P Finite Automata and Regular Languages over Countably Infinite Alphabets*

Jürgen Dassow[1] and György Vaszil[2]

[1] Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
PSF 4120, D-39016 Magdeburg, Germany
dassow@iws.cs.uni-magdeburg.de
[2] Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende utca 13-17, 1111 Budapest, Hungary
vaszil@sztaki.hu

Abstract. We examine the class of languages over countably infinite alphabets characterized by a class of restricted and simplified P automata variants, which we call P finite automata, and show that these classes possess several properties which make them perfect candidates for being the natural extension of the notion of finite automata and that of regular languages to infinite alphabets. To this aim, we also show that P finite automata are equivalent to a restricted variant of register machines, providing a more conventional automata theoretical characterization of the same infinite alphabet language class.

1 Extending the Chomsky Hierarchy to Infinite Alphabets

We wish to use the framework of P automata to combine two approaches used earlier to handle languages over infinite alphabets with devices having a finite description, that is, with devices which can be described without the necessity of specifying an infinite set of transition rules, and to define in some reasonable way the infinite alphabet counterparts of classical language classes from the Chomsky hierarchy.

One of these approaches can be summarized as enabling the device to remember a finite number of symbols from the infinite alphabet. If we think of machines accepting strings of symbols, they might be equipped with a certain kind of data structure made of memory slots capable of storing arbitrary symbols of the infinite alphabet, and with the ability to make equality checks between the input symbols and some parts of the memory contents. Then, if we also create rules to manipulate the contents of the memory, the transitions can be given by

* Research supported in part by the Hungarian Scientific Research Fund "OTKA" grant no. F037567 and by the Alexander von Humboldt Foundation of the Federal Republic of Germany.

reference to a finite subset of the memory slots and not to the input symbols themselves.

An example of this approach is [3] where the authors extend the notion of regular languages to infinite alphabets by defining them as sets of strings accepted by so called finite memory automata, finite automata equipped with a finite set of memory registers capable of storing symbols of the input. The transitions which can also manipulate in some simple way the contents of the memory, are based on the internal state of the finite control unit, and the equality (or non-equality) of the input symbol with the contents of certain registers. A similar idea is used in [1] to extend the notion of pushdown automata and of context-free grammars to infinite alphabets. Since equality check in this framework is "easy", the language $\{a_i^{2n} \mid i, n \geq 1\}$ over the alphabet $\Sigma = \{a_i \mid i \geq 1\}$, for example, can easily be accepted by finite memory automata. On the other hand, to capture relationships of the symbols other than equality or non-equality is "hard", the language $\{a_{2i} \mid i \geq 1\}$ over Σ, for example, cannot be characterized with any of the above mentioned devices.

The second approach for handling infinite alphabets can be summarized as coding the symbols using a finite alphabet and then working with the corresponding code-word language (over a finite alphabet) instead of the original language (which is over an infinite alphabet).

As an example, we could mention [5] where a symbol a_i from the infinite alphabet $\Sigma = \{a_i \mid i \geq 1\}$ is coded as the word 0^i1 over the binary alphabet $\{0, 1\}$, and then a language over the infinite alphabet Σ is defined to be regular (or context-free), if the corresponding code-word language over the binary alphabet $\{0, 1\}$ is, in the conventional sense, regular (or context-free). This approach has some advantages, the language $\{a_{2i} \mid i \leq 1\}$, for example, which cannot be characterized with finite memory automata, is clearly regular according to this definition. On the other hand, the equality check of symbols proves to be "hard" in this framework, the language $\{a_i a_i \mid i \geq 1\}$ for example, is in this sense a non-regular, or $\{a_i a_i a_i \mid i \geq 1\}$ is a non-context-free language.

In the present paper, we propose the combination of the previously described approaches: Instead of just coding or just remembering symbols, we propose to remember the codes of symbols. In some sense, this can not only be seen as a proposal which eliminates certain shortcomings of the above outlined concepts, but also as a more "realistic" description of the act of remembering a symbol from an infinite alphabet: The remembered object needs to be somehow denoted, and for this, a finite collection of signs, elements of a notation, can be used, thus, when we think of storing symbols, we really mean to think of storing code-words corresponding to symbols, and exactly this fact is expressed in our proposed model.

To explore these ideas, we will use the framework of membrane systems, or more precisely of P automata, which provide a very natural machinery to capture the above described concepts. The introduction of P automata in [2] was motivated by the idea of using P systems as language acceptors while keeping the machinery as simple as possible. The objects in a P automaton may move

through the membranes from region to region, but they may not be modified during the functioning of the systems, and furthermore, the "words" accepted by a P automaton correspond to the sequences of multisets containing the objects entering from the environment in each step of the evolution of the system.

Although the number of different objects used by the system, that is, the number of elements of the object alphabet of the system is finite, the number of possible multisets of objects entering the skin membrane in one computational step can be infinite. The reason for this property lies in the parallelism of the application of the evolution rules (which are communication rules in this case). The number of objects manipulated by one rule is finite, but since they can be applied in any number of "copies", the number of objects affected by the rules in one computational step can be arbitrary high, thus the potential number of objects requested by the rules of the skin membrane to enter the system is unbounded. Because of the infinite number of potential input multisets, it is rather natural to consider a P automaton as a machine working with strings of symbols over infinite alphabets.

In the following we define a quite restrictive class of such P automata, which we call P finite automata, to obtain a reasonably simple, but on the other hand, a still rather complex class of languages over countably infinite alphabets, and examine how appropriate candidate this language class is for being called the "infinite alphabet counterpart" of regular languages. To explore the relationship of our P automaton based model and the ones based on more conventional type of automata, we also define restricted variants of register machines which have additional capabilities for dealing with countably infinite alphabets, but a very restricted way of using the registers, the motivation of these restrictions being to formalize the properties of the P automaton based model in a more conventional-like automata theoretical framework.

We will show that the languages over finite alphabets contained by the language class accepted by P finite automata are precisely the regular languages, thus, since all infinite alphabet languages mentioned above are also contained in this class, with P finite automata we can obtain a more adequate generalization of finite automata, and of regular languages, to infinite alphabets, then with any of the above mentioned approaches.

