P Automata: Concepts, Results and New Aspects*

Erzsébet Csuhaj-Varjú

Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende u. 13-17, 1111 Budapest, Hungary
and
Department of Algorithms and Their Applications
Faculty of Informatics
Eötvös Loránd University
Pázmány Péter sétány 1/c, 1117 Budapest, Hungary
csuhaj@sztaki.hu

Summary. In this paper we discuss P automata, constructs combining properties of classical automata and P systems being in interaction with their environments. We describe the most important variants and their properties, and propose new topics and open problems for future research.

1 Introduction

Observing natural systems and processes, concepts for reconsidering fundamentals of computation can be obtained, and based on the new ideas unconventional computational devices can be built. When such a new construct is defined, its benefits for computing usually are demonstrated by a comparison to its conceptual predecessors or to other classical computational models having features similar to the new one. This procedure is taking place in the theory of P automata, a framework consisting of accepting variants of P systems which combine features of classical automata and P systems being in interaction with their environments. Shortly, a P automaton is a P system receiving input in each computational step from its environment which influences its operation, by changing its configuration and thus affecting its functioning. The sequences of inputs are distinguished as accepted or rejected input sequences. The input is given as a multiset of objects, where objects can be elementary ones, i.e., without any structure (for example, symbols) or non-elementary, structured ones (for example, a P system). The P system that receives the input is called the underlying P system of the P automaton.

* Research supported in part by the Hungarian Scientific Research Fund (OTKA), Grant no. K75952

Similarities between P automata and classical automata can immediately be observed, but the reader easily may notice differences between the two constructs as well: for example, conventional automata have separate state sets while in the case of P automata the actual state is represented by the actual configuration of the underlying P system. Another property which makes P automata different from classical automata is that the workspace that they can use for computation is provided by the objects of the already consumed input multisets. The objects which enter the system become part of the description of the machine, that is, the input, the object of the computation and the machine which executes the computation cannot be separated as it can be done in the case of customary automata.

The first variant of P automata, introduced in [14, 15], was the so-called *one-way P automaton* where the underlying P system had only top-down symport rules with promoters (and implicitly inhibitors). Almost at the same time, a closely related notion, the *analyzing P system* was defined in [21] providing a slightly different concept of an automaton-like P system. Both models describe the class of recursively enumerable languages. The property that purely communicating accepting P systems may represent computationally complete classes of computing devices gave an impetus to the research in the theory of P automata, resulting in a detailed study of automaton-like P systems.

Since that time, several variants of P automata have been introduced and investigated, which differ from each other in the main ingredients of these systems: the objects the P system operates with, the way of defining the acceptance, the way of communication with the environment, the types of the communication rules used by the regions, the types of the rules associated with the regions (whether or not evolution rules are allowed to be used), and whether or not the membrane structure changes in the course of the computation. Summaries on these constructs and their properties can be found in [32, 10, 13, 41].

Due to the power of the underlying P system, several of the above variants of P automata determine the class of recursively enumerable languages, even with limited size parameters. Although these constructs offer alternative models for Turing machines, P automata with significantly less computational power are of special interest as well, since they provide descriptions of natural systems, with low complexity. An adequate example of the latter systems is the standard, generic variant of P automata, based on antiport rules with promoters or inhibitors, functioning with sequential rule application, and accepting with final states. By appropriately chosen mappings for defining the language of the P automaton, these constructs determine a language class with sub-logarithmic space complexity.

In the following sections we describe the most important variants of P automata and their properties. We also discuss how some *classical variants of automata can be represented in terms of P automata*. Special emphasis is put on *non-standard features of P automata*, namely, that the same construct is able to operate over both finite and infinite alphabets, the underlying membrane structure may remain unchanged but it also may dynamically vary under functioning, and that to obtain large computational power they do not need workspace overhead.

We also propose new topics and problems for future research.

2 P automaton - the basic model

2.1 The formal concept

In order to provide the reader with sufficient information to follow the discussion on P automata and its different variants, we present some formal details concerning the basic model, following the terms and notations from [13]. For more information on membrane computing we refer to [37] and for more details on formal language and automata theory to [38].

Throughout the paper, we denote the class of regular, context-free, context-sensitive, and recursively enumerable languages by REG, CF, CS, and RE, respectively.

We designate the set of finite multisets over a set V by V°, and the set of their sequences by $(V^\circ)^*$. We also denote $u \in V^\circ$ by the corresponding string $a_1^{u(a_1)}a_2^{u(a_2)}\dots a_t^{u(a_t)} \in V^*$, $V = \{a_1, a_2, \dots, a_t\}$, or in the form $\{(a_1, u(a_1)), (a_2, u(a_2)), \dots, (a_t, u(a_t))\}$.

The underlying membrane system of a P automaton is an antiport (symport) P system possibly having promoters and/or inhibitors. For details on symport/antiport the reader is referred to [35], for the use of promoters to [31].

