On Reversibility and Determinism in P Systems

Artiom Alhazov[1,2], Kenichi Morita[1]

[1] IEC, Department of Information Engineering
Graduate School of Engineering, Hiroshima University
Higashi-Hiroshima 739-8527 Japan
morita@iec.hiroshima-u.ac.jp

[2] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Academiei 5, Chişinău MD-2028 Moldova
artiom@math.md

Summary. Membrane computing is a formal framework of distributed parallel computing. In this paper we study the reversibility and maximal parallelism of P systems from the computability point of view. The notions of reversible and strongly reversible systems are considered. The universality is shown for one class and a negative conjecture is stated for a more restricted class of reversible P systems. For one class of strongly reversible P systems, a very strong limitation is shown, and it is shown that this limitation does not hold for a less restricted class.

Another concept considered is strong determinism, which is a syntactic property, as opposed to the determinism typically considered in membrane computing. A limitation is shown of one class, while a less restricted class is universal.

1 Introduction

Reversibility is an important property of computational systems. It has been well studied for circuits of logical elements ([4]), circuits of memory elements ([8]), cellular automata ([9]), Turing machines ([2], [11]), register machines ([7]). Reversibility as a syntactical property is closely related to the microscopic physical reversibility, and hence it assumes better miniaturization possibilities for potential implementation.

A slightly different view on reversible systems is given for type-0 grammars ([10]). In this case, the so-called uniquely parsable grammars are studied. In very simple words, this property (still being syntactical) implies that the generation of any word in the language is unique (modulo the order of applying the rules in case when the composition of applying them is commutative). The advantage of having such a property is that it is easier to analyze their behavior.

Clearly, this reason is valid even if the property of reversibility becomes undecidable (just like the property of determinism in certain membrane systems).

Moreover, reversibility essentially is backward determinism. Reversible P systems already were considered ([5]), but the model is energy-based (so the parallelism is invariant-driven rather than maximal) and the main result is the simulation of the Fredkin gate and thus of reversible circuits (so construction of a universal system in this way would use an infinite structure). In this paper we focus on the interplay between maximal parallelism and such fundamental notions as reversibility and determinism, from the viewpoint of computability.

It is interesting that the description of some computational systems includes the initial configuration (grammars, membrane systems), while it is not the case for many others (cellular automata, Turing machines). We generalize reversibility and determinism in such a way that these properties do not depend on the initial configurations, and call them strong. Finally, we present a number of results. In particular, we show that the power of strongly deterministic systems is weaker than that of deterministic systems, and we conjecture that also the power of strongly reversible systems is weaker than that of reversible systems.

2 Definitions

In this paper we illustrate the reversibility and determinism concepts on P systems with symport/antiport rules and one membrane, sometimes with promoters, inhibitors or priorities. For simplicity, we also assume that the environment contains an unbounded supply of all objects[3]. The system thus can be defined by the alphabet, the initial multiset, the set of rules associated to the membrane and the set of terminal objects. Throughout this paper we represent multisets by strings. The union of multisets is defined by adding multiplicities of the symbols. A comprehensive bibliography of membrane computing can be found at [13].

We write an antiport rule sending a multiset x out and bringing a multiset y in as x/y, and the symport case corresponds to $y = \lambda$. If a rule has a promoter a, we write it as $x/y|_a$. If a rule has an inhibitor a, we write it as $x/y|_{\neg a}$. The priority relationship is denoted by $>$. It is not difficult to generalize the definitions for the models with multiple membranes and changing membrane structure, but it is not important here.

We can define a P system in the above-mentioned normal form as

$$\Pi = (O, T, w, R),$$

where O is the object alphabet, T is the terminal subalphabet, w is the initial multiset, and R is the set of rules. In the accepting case, T is replaced by Σ, which

[3] It is well-known that symport/antiport systems can be represented as cooperative rewriting on objects of the form (object,region). It is also known that, in case the environment contains an unbounded supply of all objects, a rewriting rule $u \to v$ is equivalent to a symport/antiport rule u/v. Therefore, one-membrane full-environment is a normal form for symport/antiport P systems. Clearly, symport-in rules are not allowed. Moreover, transition into this normal form preserves properties we consider in this paper, so in the following we only consider this case.

is the input subalphabet, the computation starts when an arbitrary multiset over Σ is added to w.

