Natural Computing (2023) 22:27–39
https://doi.org/10.1007/s11047-022-09922-1

About reversibility in sP colonies and reaction systems

Ludek Cienciala[1] · Lucie Ciencialová[1] · Erzsébet Csuhaj-Varjú[2]

Accepted: 10 September 2022 / Published online: 18 October 2022

Abstract
In this paper, we study reversibility in sP colonies and in reaction systems. sP colony is a bio-inspired computational model formed from an environment and a finite set of agents. The current state of the environment is represented by a finite set of objects and the current state of the agent is given by a finite multiset of objects. By execution of a program from a set of programs associated with the agent, the agent can change the objects in its own state and possibly in the environment, too. Reaction systems are a bio-inspired computational model where reactants are transformed into products only if some inhibitors are not present. We define sP colonies without input influence and prove that to any reversible sP colony of such type an inverse sP colony can be constructed that performs inverse computation. In the second part of the paper, we show that the concept of a reversible reaction system and the notion of an inverse reaction system can be defined in a similar way, and partially reversible reaction systems can simulate reversible logic gates and reversible Turing machines.

Keywords P colonies · Reaction systems · Reversibility · Inverse computation

1 Introduction

P colonies (introduced in Kelemen et al. (2004)) are particular variants of P systems (or membrane systems), inspired by collections of reactive agents acting in a joint, dynamically changing environment. The notion was also motivated by colonies of formal grammars. The interested reader can be referred to Păun et al. (2010) for detailed information on P systems and to Kelemen and Kelemenová (1992) as well as Csuhaj-Varjú et al. (1994) for more information on colonies in grammar systems theory. Detailed summaries on P colonies can be found in Kelemenová (2010) and Ciencialová et al. (2019).

✉ Erzsébet Csuhaj-Varjú
csuhaj@inf.elte.hu

Ludek Cienciala
ludek.cienciala@fpf.slu.cz

Lucie Ciencialová
lucie.ciencialova@fpf.slu.cz

[1] Institute of Computer Science & Research Institute of the IT4Innovations Centre of Excellence, Faculty of Philosophy and Science, Silesian University, Opava, Czech Republic

[2] Faculty of Informatics, ELTE Eötvös Loránd University, Budapest, Hungary

A P colony consists of a finite number of agents and their shared environment. Both the current state of the environment and the current state of each agent are represented by a finite multiset of objects (over the same alphabet). The agents are equipped with programs consisting of rules. The rules are of two types: the evolution rules which change the objects of the agents, and the communication rules which are used for an interaction between the agent and the environment.

In this paper, we deal with variants of P colonies where the current state of the environment is a set (instead of a multiset). They are called sP colonies and they were introduced in Ciencialová et al. (2020). Notice that while in case of P colonies, quantitative properties of the environment are significant (the number of copies of the same object), in case of sP colonies, only the occurrence of an object is important, which is a qualitative property.

The computation of an sP colony starts with its initial configuration where the environment and each agent is in its own initial state. The computation step means a maximally parallel action of the active agents: the agent is active if it is able to perform at least one of its programs, and the joint action of the agents is maximally parallel when no more active agent can be added to the synchronously acting agents. The computation ends if there is no more applicable program in the system.

Reaction systems (or R systems) were proposed as a computational device with the components that represent basic chemical reactions in Ehrenfeucht and Rozenberg (2005). Informally, a reaction system is composed of a finite set of objects that can be considered as chemicals and a finite set of reactions. Each reaction is a triplet of sets, that is, the reactants, inhibitors, and products. Let T be a set of reactants. A reaction can be applied to T if all reactants are present in T and there are no inhibitors present. Then the reactants are replaced by the products. All enabled reactions are applied in parallel. The obtained set of products is the union of all single sets of products of each reaction which is enabled in T.

It is easy too see that reaction systems are a qualitative model. Reaction systems and P colonies are related; in Ciencialová et al. (2020), it was shown that for any reaction system, a simulating P colony can be constructed.

In this paper, we continue studying sP colonies and reaction systems, with focus on their reversible variants. The notion of reversibility and inverse functions is an essential and well-studied part of mathematics and computer science (see, for example, Morita (2017)). Reversible computing systems are defined as systems for which each of their computational configurations has one predecessor at most. Therefore, they are "backward deterministic" systems.

In the paper, we introduce the sP colonies without an input influence (during computation) and the set condition of existence of an inverse sP colony to an sP colony. In the second part of the paper, we focus on the reaction systems that work in a way similar to sP colonies. As for reaction systems, we show that for every reversible reaction system, there exists an inverse reaction system. In the last part of the paper, reaction systems simulating reversible logic gates and reversible Turing machines are presented. The properties of reversible reaction systems were studied in Bagossy and Vaszil (2020b) and Bagossy and Vaszil (2020).

2 Preliminaries

Throughout the paper, we assume the reader to be familiar with the basics of the formal language theory by Rozenberg and Salomaa (1997). We introduce the notions and notations used in the sequel.