Briefly, a symport rule is of the form (x, in) or $(x, out), x \in V^\circ$. When such a rule is applied in a region of a P system, then the objects of the multiset x enter the region from the parent region (in) or they leave to the parent region (out). An antiport rule is of the form $(x, out; y, in), x, y \in V^\circ$. In this case, the objects of y enter the region from the parent region and in the same step the objects of x leave to the parent region. Notice that the parent region of the skin region is the environment. All types of these rules might be associated with a promoter or an inhibitor multiset, denoted as $(x, in)|_Z, (x, out)|_Z$, or $(x, out; y, in)|_Z, x, y \in V^\circ, Z \in \{z, \neg z \mid z \in V^\circ\}$. If $Z = z$, then the rule can only be applied if the region contains all objects of multiset z, and if $Z = \neg z$, then z must not be a submultiset of the multiset of objects present in the region. To simplify the notations, we denote symport and antiport rules with or without promoters/inhibitors as $(x, out; y, in)|_Z$, $x, y \in V^\circ, Z \in \{z, \neg z \mid z \in V^\circ\}$ where we also allow x, y, z to be the empty multiset. If $y = \lambda$ or $x = \lambda$, then the notation above denotes the symport rule $(x, in)|_Z$ or $(y, out)|_Z$, respectively, if $Z = \lambda$, then the rules above are without promoters or inhibitors.

Definition 1. *A P automaton (with n membranes) is an* $(n + 4)$*-tuple,* $n \geq 1$, $\Pi = (V, \mu, P_1, \dots, P_n, c_0, \mathcal{F})$, *where*

- V *is a finite alphabet of objects,*
- μ *is a membrane structure of* n *membranes with membrane* 1 *being the skin membrane,*

- *P_i is a finite set of antiport rules with promoters or inhibitors associated to membrane i for all $i, 1 \leq i \leq n$,*
- *$c_0 = (w_1, \ldots, w_n)$ is called the initial configuration (or the initial state) of Π where each $w_i \in V^\circ$ is called the initial contents of region i, $1 \leq i \leq n$,*
- *$\mathcal{F}$ is a computable set of n-tuples $(v_1, \ldots, v_n)$ where $v_i \subseteq V^\circ$, $1 \leq i \leq n$; it is called the set of accepting configurations of Π.*

An n-tuple $(u_1, \ldots, u_n)$ of finite multisets of objects over V present in the n regions of the P automaton Π is called a (possible) *configuration* of Π; u_i is the contents of region i in this configuration, $1 \leq i \leq n$.

A P automaton functions as a standard antiport (symport) P system (with promoters and/or inhibitors), changes its configurations by applying rules according to a certain type of working mode. In the case of P automata, the two most commonly used variants are the sequential rule application, introduced in [14, 15] (also called 1-restricted minimally parallel in [26]), and the maximally parallel rule application. In the case of sequential rule application, at any step of the computation the rule set to be applied is chosen in such a way that exactly one rule is applied in each region where the application of at least one rule is possible. When the the maximally parallel working mode is used, at every computational step as many rule application is performed simultaneously in each region as it is possible.

The set of the different types of working modes is denoted by $MODE$, we use *seq* and *maxpar* for the *sequential* and the *maximally parallel* rule application, respectively.

Definition 2. *Let $\Pi = (V, \mu, P_1, \ldots, P_n, c_0, \mathcal{F})$, $n \geq 1$, be a P automaton working in the X-mode of rule application, where $X \in MODE$. The transition mapping of Π is defined as a partial mapping $\delta_X : V^\circ \times (V^\circ)^n \to 2^{(V^\circ)^n}$ as follows:*

For two configurations $c, c' \in (V^\circ)^n$, we say that $c' \in \delta_X(u, c)$ if Π enters configuration c' from configuration c by applying its rules in the X-mode while reading the input $u \in V^\circ$, i.e., if u is the multiset of objects that enter the skin membrane from the environment while the underlying P system changes configuration c to c' by applying its rules in mode X.

The sequence of multisets of objects accepted by a P automaton is defined as the input sequence which is consumed by the skin membrane until the system reaches an accepting configuration.

Definition 3. *Let $\Pi = (V, \mu, P_1, \ldots, P_n, c_0, \mathcal{F})$, $n \geq 1$, be a P automaton. The set of input sequences accepted by Π with X-mode of rule application, $X \in MODE$, is defined as*

$$A_X(\Pi) = \{v_1 \ldots v_s \in (V^\circ)^* \mid \text{ there are } c_0, c_1, \ldots, c_s \in (V^\circ)^n, \text{ such that } c_i \in \delta_X(v_i, c_{i-1}), 1 \leq i \leq s, \text{ and } c_s \in \mathcal{F}\}.$$

A P automaton Π, as above, is said to be accepting by final states if $\mathcal{F} = E_1 \times \ldots \times E_n$ for some $E_i \subseteq V^\circ$, $1 \leq i \leq n$, where E_i is either a finite set of

finite multisets or $E_i = V^\circ$. Thus, a configuration $c = (u_1, \ldots, u_n)$ is final, if for all regions of Π, $u_i \in E_i$, $1 \leq i \leq n$.

If Π accepts by halting, then $\mathcal{F}$ contains all halting configurations of Π, that is, all configurations c with no $c' \in (V^\circ)^n$ such that $c' \in \delta_X(v, c)$ for some $v \in V^\circ$, $X \in MODE$.

The accepted multiset sequences of a P automaton can be encoded to strings, thus making possible to assign languages to the P automaton. In the case of sequential rule application, the set of multisets that may enter the system is finite, thus the input multisets can obviously be encoded by a finite alphabet. This implies that any accepted input sequence can be considered as a string over a finite alphabet. In the case of parallel rule application, the number of objects which may enter the system in one step is not necessarily bounded by a constant. Therefore, in this case the accepted input sequences correspond to strings over infinite alphabets.