2 Preliminaries and Definitions

We first recall the notions and the notations we use. The reader is assumed to be familiar with the basics of formal language theory, for details see [8]. Let Σ be a not necessarily finite, but countable set of symbols called alphabet. Let Σ^* be the set of all words over Σ, that is, the set of finite strings of symbols from Σ, and let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$ where ε denotes the empty word. For any set $A \subseteq \Sigma$ let $erase_A : \Sigma^* \to (\Sigma - A)^*$ with $erase_A(x) = x$ for $x \in \Sigma - A$, and $erase_A(x) = \varepsilon$ for $x \in A$.

Let V be a set of objects, and let $\mathbb{N}$ denote the set of nonnegative integers. A multiset is a mapping $M : V \to \mathbb{N}$ which assigns to each object $a \in V$ its

multiplicity $M(a)$ in M. The support of M is the set $supp(M) = \{a \mid M(a) \geq 1\}$. If $supp(M)$ is a finite set, then M is called a finite multiset. The set of all finite multisets over the set V is denoted by V°.

We say that $a \in M$ if $M(a) \geq 1$. For $M_1, M_2 : V \to \mathbb{N}$, the containment relation $M_1 \subseteq M_2$ holds if for all $a \in V$, $M_1(a) \leq M_2(a)$. The union of M_1 and M_2 is defined as $(M_1 \cup M_2) : V \to \mathbb{N}$ with $(M_1 \cup M_2)(a) = M_1(a) + M_2(a)$ for all $a \in V$, the difference is defined for $M_2 \subseteq M_1$ as $(M_1 - M_2) : V \to \mathbb{N}$ with $(M_1 - M_2)(a) = M_1(a) - M_2(a)$ for all $a \in V$, and the intersection is $(M_1 \cap M_2) : V \to \mathbb{N}$ with $(M_1 \cap M_2)(a) = min(M_1(a), M_2(a))$ for $a \in V$, where $min(x, y)$ denotes the minimum of $x, y \in \mathbb{N}$. We say that M is empty if its support is empty, $supp(M) = \emptyset$.

A multiset M over the finite set of objects V can be represented as a string w over the alphabet V with $|w|_a = M(a)$ where $a \in V$ and $|w|_a$ denotes the number of occurrences of the symbol a in the string w, and with ε representing the empty multiset. In the following we sometimes identify the finite multiset of objects $M : V \to \mathbb{N}$ with the word w over V representing M, thus we write $w \in V^\circ$, or sometimes we enumerate the not necessarily distinct elements $a_1, \ldots, a_n$ of a multiset as $M = \{\{a_1, \ldots, a_n\}\}$, by using double brackets to distinguish from the usual set notation.

A P system is a structure of hierarchically embedded membranes, each having a label and enclosing a region containing a multiset of objects and possibly other membranes. The out-most membrane which is unique and usually labeled with 1, is called the skin membrane. The membrane structure is denoted by a sequence of matching parentheses where the matching pairs have the same label as the membranes they represent. If membrane i contains membrane j, and there is no other membrane, k, such that k contains j and i contains k, then we say that membrane i is the parent membrane of j, denoted as $parent(j) = i$.

The evolution of the contents of the regions of a P system is described by rules associated to the regions. Applying the rules synchronously in each region, the system performs a computation by passing from one configuration to another one. Several variants of the basic notion have been introduced and studied proving the power of the framework, see the monograph [7] for a summary of notions and results of the area. In the following we concentrate on communication rules called symport or antiport rules.

A symport rule is of the form (x, in) or $(x, out), x \in V^\circ$. If such a rule is present in a region i, then the objects of the multiset x can enter from the parent region or can leave to the parent region, respectively. An antiport rule is of the form $(x, in; y, out), x, y \in V^\circ$, in this case, objects of x enter from the parent region and in the same step, objects of y leave to the parent region. All types of these rules might be equipped with a promoter or inhibitor multiset, denoted as $(x, in)|_Z, (x, out)|_Z$, or $(x, in; y, out)|_Z$, with $x, y \in V^\circ, Z \in \{z, \neg z \mid z \in V^\circ\}$, where if $Z = z$ then the rules can only be applied if region i contains the objects of multiset z, or if $Z = \neg z$, then region i must not contain any of the elements of z. (For more on symport/antiport see [6], for the use of promoters see [4].)

A *P automaton* is $\Gamma = (V, \mu, (w_1, P_1, F_1), \ldots, (w_n, P_n, F_n))$ where $n \geq 1$ is the number of membranes, V is a finite set of objects, μ is a membrane structure of n membranes with membrane 1 being the skin membrane, and for all $i, 1 \leq i \leq n$,

- $w_i \in V^\circ$ is the initial contents (state) of region i, that is, it is the finite multiset of all objects contained initially by region i,
- P_i is a finite set of communication rules associated to membrane i, they can be symport rules or antiport rules, with or without promoters or inhibitors, as above, and
- $F_i \subseteq V^\circ$ is a finite set of finite multisets over V called the set of final states of region i. If $F_i = \emptyset$, then all the states of membrane i are considered to be final.

To simplify the notations we denote symport and antiport rules with or without promoters/inhibitors as $(x, in; y, out)|_Z$, $x, y \in V^\circ, Z \in \{z, \neg z \mid z \in V^\circ\}$, thus we also allow x, y, z to be the empty multiset/empty string. If $y = \varepsilon$ or $x = \varepsilon$, then the notation above denotes the symport rule $(x, in)|_Z$ or $(y, out)|_Z$, respectively, if $z = \varepsilon$, then the rules above are without promoters or inhibitors, they can also be denoted as $(x, in; y, out)$. We might also exchange the order of multisets to be sent out and to be imported, that is, the rule $(x, out; y, in)|_Z$ is equivalent to $(y, in; x, out)|_Z$, with x, y, Z as above.