The family of recursively enumerable sets of natural numbers is denoted by $\mathbb{N}\cdot\mathsf{RE}$ and $\mathbb{N}$ denotes the set of natural numbers.

An alphabet is a finite non-empty set. Let Σ^* be the set of all words over alphabet Σ (including the empty word, ε). For the length of the word $w \in \Sigma^*$, we use the notation $|w|$ and the number of occurrences of symbol $a \in \Sigma$ in w is denoted by $|w|_a$.

A multiset of objects M is a pair $M = (V,f)$, where V is an arbitrary (not necessarily finite) set of objects and f is a mapping where $f : V \to \mathbb{N}$. The mapping f assigns to each object in V its multiplicity in M. The set of all multisets over the set of objects V is denoted by V^*. The set V' is called the support of M and is denoted by $supp(M)$ if $f(x) \neq 0$ for all $x \in V'$. The cardinality of M, denoted by $card(M)$, is defined by $card(M) = \sum_{a \in V} f(a)$.

Any multiset of objects $M = (V,f)$ where $V = \{a_1, \ldots, a_n\}$ can be represented as a string w over the alphabet V with $|w|_{a_i} = f(a_i)$, $1 \leq i \leq n$. Obviously, all words obtained from w by permuting the letters also represent M, and ε represents the empty multiset.

Let g be a mapping $g : A \to M$ where A is a finite set, and $M = (A,f)$ is a finite multiset where $f(a) = 1$ for every $a \in A$. Informally, g is a mapping that "converts" a set to a multiset with objects with multiplicity 1. If no confusion arises, a finite set X can also be represented by a word and notations $|X|$, and $|X|_a$ can also be used.

A directed graph is an ordered pair $G = (V, E)$ according to Diestel (2005) where V is a set whose elements are called vertices, nodes, or points. A is a set of ordered pairs of vertices, called arrows, directed edges, directed arcs, or directed lines. With each edge e of G, let there be associated a real number $w(e)$, called its weight. Then G, together with these weights on its edges, is called a weighted graph. If the edge is associated with a unique label, then we speak of a labeled graph. A vertex x_k is connected with vertex x_0 if there exists a sequence of vertices $x_0x_1 \ldots x_k$ such that for two vertices $e_i = (x_ix_{i+1}) \in E$ for each $i \in \{0, 1, \ldots, k-1\}$. The out-degree is the number of the outgoing edges from a vertex, and the in-degree is the number of the incoming edges onto a vertex.

3 sP colonies

The notion of an sP colony was introduced in Ciencialová et al. (2020). In this section, we define a slightly modified version of the original concept. We will also introduce some further notions, such as a deterministic, inverse, and reversible sP colony, together with a transition graph of a reversible sP colony. We show that to any reversible sP colony, we can effectively construct an inverse sP colony performing inverse computation.

First, we give the definition of the sP colony, then we focus on explaining how the sP colony performs the computation.

Definition 1 An sP colony (of degree n and capacity k, where $n, k \geq 1$) is a construct

$\Pi = (V, f, V_{E,0}, B_1, \ldots, B_n)$,

where

- V is an alphabet, its elements are called objects,
- $f \in V$, called the final object of the sP colony,
- $V_{E,0} \subseteq V$, called the initial state (or initial content) of the environment,
- B_i, $1 \leq i \leq n$, is the agent of the sP colony. Each agent $B_i = (w_{i,0}, P_i)$ is defined as follows:
  - $w_{i,0}$ is a multiset over V with k elements, called the initial state (or initial content) of B_i,
  - $P_i = \{p_{i,1}, \ldots, p_{i,m_i}\}$, $m_i \geq 1$, is a finite set of programs, where each program consists of k rules. Each rule is in one of the following forms:

    $a \to b$, $a, b \in V$, called an evolution rule,
    $c \leftrightarrow d$, $c, d \in V$, called a communication rule.

A computation of the sP colony Π is based on changing its configuration. A configuration of the sP colony consists of the current state of the environment and the current states of the agents.

Formally, the configuration of the sP colony Π is given by $(w_1, \ldots, w_n, V_E)$, where w_i is a multiset of objects in V, and V_E is a non-empty subset of V. The multiset w_i represents all the objects present inside the i-th agent, and $V_E \in 2^V$ represents the set of all objects in the environment. Furthermore, $|w_i| = k$, $1 \leq i \leq n$. The set of all possible configurations of Π is denoted by C, and $C = \underbrace{V_k \times \ldots \times V_k}_{n} \times 2^V$, where $V_k = \{v \in V^* \mid |v| = k\}$.

Notice that C can be the set of all possible configurations of more than one sP colony. While standard P colonies may have an infinite number of configurations, the number of all possible configurations of any sP colony is finite.

The functioning of an sP colony starts from its initial configuration (an initial state). The initial configuration of the sP colony is the $(n+1)$-tuple $(w_{1,0}, \ldots, w_{n,0}, V_{E,0})$, where $w_{i,0}$, $1 \leq i \leq n$ is the initial state of the agent B_i, and $V_{E,0}$ is the initial state of the environment.

