Array Insertion and Deletion P Systems

Henning Fernau[1], Rudolf Freund[2], Sergiu Ivanov[3]
Marion Oswald[2], Markus L. Schmid[1], and K.G. Subramanian[4]

[1] Fachbereich 4 – Abteilung Informatikwissenschaften, Universität Trier
[2] Technische Universität Wien, Institut für Computersprachen
[3] Laboratoire d'Algorithmique, Complexité et Logique, Université Paris Est
[4] School of Computer Sciences, Universiti Sains Malaysia

Array insertion grammars have already been considered as *contextual array grammars* in [5], whereas the inverse interpretation of a contextual array rule as a deletion rule has newly been introduced in [2] and [3]. The results described in this extended abstract were elaborated in [3] for one-dimensional arrays and in [2] for two-dimensional arrays.

Sequential Grammars

A *(sequential) grammar* G (see [4]) 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 induced by the rules in P.

$L_* (G) = \left\{ v \in O_T \mid w \overset{*}{\Longrightarrow}_G v \right\}$ is the *language generated by* G (in the $*$-mode); $L_t (G) = \left\{ v \in O_T \mid \left(w \overset{*}{\Longrightarrow}_G v \right) \wedge \nexists z \, (v \Longrightarrow_G z) \right\}$ is the *language generated by* G *in the* t*-mode,* i.e., the set of all terminal objects derivable from the axiom in a halting computation. The family of languages generated by grammars of type X in the derivation mode δ, $\delta \in \{*, t\}$, is denoted by $\mathcal{L}_\delta (X)$. The family of recursively enumerable d-dimensional array languages is denoted by $\mathcal{L}_* (d\text{-}ARBA)$.

Arrays and array grammars

Let $d \in \mathbb{N}$; then a *d-dimensional array* $\mathcal{A}$ over an alphabet V is a function $\mathcal{A} : \mathbb{Z}^d \to V \cup \{\#\}$, where $shape (\mathcal{A}) = \left\{ v \in \mathbb{Z}^d \mid \mathcal{A}(v) \neq \# \right\}$ is finite and $\# \notin V$ is called the *background* or *blank symbol*. The set of all d-dimensional arrays over V is denoted by V^{*d}. For $v \in \mathbb{Z}^d$, $v = (v_1, \ldots, v_d)$, the norm of v is $\|v\| = \max \{ |v_i| \mid 1 \leq i \leq d \}$. The *translation* $\tau_v : \mathbb{Z}^d \to \mathbb{Z}^d$ is defined by $\tau_v (w) = w + v$ for all $w \in \mathbb{Z}^d$. For any array $\mathcal{A} \in V^{*d}$, $\tau_v (\mathcal{A})$, the corresponding d-dimensional array translated by v, is defined by $(\tau_v (\mathcal{A}))(w) = \mathcal{A}(w - v)$ for all $w \in \mathbb{Z}^d$. For a (non-empty) finite set $W \subset Z^d$ the norm of W is defined as $\|W\| = \max \{ \|v - w\| \mid v, w \in W \}$.

$[\mathcal{A}] = \left\{ \mathcal{B} \in V^{*d} \mid \mathcal{B} = \tau_v (\mathcal{A}) \text{ for some } v \in \mathbb{Z}^d \right\}$ is the equivalence class of arrays with respect to linear translations containing $\mathcal{A}$. The set of all equivalence

classes of d-dimensional arrays over V with respect to linear translations is denoted by $\left[V^{*d}\right]$ etc.

$G_A = \left((N \cup T)^{*d}, T^{*d}, \mathcal{A}_0, P, \Longrightarrow_{G_A}\right)$ is called a *d-dimensional array grammar*, 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, $V := N \cup T$, and $\Longrightarrow_{G_A} \subseteq (N \cup T)^{*d} \times (N \cup T)^{*d}$ is the derivation relation induced by the array rules in P.

A *d-dimensional contextual array rule* (see [5]) over the alphabet V is a pair of finite d-dimensional arrays $(\mathcal{A}_1, \mathcal{A}_2)$ with $dom\,(\mathcal{A}_1) \cap dom\,(\mathcal{A}_2) = \emptyset$ and $shape\,(\mathcal{A}_1) \cup shape\,(\mathcal{A}_2) \neq \emptyset$; 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 $\|dom\,(\mathcal{A}_1) \cup dom\,(\mathcal{A}_2)\|$. 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-$D^m I^k A$; without specific bounds, we write d-DIA.