The n-tuple of finite multisets of objects present in the n regions of the P automaton Γ describes a *configuration* of Γ, the n-tuple $(w_1, \ldots, w_n) \in (V^\circ)^n$ is the initial configuration.

We say that a promoter or inhibitor Z of a rule $(x, in; y, out)|_Z \in P_i$ is consistent with a configuration $(w_0, w_1, \ldots, w_n)$, if it permits the application of the rule, that is, either $Z = z \in V^\circ$ and $z \subseteq w_i$, or $Z = \neg z$, $z \in V^\circ$ and $z \cap w_i = \varepsilon$.

The application of the rules can take place in a sequential, or in a maximally parallel manner. Here we only consider parallel rule application, but we keep the subscript "*par*" in order to emphasize that maximal parallel application is just one of the different possibilities.

The transition mapping of a P automaton is a partial mapping $\delta_{par} : V^\circ \times (V^\circ)^n \to 2^{(V^\circ)^n}$. These mappings are defined implicitly by the rules of the sets P_i, $1 \leq i \leq n$. For a configuration $(u_1, \ldots, u_n)$,

$$(u'_1, \ldots, u'_n) \in \delta_{par}(u, (u_1, \ldots, u_n))$$

holds, that is, while reading the input $u \in V^\circ$ the automaton may enter the new configuration $(u'_1, \ldots, u'_n) \in (V^\circ)^n$, if there exist rules as follows.

- For all $i, 1 \leq i \leq n$, there is a multiset of rules $R_i = \{\{r_{i,1}, \ldots, r_{i,m_i}\}\}$, where $r_{i,j} = (x_{i,j}, in; y_{i,j}, out)|_{Z_{i,j}} \in P_i$ and $Z_{i,j}$, $1 \leq j \leq m_i$, is consistent with u_i, satisfying the conditions below, where x_i, y_i denote the multisets $\bigcup_{1 \leq j \leq m_i} x_{i,j}$ and $\bigcup_{1 \leq j \leq m_i} y_{i,j}$, respectively. Furthermore, there is no rule occurrence $r \in P_j$, being consistent with u_j, $1 \leq j \leq n$, with the additional restriction that if $r \in P_1$ then $r \neq (x, in)|_Z$, such that the rule multisets R'_i with $R'_i = R_i$ for $i \neq j$ and $R'_j = \{\{r\}\} \cup R_j$, also satisfy the conditions.

The conditions are given as

1. $x_1 = u$, and
2. $(\bigcup_{parent(j)=i} x_j) \cup y_i \subseteq u_i,\ 1 \leq i \leq n,$

and then the new configuration is obtained by

$$u'_i = u_i \cup x_i - y_i \cup \bigcup_{parent(j)=i} y_j - \bigcup_{parent(j)=i} x_j,\ 1 \leq i \leq n.$$

Note that we allow the use of rules of type $(x, in)|_Z$ in the skin membrane in such a way that the application of any number of copies of these rules is considered to be maximally parallel.

Let us extend δ_{par} to $\bar{\delta}_{par}$, a function mapping $(V^\circ)^*$, the sequences of finite multisets over V, and $(V^\circ)^n$, the configurations of Γ, to new configurations.

1. $\bar{\delta}_{par}(v, (u_1, \ldots, u_n)) = \delta_{par}(v, (u_1, \ldots, u_n)),\ v, u_i \in V^\circ,\ 1 \leq i \leq n$, and
2. $\bar{\delta}_{par}((v_1) \ldots (v_{s+1}), (u_1, \ldots, u_n)) = \bigcup \delta_{par}(v_{s+1}, (u'_1, \ldots, u'_n))$
   for all $(u'_1, \ldots, u'_n) \in \bar{\delta}_{par}((v_1) \ldots (v_s), (u_1, \ldots, u_n)),\ v_j, u_i, u'_i \in V^\circ,$
   $1 \leq i \leq n,\ 1 \leq j \leq s+1.$

Note that we use brackets in the multiset sequence $(v_1) \ldots (v_{s+1}) \in (V^\circ)^*$ in order to distinguish it from the multiset $v_1 \cup \ldots \cup v_{s+1} \in V^\circ$.

The sequence of multisets of objects accepted by the P automaton is the sequence of input multisets consumed by the skin membrane during the sequence of computational steps while the system reaches a final state, a configuration where for all j with $F_j \neq \emptyset$, the contents $u_j \in V^\circ$ of membrane j is "final", i.e., $u_j \in F_j$.

Let Γ be a P automaton as above with initial configuration $(w_1, \ldots, w_n)$, let Σ be an alphabet, and let $f : V^\circ \longrightarrow \Sigma \cup \{\varepsilon\}$ be a mapping with $f(x) = \varepsilon$ if and only if $x = \varepsilon$.

We obtain the words of the *language accepted by* Γ as the images of the accepted multiset sequences, that is,

$$L(\Gamma, f) = \{f(v_1) \ldots f(v_s) \in \Sigma^* \mid (u_1, \ldots, u_n) \in \bar{\delta}_{par}((v_1) \ldots (v_s), (w_1, \ldots, w_n))$$
$$\text{with } u_j \in F_j \text{ for all } j \text{ with } F_j \neq \emptyset,\ 1 \leq j \leq n,\ 1 \leq s\}.$$

Obviously, the choice of the mapping f is essential. It has to be "easily" computable because the power of the P automaton should be provided by the underlying membrane system and not by f itself. The notion of "easiness", however, greatly depends on the context we are working in, so we do not give it a general specification here.

3 P Finite Automata and Restricted Register Automata

Now we introduce a restricted variant of P automata which we call P finite automata. The object alphabet of a P finite automaton contains a distinct element which is the only one that can appear in an arbitrary number of copies

inside the membrane structure and can be present only in the skin membrane. The other objects can move around through the regions, but they can only be exported, thus, their number of occurrences cannot increase during any computation. These properties are ensured by the very special and very simple form of rules which can be used by the system.