The change of the configuration is performed by actions of the agents. This means that the agents apply the rules that are contained in the executed program.

The first type of the rules associated to the programs of the agents, the evolution rules, are of the form $a \to b$. This means that an object a inside the agent is rewritten to (evolves to be) an object b.

The second type of rules, the communication rules, are of the form $c \leftrightarrow d$. If a communication rule is performed, then an object c inside the agent and an object d in the environment swap their locations. Thus, after executing the rule, the object d appears inside the agent and the object c appears in the environment.

Notice that since the environment is represented by a set, one or more rules can introduce the same object in the environment. The configuration change does not have to be done by agents only. At each step, some objects may be added to the environment as input. The notion of input of the sP colony will be explained later.

Next, we define the notion of an applicable program. Let us consider that each program in the sP colony Π has its own unique identifier (a unique label) $p_{i,j}$. This notation refers to the j-th program in the set of programs P_i of the ith agent, B_i.

To define a configuration transition, we set the following mappings:

Let $p_j = \langle r_{j1}, \ldots, r_{jk} \rangle$ be a program.

For every rule $r_{jl}, 1 \leq l \leq k$, we construct functions $\psi : X \to Y$, where $Y \subseteq V$, X is the set of all rules over V in p_j, and $\psi \in \{orig, new, in, out\}$ is defined as follows:

For the evolution rule, $r_{jl} = a \to b$

$$orig(r_{j,l}) = \{a\},\ new(r_{j,l}) = \{b\},$$
$$in(r_{jl}) = \emptyset,\ out(r_{j,l}) = \emptyset,$$

and for the communication rule, $r_{j,l} = a \leftrightarrow b$

$$orig(r_{j,l}) = \emptyset,\ new(r_{j,l}) = \emptyset,$$
$$in(r_{j,l}) = \{b\},\ out(r_{j,l}) = \{a\}.$$

For a program $p_{i,j} = \langle r_{i,j,1}, \ldots, r_{i,j,k} \rangle \in P_i$ and a configuration $c = (w_1, \ldots, w_n, w_e)$, we define

$$app_{p_{i,j}}(c) \Leftrightarrow w_i = \bigcup_{l=1}^{k} \{orig(r_{i,j,l}) \cup out(r_{i,j,l})\} \wedge \bigcup_{l=1}^{k} in(r_{i,j,l}) \subseteq V_e$$

$app_{p_{i,j}}(c)$ is true if and only if the program $p_{i,j}$ is applicable in configuration c.

A set of applicable programs associated with the agent $B_i = (w_i, P_i)$ in a configuration c, denoted by $App_i(c)$, is defined as

$$App_i(c) = \{p_{i,j} \in P_i \mid app_{p_{i,j}}(c)\}.$$

If $App_i(c)$ contains more than one program, then a non-deterministically chosen one is applied.

For the configuration c, let P_c be the set of programs applicable in the configuration c satisfying the following conditions: there is at most one program from $App_i(c)$ for every i, $1 \leq i \leq n$, and $P_c \cup App_i(c) = \emptyset$ if and only if $App_i(c)$ is equal to the empty set.

The sP colony Π is called deterministic if and only if $|App_i(c)| \leq 1$ for all $c \in C$ and for all agents B_i, $1 \leq i \leq n$, in Π. As a result, there exists only one possible P_c for every configuration c.

Next, we define the notion of computation of (length m) in the sP colony working with an input sequence (with an input, for short).

A computation of a length m, $m \geq 1$, in the sP colony Π working with input sequence

$$\gamma = in_0,\ in_1,\ in_2, \ldots, in_m$$

is a sequence of $(n+1)$-tuples $(w_{1,j}, \ldots, w_{n,j}, (V_{E,j} \cup in_j))$, $0 \leq j \leq m$, where $w_{1,j}, \ldots w_{n,j}, V_{E,j}$ is a configuration of Π, $in_j \subseteq V, 1 \leq j \leq m$. For $j = 0$ and $in_0 = \emptyset$, the configuration is the initial configuration. Therefore, the initial configuration of Π is available at the beginning of the computation and then in each of the first m steps, an input set of elements of V is inserted into the environment. Notice that in_j can be empty for some j, $1 \leq j \leq n$.

Also notice that in case of standard P colonies, no input is inserted in the environment, as it is changed only by the actions of the agents.

As we mentioned above, the computation of the sP colony passes from one configuration to another one. This change is done by an execution of at most one applicable program per agent and by inserting a set of input objects into the environment from an input sequence. The computation is defined to be collision-free. All agents having an applicable program should perform exactly one of its applicable programs.

We distinguish between two computational modes: a forgetting mode and a non-forgetting mode. If the new content (state) of the environment obtained by a transition is the union of the set of the objects put into the environment by agents, and the input in the current step, then we speak of the computation in the forgetting mode (or the forgetting computation, f mode). The computation is non-forgetting (in the non-forgetting mode, nf mode) if those elements of the environment that did not take part the in action of any agent and were not the elements of the input, remain the elements of the environment.