Consider a P system Π with alphabet O. In our setting, a configuration is defined by the multiset of objects inside the membrane, represented by some string $u \in O^*$. The space $\mathcal{C}$ of configurations (i.e., of multisets over O) is essentially $|O|$-dimensional space with non-negative integer coordinates. We use the usual definitions of maximally parallel transition ([12]): no rule is applicable together with a chosen multiset of rules. It induces an infinite graph of $\mathcal{C}$. Notice that the halting configurations (and only them) have out-degree zero.

Throughout this paper by reachable we mean reachable from the initial configuration. We now define two properties; extending the requirement from reachable configurations to all configuration, we obtain their strong variants (in case of accepting systems the initial configurations are obtained by adding to a fixed multiset arbitrary multisets over a fixed subalphabet; the extension is natural).

Definition 1. *We call Π* **strongly reversible** *if every configuration has in-degree at most one. We call Π* **reversible** *if every reachable configuration has in-degree at most one. We call Π* **strongly deterministic** *if every configuration has out-degree at most one. It is common in membrane computing to call Π* **deterministic** *if every reachable configuration has out-degree at most one.*

A property equivalent to reversibility is determinism of a dual P system ([1]). We underline that the not-strong properties refer to the actual computation of the system, where the strong ones do not depend on the initial configuration.

By a computation we mean a sequence of (maximally parallel) transitions, starting in the initial configuration, and ending in some halting configuration if it is finite. The result of a halting computation is the number of terminal objects inside the membrane when the system halts (or the number of input objects when the system starts, in the accepting case). The set $N(\Pi)$ of numbers generated by a P system Π is the set of results of all its computations. The family of number sets generated by reversible P systems with features α is denoted by $NROP_1(\alpha)_T$, where $\alpha \subseteq \{sym_*, anti_*, pro, inh, Pri\}$ and the braces of the set notation are omitted. Subscript T means that only terminal objects contribute to the result of computations; if $T = O$, we omit specifying it in the description and we then also omit the subscript T in the notation. To bound the weight (i.e., maximal number of objects sent in a direction) of symport or antiport rules, the associated $*$ is replaced by the actual number. In the case of accepting systems, we write N_a instead of N, and subscript T has no meaning. For strongly reversible systems, we replace in the notation R by R_s. For deterministic (strongly deterministic) systems, we replace R by D (D_s, respectively).

2.1 Register machines

In this paper we consider register machines with increment, unconditional decrement and test instructions, [7], see also [6].

A register machine is defined by a tuple $M = (n, Q, q_0, q_f, I)$ where

- n is the number of registers;
- I is a set of instructions bijectively labeled by elements of Q;
- $q_0 \in Q$ is the initial label;
- $q_f \in Q$ is the final label.

The allowed instructions are:

- $(q : i?, q', q'')$ - jump to instruction q'' if the contents of register i is zero, otherwise proceed to instruction q';
- $(q : i+, q', q'')$ - add one to the contents of register i and proceed to either instruction q' or q'', non-deterministically;
- $(q : i-, q', q'')$ - subtract one from the contents of register i and proceed to either instruction q' or q'', non-deterministically;
- $(q_f : halt)$ - terminate the computation; it is a unique instruction with label q_f.

As for subtract instructions, the computation is blocked if the contents of the corresponding register is zero. Without restricting generality, we can assume that a test of a register always precedes its subtraction. (A popular model where test and subtraction are combined in a conditional subtraction instruction is not suitable for defining reversibility.) A configuration of a register machine is defined by the current instruction and the contents of all registers, which are non-negative integers.