In the following we consider languages over finite alphabets, therefore we apply a mapping to produce a finite set of symbols from a possibly infinite set of multisets.

Definition 4. *Let $\Pi = (V, \mu, P_1, \ldots, P_n, c_0, \mathcal{F})$, $n \geq 1$, be a P automaton, Σ be a finite alphabet, and let $f : V^\circ \to \Sigma^*$ be a mapping. The language accepted by Π with respect to f using the X-mode rule application, where $X \in MODE$, is defined as*

$$L_X(\Pi, f) = \{f(v_1) \ldots f(v_s) \in \Sigma^* \mid v_1 \ldots v_s \in A_X(\Pi)\}.$$

The class of languages accepted by P automata with respect to a class of computable mappings $\mathcal{C}$ with X-mode rule application, $X \in MODE$, is denoted by $\mathcal{L}_{X,\mathcal{C}}(PA)$.

We illustrate the notion of a P automaton by an example from [10].

Example 1. Let

$$\Pi = (\{S_1, S_2, S_3, a, b, c\}, [_1\ [_2\ [_3\ ]_3\ ]_2\ ]_1 (S_1, P_1, \{d\}), (S_2, P_2, \{S_1S_2\}), (S_3, P_3, \emptyset),$$

with

$$\begin{aligned} P_1 = \{&(a, in)|_{S_1}, (a, in)|_a, (b, in)|_a, (b, in)|_b, (c, in)|_b, (c, in)|_c, \\ &(d, in)|_c, (\varepsilon, in)|_d\}, \\ P_2 = \{&(S_1, in)|_{S_2}, (a, in)|_{S_1}, (b, in)|_{S_1}, (c, in)|_{S_1}, (\varepsilon, in)|_c\}, \\ P_3 = \{&(\varepsilon, in)|_{S_3}, (abc, in)|_{S_3}\}, \end{aligned}$$

Then, for $f(x) = x$, for $x \in \{a, b, c\}$, Π accepts words of the form $a^n b^n c^n$, $n \geq 1$, with sequential application of rules and with only symport rules with promoters. Thus, the language accepted by Π is a well-known non-context-free context-sensitive language.

2.2 Computational power

Examining the concept of a language accepted by a P automaton, the reader can immediately notice that it strongly depends on the choice of mapping f (see Definition 4). This implies that there might be cases when the power of the P automaton comes from the mapping f and not from the P automaton itself. Due to this property, the investigations on the accepting power of P automata have concentrated on the cases where the mapping f is of low complexity.

It can also easily be seen that P automata work with no workspace overhead, i.e., the workspace P automata can use for computation is provided by the objects of the already consumed input multisets. Although this property appears to significantly bound the computational power, since P automata may use maximally parallel working mode, i.e., may input an exponentially growing number of objects, the obtained computational power can be rather large.

We first recall some notations from [13]. Let $\mathbf{NSPACE}(S)$ designate the class of languages accepted by a non-deterministic Turing machine using a workspace which is bounded by a function $S : \mathbf{N} \to \mathbf{N}$ of the length of the input. We say that $L \in \mathbf{r1NSPACE}(S)$ if there is a Turing machine which accepts L by reading the input from a read only input tape once from left to right, and for every accepted word of length n, there is an accepting computation during which the number of nonempty cells on the work-tape(s) is bounded in each step by $c \cdot S(d)$ where c is an integer constant, and $d \leq n$ is the number of input tape cells that have already been read, that is, the actual *distance* of the reading head from the left end of the one-way input tape.

Let $c = (u_1, \ldots, u_n)$ be a configuration of a P automaton. We denote by $|c|$ the number of objects present inside the membrane structure, that is, $|c| = \Sigma_{i=1}^{n} |u_i|$ where $|u_i|$ denotes the number of objects of $u_i \in V^\circ$.

The following statement describes the workspace of the P automaton used for computing and its language for a non-erasing mapping f [13]. (The mapping f is non-erasing if $f : V^\circ \to \Sigma^*$ for some V, Σ with $f(u) = \lambda$ if and only if $u = \emptyset$.)

Theorem 1. *Let Π be a P automaton, let $c_0, c_1, \ldots, c_m$ be a sequence of configurations during an accepting computation, and let $S : \mathbf{N} \to \mathbf{N}$, such that $|c_i| \leq S(d)$, $0 \leq d \leq i \leq m$, where $S(d)$ bounds the number of objects inside the system in the ith step of functioning, $d \leq i$ being the number of transitions in which a nonempty multiset was imported into the system from the environment.*

If f is non-erasing and $f \in \mathbf{NSPACE}(S_f)$, then for any $X \in MODE$, $L_X(\Pi, f) \in \mathbf{r1NSPACE}(\log(S) + S_f)$.

By applying the above theorem and its proof to three-counter machines, the following theorem can be obtained (see [13]). (Three-counter machines are Turing machines with a one-way read only input tape and three work-tapes which can be used as three counters capable of storing any non-negative integer as the distance of the reading head from the only non-blank tape cell marked with the special symbol Z.)

The following results were first published in [11, 12].

Theorem 2.