Next, we provide a formal definition of a transition (a computation step).

We say that in a t-th step of computation, where $t \geq 1$, the current configuration c directly changes for configuration c' (or there is a transition between these two configurations), written as

$$c = (w_1, \ldots, w_n, V_e) \vdash (w'_1, \ldots, w'_n, V'_e) = c',$$

if $w'_i = \bigcup_{l=1}^{k} (new(r_{jl}) \cup in(r_{jl}))$ for $p_{i,j} = \langle r_{i,j,1}, \ldots, r_{i,j,k} \rangle \in P_c$ being the chosen program applied by the i-th agent or $w'_i = w_i$ if the i-th agent has no applicable program in the configuration c, i.e., $App_i(c) = \emptyset$.

Furthermore, in case of the forgetting computation,

$$V'_e = \bigcup_{p_{i,j} \in P_c} \left(\bigcup_{l=1}^{k} out(r_{i,j,l}) \right) \cup in_t$$

and, in case of the non-forgetting computation,

$$V'_e = \left[(V_e - \bigcup_{p_{i,j} \in P_c} (\bigcup_{l=1}^{k} in(r_{i,j,l}))) \cup \cup \bigcup_{p_{j} \in P_c} (\bigcup_{l=1}^{k} out(r_{jl})) \right] \cup in_t.$$

The computation ends by halting after performing more than m steps, where $m+1$ is the length of the input set sequence.

The result of a computation with input sequence γ can be defined in several ways.

The first way is based on the fact that the sP colony can be seen as an accepting device, and so the input sequence can be accepted if the computation halts after processing the whole input sequence, i.e., no more transition can be performed.

The second way is closer to the definition of the original model of the P colonies. We say that the number d is computed by the sP colony working with the input sequence γ if the computation with γ halts and $d = \bigcup_{s=1}^{t} |V_{E_s}|_f$, where $t \geq m$ is a number of steps of the computation and V_{E_s} is the content of the environment after s steps of the computation. Since the sP colony is a non-deterministic computing device, with one input sequence, there can exist halting and also non-halting computations.

The third way of defining the result of the computation is to assign a set to the computation and the input sequence; the result set is the content of the environment in a halting configuration.

The family of all sets of numbers $N(\Pi)$ computed as above by the sP colonies of a capacity at most $k \geq 0$, degree at most $n \geq 0$ and height at most $h \geq 0$ is denoted by $NsPCOL(k, n, h)$. The maximal number of programs of an agent of Π is called the height of Π. If one of the parameters n or h is not bounded, then we replace it with $*$.

Example 1 Our example is based on some ideas presented as an example in Ciencialová et al. (2020). Let m and p be natural numbers such that $1 < p \leq m/2$. We can construct an sP colony with one agent that decides whether m is divisible by p for an input sequence of the length m . The input sequence (except of the in_0) is formed of sets of objects, each set has only one object as element. It means that the sP colony will accept one object per step of the computation. The natural number m is represented as m sets $\{\circ\}$ followed by a special set $\{\#\}$. The last object that enters the system is the indicator of the end of the input. The agent in the sP colony corresponds to the divisor by the number p. The state of the agent p corresponds to the result of an operation "h mod p", where h is the number of consumed objects $\circ$ and p is the prime number. At each

step of the forgetting computation, the agent consumes the object from the input set and changes its state in the same way as a finite state machine verifies that the number of symbols in a word is divisible by a given number. After consuming the object #, the agent generates the final object (if it is in the state corresponding to 0), or it enters an infinite loop.

Let $\Pi_1 = (V, f, v_E, B_1)$ be an sP colony with the capacity 2 and with 1 agent such that

- $V = \{\circ, \#, e, f, 0'\} \cup \{0, \ldots, p-1\}$
- $v_E = \{e\}$
- $B_1 = (0'e, P_1)$

The set of programs P_1 consists of the following programs:

0. $\langle 0' \to 0; e \to e\rangle$;
1. $\langle 0 \to 1; e \leftrightarrow \circ\rangle$;
2. $\langle \circ \to (i+1); i \leftrightarrow \circ\rangle$; $0 \le i \le p-2$
3. $\langle \circ \to 0; (p-1) \leftrightarrow \circ\rangle$;
4. $\langle \circ \to f; 0 \leftrightarrow \#\rangle$;
5. $\langle \circ \to e; i \leftrightarrow \#\rangle$; $1 \le i \le p-1$
6. $\langle \# \to f; f \leftrightarrow 0\rangle$;
7. $\langle e \to e; \# \to \#\rangle$

The sP colony works as accepting device. The input sequence corresponding to a number m is accepted only if the computation halts after all input sets enter the system, which implies that m is divisible by p.

3.1 Reversible sP colonies

In this section, we will focus on the process of computation, namely, we aim to deal with the determinism and reversibility of the computation in sP colonies and define the concept of an inverse sP colony.

First, we define the notion of a transition graph for the sP colony working without an input sequence. Notice that in this case, no external additional objects are added to the set of environmental objects during the computational steps.