If $q' = q''$ for every instruction $(q : i+, q', q'')$ and for every instruction $(q : i-, q', q'')$, then the machine is called deterministic. Clearly, this is necessary and sufficient for the global transition (partial) mapping not to be multi-valued.

A register machine is called reversible if there is more than one instruction leading to some instruction q, then exactly two exist, they test the same register, one leads to q if the register is zero and the other one leads to q if the register is positive. It is not difficult to check that this requirement is a necessary and sufficient condition for the global transition mapping to be injective. Let us formally state the reversibility of a register machine: for any two different instructions $(q_1 : i_1\alpha_1, q_1', q_1'')$ and $(q_2 : i_2\alpha_2, q_2', q_2'')$, it holds that $q_1' \neq q_2'$ and $q_1'' \neq q_2''$. Moreover,

$$\text{if } q_1' = q_2'' \text{ or } q_1'' = q_2', \text{ then } \alpha_1 = \alpha_2 = ? \text{ and } i_1 = i_2.$$

It has been shown ([7]) that reversible register machines are universal (a straightforward simulation of, e.g., reversible Turing Machines [2], would not be reversible). It follows that non-deterministic reversible register machines can generate any recursively enumerable set of non-negative integers as a value of the first register by all its possible computations starting from all registers having zero value.

3 Examples and universality

We now present a few examples to illustrate the definitions.

Example 0: Consider a P system $\Pi_0 = (\{a, b\}, a, \{a/ab\})$. It is strongly reversible (for a preimage, remove as many copies of b as there are copies of a, in case it is possible and there is at least one copy of a), but no halting configuration is reachable. Therefore, $\emptyset \in NR_sOP_1(anti_2)$.

Example 1: Consider a P system $\Pi_1 = (\{a, b, c\}, a, \{a/ab,\ a/c\})$. It generates the set of positive integers since the reachable halting configurations are cb^*, and it is reversible (for the preimage, replace c with a or ab with b), but not strongly reversible (e.g., $aa \Rightarrow cc$ and $ac \Rightarrow cc$). Hence, $\mathbb{N}_+ \in NROP(anti_2)$.

Example 2: Consider a P system $\Pi_2 = (\{a, b\}, aa, \{aa/ab,\ ab/bbb\})$. It is reversible ($aa$ has in-degree 0, while ab and bbb have in-degree 1, and no other configuration is reachable), but not strongly reversible (e.g., $aab \Rightarrow abbb$ and $aabb \Rightarrow abbb$).

Example 3: Any P system containing a rule x/λ, $x \in O^+$ is not reversible. Therefore, symport rules cannot be actually used in reversible P systems with one membrane.

Example 4: Any P system containing rules x_1/y, x_2/y that applied at least one of them in some computation is not reversible.

We now show that reversible P systems with either inhibitors or priorities are universal.

Theorem 1. *$NROP_1(anti_2, Pri)_T = NROP_1(anti_2, inh)_T = NRE$.*

Proof. We reduce the theorem statement to the claim that such P systems simulate the work of any reversible register machine $M = (n, Q, q_0, q_f, I)$. Consider a P system

$$\begin{aligned}
\Pi &= (O, \{r_1\}, q_0, R), \text{ where}\\
O &= \{r_i \mid 1 \le i \le n\} \cup Q,\\
R &= \{q/q'r_i, q/q''r_i \mid (q : i{+}, q', q'') \in I\}\\
&\quad \cup \{qr_i/q', qr_i/q'' \mid (q : i{-}, q', q'') \in I\} \cup R_t,\\
R_t &= \{q/q''|_{\neg r_i}, qr_i/q'r_i \mid (q : i?, q', q'') \in I\}.
\end{aligned}$$

Inhibitors can be replaced by priorities by redefining R_t as follows.

$$R_t = \{qr_i/q'r_i > q/q'' \mid (q : i?, q', q'') \in I\}.$$

Since there is a bijection between the configurations of Π containing one symbol from Q and the configurations of M, the reversibility of Π follows from the correctness of the simulation, the reversibility of M and from the fact that the number of symbols from Q is preserved by transitions of Π.