1. $\mathcal{L}_{seq,\mathcal{C}}(PA) = \mathbf{r1NSPACE}(\log(n))$ *for any class* $\mathcal{C}$ *of non-erasing mappings with a finite domain, and*
2. $\mathcal{L}_{maxpar,\mathcal{C}}(PA) = CS$ *for any class* $\mathcal{C}$ *of non-erasing linear space computable mappings.*

By the simulation of the three-counter machine which is used to prove the previous theorem, it follows that if we allow arbitrary linear space computable functions for mapping the input multisets of the P automaton to the alphabet of the accepted language, then we can obtain a characterization of the class of recursively enumerable languages.

Corollary 1. $\mathcal{L}_{maxpar,\mathcal{C}}(PA) = RE$ *for any class* $\mathcal{C}$ *of linear space computable mappings.*

2.3 Discussion of the basic model

In the following we briefly discuss the *main ingredients* of P automata and propose topics for future research.

If we consider a P automaton as a *P system being in interaction with its environment*, then not only input sequences but also output sequences are of interest to study. While an *input sequence* can be considered as a *representation of a sequence of impulses obtained from the environment*, a sequence of *outputs*, i.e., a sequence of multisets of objects that were sent to the environment at the steps of the computation, correspond to *reactions* to the effect of the previously obtained impulses and the change they caused in the behavior of the system. By obvious modifications of Definitions 2, 3, 4, we can assign a so-called *output language* to the P automaton. Output languages of P automata, supposing that the underlying P system issues at any computation step at least one object to the environment, would be of particular interest topic of investigations.

The concept of an (accepted) output sequence of a P automaton opens *several further topics to be examined*. For example, if u_i denotes the input and v_i the output of the P automaton Π at the ith computation step of a computation, then $diff(i) =\mid card(u_i) - card(v_i) \mid$, i.e., the difference in the number of objects entering and leaving the system, describes the *volume of information exchange* at the given computation step and it is a characteristics of the P system. Based on this parameter, several complexity measures can be defined: $maxdiff(\Pi)$, i.e., the supremum, or $mindiff(\Pi)$, i.e., the minimum of the difference of the volume of information exchange with respect to any accepting computation. We also can consider the difference of these two measures as well.

Based on the above measures, we can define *a P automaton* Π to be *monotone* or *strictly monotone* if for any accepting computation in Π (or, for an accepting computation for any word in the language accepted by Π) $m(\Pi) \geq 0$ or $m(\Pi) > 0$ holds, respectively, where $m \in \{maxdiff, mindiff\}$. Monotone P automata represent systems which are able to *tolerate more and more impulses* from

the environment. Especially *interesting topic for future research* would be the description of language classes of P automata classes where the value of measures $maxdiff$ and $mindiff$ regarding the P automata in the class can be *bounded by linear, polynomial,* and *exponential functions*, respectively.

The concepts of an input and an output of a P automaton raise *another issue*. As we have seen, unlike classical automata, the *whole input sequence* is not given at the beginning of the computation, but it will be *available step by step*. Moreover, the input is not given in advance but it is determined by the actual configuration (state) of the underlying P system. It is an obvious question, what happens if we *present an input sequence of multisets of objects in advance* and we consider it as an accepted sequence if after consuming the elements of the sequence the underlying P system enters an accepting state. Obviously, *this model needs to be elaborated*, since the multisets in the sequence need not to coincide with the multiset of objects the underlying P system is able to consume. However, this direction of research would be of certain interest.

Some steps, although in a bit different manner, have already been made in this direction, see, for example, [20]. We note that the existence of a designated input membrane does not necessarily alter the computational power.

2.4 Non-standard features of P automata

P automata over infinite alphabets

One of the important characteristics of P automata is that the basic model is suitable for describing languages over infinite alphabets, without any extension or additional component added to the construct. This property arises from the fact that the language accepted by these systems corresponds to the sequence of multisets entering during a successful computation, and the number of possible symbols which constitute the accepted string can be arbitrarily large.

An example of this approach is the notion of a *P finite automaton*, introduced in [19].

This construct is a P automaton $\Pi = (V \cup \{a\}, \mu, P_1, \ldots, P_n, c_0, \mathcal{F})$ which applies the rules in the maximally parallel manner, accepts by final states, the object alphabet $V \cup \{a\}$ contains a distinguished symbol a; P_1 (the skin region) contains rules of the form $(x, out; y, in)|_Z$ with $x \in (V \cup \{a\})^\circ$, $y \in \{a\}^\circ$, $Z \in \{z, \neg z\}$, $z \in V^\circ$; and if $i \neq 1$, the set P_i contains rules of the form $(x, out; y, in)|_Z$ with $Z \in \{z, \neg z\}$, $x, y, z \in V^\circ$. We also allow the use of rules of the form $(x, in)|_Z$ in the skin membrane in such a way, that the application of any number of copies of the rule is considered in maximally parallel manner.

Notice that the domain of the mapping f is infinite, so its range could also easily be defined to be infinite, as $f : \{a\}^\circ \to \Sigma \cup \{\lambda\}$ for an infinite alphabet $\Sigma = \{a_1, a_2, \ldots\}$ with $f(a^k) = a_k$ for any $k \geq 1$, and $f(\emptyset) = \lambda$.

The language accepted by a P finite automaton Π is $L(\Pi) = L_{maxpar}(\Pi, f)$ for f as above.

