Contextual Array Grammars with Matrix and Regular Control

Henning Fernau, Rudolf Freund, Rani Siromoney, K. G. Subramanian

Contextual Array Grammars with Matrix and Regular Control

Henning Fernau [1], Rudolf Freund [2],
Rani Siromoney [3], and K.G. Subramanian [4]

[1] Universität Trier, FB 4 – Abteilung Informatikwissenschaften,
D-54296 Trier, Germany; E-mail: fernau@uni-trier.de
[2] Technische Universität Wien, Institut für Computersprachen,
A-1040 Wien, Austria; E-mail: rudi@emcc.at
[3] Chennai Mathematical Institute,
Kelambakkam 603103, India; E-mail: siromoney@cmi.ac.in
[4] Department of Mathematics and Computer Science, Faculty of Science,
Liverpool Hope University Liverpool, L16 9JD UK
E-mail: kgsmani1948@yahoo.com

Abstract. We investigate the computational power of d-dimensional contextual array grammars with matrix control and regular control languages. For $d \geq 2$, d-dimensional contextual array grammars are less powerful than matrix contextual array grammars, which themselves are less powerful than contextual array grammars with regular control languages. Yet in the 1-dimensional case, for a one-letter alphabet, the family of 1-dimensional array languages generated by contextual array grammars with regular control languages coincides with the family of regular 1-dimensional array languages, whereas for alphabets with more than one letter, we obtain the array images of the linear languages.

1 Introduction

Contextual string grammars were introduced by Solomon Marcus [14] with motivations arising from descriptive linguistics. A contextual string grammar consists of a finite set of strings (*axioms*) and a finite set of productions, which are pairs (s, c) where s is a string, the *selector,* and c is the *context*, i. e., a pair of strings, $c = (u, v)$, over the alphabet under consideration. Starting from an axiom, contexts iteratively are added as is indicated by the productions, which yields new strings. In contrast to usual sequential string grammars in the Chomsky hierarchy (e.g., see [20]), these contextual string grammars are pure grammars where new strings are not obtained by rewriting, but by adjoining strings. Several classes of contextual grammars have been introduced and investigated, e.g., see [3] and [17] for surveys on the area.

The idea of contextual productions then was also introduced for multi-dimensional array grammars, for instance, to carry over ideas from formal languages to the processing of digital images. In the area of two-dimensional picture

languages, e.g., see [12, 16, 18, 19], different kinds of array grammars, both isometric and non-isometric ones, have been proposed, motivated by many applications such as character recognition (also confer [4]), cluster analysis of patterns, and so on. Isometric contextual array grammars were introduced in [11].

Regulated rewriting with different control mechanisms has been studied extensively especially for string grammars (e.g., see [2]), for example, grammars with control languages and matrix grammars, but then also for array grammars, e.g., see [9]. Non-isometric contextual array grammars (with regulation) were considered in [8, 7, 13].

In this paper we consider matrix contextual array grammars and contextual array grammars with regular control and examine their generative power. In the 1-dimensional case, we obtain special results: the family of 1-dimensional array languages generated by contextual array grammars with regular control languages coincides with the family of regular 1-dimensional array languages over unary alphabets and with array images of the linear languages over alphabets with more than one letter; already for binary alphabets, regular control is strictly more powerful than matrix control, a phenomenon rarely observed in regulated rewriting (confer [10]).

2 Definitions

For notions and notations as well as results related to formal language theory we refer to books like [2]. The families of λ-free (λ denotes the empty string) regular string languages (over a k-letter alphabet) is denoted by $\mathcal{L}(REG)$ ($\mathcal{L}\left(REG^k\right)$). For the definitions and notations for arrays and sequential array grammars we refer to [9, 18, 22].

Let $\mathbb{Z}$ be the set of integers and $\mathbb{N}$ be the set of positive integers. Let $d \in \mathbb{N}$. A *d-dimensional array* $\mathcal{A}$ over the alphabet V is a mapping $\mathcal{A} : \mathbb{Z}^d \to V \cup \{\#\}$ where $shape(\mathcal{A}) = \left\{v \in \mathbb{Z}^d \mid \mathcal{A}(v) \neq \#\right\}$ is finite and $\# \notin V$ is called the *blank symbol*. We usually write $\mathcal{A} = \{(v, \mathcal{A}(v)) \mid v \in shape(\mathcal{A})\}$. The set of all d-dimensional arrays over V is denoted by V^{*d}. The *empty array* Λ_d in V^{*d} satisfies $shape(\Lambda_d) = \emptyset$. Moreover, we define $V^{+d} = V^{*d} \setminus \{\Lambda_d\}$.

Let $v \in \mathbb{Z}^d$. Then the *(linear) translation* $\tau_v : \mathbb{Z}^d \to \mathbb{Z}^d$ is defined by $\tau_v(w) = w + v$ for all $w \in \mathbb{Z}^d$, and for any array $\mathcal{A} \in V^{*d}$ we define $\tau_v(\mathcal{A})$, the corresponding d-dimensional array translated by v, by $(\tau_v(\mathcal{A}))(w) = \mathcal{A}(w - v)$ for all $w \in \mathbb{Z}^d$. The vector $(0, ..., 0) \in \mathbb{Z}^d$ is denoted by Ω_d.

Usually (see [18]) arrays are regarded as equivalence classes of arrays with respect to linear translations. The equivalence class $[\mathcal{A}]$ of an array $\mathcal{A} \in V^{*d}$ satisfies $[\mathcal{A}] = \left\{\mathcal{B} \in V^{*d} \mid \mathcal{B} = \tau_v(\mathcal{A}) \text{ for some } v \in \mathbb{Z}^d\right\}$. The set of all equivalence classes of d-dimensional arrays over V with respect to linear translations is denoted by $\left[V^{*d}\right]$, and this bracket notation carries over to classes of array languages, as well.