Definition 2 Let Π be the sP colony working without the input sequence in an x-mode, where the $x \in \{nf, f\}$, and let C be the set of of all configurations of Π. A directed graph $G_x(\Pi) = (V_G, E)$ is called the transition graph of Π working in mode x, if $V_G = C$, and $e = (c_1, c_2) \in E$ if and only if there is the transition $c_1 \vdash c_2$ in some step of the x-mode computation of Π.

If the out-degree of each edge of the transition graph of the sP colony is at most one, then it is called deterministic. Formally,

Definition 3 Let Π be the sP colony working without the input sequence in an x-mode, where the $x \in \{nf, f\}$, let C be the set of all possible configurations of Π and let $G_x(\Pi) = (V_G, E)$ be the transition graph of Π. Then Π is called deterministic (in the x-mode) if and only if

$$(c, c') \in E \wedge (c, c'') \in E \Rightarrow c' = c''.$$

Next, we provide the conditions for reversibility in the sP colonies working without the input sequence.

Definition 4 Let $\Pi = (V, f, V_E, B_1, \ldots, B_n)$ be the sP colony with a capacity k, working without the input sequence in an x mode, where the $x \in \{f, nf\}$. Let C be the set of all of its possible configurations and let $G_x(\Pi) = (V_G, E)$ be the transition graph of Π. Then Π is called reversible (in the x-mode) if and only if the following conditions are satisfied:

1. $|App_i(c)| \le 1$ for $1 \le i \le n$ and for all $c \in C$, i.e., Π is deterministic;
2. for every $c \in C$ there is at most one $(c', c) \in E$, i.e., the in-degree of each node is at most one.

The conditions in Definition 4 ensure that the transition is a bijective relation in reversible sP colonies.

Definition 5 Let Π be the sP colony working without the input sequence and in an x mode, where the $x \in \{f, nf\}$. If there exists the sP colony $\hat{\Pi}$ working without the input sequence and in the x mode such that for any computation $c_1 \vdash c_2 \cdots \vdash c_m$ in Π, where $m \ge 1$, there exists a computation $c_m \vdash c_2 \cdots \vdash c_1$ in $\hat{\Pi}$; and, reversely, then $\hat{\Pi}$ is called an inverse (sP colony) of Π.

If an sP colony Π working without input is reversible, then we can effectively construct an inverse sP colony $\hat{\Pi}$ to Π.

Theorem 1 *Let $\Pi = (V, f, V_E, B_1, \ldots, B_n)$ be a reversible sP colony with a capacity k and working without the input and in the x mode, where the $x \in \{f, nf\}$. Then there exists an inverse sP colony $\hat{\Pi} = (V, f, V_E, \hat{B}_1, \ldots, \hat{B}_n)$ such that the set of program $\hat{P}_i$ of agent $\hat{B}_i$, $1 \le i \le n$, is given as follows: if $r_{i,j,l} = a \to b$ is a rule in the program $p_{i,j} \in P_i$, $1 \le j \le |P_i|$, then $\hat{r}_{i,j,l} = b \to a$ is a rule in the program $\hat{p}_{i,j} \in \hat{P}_i$; if $r_{i,j,l} = c \leftrightarrow d$ is a rule in the program $p_{i,j} \in P_i$, then $\hat{r}_{i,j,l} = d \leftrightarrow c$ is a rule in the program $\hat{p}_{i,j} \in \hat{P}_i$.*

Proof From the construction of rules of $\hat{\Pi}$, it is easy to see that:

If $r_{i,j,l} = a \to b \Rightarrow \hat{r}_{i,j,l} = b \to a \Rightarrow \quad orig(r_{i,j,l}) = new(\hat{r}_{i,j,l}); new(r_{i,j,l}) = orig(\hat{r}_{i,j,l}); out(r_{i,j,l}) = in(\hat{r}_{i,j,l}) = \emptyset; in(r_{i,j,l}) = out(\hat{r}_{i,j,l}) = \emptyset$

$r_{i,j,l} = c \leftrightarrow d \Rightarrow \hat{r}_{i,j,l} = d \leftrightarrow c \Rightarrow \quad out(r_{i,j,l}) = in(\hat{r}_{i,j,l}); in(r_{i,j,l}) = out(\hat{r}_{i,j,l}); \quad orig(r_{i,j,l}) = new(\hat{r}_{i,j,l}) = \emptyset; new(r_{i,j,l}) = orig(\hat{r}_{i,j,l}) = \emptyset$

We have to prove that $c_1 \vdash c_2 \vdash \ldots \vdash c_m$ is a computation in the sP colony Π if and only if there exists the computation $\hat{c}_1 \vdash \hat{c}_2 \vdash \ldots \vdash \hat{c}_m$ such that $\hat{c}_s = c_{m-s+1}; \vdash \ldots \vdash c_2 \vdash c_1$ in the inverse sP colony $\hat{\Pi}$.