In [19] it was shown that *for any $L \subseteq \Sigma^*$ over a finite alphabet Σ, $L \in REG$ if and only if $L = L(\Pi)$ for some P finite automaton Π.*

Because of these properties, the *infinite alphabet languages accepted by P finite automata* can be considered as the extension of the class of regular languages to infinite alphabets. In [19] it is also shown that this construction significantly differs from other infinite alphabet extensions of regular languages defined by, for example, the machine model called finite memory automata from [29] or the infinite alphabet regular expressions introduced in [34].

P automata models for *extensions of further language classes to infinite alphabets*, for example, to context-free languages, would also be an interesting research direction.

ω-P automata

P automata also provide possibilities of describing (possibly) infinite runs (sequences of configurations). This feature is of particular importance, since if we consider a P automaton as a system being in interaction with its environment, we also should consider communication processes not limited in time.

Variants of P automata, motivated by these considerations, are the so-called *ω-P automata* [25]. These constructs (having also so-called membrane channels) were introduced to simulate the functioning of ω-Turing machines, that is, actions of Turing machines on *infinite words*.

It was proved that *for any well-known variant of acceptance mode of ω-Turing machines one can construct an ω-P automaton with two membranes which simulates the computations of the corresponding ω-Turing machine.*

2.5 Variants of P automata

During the years, several types of automaton-like P systems were introduced with the aim of studying their boundaries as computational devices and exploring their relations to classical automata.

A lot of efforts have been devoted to describe the recursively enumerable language class in terms of P automata. To be conform with formal language theoretic constructs, several variants have been introduced, where input objects and auxiliary objects , i.e., terminal objects and nonterminal objects of the P automaton are distinguished. Then, the accepted language is defined as the sequence of terminal strings of the input multisets during an accepting computation (where the set of terminal strings of a multiset consists of all permutations of its terminal symbols). An example for this *extended P automaton* is the *analyzing P system* [21], which has only antiport rules, works with the maximally parallel rule application and accepts by halting. As we mentioned in the Introduction, the authors have shown that these systems, even with small size parameters, are able to recognize any recursively enumerable language.

In the case of these extended P automata, the workspace to obtain the computational completeness of P automata model, is due to the nonterminal objects which

can be available in a number not restricted by the length of the input string. In [28] interesting results were obtained for automaton-like P systems, called *exponential-space symport/antiport acceptors*, working with other types of bounded resources, with a set of terminal objects Σ containing a distinguished symbol \$, and four types of rules of restricted forms. These systems work with maximally parallel application of rules and accept by final states; the language accepted by them is defined in a slightly different way from the one that is used in the case of an extended P automaton. The term "exponential-space symport/antiport acceptor" comes from the fact that due to the restricted form of the rules, the workspace which can be used by such a construct is not arbitrarily large, the membrane system contains no more than an exponential number of objects (up to some constant) at any time during the computation. Working with the maximally parallel rule application, these systems describe the class of context-sensitive languages [28].

The original motivation of the introduction of the concept of the P automaton was to study the power of purely communicating accepting P systems. For this reason, the question whether or not any change in the underlying communicating P system implies changes in the power and the size complexity of the respective new class of P automata is of particular interest. During the years, several models have been introduced to approach this problem.

Additional constraints given by a partial binary relation were posed to the application of the communication rules of the basic model in the case of *P automata with priorities* in [6], where the rules with the highest priority must be applied in configuration change. Two other variants, with conditional symport/antiport rules, are *P automata with membrane channels* [32, 22, 23], motivated by certain natural processes taking place in cells, and *P automata with conditional communication rules associated with the membranes* [32, 24]. All these models are computationally complete devices, in the latter two cases optimal results on their size parameters have also been obtained.

Another feature in which P automata differ from classical automata is the property that they have no separate internal state sets, the states are represented by the (possibly infinite) set of configurations. *P automata with states* attempted to make the basic concept resemble more to conventional automata [30]. In this model, both states and objects are considered, the states, together with the objects, govern the communication. The device is computationally complete, moreover, any recursively enumerable language can be described by these systems with very restricted form [20].

Although most of the variants of P automata realize purely communicating, accepting P systems, the concept can be extended to be suitable for *describing complex evolving systems. Evolution-communication P automata*, having both communication and evolution rules, are examples for such models [1]. The construct can be considered as a variant of extended P automata, and as it is expected, it provides a description of the class of recursively enumerable languages.

3 Further developments

3.1 P automata computing by structure

The models that have been discussed so far have a static membrane structure, that is, the membrane structure is not altered during the functioning of the system. Considering P automata as models of complex biological systems, this condition is rather restrictive, since the architecture of natural systems may change in the course of their functioning.

A P automaton-like system working with a dynamically changing membrane structure is the *P automaton with marked membranes* ([16]), or a P_{pp} *automaton*, for short. The concept is motivated by *the theory of P systems, brane calculi [5], and traditional automata theory.* The underlying P system models the situation when proteins are allowed to move through the membranes and to attach onto or to detach from the membranes, in such a way that their moves may also imply changes in the membrane structure. P automata with marked membranes are able to consume inputs from their environment, i.e., multisets of proteins, which might influence the behavior of the system, and correspond to the result of a computation if the P_{pp} automaton starts in the initial configuration and halts in a final configuration. As in the previous cases, the model is computationally complete. Its importance lies in the bridge built between important research areas.