As many results for d-dimensional arrays for a specific d can be taken over immediately for higher dimensions, we introduce special notions:

Let $n, m \in \mathbb{N}$ with $n \leq m$. For $n < m$, *the natural embedding* $i_{n,m} : \mathbb{Z}^n \to \mathbb{Z}^m$ is defined by $i_{n,m}(v) = (v, \Omega_{m-n})$ for all $v \in \mathbb{Z}^n$; for $n = m$ we define $i_{n,n} : \mathbb{Z}^n \to \mathbb{Z}^n$ by $i_{n,n}(v) = v$ for all $v \in \mathbb{Z}^n$. To an n-dimensional array $\mathcal{A} \in V^{+n}$ with $\mathcal{A} = \{(v, \mathcal{A}(v)) \mid v \in \mathit{shape}(\mathcal{A})\}$ we assign the m-dimensional array $i_{n,m}(\mathcal{A}) = \{(i_{n,m}(v), \mathcal{A}(v)) \mid v \in \mathit{shape}(\mathcal{A})\}$.

We can use the well-known graph-theoretic notion of a connected graph to define connected arrays. Let W be a non-empty finite subset of $\mathbb{Z}^d$. We associate a graph $g(W)$ to W with vertex set W and an edge between $v, w \in W$ if and only if $\|v - w\| = 1$, where the norm $\|u\|$ of a vector $u \in \mathbb{Z}^d$, $u = (u(1), ..., u(d))$, is defined by $\|u\| = \max\{|u(i)| \mid 1 \leq i \leq d\}$. Then W is said to be *connected* if $g(W)$ is connected. There is a natural bijection between the (equivalence classes of) 1-dimensional connected arrays and strings: for any equivalence class of 1-dimensional arrays $\mathcal{A} = [\{((i-1), a_i) \mid 1 \leq i \leq n\}]$ we define its *string image* as $str(\mathcal{A}) = a_1 \dots a_n$; the string $w = a_1 \dots a_n$ can be interpreted as the array $arr(w) = \{\{((i-1), a_i)\} \mid 1 \leq i \leq n\}$. In the standard way, these notions are extended from strings and arrays to sets of strings and arrays.

Example 1. Consider the language L_1 of connected 2-dimensional arrays

$$L_1 = \left\{ \{((0,i), a) \mid 0 \leq i \leq n\} \cup \{((j,0), a) \mid 1 \leq j \leq m\} \;\middle|\; n, m \in \mathbb{N} \right\}.$$

$$\begin{array}{l} a \\ a \\ a \\ a\ a\ a\ a\ a \end{array}$$

An example of these L-shaped arrays (for $n = 3$ and $m = 4$) from $[L_1]$ can be depicted as shown on the left. Observe that both arms of these arrays can have arbitrary lengths. □

Definition 1. *A regular* d-dimensional array grammar *is specified as* $G = (d, N, T, \#, P, \{(v_S, S\}))$ *where* N *is the alphabet of* non-terminal symbols, T *is the alphabet of* terminal symbols, $N \cap T = \emptyset$, $\# \notin N \cup T$; P *is a finite non-empty set of regular* d*-dimensional array productions over* $N \cup T$, *as well as* $v_S \in \mathbb{Z}^d$ *and* $S \in N$ *is the* start symbol. *A regular* d -dimensional array production *either is of the form* $A \to b$, $A \in N$, $b \in T$, *or* $Av\# \to bC$, $A, C \in N$, $b \in T$, $v \in \mathbb{Z}^d$ *with* $\|v\| = 1$. *The application of* $A \to b$ *means replacing* A *by* b *in a given array.* $Av\# \to bC$ *can be applied if in the underlying array we find a position* u *occupied by* A *and a blank symbol at position* $u + v$; A *then is replaced by* b, *and* $\#$ *by* C. *The array language generated by* G *is the set of all* d*-dimensional arrays derivable from the* initial array $\{(v_S, S)\}$. *The family of* Λ*-free* d*-dimensional array languages (of equivalence classes) of arrays over a* k*-letter alphabet generated by regular* d*-dimensional array grammars is denoted by* $\mathcal{L}\left(d\text{-}REGA^k\right)$ *(*$\left[\mathcal{L}\left(d\text{-}REGA^k\right)\right]$*). For arbitrary alphabets, we omit the superscript* k.

The following results for 1-dimensional array languages are folklore:

Theorem 1. *For all* $k \geq 1$, $\left[\mathcal{L}\left(1\text{-}REGA^k\right)\right] = \left[arr\left(\mathcal{L}\left(REG^k\right)\right)\right]$ *and* $str\left(\left[\mathcal{L}\left(1\text{-}REGA^k\right)\right]\right) = \mathcal{L}\left(REG^k\right)$.

Let us mention the close similarities of the work of 1-dimensional regular array grammars and Lindenmayer systems with apical growth [21]. Another similar development can be found within Watson-Crick systems [15].

3 Contextual Array Grammars

We now turn our attention to the main variants of contextual array grammars considered in this paper.

Definition 2. *A* d-dimensional contextual array grammar *($d \in \mathbb{N}$) is a construct* $G = (d, V, \#, P, A)$ *where* V *is an alphabet not containing the blank symbol* $\#$, A *is a finite set of axioms, i. e., of* d*-dimensional arrays in* V^{+d}, *and* P *is a finite set of rules of the form* $(U_\alpha, \alpha, U_\beta, \beta)$ *where*

(i) $U_\alpha, U_\beta \subseteq \mathbb{Z}^d$, $U_\alpha \cap U_\beta = \emptyset$, *and* U_α, U_β *are finite and non-empty;*
(ii) $\alpha : U_\alpha \to V$ *and* $\beta : U_\beta \to V$.