The universality leads to the following undecidability.

Corollary 1. *It is undecidable whether a system from the class of P systems with either inhibitors or priorities is reversible.*

Proof. We recall that the halting problem for register machines is undecidable. Add instructions q_f/F_1, q_f/F_2, F_1/F, F_2/F to the construction presented above, where F_1, F_2, F are new objects; the system is now reversible if and only if some configuration containing F is reachable, i.e., when the underlying register machine does not halt, which is undecidable.

A more restricted property of strong reversibility is much easier to check, since checking that at most one preimage exists for any configuration is no longer related to the reachability. However, the problem of specifying an algorithmic criterion for strong reversibility is currently open.

4 Limitations

The construction in the theorem above uses both cooperation and additional control. It is natural to ask whether both inhibitors and priorities can be avoided. Yet, consider the following situation. Let $(p : i?, s, q''), (q : i?, q', s) \in I$. It is usual for reversible register machines to have this, since the preimage of configuration containing a representation of instruction s depends on register i. Nevertheless, P systems with maximal parallelism without additional control can only implement a zero-test by try-and-wait-then-check strategy. In this case, the object containing the information about the register p finds out the result of checking after a possible action of the object related to the register. Therefore, when the instruction represented in the configuration of the system changes to s, it obtains an erroneous preimage representing instruction q. This leads to the following

Conjecture 1. Reversible P systems without priorities and without inhibitors are not universal.

Now consider strongly reversible P systems. The following theorem establishes a very serious limitation on such systems if no additional control is used.

Theorem 2. *In strongly reversible P systems without inhibitors and without priorities, every configuration is either halting or induces only infinite computation(s).*

Proof. If the right-hand side of every rule contains a left-hand side of some rule, then the claim holds. Otherwise, let x/y be a rule of the system such that y does not contain the left-hand side of any rule. Then $x \Rightarrow y$ and y is a halting configuration. It is not difficult to see that $xy \Rightarrow yy$ (objects y are idle) and $xx \Rightarrow yy$ (the rule can be applied twice). Therefore, such a system is not strongly reversible, which proves the theorem.

Therefore, the strongly reversible systems without additional control can only generate singletons, i.e., $NR_sOP_1(anti_*)_T = \{\{n\} \mid n \in \mathbb{N}\}$, and only in a degenerate way, i.e., without actual computing.

It turns out that the theorem above does not hold if inhibitors are used. Consider a system $\Pi_3 = (\{q, f, a\}, q, \{q/qaa|_{\neg f}\}, \{q/f|_{\neg f}\})$. If at least one object f is present or no objects q are present, such a configuration is a halting one. Otherwise, all objects q are used by the rules of the system. Therefore, the only possible transitions in the space of all configurations are of the form $q^{m+n}a^{p-2m} \Rightarrow q^m f^n a^p$, $m + n > 0$, $p \geq 2m$ and the system is strongly reversible. Notice that $N(\Pi) = \{2k+1 \mid k \geq 0\}$, since starting from q we apply the first rule for $k \geq 0$ steps and eventually the second rule.

5 Strong determinism

The concept of determinism common to membrane computing essentially means that such a system, starting from the fixed configuration, has a unique computation. As it will be obvious later, this property is often not decidable. Of course, this section only deals with accepting systems.

First, we recall from [3] that deterministic symport/antiport P systems with restrictions mentioned in the preliminaries (one membrane, infinite supply of all objects in the environment) are still universal, by simulation of register machines.

In general, if a certain class of non-deterministic P systems is universal even in a deterministic way, then the determinism is undecidable for that class. This applies to our model of one-membrane all-objects-in-environment P systems with symport/antiport, similarly to Corollary 1.

Corollary 2. *It is undecidable whether a given P system with symport/antiport rules is deterministic.*

Proof. Consider an arbitrary register machine M. There is a deterministic P system Π simulating M. Without restricting generality we assume that an object q_f appears in the configuration of Π if and only if it halts. Add instructions q_f/F_1 and q_f/F_2 to the set of rules, where F_1, F_2 are new objects; the system is now deterministic if and only if some configuration with q_f is reachable, i.e., when the underlying register machine does not halt, which is undecidable.