The initial configurations of the computations in the sP colonies are c_1 and $c_m = \hat{c_1}$, respectively. Let us focus on one step of the computation in both systems. Let us consider $c_s \vdash c_{s+1}, 1 \leq s \leq m$, i.e. the s-th step of the computation in the sP colony Π. Since Π is reversible, to perform the transition, there is only one set of programs P_{c_s} where there exists at most one program for each agent in Π. Let $c_s = (w_1, \ldots w_n, V_E)$ and $c_s = (w'_1, \ldots w'_n, V'_E)$. Then

$$\bigcup_{l=1}^{k} \{orig(r_{i,j,l}) \cup out(r_{i,j,l})\} =$$
$$= w_i \wedge \bigcup_{l=1}^{k} \{new(r_{i,j,l}) \cup in(r_{i,j,l})\} = w'_i$$

for $p_{i,j} \in P_{c_s}$ and $1 \leq i \leq n$ (if for given i $p_{i,j} \in P_{c_s}$, otherwise $App_i(c_s) = \emptyset$ for agent B_i). By substituting the functions, we obtain

$$\bigcup_{l=1}^{k} \{new(\hat{r}_{i,j,l}) \cup in(\hat{r}_{i,j,l})\} =$$
$$= w_i \wedge \bigcup_{l=1}^{k} \{orig(\hat{r}_{i,j,l}) \cup out(\hat{r}_{i,j,l})\} = w'_i$$

Regarding the change of the environment, we remember that P_i is without the input, consequently

$$\bigcup_{p_{i,j} \in P_{c_s}} \left(\bigcup_{l=1}^{k} in(r_{i,j,l}) \right) \subseteq V_E.$$

Then, in the case of the forgetting computation, we obtain

$$V'_E = \bigcup_{p_{i,j} \in P_{c_s}} \left(\bigcup_{l=1}^{k} out(r_{i,j,l}) \right) =$$
$$= \bigcup_{\hat{p}_{i,j} \in P_{c_{s+1}}} \left(\bigcup_{l=1}^{k} in(\hat{r}_{i,j,l}) \right)$$

and, in case of the non-forgetting computation, we get

$$\begin{aligned} V'_E &= \left[\left(V_e - \bigcup_{p_{i,j} \in P_{c_s}} \left(\bigcup_{l=1}^{k} in(r_{i,j,l}) \right) \right) \cup \right. \\ &\quad \left. \cup \bigcup_{p_j \in P_{c_s}} \left(\bigcup_{l=1}^{k} out(r_{jl}) \right) \right] = \\ &= \left[\left(V_E - \bigcup_{\hat{p}_{i,j} \in \hat{P}_{c_{s+1}}} \left(\bigcup_{l=1}^{k} in(\hat{r}_{i,j,l}) \right) \right) \cup \right. \\ &\quad \left. \cup \bigcup_{\hat{p}_j \in \hat{P}_{c_{s+1}}} \left(\bigcup_{l=1}^{k} out(\hat{r}_{jl}) \right) \right] \end{aligned}$$

The previous relations are true only if sets P_{c_s} and $\hat{P}_{c_{s+1}}$ are formed from the corresponding programs. If the program $p_{i,j}$ is in P_{c_s}, then the program $\hat{p}_{i,j}$ is applicable in the configuration c_{s+1} and, as a result, it is in $\hat{P}_{c_{s+1}}$. If the program $\hat{p}_x \in P_i$ is applicable in the configuration c_{s+1} and the corresponding program p_x is not applicable in c_s then - because Π is deterministic - there exists a configuration c_x for which $c_x \vdash c_{s+1}$ holds. But this is not possible because Π is reversible, and therefore Condition 2 of Definition 4 should hold. □

4 Reaction systems

In this section we recall the basic notions concerning reaction systems, introduced in Ehrenfeucht and Rozenberg (2005).

Let S be a finite non-empty set (its elements are defined as objects or molecules). A triplet $a = (R, I, P)$ where R, I, P are non-empty subsets of S and $R \cap I = \emptyset$ is called a reaction in S. S is called the background set, R is called the set of reactants (or the reactant set), I is the set of inhibitors (or the inhibitor set), and P is called the set of products (or the product set) of reaction a. The set of all reactions in S is denoted by $rac(S)$. The sets R, I, P are also denoted by R_a, I_a, P_a, respectively.

A reaction system is an ordered pair $\mathcal{A} = (S, A)$, where S is the background set and A is the non-empty set of reactions in S.

Reaction systems work with performing their reactions on a non-empty set of objects from the background set that is called the current state of the reaction system.

Let S be a background set, let $X \subseteq S$, and let a be a reaction in S. Then, a is enabled by X, denoted by $en_a(X)$, if $R_a \subseteq X$ and $I_a \cap X = \emptyset$ holds. The result of a on X, denoted by $res_a(X)$, is defined by $res_a(X) = P_a$ if $en_a(X)$, and $res_a(X) = \emptyset$, otherwise.

The effect of a set of reactions on a state is cumulative.