Definition 1 A *P finite automaton* (PFA in short) is a P automaton $(V \cup \{a\}, \mu, (w_1, P_1, F_1), \ldots, (w_n, P_n, F_n))$ where $V \cup \{a\}$ is a finite alphabet with a distinct element denoted by a, μ is a membrane structure of n membranes, and for $1 \leq i \leq n$, $w_i \in V^\circ$ is the initial multiset of region i, F_i is the set of final states for region i, and P_i is a finite set of rules associated to region i where

- if $i \neq 1$, P_i contains rules of the form $(x, in; y, out)|_Z$ with $Z \in \{z, \neg z\}$, $x, y, z \in V^\circ$, and
- P_1 contains rules of the form $(x, in; y, out)|_Z$ where $x \in \{a\}^\circ$, $y \in (V \cup \{a\})^\circ$, $Z \in \{z, \neg z\}$, $z \in V^\circ$.

Thus, a PFA can only input multisets of the symbol a from the environment and these symbols can only remain in the skin region or be sent back to the environment, but they cannot enter regions i with $i \geq 2$. The symbol a is the only one which can appear in arbitrary many copies inside the system, the number of the other letters is at most as many as in the initial configuration, it might only decrease during the computation.

To obtain the accepted word we set up a correspondence between the set of possible input multisets and the countably infinite alphabet $\Sigma = \{a_i \mid i \geq 1\}$ based on the number of a symbols in the input. If $M \in \{a\}^\circ$ is an input multiset containing k copies of the symbol a, that is, $M(a) = k$, then M is mapped to $a_k \in \Sigma$. Thus, a sequence of input multisets corresponds to a sequence of letters from Σ. This is formalized in the following definition.

Definition 2 Let $\Sigma = \{a_i \mid i \geq 1\}$ be a countably infinite alphabet, and let Γ be a PFA as above. The language over Σ accepted by Γ, denoted as $L(\Gamma)$, is defined as follows.

$$L(\Gamma) = \{a_{i_1} a_{i_2} \ldots a_{i_s} \in \Sigma^* \mid (u_1, \ldots, u_n) \in \bar{\delta}_{par}((v_1) \ldots (v_s), (w_1, \ldots, w_n)) \text{ with } u_j \in F_j \text{ for all } j \text{ with } F_j \neq \emptyset,\ 1 \leq j \leq n, \text{ and } f(v_j) = a_{i_j},\ 1 \leq j \leq s\}$$

where f is defined as $f : \{a\}^\circ \to \Sigma \cup \{\varepsilon\}$, with $f(M) = a_i$ where $M(a) = i$ for a multiset $M(a) > 0$, and $f(M) = \varepsilon$ for M with $M(a) = 0$.

Let $\mathcal{L}(PFA)$ denote the class of languages accepted by P finite automata.

Now we define restricted two register automata, or RRA in short, an other type of machine model to capture the capabilities of P finite automata. As we will show, the two models are equivalent, that is, they characterize the same class of infinite alphabet languages.

An RRA has a finite control unit and two registers holding nonnegative integer values. The machine is capable of subtracting certain values from the first register and adding other values to the second one, based on rules specifying an internal state and two integers, until no more modifications of this type are possible. At this point, if the value of the second register is k, the machine reads an input symbol a_k from the countably infinite input alphabet while changing its state, empties the second register, and adds its value to the contents of the first register.

Definition 3 A *restricted register finite automaton*, or RRA in short, is a construct $M = (\Sigma, Q, P, \leq, q_0, r_0, F)$ where $\Sigma = \{a_i \mid i \geq 1\}$ is a countably infinite alphabet, Q is a finite set of states, $q_0 \in Q$ is the initial state, $r_0 \in \mathbb{N}$ is a nonnegative integer, the initial contents of the first register, $F \subseteq Q$ is the set of final states, P is a finite set of instructions of the form $(q; i_1, i_2)$, or $(q, q'; i_1, i_2)$ where $q, q' \in Q,\ i_1, i_2 \in \mathbb{N}$, and $\leq$ is a transitive relation defined on the rules of the form $(q, q'; i_1, i_2) \in P$ with the property that $(q_1, q_2; i_1, i_2) \leq (q_1', q_2'; i_1', i_2')$ implies $q_1 = q_1'$ and $i_1 \leq i_1'$ (but not the other way around, i.e., rule pairs with the above properties are not necessarily elements of the relation $\leq$). Furthermore, $(*)$ if $(q, q'; i_1, i_2) \in P$ with $i_1 > 0$ or $i_2 > 0$, then the internal state q is not reachable from q', that is, $q' \neq q$ and there is no sequence of states

$$q' = q_1, q_2, \ldots, q_t = q,\ q_j \in Q,\ 1 \leq j \leq t,$$

for any $t \geq 2$, such that $(q_j, q_{j+1}; i_{j_1}, i_{j_2}) \in P$, $i_{j_1}, i_{j_2} \in \mathbb{N}$, for all $1 \leq j \leq t-1$.

A configuration of M is a triple (q, r_1, r_2) with the current state $q \in Q$, and the current register contents $r_1, r_2 \in \mathbb{N}$. Given a configuration (q, r_1, r_2), the register contents can be modified obtaining (q, r_1', r_2'), denoted as

$$(q, r_1, r_2) \Rightarrow (q, r_1', r_2'),$$

if there is an instruction $(q; i_1, i_2) \in P$ with $r_1 - i_1 \geq 0$, and $r_1' = r_1 - i_1$, $r_2' = r_2 + i_2$, thus, the value of i_1 is subtracted from the first register, and i_2 is added to the second register.

The internal state of the machine can be modified while reading the input symbol a_j, denoted as

$$(q, r_1, r_2) \Rightarrow^{a_j} (q', r_1', 0),$$

provided, that there is an instruction $(q, q'; i_1, i_2) \in P$, where $r_1 - i_1 \geq 0$. Then $j = r_2 + i_2$ and $r_1' = r_1 - i_1 + j$. Thus, when an instruction of this type is used, then after modifying the register contents using the values i_1, i_2, the resulting contents of the second register, denoted above as j, is added to the value of the first register, the symbol a_j is read from the input, and the contents of the second register is changed to 0. Note that if $j = r_2 + i_2 = 0$, then the automaton may change its state without reading any input symbol, or in other words, $a_0 = \varepsilon$. Note also, that if any of i_1 or i_2 in an instruction $(q, q'; i_1, i_2) \in P$ is not zero, then the instruction can only be used once during any computation of the automaton

since, due to the constraint marked with $(*)$ and described above, after being in state q', the internal control can never enter state q again.