On the contrary, the strong determinism we now consider means that a system has no choice of transitions from any configuration. We now claim that it is a syntactic property. To formulate the claim, we need the following notions. We call the *domain* of a rule x/y, $x/y|_a$ or $x/y|_{\neg a}$ the set of objects in x (the multiplicities of objects in x are not relevant for the results in this paper). We say that two rules are mutually excluded by promoter/inhibitor conditions if the inhibitor of one is either the promoter of the other rule, or is in the domain of the other rule.

Theorem 3. *A P system is strongly deterministic if and only if any two rules with intersecting domains are either mutually excluded by promoter/ inhibitor conditions, or are in a priority relation.*

Proof. Clearly, any P system with only one rule is strongly deterministic, because the degree of parallelism is defined by exhausting the objects from the domain of this rule.

The forward implication of the theorem holds because the rules with non-intersecting domains do not compete for the objects, while mutually excluding promoter/inhibitor conditions eliminate all competing rules except one, and so does the priority relation. In the result, for any configuration the set of objects is partitioned in disjoint domains of applicable rules, and the number of applications of different rules can be computed independently.

We now proceed with the converse implication. Assume that rules p, p' of the system intersect in the domain, are not in a priority relation, and are not mutually excluded by the promoter/inhibitor conditions. Let x, x' be the multisets of objects to be sent out by rules p, p', respectively. Then consider the multiset C, which is the minimal multiset including x, x', and the configuration C', defined as the minimal multiset including C' and promoters of p, p', if any.

Starting from C', there are enough objects for applying either p or p'. Since the rules neither are mutually excluded nor are in a priority relation, both rules are applicable. However, both cannot be applied together because the rules intersect in the domain and thus the multiset C is strictly included in the union of x, x' (and C' is only different from C if either promoter of p, p' does not belong to C). The sufficiency of the condition of this theorem follows from contradicting the strong determinism.

Corollary 3. *A P system without promoters, inhibitors, and without priority is strongly deterministic if and only if the domains of all rules are disjoint.*

We show an interesting property of strongly deterministic P systems without additional control. To define it, we use the following notion for deterministic P systems. Let $C \Rightarrow^{\rho_1} C_1 \Rightarrow^{\rho_2} C_2 \cdots \Rightarrow^{\rho_n} C_n$, where ρ_i are multisets of applied rules, $1 \leq i \leq n$. We define the multiset of rules applied starting from configuration C in n steps as

$$m(C, n) = \bigcup_{i=1}^{n} \rho_i.$$

We write $lhs(x/y) = x$ and $rhs(x/y) = y$, and extend this notation to the multiset of rules by taking the union of the correspoding multisets. For instance, if $C \Rightarrow^{\rho} C_1$, then $C_1 = C \cup rhs(\rho) \setminus lhs(\rho)$.

Lemma 1. *Consider a strongly deterministic P system Π without promoters, inhibitors and without priorities. Consider also two configurations C, C' with $C \subsetneq C'$ and a number n. Then, $m(C, n) \subseteq m(C', n)$.*

Proof. We prove the statement by induction. It holds for $n = 1$ step because strongly deterministic systems are deterministic, and if the statement did not hold, then neither would the determinism.

Assume the statement holds for $n - 1$ steps, and

$$C \Rightarrow^{\rho_1} C_1 \Rightarrow^{\rho_2} C_2 \cdots \Rightarrow^{\rho_n} C_n,$$
$$C' \Rightarrow^{\rho'_1} C'_1 \Rightarrow^{\rho'_2} C'_2 \cdots \Rightarrow^{\rho'_n} C'_n.$$

Then, after $n-1$ steps the difference between the configurations can be described by $C'_{n-1} = C_{n-1} \cup D_1 \cup D_2 \setminus D_3$, where

- $D_1 = C' \setminus C$,
- $D_2 = rhs(m(C', n-1) \setminus m(C, n-1))$,
- $D_3 = lhs(m(C', n-1) \setminus m(C, n-1))$.

