

Array Insertion and Deletion P Systems

Henning Fernau¹ Rudolf Freund²

Sergiu Ivanov³ Marion Oswald²

Markus L. Schmid¹ K.G. Subramanian⁴

¹ Universität Trier, D-54296 Trier, Germany

Email: {fernau,MSchmid}@uni-trier.de

² Vienna Univ. of Technology, Austria

Email: {rudi,marion}@emcc.at

³ LACL, Université Paris Est, France

Email: sergiu.ivanov@u-pec.fr

⁴ Universiti Sains Malaysia, 11800 Penang, Malaysia

Email: kgsmani1948@yahoo.com

Overview

Basic Definitions and Results

A General Model for Sequential Grammars

String Rewriting Grammars

Arrays and Array Grammars

P Systems

Undecidability Results for Array Grammars

Computational Completeness Results

P Systems with Minimal String


Insertion, Deletion, and Substitution Rules

P Systems with One-/Two-dimensional

Array Insertion and Deletion Rules

Future Research

A General Model for Sequential Grammars

-  R. Freund, M. Kogler, M. Oswald, A general framework for regulated rewriting based on the applicability of rules, in J. Kelemen and A. Kelemenová, Eds., *Computation, Cooperation, and Life - Essays Dedicated to Gheorghe Păun on the Occasion of His 60th Birthday*, LNCS **6610**, Springer, 2011, pp. 35-53.

A *(sequential) grammar* G is a construct

$(O, O_T, w, P, \Longrightarrow_G)$ where

- ▶ O is a set of *objects*,
- ▶ $O_T \subseteq O$ is a set of *terminal objects*,
- ▶ $w \in O$ is the *axiom (start object)*,
- ▶ P is a finite set of *rules*, and
- ▶ $\Longrightarrow_G \subseteq O \times O$ is the *derivation relation* of G .

A General Model for Sequential Grammars - Derivations

We assume that each of the rules $p \in P$ induces a relation $\Longrightarrow_p \subseteq O \times O$ with respect to \Longrightarrow_G fulfilling at least the following conditions:

- (i) for each object $x \in O$, $(x, y) \in \Longrightarrow_p$ for only finitely many objects $y \in O$;
- (ii) there exists a finitely described mechanism (as, for example, a Turing machine) which, given an object $x \in O$, computes all objects $y \in O$ such that $(x, y) \in \Longrightarrow_p$.

A General Model for Sequential Grammars – Applicability of Rules, Derivations

A rule $p \in P$ is called *applicable* to an object $x \in O$ if and only if there exists at least one object $y \in O$ such that $(x, y) \in \Longrightarrow_p$; we also write $x \Longrightarrow_p y$.

The *derivation relation* \Longrightarrow_G is the union of all \Longrightarrow_p , i.e., $\Longrightarrow_G = \bigcup_{p \in P} \Longrightarrow_p$. The reflexive and transitive closure of \Longrightarrow_G is denoted by \Longrightarrow_G^* .

A General Model for Sequential Grammars – Generated Languages

$$L_*(G) = \left\{ v \in O_T \mid A \xRightarrow{*}_G v \right\}$$

*language generated by G in the *-mode.*

$$L_t(G) = \left\{ v \in O_T \mid A \xRightarrow{*}_G v \wedge \neg \exists w : v \xRightarrow{*}_G w \right\}$$

language generated by G in the t -mode.

$\mathcal{L}_*(X)$: family of languages generated by grammars of type X in the *-mode.

$\mathcal{L}_t(X)$: family of languages generated by grammars of type X in the t -mode.

String Rewriting Grammar of Type X

$G = (V^*, T^*, A, P)$ where

- ▶ V is a (finite) set of *symbols*,
- ▶ $T \subseteq V$ is a set of *terminal symbols*,
- ▶ $A \in V^+$ is the *axiom*, and
- ▶ P is a finite set of *rules* of type X .

$$L(G) = L_*(G) = \left\{ v \in T^* \mid A \xRightarrow{*}_G v \right\}$$

language generated by G .

$\mathcal{L}(X)$: family of languages generated by grammars of type X .