(U_α, α) *corresponds with the* selector *and* (U_β, β) *with the* context *of the production* $(U_\alpha, \alpha, U_\beta, \beta)$; U_α *is called the* selector area, *and* U_β *is the* context area. *As the sets* U_α *and* U_β *are uniquely determined by* α *and* β, *we will also represent* $(U_\alpha, \alpha, U_\beta, \beta)$ *by* (α, β) *only.*

For $\mathcal{C}_1, \mathcal{C}_2 \in V^{+d}$ *we say that* $\mathcal{C}_2$ *is directly derivable from* $\mathcal{C}_1$ *by the contextual array production* $p \in P$, $p = (U_\alpha, \alpha, U_\beta, \beta)$ *(we write* $\mathcal{C}_1 \Longrightarrow_p \mathcal{C}_2$*), if there exists a vector* $v \in \mathbb{Z}^d$ *such that*

- $\mathcal{C}_1(w) = \mathcal{C}_2(w) = \alpha(\tau_{-v}(w))$ *for all* $w \in \tau_v(U_\alpha)$,
- $\mathcal{C}_1(w) = \#$ *for all* $w \in \tau_v(U_\beta)$,
- $\mathcal{C}_2(w) = \beta(\tau_{-v}(w))$ *for all* $w \in \tau_v(U_\beta)$,
- $\mathcal{C}_1(w) = \mathcal{C}_2(w)$ *for all* $w \in \mathbb{Z}^d \setminus \tau_v(U_\alpha \cup U_\beta)$.

Hence, if in $\mathcal{C}_1$ *we find a subpattern that corresponds with the selector* α *and only blank symbols at the places corresponding with* β, *we can add the context* β *thus obtaining* $\mathcal{C}_2$. *For every* $\mathcal{B}_1, \mathcal{B}_2 \in \left[V^{+d}\right]$ *we say that* $\mathcal{B}_2$ *is directly derivable from* $\mathcal{B}_1$ *by the contextual array production* $p \in P$, $p = (U_\alpha, \alpha, U_\beta, \beta)$, *denoted* $\mathcal{B}_1 \Longrightarrow_p \mathcal{B}_2$, *if and only if* $\mathcal{C}_1 \Longrightarrow_p \mathcal{C}_2$ *for some* $\mathcal{C}_1 \in \mathcal{B}_1$ *and* $\mathcal{C}_2 \in \mathcal{B}_2$. $\mathcal{C}_1 \Longrightarrow_G \mathcal{C}_2$ *(*$\mathcal{B}_1 \Longrightarrow_G \mathcal{B}_2$*) means that* $\mathcal{C}_1 \Longrightarrow_p \mathcal{C}_2$ *(*$\mathcal{B}_1 \Longrightarrow_p \mathcal{B}_2$*) for some* $p \in P$.

The array language generated by G *is defined as*

$$L(G) = \left\{\mathcal{C} \in V^{+d} \mid \mathcal{A} \Longrightarrow_G^* \mathcal{C} \text{ for some } \mathcal{A} \in A\right\}.$$

The special type of d*-dimensional contextual array grammars where axioms are connected and rule applications preserve connectedness is denoted by* d-$ContA$, *the corresponding family of* d*-dimensional array languages by* $\mathcal{L}(d\text{-}ContA)$; *by* $\mathcal{L}\left(d\text{-}ContA^k\right)$ *we denote the corresponding family of* d*-dimensional array languages over a* k*-letter alphabet.*

Remark 1. As we mostly are interested in (families of) equivalence classes of arrays, a d-dimensional contextual array grammar $[G]$ for generating $[L]$ for $L \in \mathcal{L}(d\text{-}ContA)$ being generated by a d-dimensional contextual array grammar $G = (d, V, \#, P, A)$ with $A = \{\mathcal{A}_i \mid 1 \leq i \leq n\}$ will be specified by writing $[G] = (d, V, \#, P, A')$ where $A' = \{\mathcal{A}'_i \mid 1 \leq i \leq n\}$ such that $\mathcal{A}'_i \in [\mathcal{A}_i]$, $1 \leq i \leq n$, which means specifying an axiom $\mathcal{A}_i$ by one array from $[\mathcal{A}_i]$.

Example 2. Any finite d-dimensional array language of connected arrays $L \subset T^{+d}$ is in $\mathcal{L}(d\text{-}ContA)$ as $L = L(G_L)$ where $G_L = (d, T, \#, \emptyset, L)$. □

Example 3. We now show how the language L_1 from Example 1 can be generated by the contextual array grammar G_1, i.e., $L_1 \in \mathcal{L}\left(2\text{-}ContA^1\right)$: $G_1 = (2, \{a\}, \#, P_1, \{\mathcal{A}_1\})$ where $\mathcal{A}_1 = \{((0,0), a), ((0,1), a), ((1,0), a)\}$ is the only axiom and P_1 consists of the two productions p_u and p_r:

$$p_u = (\{(0,0), (0,1)\}, \{((0,0), a), ((0,1), a)\}, \{(0,2)\}, \{((0,2), a)\}),$$
$$p_r = (\{(0,0), (1,0)\}, \{((0,0), a), ((1,0), a)\}, \{(2,0)\}, \{((2,0), a)\}).$$

As the selector area U_α and the context area U_β in a contextual array production of the form $(U_\alpha, \alpha, U_\beta, \beta)$ are disjoint, both α and β can be represented within only one pattern, i. e., p_u and p_r can be represented in a more depictive way by the patterns shown on the right (the symbols of the selector are enclosed in boxes).

$$p_u = \begin{array}{c} a \\ \boxed{a} \\ \boxed{a} \end{array}, \quad p_r = \boxed{a}\,\boxed{a}\,a\,.$$

The example of the L-shaped array for $n = 3$ and $m = 4$ then is generated by twice applying rule p_u and three times applying rule p_r, in any order. We also observe that every intermediate array obtained by applying these rules is in L_1, too. Obviously, by the definition of equivalence classes of arrays, we also have $[L(G_1)] = [L_1] \in \left[\mathcal{L}\left(2\text{-}ContA^1\right)\right]$.

