academic Press, Reading, MA, 1979).
22. A. Salomaa, *Formal Languages.* (Academic Press, Reading, MA, 1973).
23. G. Siromoney, R. Siromoney, K. Krithivasan, Abstract families of matrices and picture languages, *Computer Graphics and Image Processing* 1 (1972) pp. 234 – 307.
24. K. G. Subramanian, L. Revathi, R. Siromoney, "Siromoney array grammars and applications", pp. 55 – 73, in: [26].
25. P. S.-P. Wang, An application of array grammars to clustering analysis for syntactic patterns, *Pattern Recognition* **17**, 4 (1984) pp. 441 – 451.
26. P. S.-P. Wang (ed.), *Array Grammars, Patterns and Recognizers*, World Scientific Series in Computer Science **18**, World Scientific Publ., Singapore, 1989.