Rules Working at the Ends of a String

Post rewriting rule: $P[x/y]$ with $x, y \in V^*$:

$$P[x/y](wx) = yw \text{ for } w \in V^*.$$

Left substitution: $S_L[x/y]$ with $x, y \in V^*$:

$$S_L[x/y](xw) = yw \text{ for } w \in V^*.$$

Right substitution: $S_R[x/y]$ with $x, y \in V^*$:

$$S_R[x/y](wx) = wy \text{ for } w \in V^*.$$

left insertion: $S_L[\lambda/y]$ is denoted by $I_L[y]$

right insertion: $S_R[\lambda/y]$ is denoted by $I_R[y]$

left deletion: $S_L[x, \lambda]$ is denoted by $D_L[x]$

right deletion: $S_R[x, \lambda]$ is denoted by $D_R[x]$

Types of String Grammars

$S_L^{k,m} / S_R^{k,m}$:

type of grammars using only substitution rules

$S_R[x/y] / S_L^{k,m}$ with $|x| \leq k$ and $|y| \leq m$.

$I_L^m, I_R^m, D_L^k, D_R^k$:

left/right insertion/deletion of strings with lengths at most m/k .

$D^k I^m S^{k'm'}$:

deletion/insertion/substitution of strings with lengths at most $k/m/k', m'$.

Post System

grammar $G = (V, T, A, P)$ of type *PS*:
Post rewriting rules $P[x/y]$ in P .

Post system *normal form* (type *PSNF*):
Post rewriting rules $P[x/y]$ in P are only of the following forms, with $a, b, c \in V$:

- ▶ $P[ab/c]$,
- ▶ $P[a/bc]$,
- ▶ $P[a/b]$,
- ▶ $P[a/\lambda]$.

Post System

A Post system (V, T, A, P) is in *Z-normal form* (type *PSZNF*) if it is in normal form and there exists a special symbol $Z \in V \setminus T$ such that

- ▶ $P[Z/\lambda]$ is the only rule where Z appears;
- ▶ if $P[Z/\lambda]$ is applied, the derivation stops yielding a terminal string;
- ▶ applying $P[Z/\lambda]$ is the only way to obtain a terminal string.

Theorem

$$\mathcal{L}(PS) = \mathcal{L}(PSNF) = \mathcal{L}(PSZNF) = RE.$$

d-dimensional Array

Let $d \in \mathbb{N}$; then a *d-dimensional array* \mathcal{A} over an alphabet V is a function $\mathcal{A} : \mathbb{Z}^d \rightarrow V \cup \{\#\}$, where $\text{shape}(\mathcal{A}) = \{v \in \mathbb{Z}^d \mid \mathcal{A}(v) \neq \#\}$ is finite and $\# \notin V$ is called the *background* or *blank symbol*.

The set of all d -dimensional arrays over V is denoted by V^{*d} . For $v \in \mathbb{Z}^d$, $v = (v_1, \dots, v_d)$, the norm of v is $\|v\| = \max \{|v_i| \mid 1 \leq i \leq d\}$. For a (non-empty) finite set $W \subset \mathbb{Z}^d$ the norm of W is defined as $\|W\| = \max \{ \|v - w\| \mid v, w \in W \}$.

d-dimensional Array Grammar

$$G_A = \left((N \cup T)^{*d}, T^{*d}, \mathcal{A}_0, P, \Longrightarrow_{G_A} \right)$$

where

- ▶ N is the alphabet of *non-terminal symbols*,
- ▶ T is the alphabet of *terminal symbols*,
 $N \cap T = \emptyset$,
- ▶ $\mathcal{A}_0 \in (N \cup T)^{*d}$ is the *start array*,
- ▶ P is a finite set of *d-dimensional array rules* over V , $V := N \cup T$,
- ▶ $\Longrightarrow_{G_A} \subseteq (N \cup T)^{*d} \times (N \cup T)^{*d}$ is the derivation relation induced by the array rules in P .