$[\mathcal{A}_1]$ can be described in a more depictive way by $\begin{smallmatrix} a \\ a\ a \end{smallmatrix}$, i.e., the contextual array grammar $[G_1]$ for $[L(G_1)]$ can also be written as $[G_1] = \left(2, \{a\}, \#, P_1, \left\{\begin{smallmatrix} a \\ a\ a \end{smallmatrix}\right\}\right)$ (see Remark 1). In the following, the axiom(s) often will just be given in such a pictorial variant. □

Example 4. For the singleton language $L_\perp = \left\{\begin{smallmatrix} & a & \\ & a & \\ a & a & a & a & a \end{smallmatrix}\right\} \subset \left[\{a\}^{+2}\right]$, we have $L_\perp \in [\mathcal{L}(2\text{-}ContA)] \setminus [\mathcal{L}(2\text{-}REGA)]$. As we can take $L_\perp$ (as any finite language) as a set of axioms, containment in $[\mathcal{L}(2\text{-}ContA)]$ is clear. Conversely, any regular array grammar has to scan the non-blank symbols of the array A, which is impossible, as the underlying graph $g(shape(A))$ is not Hamiltonian. □

Theorem 2. $\left[\mathcal{L}\left(1\text{-}REGA^1\right)\right] \subseteq \left[\mathcal{L}\left(1\text{-}ContA^1\right)\right]$.

Proof. Due to the results from Theorem 1, it only remains to show that $\left[arr\left(\mathcal{L}\left(REG^1\right)\right)\right] \subseteq \left[\mathcal{L}\left(1\text{-}ContA^1\right)\right]$.

From [1, Theorem 4.4], we deduce that any infinite language $L \subseteq \{a\}^+$ in $\mathcal{L}\left(REG^1\right)$ can be written in the form $L = \{a^{s_1}, a^{s_2}, \ldots, a^{s_t}\} \cup \bigcup_{i=1}^{m} \{a^{k \cdot n + d_i} \mid n \geq 0\}$ for some numbers $k, k \leq d_1 < d_2 < ... < d_m < 2k, 0 \leq s_1 < s_2 < ... < s_t < k$. The 1-dimensional contextual array grammar now is constructed using a context of length k and putting the words a^{s_j}, $1 \leq j \leq t$, and a^{d_i}, $1 \leq i \leq m$, into the set of axioms, i.e., we define the 1-dimensional contextual array grammar $G(L) = (1, \{a\}, \#, P, A)$ with $A = \{arr(a^{s_j}) \mid 1 \leq j \leq t\} \cup \left\{arr\left(a^{d_i}\right) \mid 1 \leq i \leq m\right\}$ and

$P = \left\{ \boxed{a}^k a^k \right\}$. Obviously, $[L(G(L))] = [arr(L)]$. The 1-dimensional contextual array grammar $[G(L)]$ for $[L(G(L))]$ can also be written as $[G(L)] = (1, \{a\}, \#, P, A')$ with $A' = \{a^{s_j} \mid 1 \leq j \leq t\} \cup \left\{a^{d_i} \mid 1 \leq i \leq m\right\}$ (compare with Remark 1).

For the sake of completeness we mention that every finite array language $A = \{arr(a^{s_j}) \mid 1 \leq j \leq t\}$ is generated by the 1-dimensional contextual array grammar $G(L) = (1, \{a\}, \#, P, A)$ with $P = \emptyset$. □

Remark 2. Following the definition already given in [11], our d-dimensional extension of (external) contextual grammars only appends at one location, while external contextual string grammars as originally defined by Solomon Marcus, see [14], append to both ends of a string at the same time. This design decision has two main reasons. First, it is not quite clear what the d-dimensional counterpart of external contextual grammars would really mean: for instance, for $d = 2$, should we allow appending on both ends of a row or column at the same time, as we did in [8] for the case of non-isometric contextual array grammars? Or, should we rather append on 'all ends'? Obviously, this situation becomes even more intricate for higher dimensions. Yet second and even more important, appending at both sides of a string, i.e., a 1-dimensional array, in parallel can easily be simulated sequentially by a matrix with two components. It is therefore easy to see that in the 1-dimensional case, the string images of the arrays generated by contextual array grammars with matrix control exactly correspond with the string languages generated by external contextual string grammars. This means that for the regulated variants discussed in the following, any variant that can be conceivably defined for the d-dimensional analogue of external contextual grammars, in the 1-dimensional case should lead to the same results as the original variant of contextual array grammars defined in [11] and taken as the basis in this paper, too.

3.1 Matrix Contextual Array Grammars

Definition 3. *A d-dimensional matrix contextual array grammar is a pair $G_M = (G, M)$ where $G = (d, V, \#, P, A)$ is a d-dimensional contextual array grammar and M is a finite set of sequences, called* matrices*, of rules from P, i.e., each element of M is of the form $\langle p_1, \cdots, p_n \rangle$, $n \geq 1$, where $p_i \in P$ for $1 \leq i \leq n$. Derivations in a matrix contextual array grammar are defined as in a contextual array grammar except that a single derivation step now consists of the sequential application of the rules of one of the matrices in M, in the order in which the rules are given in the matrix. The array language generated by G_M is the set of all d-dimensional arrays which can be derived from any of the axioms in A. The family of d-dimensional array languages of arrays generated by d-dimensional matrix contextual array grammars (over a k-letter alphabet) is denoted by $\mathcal{L}(d\text{-}MContA)$ ($\mathcal{L}\left(d\text{-}MContA^k\right)$).*

Example 5. Consider the language L_2 of connected arrays given by

$$L_2 = \left\{ \left\{ ((0,0), a) \right\} \cup \left\{ ((0,i), a), ((i,0), a) \mid 1 \leq i \leq n \right\} \;\middle|\; n \in \mathbb{N} \right\},$$

which contains L-shaped arrays as L_1 from Example 1, but now with both arms having the same length. $L_2 \in \mathcal{L}\left(2\text{-}MContA^1\right)$, as it can be generated by the 2-dimensional matrix contextual array grammar $G_M = (G_1, M)$ where G_1 is the 2-dimensional contextual array grammar from Example 3 and $M = \{\langle p_u, p_r\rangle\}$. The only derivations possible in G'_M for $[L_2] \in \left[\mathcal{L}\left(2\text{-}MContA^1\right)\right]$ (see Remark 1) are:

$$\begin{array}{c} a \\ a\,a \end{array} \Longrightarrow_{G'_M} \begin{array}{c} a \\ a \\ a\,a\,a \end{array} \Longrightarrow_{G'_M} \begin{array}{c} a \\ a \\ a \\ a\,a\,a\,a \end{array} \Longrightarrow_{G'_M} \cdots$$