Therefore, $C_{n-1} \setminus C'_{n-1} \subsetneq D_3$. Because of the strong determinism property, these objects will either be consumed by some rules from $m(C', n-1) \setminus m(C, n-1)$, or remain idle. Therefore, $m(C_{n-1}, 1) \subseteq m(C'_{n-1}, 1) \cup (m(C', n-1) \setminus m(C, n-1))$. It follows that $m(C, n) \subseteq m(C', n)$, concluding the proof.

Example 5: For a P system $\Pi = (\{a\}, a, \{p : a^3/a\})$,

$$a^{15} \Rightarrow^{p^5} a^5 \Rightarrow^{p} a^4 \Rightarrow^{p} a.$$
$$a^{14} \Rightarrow^{p^4} a^6 \Rightarrow^{p^2} a^2.$$

We now establish an upper bound for the power of strongly deterministic P systems without additional control: any P system without promoters, inhibitors or priorities accepts either the set of all non-negative integers, or a finite set of all numbers bounded by some number.

Theorem 4. $N_aD_sOP_1(sym_*, anti_*) = \{\{k \mid 0 \leq k \leq n\} \mid n \in \mathbb{N}\} \cup \{\emptyset, \mathbb{N}\}$.

Proof. A computation starting from a configuration C is not accepting if it does not halt, i.e., if $\lim_{n\to\infty} m(C, n) = \infty$. Due to Lemma 1, if the computation starting from C is accepting, then any computation starting from a submultiset $C' \subseteq C$ would also be accepting. This also implies that if the computation starting from C is not accepting, then neither is any computation starting from a multiset containing C. Therefore, the set of numbers accepted by a strongly deterministic P system without additional control can be identified by the largest number of input objects leading to acceptance, unless the system accepts all numbers or none.

The converse can be shown by the following P systems.

- System $(\{a\}, \{a\}, a, \{a/a\})$ accepts $\emptyset$ because of the infinite loop in its computation;
- system $(\{a\}, \{a\}, a, \{a/\lambda\})$ accepts $\mathbb{N}$, i.e., anything, because it halts after erasing everything in one step; and
- for any $n \in \mathbb{N}$ there is a system $(\{a\}, \{a\}, \lambda, \{a^{n+1}/a^{n+1}\})$ accepting $\{k \mid 0 \leq k \leq n\}$, because the system starts in a final configuration if and only if the input does not exceed n, and enters an infinite loop otherwise.

Theorem 4 shows that the computational power of strongly deterministic P systems without additional control is, in a certain sense, degenerate (it is subregular). We now show that the use of promoters and inhibitors lead to universality of even the strongly deterministic P systems.

Theorem 5. $N_aD_sOP_1(sym_*, anti_*, pro, inh) = NRE$.

Proof. We reduce the theorem statement to the claim that such P systems simulate the work of any deterministic register machine $M = (n, Q, q_0, q_f, I)$. Without restricting generality, we assume that every subtracting instruction is preceded by the testing instruction. Consider a P system

$$\begin{aligned}
\Pi &= (O, \{r_1\}, q_0, R), \text{ where} \\
O &= \{r_i, d_i \mid 1 \le i \le n\} \cup \{q, q_1 \mid q \in Q\}, \\
R &= \{q/q'r_i \mid (q : i+, q', q') \in I\} \\
&\cup \{q/q_1d_i, q_1/q',\ d_ir_i/\lambda \mid (q : i-, q', q') \in I\} \\
&\cup \{q/q'|_{r_i}, q/q''|_{\neg r_i} \mid (q : i?, q', q'') \in I\}.
\end{aligned}$$

All rules using objects q, q' have disjoint domains, except the ones in the last line, simulating the zero/non-zero test. However, they exclude each other by the same object which serves as promoter and inhibitor. Subtraction of register i is handled by producing object d_i, which will "annihilate" (i.e., be deleted together with) with r_i. Therefore, different instructions subtracting the same r_i are implemented by the same rule d_ir_i/λ, hence all rules using objects d_i, r_i have different domains. It follows from Theorem 3 that the system is strongly deterministic, concluding the proof.