Types of Array Rewriting Rules

A *d-dimensional contextual array rule* over the alphabet V is a pair of finite d -dimensional arrays $(\mathcal{A}_1, \mathcal{A}_2)$ with $\text{dom}(\mathcal{A}_1) \cap \text{dom}(\mathcal{A}_2) = \emptyset$ and $\text{shape}(\mathcal{A}_1) \cup \text{shape}(\mathcal{A}_2) \neq \emptyset$; we also call it an *array insertion rule*, as its effect is that in the context of \mathcal{A}_1 we insert \mathcal{A}_2 ; hence, we write $I(\mathcal{A}_1, \mathcal{A}_2)$. The pair $(\mathcal{A}_1, \mathcal{A}_2)$ can also be interpreted as having the effect that in the context of \mathcal{A}_1 we delete \mathcal{A}_2 ; in this case, we speak of an *array deletion rule* and write $D(\mathcal{A}_1, \mathcal{A}_2)$. For any (contextual, insertion, deletion) array rule we define its **norm** by $\|\text{dom}(\mathcal{A}_1) \cup \text{dom}(\mathcal{A}_2)\|$.

Types of Array Grammars

The types of d -dimensional array grammars using array insertion rules of norm $\leq k$ and array deletion rules of norm $\leq m$ are denoted by $d-D^m I^k A$.

If only array insertion (i.e., contextual) rules are used, we have the case of pure grammars, and the type is denoted by $d-CA$.

Contextual Array Grammar Generating a Special Line

Example

Consider the contextual array grammar

$$G_{line} = (\{\bar{S}, E, L, R\}, E\bar{S}E, P) \text{ with}$$

$$P = \left\{ \boxed{E}E, E\boxed{E}, \boxed{E}R, L\boxed{E} \right\}.$$






Then

$$L_t(G_{line}) = \{LE^n\bar{S}E^mR \mid n, m \geq 1\},$$

whereas

$$L_*(G_{line}) = \{E^n\bar{S}E^m, E^n\bar{S}E^mR, LE^n\bar{S}E^m, LE^n\bar{S}E^mR \mid n, m \geq 1\}.$$

d-dimensional Arrays - Literature

-  C. R. Cook and P. S.-P. Wang, A Chomsky hierarchy of isotonic array grammars and languages, *Computer Graphics and Image Processing* **8** (1978), pp. 144–152.
-  H. Fernau, R. Freund, M.L. Schmid, K.G. Subramanian, P. Wiederhold, Contextual array grammars and array P systems, *submitted*.
-  R. Freund, Gh. Păun, G. Rozenberg, Contextual array grammars, in K.G. Subramanian, K. Rangarajan, and M. Mukund, Eds., *Formal Models, Languages and Applications*, Series in Machine Perception and Artificial Intelligence **66**, World Scientific, 2007, pp. 112–136.
-  A. Rosenfeld, *Picture Languages*, Academic Press, Reading, MA, 1979.
-  P. S.-P. Wang, Some new results on isotonic array grammars, *Information Processing Letters* **10** (1980), pp. 129–131.

P System of Type X

$\Pi = (G, \mu, R, i_0)$ where

- ▶ $G = (V, T, A, P)$: grammar of type X ;
- ▶ μ : membrane structure (tree);
the nodes of the tree representing μ are uniquely labelled by labels from a set Lab ;
- ▶ R : set of rules of the form (h, r, tar) ;
 $h \in Lab$, $r \in P$, and
 $tar \in \{here, in, out\} \cup \{in_j \mid 1 \leq j \leq n\}$;
- ▶ i_0 : initial membrane; the axiom A is put in there at the beginning of a computation.

Computations in a P System

$(w_1, h_1) \Longrightarrow_{\Pi} (w_2, h_2)$ (*computation step*):

for some $(h_1, r, tar) \in R$, $w_1 \Longrightarrow_r w_2$ and w_2 is sent from membrane h_1 to membrane h_2 indicated by tar .

halting computation: sequence $(A, i_0) \Longrightarrow_{\Pi}^* (w, h)$ of computation steps ending with a configuration (w, h) to which no rule from R can be applied; $w (\in O_{\mathcal{T}})$ is the result of this computation.

$L(\Pi)$ (language generated by Π):

consists of all objects from $O_{\mathcal{T}}$ which are results of a halting computation in Π .