The single matrix $\langle p_u, p_r\rangle$, $p_u = \begin{array}{c} a \\ \boxed{a} \\ \boxed{a} \end{array}$, $p_r = \boxed{a}\,\boxed{a}\,a$, guarantees that both arms of the array grow in a synchronized way. □

Theorem 3. *For any $d \geq 2$ and any $k \geq 1$, we have $\mathcal{L}\left(d\text{-}ContA^k\right) \subsetneqq \mathcal{L}\left(d\text{-}MContA^k\right)$ and $\left[\mathcal{L}\left(d\text{-}ContA^k\right)\right] \subsetneqq \left[\mathcal{L}\left(d\text{-}MContA^k\right)\right]$.*

Proof. The inclusion $\mathcal{L}\left(d\text{-}ContA^k\right) \subseteq \mathcal{L}\left(d\text{-}MContA^k\right)$ and therefore also $\left[\mathcal{L}\left(d\text{-}ContA^k\right)\right] \subseteq \left[\mathcal{L}\left(d\text{-}MContA^k\right)\right]$ is obvious from general results for grammars working on various kinds of objects and with specific regulating mechanisms, see [10].

For showing the strictness of the inclusion, we prove that the array language L_2 from Example 5 cannot be generated by a 2-dimensional contextual array grammar; for dimensions $d > 2$, we just take $[i_{2,d}(L_2)]$.

Now assume we could find a 2-dimensional contextual array grammar $[G = (2, \{a\}, \#, P, A)]$ that generates $[L_2]$. As contextual grammars are pure grammars, $[A]$ is a finite subset of $[L(G)]$. As $[L(G)]$ is infinite, we would need an infinite number of rules to get $[L_2]$ which resembles the case of *external* contextual string grammars; in fact, as soon as the arms get long enough, we have to apply a rule which only grows the arm going up or only grows the arm going to the right, resulting in an array which contradicts the definition of $[L_2]$. It is obvious that we also have $[i_{2,d}(L_2)] \in \left[\mathcal{L}\left(d\text{-}MContA^k\right)\right] \setminus \left[\mathcal{L}\left(d\text{-}ContA^k\right)\right]$; this observation completes the proof. □

In the 1-dimensional case, the situation is different: as we shall prove later, see Theorem 6, $\left[\mathcal{L}\left(1\text{-}ContA^1\right)\right] = \left[\mathcal{L}\left(1\text{-}MContA^1\right)\right]$, but for $k \geq 2$, we still have $\left[\mathcal{L}\left(1\text{-}ContA^k\right)\right] \subsetneqq \left[\mathcal{L}\left(1\text{-}MContA^k\right)\right]$, as the following example shows.

Example 6. Consider the non-regular language $L_n = \{a^n b a^n \mid n \geq 1\}$. By Theorem 1, there cannot exist an array grammar G of type 1-$REGA^2$ such that $[L(G)] = [arr(L_n)]$. Even more, there is no 1-dimensional contextual array grammar for L_n. Namely, if this would be the case, then first observe that there must be rules that append something to the right, as well as to the left of the array, and this should be possible infinitely often. Otherwise, the sequence of context additions would happen (finally) only on one side, which means that this behavior can again be simulated by some regular array grammar, contradicting our previous reasoning. Hence, there must be a rule that contains a sequence of

a's as its selector, say, $arr(a^{r_s})$, and also a sequence of a's, say, $arr(a^{r_c})$ as its context in order to append a^{r_c} to the right of the current array, and likewise, there must be a rule that contains a sequence of a's as its selector, say, $arr(a^{\ell_s})$, and also a sequence of a's, say, $arr(a^{\ell_c})$ as its context in order to append a^{l_c} to the left of the current array. For sufficiently long arrays $arr(a^n ba^n)$, both rules can be applied, and arrays like $arr(a^n ba^{n+r_c})$ can generated that do not belong to L_n. Hence, $L_n \notin \mathcal{L}(1\text{-}ContA)$.

Yet for the 1-dimensional matrix contextual array grammar $[G_M] = (G_n, M_n)$ with $[G_n] = (1, \{a, b\}, \#, P, \{aba\})$ where $p_l = a\,\boxed{a}$, $p_r = \boxed{a}\,a$, and $M_n = \{\langle p_l, p_r\rangle\}$, we have $[L(G_n)] = [arr(L_n)]$. The single matrix $\langle p_l, p_r\rangle$ guarantees that the number of symbols a grows to the left and to the right in a synchronized way. □

In addition, the following example even yields that for any $k \geq 2$, $\left[\mathcal{L}\left(1\text{-}MContA^k\right)\right]$ is incomparable with $\left[\mathcal{L}\left(1\text{-}REGA^k\right)\right]$.

Example 7. Consider the regular string language $L_r = \{ba^n b \mid n \geq 1\}$. Due to Theorem 1, there exists an array grammar of type 1-$REGA^2$ G_r such that $[L(G_r)] = [arr(L_r)]$. Yet on the other hand, there cannot exist an array grammar of type 1-$MContA^2$ $[G]$ such that $L([G]) = [arr(L_r)]$, which can be proved by a simple pumping argument: The number of symbols a between the two symbols b can become arbitrarily large, but we only have a finite set of axioms A; as $[G]$ is a pure grammar, $[A] \subset [L]$; yet $[G]$ can only grow these arrays in an external way, i.e., by adding symbols on the left or on the right, but in this way we are not able to grow the number of symbols a in the middle. □

3.2 Contextual Array Grammars with Regular Control

Definition 4. *A d-dimensional contextual array grammar with regular control is a pair $G_C = (G, L)$ where $G = (d, V, \#, P, A)$ is a d-dimensional contextual array grammar and L is a regular string language over P. Derivations in a d-dimensional contextual array grammar with regular control are defined as in the contextual array grammar G except that in a successful derivation the sequence of applied rules has to be a word from L. The array language generated by G_C is the set of all d-dimensional arrays which can be derived from any of the axioms in A following a control word from L. The family of d-dimensional array languages of arrays generated by d-dimensional contextual array grammars over a k-letter alphabet with regular control is denoted by $\mathcal{L}((d\text{-}ContA, REG))$. The corresponding family of array languages of equivalence classes of arrays is denoted by using brackets in the notations.*