Let S be a background set, let $X \subseteq S$, and let A be a non-empty set of reactions in S. A is called enabled by X, denoted by $en(A, X)$, and it is defined by $en(A, X) = \{a \in A \mid en_a(X)\}$. The result of A on X, denoted by $res(A, X)$, is defined by $res(A, X) = \{res_a(X) \mid a \in A\}$. The set of those reactions in A that generate a product set included in X is defined as $prod(A, X) = \{a \in A \mid P_a \subseteq X)\}$.

The behavior of the reaction system is described by an interactive process defined as follows.

Let $\mathcal{A} = (S, A)$ be a reaction system. An interactive process in $\mathcal{A}$ is a pair $\pi = (\gamma, \varphi)$ of finite sequences such that $\gamma = C_0, C_1, \ldots, C_{n-1}$, $\varphi = D_1, \ldots, D_n$ with $n \geq 1$, where $C_0, \ldots, C_{n-1}, D_1, \ldots, D_n \subseteq S$, $D_1 = res(A, C_0)$, and $D_i = res(A, D_{i-1} \cup C_{i-1})$ for each $2 \leq i \leq n$.

The sequences $C_0, \ldots, C_{n-1}$, and $D_1, \ldots, D_n$ are the context and result sequences of π, respectively.The context C_0 represents the initial state of π (the state in which the interactive process starts), and the contexts $C_1, \ldots, C_{n-1}$ represent the influence of the environment on the computation.

If $C_0 \neq \emptyset$ and $C_i = \emptyset$ for all $i \geq 1$, then the reaction system is said to be working without the influence of the environment (or without the environment influence).

The sequence $sts(\pi) = W_0, \ldots, W_n$ denotes the state sequence of π, where $W_0 = C_0$ (the initial state), and $W_i = D_i \cup C_i$ for all $1 \leq i \leq n$. The sequence $act(\pi) = E_0, \ldots, E_{n-1}$ of subsets of A such that $E_i = en(A, W_i)$ for all $0 \leq i \leq n-1$ represents the activity sequence of π.

Consequently, the evolution of the state sequence of $\mathcal{A}$ starting from W_0 is

$$W_0 \xrightarrow{E_0} W_1 \xrightarrow{E_1} \ldots \xrightarrow{E_{n-1}} W_n.$$

If $E_n = en(A, W_n) = \emptyset$, then the interactive process terminates.

To describe the evolution, we can define a mapping $\delta : 2^S \times 2^S \to 2^S$ such that the following conditions are satisfied: $\delta(D_i, C_i) = D_{i+1}$ if and only if there exists (just one) set $E_i \subseteq A$ such that $E_i = en(A, D_i \cup C_i)$ and $D_{i+1} = res(E_i, D_i \cup C_i)$, and, furthermore, $\delta(\emptyset, C_0) = D_1$ holds.

If a reaction system works without the environment influence, then for every $i \geq 1$ $\delta(D_i, \emptyset) = D_{i+1}$.

In this case, we can simplify the definition of the transition mapping as follows: $\delta' : 2^S \to 2^S$ where $\delta'(W) = W'$. If we have to indicate that the transition mapping is associated with a certain reaction system, then we write $\delta'_{\mathcal{A}}(W') = \delta'_{\mathcal{A}}(W'')$.

For every reaction system working without the environment influence, we can construct a labeled directed graph $G_{\mathcal{A}} = (V, E, lab)$ called the transition graph with a set of nodes $V \subseteq 2^S - \{\emptyset\}$ and a set of directed edges E such that there is a directed edge $e = (X, Y)$ if and only if $en(A, X) \neq \emptyset$, $res(A, X) = Y$ and $lab(e) = \bigcup_{a \in en(A,X)} I_a$. Because of the determinism of the evolution, every node of the transition graph has out-degree at most one.

4.1 Reversible reaction systems

In this section, we deal with the reversibility of reaction systems working without the environment influence.

First, we present some notions we will use in the rest of the paper.

Let S be a background set and let $a = (R_a, I_a, P_a)$ be a reaction in S. If $\hat{\mathcal{A}} = (P_a, I_a, R_a)$ is a reaction in S, then $\hat{\mathcal{A}}$ is called an inverse reaction of a. For a reaction $a = (R_a, I_a, P_a)$, we also may refer to R_a and P_a by $lhs(a)$ and $rhs(a)$, respectively.

For the reaction system $\mathcal{A} = (A, S)$ and for the set of reactants $X \subset S$, let us define $reac(A, X) = \{a \in A \mid R_a \subseteq X\}$.

Now we define the notion of a reversible reaction system.

Definition 6 A reaction system $\mathcal{A} = (A, S)$ working without the environment influence is reversible if the following conditions are satisfied:

1. $P_a \cap I_a = \emptyset$ for every reaction $a \in A$,
2. for every state X of the reaction system $\mathcal{A}$ if $prod(A, X) \neq \emptyset$, then
$$\bigcup_{a \in prod(A,X)} I_a \cap \bigcup_{a \in prod(A,X)} R_a = \emptyset,$$
3. for every state X of the reaction system $\mathcal{A}$ if $reac(A, X) \neq \emptyset$, then
$$\bigcup_{a \in reac(A,X)} P_a \cap \bigcup_{a \in reac(A,X)} I_a = \emptyset,$$
4. for every node Y of the transition graph $G_{\mathcal{A}} = (V, E, w)$ of the reaction system $\mathcal{A}$, there is at most one edge $e = (X, Y)$; i.e., each node of $G_{\mathcal{A}}$ has at most one in-degree.