Language Families Generated by P Systems

$\mathcal{L}(X-LP)$, $(\mathcal{L}(X-LP^{(n)}))$:

family of languages generated by P systems using rules of type X (of tree height at most n).

$\mathcal{L}(X-LsP)$, $(\mathcal{L}(X-LsP^{(n)}))$: $s = \text{simple}$;

family of languages generated by P systems using rules of type X (of tree height at most n);
only the targets *here*, *in*, *out* are used.

$\mathcal{L}(X-LcP)$, $(\mathcal{L}(X-LcP^{(n)}))$: $c = \text{channel type}$;

family of languages generated by P systems using rules of type X (of tree height at most n);
only the targets *in* and *out* are used.

P System of Type X as Acceptors

$$\Pi = (G, \mu, R, i_0)$$

- ▶ i_0 : initial membrane; the **input** is put in there at the beginning of a computation.

$L_a(\Pi)$ (language accepted by Π):

consists of all objects from O_T which are accepted by a halting computation in Π .

Language Families Accepted by P Systems

$\mathcal{L}_a(X-LP)$, $(\mathcal{L}_a(X-LP^{(n)}))$:

family of languages accepted by P systems using rules of type X (of tree height at most n).

$\mathcal{L}_a(X-LsP)$, $(\mathcal{L}_a(X-LsP^{(n)}))$: $s = \text{simple}$;

family of languages accepted by P systems using rules of type X (of tree height at most n);
only the targets *here*, *in*, *out* are used.

$\mathcal{L}_a(X-LcP)$, $(\mathcal{L}_a(X-LcP^{(n)}))$: $c = \text{channel type}$;

family of languages accepted by P systems using rules of type X (of tree height at most n);
only the targets *in* and *out* are used.

Undecidability for One-dimensional Array Grammars with Array Insertion and Deletion Rules

Lemma

Let $I = ((u_1, \dots, u_n), (v_1, \dots, v_n))$ be an instance of the PCP over T . Then we can effectively construct a one-dimensional array insertion P system Π such that

$$L(\Pi) = \{LL'h_T(w)RR' \mid w \in L((u_1, \dots, u_n), (v_1, \dots, v_n))\}.$$


As the Post Correspondence Problem is undecidable, the emptiness problem for $\mathcal{L}_t(1\text{-DIA-LP}^{\langle k \rangle})$ is undecidable:

Corollary

For any $k \geq 1$, the emptiness problem for $\mathcal{L}_t(1\text{-DIA-LP}^{\langle k \rangle})$ is undecidable.

Undecidability for More-dimensional Contextual Array Grammars

Every recursively enumerable one-dimensional array language can be characterized as the projection of an array language generated by a two-dimensional contextual array grammar using rules of norm one only, see:

 H. Fernau, R. Freund, and M. Holzer:
Representations of recursively enumerable array languages by contextual array grammars,
Fundamenta Informaticae **64** (2005), pp. 159–170.

Hence, for $d \geq 2$, even the emptiness problem for $\mathcal{L}_t(d\text{-CA})$ is undecidable.

P Systems with Minimal Left and Right Insertion, Deletion, and Substitution Rules

-  R. Freund, Yu. Rogozhin, S. Verlan: P systems with minimal left and right insertion and deletion. In: J. Durand-Lose, N. Jonoska (eds.): *Unconventional Computation and Natural Computation, 11th International Conference, UCNC 2012*. Orleans, France, September 3–7, 2012. Lecture Notes in Computer Science **7445**, 82–93, Springer (2012).
-  R. Freund, Yu. Rogozhin, S. Verlan: Generating and accepting P systems with minimal left and right insertion and deletion. To appear in *Natural Computing*.

Computational Power of String Grammars with Minimal Left and Right Insertion and Deletion Rules

Theorem

Every language $L \subseteq T^*$ in $\mathcal{L}(D^1 I^1 S^{1,1})$ is of the form $T_l^* S T_r^*$ where $T_l, T_r \subseteq T$ and $S \subset_{fin} T^*$.