As a general result (following [10]) we can state:

Theorem 4. *For any $d \geq 1$ and any $k \geq 1$,*

$$\left[\mathcal{L}\left(d\text{-}ContA^k\right)\right] \subseteq \left[\mathcal{L}\left(d\text{-}MContA^k\right)\right] \subseteq \left[\mathcal{L}\left(\left(d\text{-}ContA^k, REG\right)\right)\right].$$

Example 8. Consider the regular string language $L_r = \{ba^n b \mid n \geq 1\}$ from Example 7. We have shown that $[arr\,(L_r)] \in \left[\mathcal{L}\left(1\text{-}REGA^2\right)\right] \setminus \left[\mathcal{L}\left(1\text{-}MContA^2\right)\right]$. Moreover, $[arr\,(L_r)] \in \left[\mathcal{L}\left(\left(1\text{-}ContA^1, REG\right)\right)\right] \setminus \left[\mathcal{L}\left(1\text{-}MContA^2\right)\right]$: Consider $G'_r = (G_r, C_r)$ with $G_r = (1, \{a, b\}, \#, P, \{arr\,(ba)\})$ and $P = \{p_{aa}, p_{ab}\}$ with $p_{aa} = \boxed{a}\, a$, and $p_{ab} = \boxed{a}\, b$, as well as $C_r = \{p_{aa}\}^* \{p_{ab}\}$. It is easy to see that $[L\,(G'_r)] = [arr\,(L_r)]$. □

Theorem 5. *For any $d \geq 1$ and any $k \geq 2$, we have:*

$$\left[\mathcal{L}\left(d\text{-}ContA^k\right)\right] \subsetneq \left[\mathcal{L}\left(d\text{-}MContA^k\right)\right] \subsetneq \left[\mathcal{L}\left(\left(d\text{-}ContA^k, REG\right)\right)\right].$$

Proof. The inclusions directly follow from Theorem 4. The strictness of the first inclusion follows from Example 6 by taking the non-regular string language $L_n = \{a^n b a^n \mid n \geq 1\}$. Then $[i_{1,d}\,(arr\,(L_n))] \in \left[\mathcal{L}\left(d\text{-}MContA^2\right)\right] \setminus \left[\mathcal{L}\left(d\text{-}ContA^k\right)\right]$. The strictness of the second inclusion follows from Example 8 by taking $[i_{1,d}\,(arr\,(L_r))]$. □

On the other hand, in the 1-dimensional case, the following theorem says that even with the regulating mechanisms of matrix control or regular control languages, with 1-dimensional contextual array grammars over a one-letter alphabet we cannot go beyond regularity, i.e., beyond $\left[\mathcal{L}\left(1\text{-}REGA^1\right)\right]$.

Theorem 6. $\left[\mathcal{L}\left(1\text{-}REGA^1\right)\right] =$
$\left[\mathcal{L}\left(1\text{-}ContA^1, REG\right)\right] = \left[\mathcal{L}\left(1\text{-}MContA^1\right)\right] = \left[\mathcal{L}\left(1\text{-}ContA^1\right)\right]$.

Proof. (Sketch) According to Theorems 4 and 2, we only have to show that $\left[\mathcal{L}\left(1\text{-}REGA^1\right)\right] \supseteq \left[\mathcal{L}\left(\left(1\text{-}ContA^1, REG\right)\right)\right]$. The main ideas of the corresponding technically non-trivial proof can be described as follows:

- Without loss of generality, right-hand sides of rules have the form $\boxed{a}^m a^n$.
- Context information is irrelevant for the unary 1-dimensional case, assuming that the set of axioms collects all arrays of sufficient size.
- The state information of the regular control is then encoded in the nonterminals of the regular array grammar. □

Allowing for more than one symbol, 1-dimensional contextual array grammars can generate exactly the array images of linear languages. The proof is based on the following normal form:

Lemma 1. *For any 1-dimensional contextual array grammar with regular control $G_C = (G, L)$, where $G = (1, V, \#, P, A)$, $L \subseteq P^*$, we can construct an equivalent 1-dimensional contextual array grammar with regular control $G'_C = (G', L')$ with $G' = (1, V, \#, P', A')$, $L' \subseteq P'^*$, such that for P' we have:*

- *All rules in P' are of the form $\boxed{a}\, b$ or $b\, \boxed{a}$ for some $a, b \in V$, i.e., we only have the minimal non-empty size of selectors and minimal contexts of size 1.*
- *If there is a rule of the form $\boxed{a}\, b \;/\; b\, \boxed{a}$ in P', then also all rules of the form $\boxed{c}\, b$ or $b\, \boxed{c}$ are in P', for any $c \in V$, i.e., the selector contents is irrelevant, only direction of growth of the array is important.*

The rules in this normal form nicely correspond with the operations of left and right insertions for strings, which operations together with regular control languages also characterize the family of linear languages.

Theorem 7. $[\mathcal{L}\,(1\text{-}ContA, REG)] = arr\,(\mathcal{L}\,(LIN))$.

Proof. (Sketch) The main ideas of the proof can be described as follows:

- Adding strings in a controlled way "on both ends" corresponds to applying linear rules, but in reverse order.
- The information about the finitely many selectors possible can be stored in the nonterminal; on the other hand, the nonterminal can be stored in the state of the finite automaton of the control language. □

For $d \geq 2$, i.e., in the case of at least two symbols, we can prove the incomparability of the families of array languages generated by contextual array grammars and those equipped with control mechanisms:

Theorem 8. *For any $d \geq 2$ and any $k \geq 1$, all the three families* $\left[\mathcal{L}\left(d\text{-}ContA^k\right)\right]$, $\left[\mathcal{L}\left(d\text{-}MContA^k\right)\right]$, *and* $\left[\mathcal{L}\left(\left(d\text{-}ContA^k, REG\right)\right)\right]$ *are incomparable with* $\left[\mathcal{L}\left(d\text{-}REGA^k\right)\right]$.