(Sequential) P Systems

For the the area of P systems, we refer the reader to [6] and the P page [7].

A *(sequential) P system of type X with tree height n* is a construct $\Pi = (G, \mu, R, i_0)$ where $G = (O, O_T, A, P, \Longrightarrow_G)$ is a sequential grammar of type X; μ is the membrane (tree) structure of the system with the height of the tree being n, the membranes are uniquely labelled by labels from a set Lab; R is a set of rules of the form (h, r, tar) where $h \in Lab$, $r \in P$, and tar, called the *target indicator*, is taken from the set $\{here, in, out\} \cup \{in_h \mid h \in Lab\}$; i_0 is the initial membrane containing the axiom A.

A configuration of Π is a pair (w, h) where w is the current object (e.g., string or array) and h is the label of the membrane currently containing the object w. A sequence of transitions between configurations of Π, starting from the initial configuration (A, i_0), is called a *computation* of Π. A *halting computation* is a computation ending with a configuration (w, h) such that no rule from R_h can be applied to w anymore; w is called the *result* of this halting computation if $w \in O_T$. The language generated by Π, $L_t\,(\Pi)$, consists of all terminal objects from O_T being results of a halting computation in Π.

By $\mathcal{L}_t\,(X\text{-}LP)$ $(\mathcal{L}_t\left(X\text{-}LP^{\langle n\rangle}\right))$ we denote the family of languages generated by P systems (of tree height at most n) using grammars of type X. If only the targets *here*, *in*, and *out* are used, then the P system is called *simple*, and the corresponding families of languages are denoted by $\mathcal{L}_t\,(X\text{-}LsP)$ $(\mathcal{L}_t\left(X\text{-}LsP^{\langle n\rangle}\right))$.

Undecidability and Computational Completeness Results

An instance of the *Post Correspondence Problem (PCP)* is a pair of sequences of non-empty strings $(u_1, \ldots, u_n)$ and $(v_1, \ldots, v_n)$ over an alphabet T. A solution of

this instance is a sequence of indices $i_1, \ldots, i_k$ such that $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$; we call $u_{i_1} \ldots u_{i_k}$ the result of this solution. Let $L((u_1, \ldots, u_n), (v_1, \ldots, v_n))$ denote the set of results of all solutions of the instance $((u_1, \ldots, u_n), (v_1, \ldots, v_n))$ of the PCP, and let the homomorphism h_Σ be defined by $h_\Sigma : \Sigma \to \Sigma^+$ with $h_\Sigma(a) = aa'$ for all $a \in \Sigma$.

Lemma 1. *([3]) Let $I = ((u_1, \ldots, u_n), (v_1, \ldots, v_n))$ be an instance of the PCP. Then we can effectively construct a one-dimensional array insertion P system Π such that*

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

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

For $d \geq 2$, even the emptiness problem for $\mathcal{L}_t(d\text{-}CA)$ is undecidable, see [1].

Theorem 2. *([3]) $\mathcal{L}_t\left(1\text{-}D^1I^1A\text{-}LsP^{\langle 2 \rangle}\right) = \mathcal{L}_t\left(1\text{-}D^2I^2A\right) = \mathcal{L}_*(1\text{-}ARBA)$.*

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; at least for dimension two, in [2] 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 3. *([2]) $\mathcal{L}_t\left(2\text{-}D^2I^2A\text{-}LsP^{\langle 2 \rangle}\right) = \mathcal{L}_*(2\text{-}ARBA)$.*

References

1. H. Fernau, R. Freund, and M. Holzer, Representations of recursively enumerable array languages by contextual array grammars, *Fundamenta Informaticae* **64** (2005), pp. 159–170.
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.
3. R. Freund, S. Ivanov, M. Oswald, and K. G. Subramanian, One-dimensional array grammars and P systems with array insertion and deletion rules, *accepted for MCU 2013.*
4. R. Freund, M. Kogler, and 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.
5. R. Freund, Gh. Păun, and 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.
6. Gh. Păun, G. Rozenberg, A. Salomaa, Eds., *The Oxford Handbook of Membrane Computing*, Oxford University Press, 2010.
7. The P systems Web page, `http://ppage.psystems.eu/`.