Corollary

$$\mathcal{L}(A-D^1 I^1 S^{1,1}) = \mathcal{L}(A-I^1) \subset REG.$$

The prefix A in front of the types indicates that we consider a finite subset of axioms instead of a single axiom.

Computational Completeness of P Systems with Minimal Insertion, Deletion, and Substitution Rules

Theorem

$$RE = \mathcal{L} \left(D_R^1 I_L^1 S_R^{1,1} - LP^{\langle 1 \rangle} \right) = \mathcal{L}_a \left(D_R^1 I_L^1 S_R^{1,1} - LP^{\langle 1 \rangle} \right).$$

Theorem

$$RE = \mathcal{L} \left(D^1 I^1 - LsP^{\langle 8 \rangle} \right) = \mathcal{L}_a \left(D^1 I^1 - LsP^{\langle 8 \rangle} \right).$$

Theorem

$$RE = \mathcal{L} \left(D^1 I^1 - LcP^{\langle 8 \rangle} \right) = \mathcal{L}_a \left(D^1 I^1 - LcP^{\langle 8 \rangle} \right).$$

Computational Completeness of P Systems with One-dimensional Array Insertion and Deletion Rules

Theorem

$$\mathcal{L}_* (1\text{-ARBA}) = \mathcal{L}_t (1\text{-}D^1I^1A\text{-}LsP^{\langle 2 \rangle}).$$

Allowing norm two, we even do not need the regulating mechanism of membranes:

Theorem

$$\mathcal{L}_* (1\text{-ARBA}) = \mathcal{L}_t (1\text{-}D^2I^2A).$$


It remains as an interesting question for future research whether this result for array grammars only using array insertion and deletion rules with norm at most two can also be achieved in higher dimensions.

Computational Completeness of P Systems with Two-dimensional Array Insertion and Deletion Rules

The corresponding computational completeness result has been shown for 2-dimensional array insertion and deletion P systems using rules with norm at most two.

Theorem

$$\mathcal{L}_*(2\text{-ARBA}) = \mathcal{L}_t(2\text{-D}^2\text{I}^2\text{A-LS}P^{(2)}).$$

-  H. Fernau, R. Freund, S. Ivanov, M. L. Schmid, and K. G. Subramanian, Array insertion and deletion P systems, in G. Mauri, A. Dennunzio, L. Manzoni, and A. E. Porreca, Eds., *UCNC 2013*, Milan, Italy, July 1–5, 2013, LNCS **7956**, Springer 2013, pp. 67–78.

Proof Idea for P Systems with Two-dimensional Array Insertion and Deletion Rules

The main idea for showing computational completeness for one-/two-dimensional array insertion and deletion P systems using rules with norm one/two is to generate a line and a two-dimensional cube (rectangle) in which the rules of an array grammar of specific normal form can be simulated by erasing one symbol and inserting another one at the same position.

Theorem

$$\mathcal{L}_*(k\text{-ARBA}) = \mathcal{L}_t(k\text{-D}^k\text{I}^k\text{A-LS}P^{\langle 2 \rangle}), k \in \{1, 2\}.$$

Future Research

(1) Array Grammars with Array Insertion and Deletion Rules of Norm 2

For obtaining computational completeness, in general, i.e., for any dimension, we only need array grammars using array insertion and deletion rules with norm at most two:

Theorem

For any $k \geq 1$,

$$\mathcal{L}_*(k\text{-ARBA}) = \mathcal{L}_t(k\text{-D}^2\text{I}^2\text{A}).$$

Future Research

(2) P Systems with Array Insertion and Deletion Rules of Norm 1

When using only array insertion and deletion rules with norm at most one, we conjecture that we still need the control mechanism of P systems.

For higher dimensions, the constructions needed for showing computational completeness are very complicated, even for the case $k = 2$, hence, currently we only may show:

Theorem

For $k \in \{1, 2\}$,

$$\mathcal{L}_*(k\text{-ARBA}) = \mathcal{L}_t(k\text{-D}^1\text{I}^1\text{A-LsP}^{\langle 2 \rangle}).$$

THANK YOU VERY MUCH
FOR YOUR ATTENTION!

Requests for NCMA Proceedings:

rudi@emcc.at