Let us denote the reflexive and transitive closure of $\Rightarrow$ by $\Rightarrow^*$, and let a transition of M be defined as

$$(q, r_1, 0) \vdash^{a_j} (q', r_1', 0),$$

if $(q, r_1, 0) \Rightarrow^* (q, r_1'', r_2'') \Rightarrow^{a_j} (q', r_1', 0)$, and the following properties hold: If $(q, q'; i_1, i_2)$ is the rule applied in $(q, r_1'', r_2'') \Rightarrow^{a_j} (q', r_1', 0)$ above, then there is no rule $(q; i_1', i_2') \in P$ with $i_1 + i_1' \leq r_1''$, and there is no $(q, q''; i_1'', i_2'') \in P$ such that $(q, q'; i_1, i_2) \leq (q, q''; i_1'', i_2'')$ and $i_1'' \leq r_1''$.

Note, that if $(q, r_1, 0) \vdash^{a_j} (q, r_1', 0)$ holds and there is a rule of the form $(q; 0, i_2) \in P$, then $(q, r_1, 0) \vdash^{a_{j+k \cdot i_2}} (q', r_1' + k \cdot i_2, 0)$ also holds, for any $k \in \mathbb{N}$.

Definition 4 Let M be a RRA as above. The *language* accepted by M is defined as

$$L(M) = \{w = x_1 \dots x_n \in \Sigma^* \mid (q_0, r_0, 0) = C_0 \vdash^{x_1} C_1 \vdash^{x_2} \cdots \vdash^{x_{n-1}} C_{n-1} \vdash^{x_n} C_n = (q_f, r, 0) \text{ where } q_f \in F\}.$$

Let the class of languages (over countably infinite alphabets) accepted by RRA be denoted by $\mathcal{L}(RRA)$.

Now we show that P finite automata and restricted register automata are equivalent, that is, they accept the same class of infinite alphabet languages.

Lemma 1 $\mathcal{L}(RRA) = \mathcal{L}(PFA)$.

Proof. We first show that $\mathcal{L}(RRA) \subseteq \mathcal{L}(PFA)$. Let $\Sigma = \{a_i \mid i \geq 1\}$ be a countably infinite alphabet, and let $L \subseteq \Sigma^*$ be a language accepted by the RRA $M = (\Sigma, Q, P, \leq, q_0, r_0, F)$. We construct a P finite automaton Γ, such that $L(M) = L(\Gamma)$.

Let $\Gamma = (V \cup \{a\}, [\ ]_2\ ]_1, (w_1, P_1, \emptyset), (w_2, P_2, F_2))$ where $V = Q \cup \{A, B\} \cup \{(q, q'; i_1, i_2) \in P\}$ for the additional symbols $A, B \notin Q$, and

$$\begin{aligned}
w_1 &= Aa^{r_0} \cup \{\{q \mid q \in Q\}\},\\
P_1 &= \{(a^{i_1}, out; a^{i_2}, in)|_{(q,q';k,l)q} \mid (q; i_1, i_2) \in P\} \cup\\
&\quad \{(qB, out)|_{(q,q';k,l)q} \mid (q, q'; k, l) \in P,\ k + l > 0\}\\
&\quad \{(a^{i_1}q, out; a^{i_2}, in)|_{(q,q';i_1,i_2)q} \mid (q, q'; i_1, i_2) \in P,\ i_1 + i_2 > 0\} \cup\\
&\quad \{(a^{i_1'-i_1}B, out)|_{(q,q';i_1,i_2)q} \mid (q, q'; i_1, i_2) \leq (q, q''; i_1', i_2') \text{ for some } (q, q''; i_1', i_2') \in P\},\\
w_2 &= \{\{(q, q'; i_1, i_2) \in P\}\} \cup \{\{B\}\},\\
P_2 &= \{((q_0, q; i_1, i_2)B, out; A, in) \mid q \in Q, (q_0, q; i_1, i_2) \in P\} \cup\\
&\quad \{((q', q''; i_1, i_2), out; (q, q'; i_1, i_2), in)\} \cup\\
&\quad \{((q, q_f; i_1, i_2)B, in) \mid q_f \in F\},\\
F_2 &= w_2AB.
\end{aligned}$$

We claim that M' accepts the same words as M. To see this, consider the following. A configuration of Γ containing $(q, q'; i_1, i_2)$, q, and a^{r_1} in the first region corresponds to a configuration $(q, r_1, 0)$ of M. Being in a configuration $(q, r_1, 0)$, M can apply its rules $(q; i_1, i_2)$, each of which has a corresponding rule $(a^{i_1}, out; a^{i_2}, in)|_{(q,q';k,l)q}$ in Γ, and then one rule $(q, q'; i_1, i_2)$ which has in Γ the counterpart $(a^{i_1}q, out; a^{i_2}, in)|_{(q,q';i_1,i_2)q}$ if $i_1 + i_2 > 0$. The repeated application of the rules of M is simulated by the maximal parallel application of the corresponding rules of Γ.

The second region of Γ is responsible for changing the rule symbols in the first region, it is achieved with $((q', q''; i_1, i_2), out; (q, q'; i_1, i_2), in)$. The export of the state symbols q to the environment makes sure that Γ cannot reach a configuration corresponding to the state q of M after simulating a rule of the form $(q, q'; i_1, i_2)$ with $i_1 + i_2 > 0$. If such a rule should be simulated and q is not sent out the environment with $(a^{i_1}q, out; a^{i_2}, in)|_{(q,q';i_1,i_2)q}$, then the rule $(qB, out)|_{(q,q';k,l)q}$ must be used, which sends B out to the environment, making it impossible for Γ to reach a final configuration. The symbol B is also sent out, thus, a final configuration cannot be reached, in the case when the simulated rule $(q, q'; i_1, i_2)$ could not be applied in M because of the existence of $(q, q'; i_1, i_2) \leq (q, q''; i'_1, i'_2)$ where $(q, q''; i'_1, i'_2)$ could also be applicable.