A variant of accepting P systems with dynamically changing membrane structure, called an *active P automaton*, was proposed and used for parsing sentences of natural languages in [2, 3]. An active P automaton starts the computation with one membrane containing the string to be analyzed together with some additional information assisting the computation. It computes with the structure of the membrane system, using operations as membrane creation, division, and dissolution. There are also rules for extracting a symbol from the left-hand end of the input string and for processing assistant objects. The computation ends with acceptance when all symbols from the string are consumed and all membranes are dissolved. It was shown that the model is suitable for recognizing any recursively enumerable language, and with restrictions in the possible types of rules, also for determining other well-known language classes, such as the regular language class and the class of context-sensitive languages. This special variant of accepting P systems resembles P automata since any symbol in the string can be considered as a multiset of objects with one element consumed from the environment.

3.2 Classical automata versus P automata

Another important research area to investigate is how models and concepts of classical automata theory can be related to models and concepts in P automata theory. As we have seen in Subsection 2.4, finite automata can be represented in terms of P automata in a natural manner.

The property that by using the maximally parallel working mode an object can appear in a region in a number of copies not bounded by a constant (obviously, depending on the underlying P automaton), implies that strings (in the form of numbers which are the values of numbers given in k-ary notation) can be represented by regions of P automata. Based on this correspondence, contents of pushdown storages or stacks can be described, which natural observation is used for characterizing the context-free language class by a restricted variant of P automata, called *stack P automata* in [40]. Obviously, a pushdown storage can also be represented as a configuration of a P system with a linear structure, where there is only one object or one object of some distinguished type (representing a symbol that belongs to the pushdown alphabet) in each region [39]. If we allow changes in the linear membrane structure, i.e., the dissolution of the skin membrane and creation of a new linear structure which embraces the remaining part of the original linear membrane structure, we can obtain a representation of a pushdown storage in some other manner. Both approaches are used in [17], where different languages classes, for example, the growing context-sensitive language class, are described in terms of variants of multi-pushdown automata.

Counterparts of other classical variants of automata are found in [7], where the so-called *Mealy multiset automata* and *elementary Mealy membrane automata* are proposed and examined. These models are inspired by the concept of a Mealy automaton. As a continuation of this research, an augmented version of the elementary Mealy membrane automaton, with extended communication capabilities, called a *simple P machine* was investigated in [8].

So far we have discussed automata with only input, although transducers, i.e., automata with input and output play outstanding role in classical automata theory. The concept of a *P transducer*, which is basically a one-membrane P automaton working with input and output objects [9], realizes such a construction. Four types of these machines were distinguished and studied, two of them are computationally complete, and two are incomparable to finite state sequential transducers. Iterating these latter P transducer classes, new characterizations of the recursively enumerable language class were obtained.

3.3 P automata and words with nested data

Since membrane systems are *nested architectures*, investigations in connections between P automata theory and the theory of data languages, a theory mainly motivated by applications in XML databases and parametrized verification, are of particular interest. Research in this direction has started in [18].

In order to briefly report on the topic, we recall some notions on words with nested data, following the notations in [4]. Let V be a finite alphabet and Δ an infinite set whose elements are called data values. For a natural number k, a word w with k layers of data is a string where every position, apart from a label in V, has k labels $d_1, \ldots, d_k \in \Delta$. The label d_i is called the ith data value of the position. Thus, $w \in (V \times \Delta^k)^*$. In w, the data values can be seen as inducing k

equivalence relations $\sim_1, \ldots, \sim_k$ on the positions of w; two positions are related by $\sim_i$ if they agree on the ith data value. A word with k layers of data is said to have nested data if for each $i = 2, \ldots, k$ the relation $\sim_i$ is a refinement of $\sim_{i-1}$. Since P automata are able to operate over infinite alphabets, for representing sets of words with k layers of data or with k layers of nested data (over some alphabets V and Δ), P automata with dynamically changing linear structure and antiport rules can be constructed.

Unlike standard questions concerning the computational power of P automata, the main questions in this case are *how much change the input implies in the structure of the underlying P system* and in the contents of certain regions.

Another important research direction can be to *develop logic for these P automata (P systems)*, since certain properties of words with (k layers of) nested data, have been described in terms of a fragment of first order logic, thus these words were considered as models for logic, with logical quantifiers ranging over word positions.

The topic is closely related to the study of shuffle expressions, since connections between words with nested data and these expressions have been explored, see for example [4]. *Shuffle expressions* are regular expressions extended with intersections and the shuffle operation. Relations between shuffle expressions and so-called *high-order multicounter automata* was analysed in [4], where it was shown that the class of languages defined by shuffle expressions, the class of languages defined by high-order multi-counter automata, and the recursively enumerable language class are equal. High-order multicounter automata are automata with several counters which can be incremented and decremented, but zero tests are only allowed at the end of the word. In [18] a new variant of P automata is defined with strong formal similarities to high-order multicounter automata. Based on the construction, results on P automata and shuffle expressions can be derived.

3.4 P automata expressions

One important research area of classical automata theory is the study of the closure with respect to certain operations, especially how to construct an automaton for languages obtained by certain operation among a given collection of automata. Questions related to *compositions of P automata* are of particular interest.