Proof. For the singleton language $L_\perp$ from Example 4, we have $i_{2,d}\,(L_\perp) \in \left(\left[\mathcal{L}\left(d\text{-}ContA^1\right)\right] \cap \left[\mathcal{L}\left(d\text{-}MContA^1\right)\right] \cap \left[\mathcal{L}\left(\left(d\text{-}ContA^1, REG\right)\right)\right]\right) \setminus \left[\mathcal{L}\left(d\text{-}REGA^1\right)\right]$. On the other hand, for L_r from Example 7 we have $i_{2,d}\left([arr\,(L_r)]\right) \in \left[\mathcal{L}\left(1\text{-}REGA^2\right)\right] \setminus \left(\left[\mathcal{L}\left(d\text{-}ContA^1\right)\right] \cup \left[\mathcal{L}\left(d\text{-}MContA^1\right)\right] \cup \left[\mathcal{L}\left(\left(d\text{-}ContA^1, REG\right)\right)\right]\right)$. Yet even for the case of one-letter alphabets we can find an array language of 2-dimensional arrays in $\left[\mathcal{L}\left(2\text{-}REGA^1\right)\right] \setminus \left[\mathcal{L}\left(\left(2\text{-}ContA^1, REG\right)\right)\right]$: we consider ⌊-shaped arrays with the left vertical line having a length being a multiple of 3 and the right vertical line having a length being a multiple of 5. These arrays can easily be generated by a regular array grammar by first generating the left vertical line from up to down, followed by the horizontal line, finally generating the right vertical line upwards. On the other hand, this set of 2-dimensional arrays cannot be generated by a contextual array grammar even when using regular control: as soon as the vertical lines have become long enough, we cannot distinguish any more between the left and the right one, so either the lengths will not necessarily fulfill the constraints of being a multiple of 3 and 5, respectively, any more, or even worse, the lines might even be prolonged below the horizontal line yielding arrays of the shape of an H. □

4 Decidability Questions

As the size of the arrays generated by contextual array grammars (even with any control mechanism) increases with every derivation step, the generated array languages are computable (i.e., recursive).

As an immediate consequence of Theorem 7, we obtain:

Corollary 1. *Emptiness is decidable for $\mathcal{L}\,(1\text{-}ContA, REG)$.*

Yet for higher dimensions, we obtain a completely different situation:

Theorem 9. *Emptiness is not decidable for* $\mathcal{L}\left(d\text{-}ContA^k, REG\right)$ *for* $d \geq 2$, *even for* $k = 1$.

Proof. (Sketch) As, for example, described in [5], the derivation carpet of a Turing machine can be described using 2-dimensional contextual array productions in the t-mode of derivation, i.e., a derivation only stops if no rule can be applied any more. The goal of only halting with specific conditions being fulfilled can also be obtained using suitable regular control languages, as we can require specific final rules to be applied. Hence, we will obtain a non-empty array language if and only if there is a derivation simulating the acceptance of a string by the given Turing machine. The proof given in [5] does not bound the number of symbols used. Yet m symbols can be encoded by $2 \times m$ rectangles with the k-th of these m symbols being encoded by leaving the k-th position in the second vertical line free, which then can be checked by the selector in the contextual array productions. Hence, simulating successful computations of the given Turing machine will result in the generation of $2k \times mn$ rectangles for accepting computations. □

5 Picture Generation

Another interesting topic is to consider the generation of geometric objects such as solid rectangles and squares, which has been used to exhibit the generative power of various array grammar variants. Both of them, i.e., the 2-dimensional array language L_{rect} of all solid rectangles of size $m \times n$, $m, n \geq 2$, made of a single symbol a and the 2-dimensional array language L_{square} of all solid squares of side length n, $n \geq 2$, made of a single symbol a are well-known to be in $\left[\mathcal{L}\left(2\text{-}REGA^1\right)\right]$, see [23], but as we are able to show they can also be generated by 2-dimensional contextual array grammars with regular control, i.e., $\{L_{rect}, L_{square}\} \subset \left[\mathcal{L}\left(\left(2\text{-}ContA^1, REG\right)\right)\right]$. We now only exhibit the contextual array grammar with regular control for the squares.

Example 9. L_{square} is generated by the 2-dimensional contextual array grammar with regular control $G_{squareRC} = (G_{square}, C_{square})$ with $G_{square} = (\{a\}, P_{square}, A_{square})$, where A_{square} collects the 2×2 and 3×3 squares,

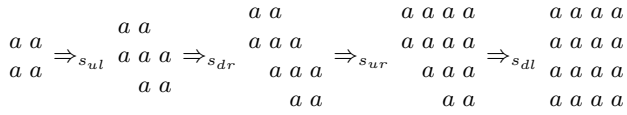
$$P_{square} = \{s_{ul}, s_{dr}, s_{ur}, s_{dl}, r_{ul}, r_{uu}, r_{dr}, r_{dd}\}, \text{ and}$$
$$C_{square} = \left(\{s_{ul}s_{dr}\}\{r_{ul}r_{dr}\}^*\{r_{uu}r_{dd}\}^*\{s_{ur}s_{dl}\}\right)^+.$$

The rules are listed in the following:

$$s_{ul} = \begin{array}{cc} a & a \\ a & \boxed{a}\ \boxed{a} \\ & \boxed{a} \end{array},\quad s_{dr} = \begin{array}{ccc} & & \boxed{a} \\ \boxed{a} & \boxed{a} & a \\ & a & a \end{array},\quad s_{ur} = \begin{array}{ccc} \boxed{a} & a & a \\ \boxed{a} & \boxed{a} & a \\ & \boxed{a} & \boxed{a} \end{array},\quad s_{dl} = \begin{array}{ccc} \boxed{a} & \boxed{a} & \\ a & \boxed{a} & \boxed{a} \\ a & a & \boxed{a} \end{array},$$

$$r_{ul} = \begin{array}{ccc} \boxed{a} & a & \\ \boxed{a} & \boxed{a} & \boxed{a} \end{array},\quad r_{uu} = \begin{array}{ccc} & \boxed{a} & \boxed{a} \\ & \boxed{a} & \boxed{a} \end{array},\quad r_{dr} = \begin{array}{ccc} \boxed{a} & \boxed{a} & \boxed{a} \\ & a & \boxed{a} \end{array},\quad r_{dd} = \begin{array}{ccc} & \boxed{a} & \\ \boxed{a} & \boxed{a} & a \end{array}.$$