Let us now show that $\mathcal{L}(PFA) \subseteq \mathcal{L}(RRA)$. Let $\Sigma = \{a_i \mid i \geq 1\}$ be a countably infinite alphabet, and let $L \subseteq \Sigma^*$ be a language accepted by the P finite automaton $\Gamma = (V \cup \{a\}, \mu, (w_1, P_1, \emptyset), \ldots, (w_n, P_n, F_n))$. We construct an RRA M, such that $L(M) = L(\Gamma)$.

First, let for any $V' \in V^\circ$ (finite) multiset of objects $DIST(V')$ denote the set of possible "distributions" of the elements of V' in the different membranes of μ, that is, let $DIST(V') = \{(v_1, \ldots, v_n) \mid \bigcup_{i=1}^n v_i = V'\}$. Let also, for any $d \in DIST(V')$, the set $NEXT(d) \subseteq DIST(V')$ denote the set of those distributions of the elements of V' which can be reached from $d \in DIST(V')$ by the maximal parallel application of rules of the form $(x, out; y, in)|_Z \in P_i,\ i \geq 2,\ Z \in \{z, \neg z\}$, $x, y, z \in V^\circ$, in the membranes $i,\ i \geq 2$, of M. Note that, given V' and $\bigcup_{i=2}^n P_i$, $DIST(V')$ and $NEXT(d)$ for any $d \in DIST(V')$ are easily constructible.

For any rule set P_i, $1 \leq i \leq n$, and distribution $d \in DIST(V')$, $V' \in V^\circ$, let $P_i(d)$ denote the set of those rules $(x, out; y, in)|_Z \in P_i$ where Z is consistent with $d \in DIST(V')$.

Now we construct the simulating RRA. Let $M = (\Sigma, Q, P, \leq, q_0, r_0, F)$. The states of the finite control correspond to the possible distributions of elements from V in the different membranes of M, the contents of the first counter corresponds to the number of a symbols in the skin membrane. M is constructed as follows.

Let $W = erase_{\{a\}}(\bigcup_{i=1}^n w_i)$, thus W is the multiset of objects from V occurring in the regions of the P finite automata Γ. Let $Q = \bigcup_{W' \subseteq W} DIST(W')$, $q_0 = (w'_1, w_2, \ldots, w_n)$, and $r_0 = k$, where $w'_1 = erase_{\{a\}}(w_1)$ and $w_1 = a^k w'_1$. Let also

$$F = \{(v_1, \ldots, v_n) \in Q \mid v_i \in F_i \text{ for all } i \text{ with } F_i \neq \emptyset,\ 1 \leq i \leq n\},$$

and let the set of instructions be the following.

$$P = \{(q, i_1, i_2) \mid (a^{i_1}, out; a^{i_2}, in)|_Z \in P_1(q),\ q \in Q\} \cup$$
$$\{(q, q', i_1, i_2) \mid \text{ there is a multiset of rules}$$
$$(a^{i_{1,j}} x_j, out; a^{i_{2,j}}, in)|_{Z_j} \in P_1(q),\ x_j \in (V)^\circ, x_j \neq \varepsilon,\ 1 \leq j \leq t,$$
$$\text{for } q = (u_1, \ldots, u_n) \text{ and } i_1 = \Sigma_{j=1}^{t} i_{j,1},\ i_2 = \Sigma_{j=1}^{t} i_{j,2}, \text{ with}$$
$$q' = (v_1, \ldots, v_n) \in NEXT(u_1 - x, u_2, \ldots, u_n) \text{ where } x = \bigcup_{j=1}^{t} x_j\}.$$

Furthermore, if for some state $q = (u_1, \ldots, u_n)$, there are two multisets of rules

$$(a^{i_{1,j}} x_j, out; a^{i_{2,j}}, in)|_{Z_j} \in P_1(q),\ 1 \leq j \leq t, \text{ and}$$
$$(a^{i'_{1,j}} x'_j, out; a^{i'_{2,j}}, in)|_{Z'_j} \in P_1(q),\ 1 \leq j \leq t',$$

such that for $x = \bigcup_{j=1}^{t} x_j,\ x' = \bigcup_{j=1}^{t'} x'_j$, the property that $x \cup x' \subseteq u_1$ holds, and there exists a state $\bar{q} = (v_1, \ldots, v_n)$ such that for $q_1 = (u_1 - x, u_2, \ldots, u_n)$ and $q_2 = (u_1 - x - x', u_2, \ldots, u_n)$,

$$\bar{q} \in \{(s_1 - x', s_2, \ldots, s_n) \mid (s_1, \ldots, s_n) \in NEXT(q_1)\} \cap NEXT(q_2),$$

then let

$$(q, (v_1 + x', \ldots, v_n); i_1, i_2) \leq (q, (v_1, \ldots, v_n); i_1 + i'_1, i_2 + i'_2)$$

for $i_1 = \Sigma_{j=1}^{t} i_{j,1},\ i_2 = \Sigma_{j=1}^{t} i_{j,2}, i'_1 = \Sigma_{j=1}^{t'} i'_{j,1},\ i'_2 = \Sigma_{j=1}^{t'} i'_{j,2}$.

To see how these rules simulate the P finite automaton Γ, consider the following. A configuration $(a^{r_1} v_1, \ldots, v_n)$, $v_i \in V^\circ,\ 1 \leq i \leq n$, of Γ corresponds to the configuration $(q, r_1, 0)$ of M where $q = (v_1, \ldots, v_n)$.

Consider now a computational step of Γ, in which the maximal parallel application of rules of the form $(a^{i_1}, out; a^{i_2}, in)|_Z$ occurs. This can be simulated by the repeated applications of rules of the form $(q; i_1, i_2)$ and then with $(q, q'; 0, 0)$ where $q' = (v'_1, \ldots, v'_n)$ and the configuration $(v'_1, \ldots, v'_n)$ is one of those which can be obtained by applying the rest of the rules in the rest of the regions of Γ.