A step in this direction has been made in [27], where so-called *P automata with communication and active membrane rules working in the initial mode (CAIP)* have been introduced. The authors presented methods for constructing automata for accepting the union, the concatenation, the Kleene closure, or the ω closure of the given languages which are represented by some P automata. Starting from these results, and considering these and other operations and these and other (restricted) variants of P automata, it would be interesting to develop further descriptions of language classes in term of so-called *P-automata expressions*.

4 Conclusions

Investigations in the theory of P automata expected to be continued in several directions. Since P automata can be considered as constructs attempting to build a *bridge between automata theory and membrane systems theory*, similarities and differences between the two fields are certainly of interest. But, as we mentioned in the Introduction, P automata are models of *dynamically changing systems which are in communication (interaction) with their environments* as well. According to this approach, the investigations of P automata as dynamical systems form similarly important research directions. We hope to have new results in both directions in the future.

References

1. A. Alhazov: Minimizing evolution-communication P systems and EC P automata. In: M. Cavaliere, C. Martín-Vide and Gh. Păun (eds.), *Brainstorming Week on Membrane Computing.* Technical Report 26/03 of the Research Group on Mathematical Linguistics, Rovira i Virgili University, Tarragona, Spain, 2003, 23-31.
2. G. Bel-Enguix and R. Gramatovici: Parsing with active P automata. In: C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg and A. Salomaa (eds.), *Membrane Computing. International Workshop, WMC 2003, Tarragona, Spain, July 17-22, 2003. Revised Papers.* Lecture Notes in Computer Science 2933, Springer, Berlin, 2004, 31-42.
3. G. Bel-Enguix and R. Gramatovici: Parsing with P automata. In: G. Ciobanu, M. Pérez-Jiménez and Gh. Păun (eds.), *Applications of Membrane Computing.* Natural Computing Series, Springer, Berlin, 2006, 389-410.
4. H. Björklund and Mikolaj Bojanczyk: Shuffle Expressions and Words with Nested Data. In: L. Kucera, Antonn Kucera (eds.): Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Cesk Krumlov, Czech Republic, August 26-31, 2007, Proceedings. Lecture Notes in Computer Science 4708, Springer, 2007, 750-761.
5. L. Cardelli: Brane Calculi. Interactions of biological membranes. In: V. Danos and V. Schacter (eds.), *Computational Methods in Systems Biology. International Conference CMSB 2004, Paris, France, May 2004, Revised Selected Papers.* Lecture Notes in Computer Science 3082, Springer-Verlag, Berlin, 2005, 257-280.
6. L. Cienciala and L. Ciencialova: Membrane automata with priorities. *Journal of Computer science and Technology* 19(1) (2004), 89-97.
7. G. Ciobanu and V. M. Gontineac: Mealy multiset automata. *International Journal of Foundations of Computer Science* 17 (2006), 111-126.
8. G. Ciobanu and V. M. Gontineac: P machines: An automata approach to membrane computing. In: H.-J. Hoogeboom, Gh. Păun, G. Rozenberg and Arto Salomaa (eds.), *Membrane Computing, 7th International Workshop, WMC 2006, Leiden, The Netherlands, July 17-21, 2006, Revised, Selected, and Invited Papers.* Lecture Notes in Computer Science 4361, Springer, Berlin, 2006, 314-329.
9. G. Ciobanu, Gh. Păun and Gh. Stefănescu: P transducers. *New Generation Computing* 24(1) (2006), 1-28.

10. E. Csuhaj-Varjú: P automata. In: G. Mauri, Gh. Păun, M. Pérez-Jiménez, G. Rozenberg and A. Salomaa (eds.), *Membrane Computing: 5th International Workshop, WMC 2004, Milan, Italy, June, 14-16, 2004. Revised Selected and Invited Papers.* Lecture Notes in Computer Science 3365, Springer, 2005, 19-35.
11. E. Csuhaj-Varjú, O.H. Ibarra and Gy. Vaszil: On the computational complexity of P automata. In: C. Ferretti, G. Mauri and C. Zandron (eds.), *DNA Computing, 10th International Workshop on DNA Computing, DNA10, Milan, Italy, June 7-10, Revised Selected Papers.* Lecture Notes in Computer Science 3384, Springer, 2005, 77-90.
12. E. Csuhaj-Varjú, O.H. Ibarra and Gy. Vaszil: On the computational complexity of P automata. *Natural Computing* 5(2) (2006), 109-126.
13. E. Csuhaj-Varjú, M. Oswald and Gy. Vaszil: P automata. Chapter in Handbook of Membrane Computing. Gh. Păun, G. Rozenberg and A. Salomaa (Eds.), Oxford University Press, to appear.
14. E. Csuhaj-Varjú and Gy. Vaszil: P automata. In: Gh. Păun and C. Zandron (eds.), *Pre-Proceedings of the Workshop on Membrane Computing WMC-CdeA 2002, Curtea de Argeş, Romania, August 19-23, 2002.* Pub. No. 1 of MolCoNet-IST-2001-32008, 2002, 177-192.
15. E. Csuhaj-Varjú and Gy. Vaszil: P automata or purely communicating accepting P systems. In: Gh. Păun, G. Rozenberg, A. Salomaa and C. Zandron (eds.), *Membrane Computing. International Worskhop, WMC-CdeA 2002, Curtea de Arges, Romania, August 19-23, 2002, Revised Papers.* Lecture Notes in Computer Science 2597, Springer, Berlin, 2003, 219-233.
16. E. Csuhaj-Varjú and Gy. Vaszil: (Mem)brane automata. *Theoretical Computer Science* 404(1-2) (2008), 52-60.
17. E. Csuhaj-Varjú and Gy. Vaszil: Representation of language classes in terms of P automata. Manuscript, 2009.
18. E. Csuhaj-Varjú and Gy. Vaszil: Logic for P automata. Manuscript, 2009.
19. J. Dassow and Gy. Vaszil: P finite automata and regular languages over countably infinite alphabets. In: H.-J. Hoogeboom, Gh. Păun, G. Rozenberg and A. Salomaa (eds.), *Membrane Computing. 7th International Workshop, WMC 2006, Leiden, The Netherlands, July 17-21, 2006, Revised, Selected, and Invited Papers.* Lecture Notes in Computer Science 4361, Springer-Verlag, Berlin, 2006, 367-381.
20. R. Freund, C. Martín-Vide, A. Obtułowicz and Gh. Păun: On three classes of automata-like P systems. In: Z. Ésik and Z. Fülöp (eds.), *Developments in Language Theory. 7th International Conference, DLT 2003, Szeged, Hungary, July 7-11, 2003. Proceedings.* Lecture Notes in Computer Science 2710, Springer, Berlin, 2003, 292-303.
21. R. Freund and M. Oswald: A short note on analysing P systems. *Bulletin of the EATCS* 78 (October 2002), 231-236.
22. R. Freund and M. Oswald: P automata with activated/prohibited membrane channels. In: Gh. Păun, G. Rozenberg, A. Salomaa and C. Zandron (eds.), *Membrane Computing. International Worskhop, WMC-CdeA 2002, Curtea de Arges, Romania, August 19-23, 2002, Revised Papers.* Lecture Notes in Computer Science 2597, Springer, Berlin, 2003, 261-269.
23. R. Freund and M. Oswald: P automata with membrane channels. *Artificial Life and Robotics* 8(2004), 186-189.