How to derive a 4×4 square is shown below:



Notice that the rules s_{ur} and s_{dl} check if a complete new border layer was actually generated, so they provide "keystones" as used in architecture, and it somehow replaces the t-mode of derivation, e.g., see [6]. □

As already with the t-mode of derivation, e.g., see [6], only eight contextual array rules were needed in Example 9 to generate the squares. This shows that the ability of contextual array grammars to insert new parts on different positions in the current array allows for a significantly smaller number of rules when using specific control mechanisms as the t-mode of derivation or regular control languages, in comparison with the construction of an extended regular array grammar as described in [23], where the construction has to be carried out along a Hamiltonian path. The inserted pieces used in [23] in fact could also be used as arrays inserted by a contextual array grammar with regular control, yet even for the subset of squares of side lengths $5k + 16$, $k \geq 0$, as exhibited in [23], 27 rules (arrays) were used. As these are in fact a kind of macro-rules, a complete list of regular array rules based on [23] would correspond to about one thousand rules. This is an example showing that contextual array grammars may allow for a succinct description of specific picture languages with rather small descriptional complexity.

References

1. Chrobak, M.: Finite automata and unary languages. Theoretical Computer Science 47, 149–158 (1986)
2. Dassow, J., Păun, Gh.: Regulated Rewriting in Formal Language Theory, EATCS Monographs in Theoretical Computer Science, vol. 18. Springer (1989)
3. Ehrenfeucht, A., Păun, Gh., Rozenberg, G.: Contextual grammars and formal languages. In: [20], vol. 2, pp. 237–293 (1997)
4. Fernau, H., Freund, R.: Bounded parallelism in array grammars used for character recognition. In: Perner, P., Wang, P., Rosenfeld, A. (eds.) Advances in Structural and Syntactical Pattern Recognition (SSPR'96). Lecture Notes in Computer Science, vol. 1121, pp. 40–49. Springer (1996)
5. Fernau, H., Freund, R., Holzer, M.: Representations of recursively enumerable array languages by contextual array grammars. Fundamenta Informaticae 64, 159–170 (2005)
6. Fernau, H., Freund, R., Schmid, M.L., Subramanian, K.G., Wiederhold, P.: Contextual array grammars and array P systems. Annals of Mathematics and Artificial Intelligence 75(1-2), 5–26 (2015)
7. Fernau, H., Freund, R., Siromoney, R., Subramanian, K.G.: Contextual array grammars with matrix and regular control. Annals of the University of Bucharest (Seria Informatica) LXII(3), 63–78 (2015)

8. Fernau, H., Freund, R., Siromoney, R., Subramanian, K.G.: Non-isometric contextual array grammars with regular control and local selectors. In: Durand-Lose, J., Nagy, B. (eds.) Machines, Computations, and Universality, MCU. Lecture Notes in Computer Science, vol. 9288, pp. 61–78. Springer (2015)
9. Freund, R., Kogler, M., Oswald, M.: Control mechanisms on #−context-free array grammars. In: Păun, Gh. (ed.) Mathematical Aspects of Natural and Formal Languages, pp. 97–136. World Scientific Publ., Singapore (1994)
10. Freund, R., Kogler, M., Oswald, M.: A general framework for regulated rewriting based on the applicability of rules. In: Kelemen, J., Kelemenovà, A. (eds.) Computation, Cooperation, and Life, Lecture Notes in Computer Science, vol. 6610, pp. 35–53. Springer (2011)
11. Freund, R., Păun, Gh., Rozenberg, G.: Contextual array grammars. In: Subramanian, K., Rangarajan, K., Mukund, M. (eds.) Formal Models, Languages and Applications, Series in Machine Perception and Articial Intelligence, vol. 66, chap. 8, pp. 112–136. World Scientific (2007)
12. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, pp. 215–267. Springer (1997)
13. Krithivasan, K., Balan, M., Rama, R.: Array contextual grammars. In: Martín-Vide, C., Păun, Gh. (eds.) Recent Topics in Mathematical and Computational Linguistics, pp. 154–168. Editura Academiei Române, Bucureşti (2000)
14. Marcus, S.: Contextual grammars. Revue Roumaine de Mathématiques Pures et Appliquées 14, 1525–1534 (1969)
15. Nagy, B.: On a hierarchy of $5' \to 3'$ sensing Watson-Crick finite automata languages. Journal of Logic and Computation 23(4), 855–872 (2013)
16. Pradella, M., Cherubini, A., Crespi-Reghizzi, S.: A unifying approach to picture grammars. Information and Computation 209, 1246–1267 (2011)
17. Păun, Gh.: Marcus contextual grammars, Studies in Linguistics and Philosophy, vol. 67. Springer, Dordrecht (1997)
18. Rosenfeld, A.: Picture Languages. Academic Press, Reading, MA (1979)
19. Rosenfeld, A., Siromoney, R.: Picture languages – a survey. Languages of Design 1(3), 229–245 (1993)
20. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, vol. 1-3. Springer (1997)
21. Subramanian, K.G.: A note on regular controlled apical growth filamentous systems. International Journal of Computer and Information Sciences 14, 235–242 (1985)
22. Wang, P.S.-P.: Some new results on isotonic array grammars. Information Processing Letters 10, 129–131 (1980)
23. Yamamoto, Y., Morita, K., Sugata, K.: Context-sensitivity of two-dimensional regular array grammars. International Journal of Pattern Recognition and Artificial Intelligence 3, 295–319 (1989)