If not only rules of the form $(a^{i_1}, out; a^{i_2}, in)|_Z$, but also rules of the form $(a^{i_1} u, out; a^{i_2}, in)|_Z$ with $u \in V^\circ,\ u \neq \varepsilon$, are applied, then instead of $(q, q'; 0, 0)$, a rule of the form $(q, q'; i_1, i_2),\ i_1 + i_2 > 0$, corresponding to a subset of the applicable rules of this form are used, q' corresponding to a possible distribution of the remaining elements of V which were not sent out to the environment by the corresponding rule set of Γ.

The construction of the relation $\leq$ ensures that M uses only rules which correspond to maximal subsets of the applicable rules of Γ. □

4 $\mathcal{L}(PFA)$ as the Extension of the Regular Language Class to Infinite Alphabets

First we show that all "conventional", finite alphabet regular languages can be accepted by P finite automata.

Lemma 2 $\mathcal{L}(REG) \subset \mathcal{L}(PFA)$.

Proof. Let $M = (\Sigma_1, Q, \delta, q_0, F)$ be a deterministic finite automaton over the finite input alphabet $\Sigma_1 = \{a_1, \ldots, a_k\}$, with set of states Q, transition relation $\delta : Q \times \Sigma \to Q$, initial state $q_0 \in Q$, and set of final states $F \subseteq Q$. Let the regular language accepted by M be denoted by $L(M)$. Let us also assume that $q_0 \notin F$, and furthermore, there is no $q \in Q,\ a_i \in \Sigma_1$, such that $\delta(q, a_i) = q_0$.

Let $TRANS = \{[q_1, a_i, q_2] \mid \delta(q_1, a_i) = q_2\}$ and let $TRANS' = \{[q_1, a_i, q_2]' \mid \delta(q_1, a_i) = q_2\}$, a primed and a non-primed set of triples corresponding to the transitions of M. Let us also denote for any $t' \in TRANS'$, by $next(t')$ the set of those non-primed transition symbols which correspond to transitions that can follow the transition denoted by the primed symbol t', that is, $next(t') = \{[q_2, a_j, q_3] \in TRANS \mid t' = [q_1, a_i, q_2]'\}$. For any non-primed $t \in TRANS$, we always denote the corresponding primed symbol from $TRANS'$ by t'.

Now we construct a P finite automaton Γ, such that $L(\Gamma) = L(M)$. Let $\Gamma = (V \cup \{a\}, [\ [\ ]_2\ ]_1, (w_1, P_1, \emptyset), (w_2, P_2, F_2))$ where $V = TRANS \cup TRANS' \cup \{\#\}$ and

$$
\begin{aligned}
w_1 &= a\#,\\
P_1 &= \{(a^i, in; a, out)|_t, (a^{i-1}, out)|_{t'} \mid t = [q_j, a_i, q_k],\ i > 1\} \cup\\
&\quad \{(a, in; a, out)|_t \mid t = [q_j, a_1, q_k]\},\\
\\
w_2 &= \{\{t, t' \mid t \in TRANS\}\},\\
P_2 &= \{(\#, in; t_0, out) \mid t_0 = [q_0, a_i, q]\} \cup\\
&\quad \{(t, in; t', out), (t', in; s, out) \mid t \in TRANS,\ s \in next(t')\},\\
F_2 &= \{\ \{\{t, t' \mid t \in TRANS\}\} - \{\{s'\}\} \mid \text{for}\\
&\qquad \text{all } s' \in TRANS' \text{ such that } s' = [q, a_i, q_f]',\ q_f \in F\}.
\end{aligned}
$$

It is not difficult to see how Γ simulates M. The rules of the second region are responsible for sending symbols representing the transitions of M into the first region in an order which is a legal transition sequence of M, and the rules of the first region import from the environment the necessary number of as corresponding to the input symbol belonging to the simulated transition. □

Now we show that the class of finite alphabet languages accepted by P finite automata is precisely the class of regular languages.

Lemma 3 *If L is a language over a finite alphabet, such that $L \in \mathcal{L}(PFA)$, then $L \in \mathcal{L}(REG)$.*

Proof. Let $M = (\Sigma, Q, P, \leq, q_0, r_0, F)$ be a RRA and let $L(M) \subseteq \Sigma_1^*$ where $\Sigma_1 \subseteq \Sigma$ is a finite alphabet. Note that the existence of a rule of the form $(q, 0, i_2) \in P$ would contradict the fact that $L(M)$ is a language over a finite alphabet, thus, we can assume that there are no rules of this form in the rule set P. Let us also assume, without the loss of generality, that $q_0 \notin F$. In the following, we construct a nondeterministic finite automaton M', such that $L(M') = L(M)$.

Let for all $q \in Q$, P_q be the set of those rules for state q which decrement the first register, $P_q = \{(q; r_1, r_2), (q, q'; r_1, r_2) \in P \mid r_1 > 0\}$. Let ∞ be a symbol, such that for all $n \in \mathbb{N}$, $n \leq \infty$, and let $rest(q) \in \mathbb{N} \cup \{\infty\}$ be

$$rest(q) \begin{cases} min(\{r_1 \mid (q; r_1, r_2), (q, q'; r_1, r_2) \in P_q\}) - 1 & \text{if } P_q \neq \emptyset, \\ \infty & \text{if } P_q = \emptyset, \end{cases}$$

that is, $rest(q)$ denotes the maximal value which can be stored in the first register when M is in sate q, such that no rule which decrements the first register can be applied because the contents of the first register is less then necessary. If for a certain $q \in Q$, the set P_q is empty, then $rest(q) = \infty$.

Let $M' = (\Sigma_1, Q', \delta, q_0', F')$ be a finite automaton with input alphabet Σ_1, state set Q', transition relation $\delta : Q' \times \Sigma_1 \to 2^{Q'}$, initial state $q_0' \in Q'$, and set of final states $F' \subseteq Q'$.