6 Conclusions

We outlined the concepts of reversibility, strong reversibility and strong determinism for P systems, concentrating on the case of symport/antiport rules (possibly with control such as priorities or inhibitors) with one membrane, assuming that the environment contains an unbounded supply of all objects, see Table 1. We added the universality of the usual deterministic systems without control from [3] for comparison.

We showed that reversible P systems with control are universal, and we conjectured that this result does not hold without control. Moreover, the strongly reversible P systems without control do not halt unless the starting configuration is halting, but this is no longer true if inhibitors are used.

We also gave a syntactic characterization for the strong determinism property. Moreover, we showed that a corresponding system without control either accepts all natural numbers, or a finite set of numbers. With the help of promoters and inhibitors the corresponding systems become universal.

Showing related characterizations might be quite interesting. Many other problems are still open, e.g., cells with "C" and "?" in Table 1. Another interesting problem is to formulate reversibility for P systems with active membranes and to characterize their power.

Property	$npro, ninh, nPri$	Pri	inh	pro, inh
D(acc)	U	U	U	U
D_s(acc)	E (Th. 4)	?	?	U (Th. 5)
R(gen)	C (Conj. 1)	U (Th. 1)	U (Th. 1)	U (Th. 1)
R_s(gen)	E (Th. 2)	C	C	C

Table 1. The power of P systems with different properties, depending on the features. U - universal, E - degenerate, ? - open, C - conjectured to be non-universal.

Acknowledgments

Artiom Alhazov gratefully acknowleges the support of the Japan Society for the Promotion of Science and the Grant-in-Aid for Scientific Research, project 20·08364. He also acknowledges the support by the Science and Technology Center in Ukraine, project 4032.

References

1. O. Agrigoroaiei, G. Ciobanu: Dual P Systems, *Membrane Computing - 9th International Workshop*, LNCS **5391**, 95–107, 2009.
2. C.H. Bennett: Logical Reversibility of Computation, *IBM Journal of Research and Development* **17**, 1973, 525-532.
3. C. Calude, Gh. Păun: Bio-steps beyond Turing, *BioSystems* **77**, 2004, 175–194.
4. E. Fredkin, T. Toffoli: Conservative Logic, *Int. J. Theoret. Phys.* **21**, 1982, 219-253.
5. A. Leporati, C. Zandron, G. Mauri: Reversible P Systems to Simulate Fredkin Circuits, *Fundam. Inform.* **74**(4), 2006, 529–548.
6. M.L. Minsky: *Computation: Finite and Infinite Machines*, Prentice-Hall, Englewood Cliffs, NJ, 1967.
7. K. Morita: Universality of a Reversible Two-Counter Machine, *Theoret. Comput. Sci.* **168** (1996) 303-320.
8. K. Morita: A Simple Reversible Logic Element and Cellular Automata for Reversible Computing, Proc. *3rd Int. Conf. on Machines, Computations, and Universality*, LNCS **2055**, Springer-Verlag, 2001, 102-113.
9. K. Morita: Simple Universal One-Dimensional Reversible Cellular Automata, *J. Cellular Automata* **2**, 2007, 159-165.
10. K. Morita, N. Nishihara, Y. Yamamoto, Zh. Zhang: A Hierarchy of Uniquely Parsable Grammar Classes and Deterministic Acceptors, *Acta Inf.* **34**(5), 1997, 389–410.
11. K. Morita, Y. Yamaguchi: A Universal Reversible Turing Machine, Proc. *5th Int. Conf. on Machines, Computations, and Universality*, LNCS **4664**, Springer-Verlag, 2007, 90-98.
12. Gh. Păun: *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.
13. P systems webpage. `http://ppage.psystems.eu/`.