24. R. Freund and M. Oswald: P systems with conditional communication rules assigned to membranes. *Journal of Automata, Languages and Combinatorics* 9(4) (2004), 387-397.
25. R. Freund, M. Oswald and L. Staiger: ω-P automata with communication rules. In: C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg and A. Salomaa (eds.), *Membrane Computing. International Workshop, WMC 2003, Tarragona, Spain, July 17-22, 2003. Revised Papers.* Lecture Notes in Computer Science 2933, Springer, Berlin, 2004, 203-217.
26. R. Freund, S. Verlan: (Tissue) P systems working in the k-restricted minimally parallel derivation mode. In: E. Csuhaj-Varjú, R. Freund, M. Oswald and K. Salomaa (eds.), *International Workshop on Computing with Biomolecules, August 27, 2008, Wien, Austria.* Österreichische Computer Gesellschaft, 2008, 43-52.
27. H. Long, Y. Fu: A general approch for building combinational P automata. *International Journal of Computer Mathematics* 84(12) (2007), 1715-1730.
28. O. H. Ibarra and Gh. Păun: Characterization of context-sensitive languages and other language classes in terms of symport/antiport P systems. *Theoretical Computer Science* 358(1) (2006), 88-103.
29. M. Kaminski and N. Francez: Finite-memory automata. *Theoretical Computer Science* 134(1994), 329-363.
30. M. Madhu and K. Krithivasan: On a class of P automata. *International Journal of Computer Mathematics* 80(9) (2003), 1111-1120.
31. C. Martín-Vide, A. Păun and Gh. Păun: On the power of P systems with symport rules. *Journal of Universal Computer Science* 8(2002), 317-331.
32. M. Oswald: *P Automata*. PhD dissertation, Vienna University of Technology, 2003.
33. M. Oswald and R. Freund: P Automata with membrane channels. In: M. Sugisaka and H. Tanaka, H. (eds): *Proc. of the Eights Int. Symp. on Artificial Life and Robotics*, Beppu, Japan, 2003, 275-278.
34. F. Otto: Classes of regular and context-free languages over countably infinite alphabets. *Discrete Applied Mathematics* 12 (1985), 41-56.
35. A. Păun and Gh. Păun. The power of communication: P systems with symport/antiport. *New Generation Computing* 20(3) (2002), 295-305.
36. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*61(1) (2000) 108-143.
37. Gh. Păun: *Membrane Computing. An Introduction.* Springer Verlag, Berlin-Heidelberg, 2002.
38. G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages*, Springer-Verlag, Berlin, Heidelberg, 1997.
39. D. Sburlan: Private communication, 2009.
40. Gy. Vaszil: A class of P automata characterizing context-free languages. In: M. A. Gutiérrez-Naranjo, Gh. Păun, A. Riscos-Núnez and F. J. Romero-Campero (eds.), *Proceedings of the Fourth Brainstorming Week on Membrane Computing, Sevilla, Spain, January 30-February 3, 2006. Volume II.* RGNC Report, 03/2006, Fénix Editora, Sevilla, 2006, 267-276.
41. Gy. Vaszil:Automata-like membrane systems - A natural way to describe complex phenomena. In: C. Campeanu, G. Pighizzini (eds.), *10th International Workshop on Descriptional Complexity of Formal Systems, July 16-18, Charlottetown, PE, Canada. Proceedings.* University of Prince Edwards Island, 2008, 26-37.