The states of M' are essentially elements of $Q \times \mathbb{N}$,

$$Q' = \{(q, i) \mid q \in Q, 0 \leq i \leq max(\{rest(q) \mid q \in Q,\ rest(q) \neq \infty\}) + |\Sigma_1| + r_0 + \Sigma_{(q,q';i_1,i_2)\in P}\ i_2\}.$$

The initial state of M' is $q_0' = (q_0, r_0)$. For the initial state, we define for all j, $0 \leq j \leq |\Sigma_1|$, that is, since in our notation $a_0 = \varepsilon$, for all symbols from $\Sigma_1 \cup \{\varepsilon\}$

$$\delta((q_0, r_0), a_j) = \{(q, i) \mid (q_0, r_0, 0) \vdash^{a_j} (q, i, 0)\}.$$

Let $Q_0' = \{(q_0, r_0)\}$, and let $Q_1 = Q_0' \cup \delta((q_0, r_0), a_j)$. Now, for all states in $(q, i) \in Q_1'$,

$$\delta((q, i), a_j) = \{(q', i') \mid (q, i, 0) \vdash^{a_j} (q', i', 0)\},$$

and $Q_2' = \bigcup_{(q,i)\in Q_1'} \delta((q, i), a_j)$.

We continue this way the definition of δ and Q_i', $i \geq 0$. That is, if we already have Q_n', then $Q_{n+1}' = \bigcup_{(q,k)\in Q_n'} \delta((q, k), a_j)$, where

$$\delta((q, k), a_j) = \{(q', k') \mid (q, k, 0) \vdash^{a_j} (q', k', 0)\}.$$

We claim that $\bigcup_{i\geq 0} Q_i'$ is a finite set, namely that $\bigcup_{i\geq 0} Q_i' \subseteq Q'$, that is, that M' is really a finite automaton.

To see this, consider a derivation

$$(q_0, r_0, 0) \vdash^{x_1} \ldots (q_i, r_i, 0) \vdash^{x_{i+1}} (q_{i+1}, r_{i+1}, 0) \ldots \vdash^{x_n} (q_n, r_n, 0) = (q_f, r_n, 0).$$

Let $(q_i, r_i, 0) \vdash^{x_{i+1}} (q_{i+1}, r_{i+1}, 0)$ be a transition with

$$(q_i, r_i, 0) \Rightarrow^* (q_i, r_i', j_1) \Rightarrow^{x_{i+1}} (q_{i+1}, r_{i+1}, 0)$$

where $x_{i+1} = a_j$, $j = j_1 + i_2$ and $r_{i+1} = r_i' + j$ if $(q_i, q_{i+1}; i_1, i_2)$ is the rule applied in the last step.

If $rest(q_i) < r_i$, then $r_i' < r_i$, thus, $r_{i+1} = r_i' + j \leq rest(q_i) + |\Sigma_1|$ which means that $(q_{i+1}, r_{i+1}) \in Q'$.

If $rest(q_i) > r_i$ then $r'_i = r_i$, $j_1 = i_1 = 0$, and $r_{i+1} = r_i + i_2$. Thus, during this transition, the value of the first register can increase with the value i_2 where $(q_i, q_{i+1}; i_1, i_2)$ with $i_2 > 0$ is the rule applied in the last step. Since these types of rules satisfy the constraint that q_i is not reachable from q_{i+1}, they can be applied only once in any transition sequence. Thus, if we start the derivation in the state (q_0, r_0) then the total increase of the first counter cannot be more then $\Sigma_{(q,q';i_1,i_2)\in P}\ i_2$, thus $r_{i+1} \leq r_0 + \Sigma_{(q,q';i_1,i_2)\in P}\ i_2$, that is, $(q_{i+1}, r_{i+1}) \in Q'$.

If we now take $F' = \{(q,i) \mid (q,i) \in Q',\ q \in F\}$, then it is clear that the finite automaton M' simulates the RRA M, since all states of F' correspond to accepting configurations of M, thus our statement is proved. □

5 Conclusion

We have presented two equivalent computing models which are able to characterize languages over infinite alphabets, we called them P finite automata and restricted register finite automata. We have shown that the languages over finite alphabets contained in the language class that these models characterize are precisely the regular languages, thus it can be seen as the extension of the class of regular languages to infinite alphabets. Without going into the details, we would like to add that the languages mentioned in the introduction as regular in the sense of [3] but not regular in the sense of [5], and vice versa, can all be accepted by our model, thus, it seems that our approach is able to eliminate at least some of the shortcomings of previous attempts to define the class of regular languages over infinite alphabets in a reasonable way. A more detailed analysis of $\mathcal{L}(PFA)$ (or equivalently $\mathcal{L}(RRA)$) remains a topic of further study.

References

1. Cheng, E. H. Y., Kaminski, M.: Context-free Languages over Infinite Alphabets. Acta Informatica **35** (1998) 245-267
2. Csuhaj-Varjú, E., Vaszil, Gy.: P Automata. In: Păun, Gh., Zandron, C. (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, and also in
   Păun, Gh., Rozenberg, G., Salomaa, A., Zandron, C. (eds.): Membrane Computing. Lecture Notes in Computer Science, Vol. 2597. Springer, Berlin (2003) 219-233
3. Kaminski, M., Francez, N.: Finite-memory Automata. Theoretical Computer Science **134** (1994) 329-363
4. Martín-Vide, C., Păun, A., Păun, Gh.: On the Power of P Systems with Symport Rules. Journal of Universal Computer Science **8**(2) (2002) 317-331
5. Otto, F.: Classes of Regular and Context-free Languages over Countably Infinite Alphabets. Discrete Applied Mathematics **12** (1985) 41-56
6. Păun, A., Păun, Gh.: The Power of Communication: P Systems with Symport/Antiport. New Generation Computing **20**(3) (2002) 295-306
7. Păun, Gh.: Membrane Computing: An Introduction. Springer, Berlin, (2002)
8. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages. Springer-Verlag, Berlin, vol. 1-3, (1997)