The first condition ensures that the inverse reaction of the reaction a will is in the required form. The second condition is met if the reactions that take place in generating of some (non-empty) subset of S are not in conflict, i.e., there is no reactant of such a reaction that is an inhibitor of some other reaction. The third condition means that if the set of reactions is enabled on some non-empty subset of S, then there is no product of any of the inverse reactions which is an inhibitor of any inverse reaction. The fourth condition assures that the transition mapping δ is injective.

Each component of the transition graph of the reversible reaction system is either a cycle or it forms a line.

Example 2 Let $\mathcal{A} = (S, A)$ be a reversible reaction system with $S = \{a, b, c, e\}$ and $A = \{a_1, a_2, a_3, a_4\}$ such that

$$a_1 = (\{a, b\}, \{e\}, \{b, c\}); a_2 = (\{b, c\}, \{e\}, \{a, c\});$$
$$a_3 = (\{a, c\}, \{e\}, \{a, b\}).$$

The transition graph of the reaction system looks like the one in Fig. 1.

Theorem 2 *Let $\mathcal{A} = (A, S)$ be a reversible reaction system without the environment influence. Then there exists a reaction system $\hat{\mathcal{A}} = (\hat{A}, S)$ such that the following conditions hold*:

1. for every reaction $a = (R_a, I_a, P_a) \in A$, there exists a reaction $\hat{a} = (P_a, I_a, R_a) \in \hat{A}$ which is an inverse reaction of a,
2. to every interactive process $\pi = (C_0, C_1, \ldots, C_{n-1};\ D_1, D_2, \ldots, D_{n-1},\ D_n)$ in $\mathcal{A}$, there

exists an interactive process $\hat{\pi} = (D_n, C_{n-1}, \ldots, C_1;\ D_{n-1}, \ldots, D_1,\ C_0)$ in $\hat{\mathcal{A}}$.

Proof We should prove that if ω_π is the interactive process in the reaction system $\mathcal{A}$, then there exists an interactive process $\omega_{\hat{\pi}}$ in the reversed reaction system meeting the following condition: if $\omega_\pi = W_0 \xrightarrow{E_0} W_1 \xrightarrow{E_1} \ldots \xrightarrow{E_{n-1}} W_n$ is an evolution in $\mathcal{A}$ (corresponds to π), then $\omega_{\hat{\pi}} = W_n \xrightarrow{\hat{E}_{n-1}} W_{n-1} \xrightarrow{\hat{E}_{n-2}} \ldots \xrightarrow{\hat{E}_0} W_0$ is an evolution in $\hat{\mathcal{A}}$ (corresponds to $\hat{\pi}$), with $\hat{E}_i = \{\hat{a} \mid a \in E_i\}, 0 \leq i \leq n-1$.

By definition, the reversible reaction system $\mathcal{A}$ is without the environment influence, and $C_0 \neq \emptyset \wedge C_i = \emptyset, i \geq 1$ for a sequence $C_0, C_1, \ldots, C_n$ of any interactive process in $\mathcal{A}$.

Because every node (state) has an in-degree at most one (the fourth condition of Definition 6), the (simplified) transition mapping $\delta'_{\mathcal{A}}$ is injective.

For an i-th step of the evolution ω_π, $i \geq 1$, there exists a transition $\delta'_{\mathcal{A}}(W_{i-1}) = W_i$ such that $en(A, W_{i-1}) = E_{i-1}$ and $res(E_{i-1}, W_{i-1}) = W_i$.

Let us assume that $\delta'_{\hat{\mathcal{A}}}(W_i) = W$. We construct a set $\hat{E}_{i-1}$ of inverse reactions from E_{i-1}. Because $rhs(E_{i-1}) = lhs(\hat{E}_{i-1})$ and the third condition of Definition 6 holds, such state exists and

$$\hat{E}_{i-1} \subseteq en(\hat{A}, W_i). \tag{1}$$

We first consider the case $W \neq W_i$. Since inclusion (1) holds, $W_i \subseteq W$ follows. Let $\hat{b}$ be a reaction from a set of reactions $\hat{A}$ such that $\hat{b} \in en(\hat{A}, W_i)$ and $\hat{b} \notin \hat{E}_{i-1}$. Since $b \in prod(A, W_i)$ and the second condition of Definition 6 holds, there is no inhibitor of reaction $b \in A$ ($b \notin E_{i-1}$) in any set of reactants of the reactions from E_{i-1}. The only reason why reaction b is not in E_{i-1} can be if $R_b - W_{i-1} \neq \emptyset$ holds. Therefore, $R_b \cup W_{i-1} = Y$ and $\delta'_{\mathcal{A}}(Y) = W_i$.

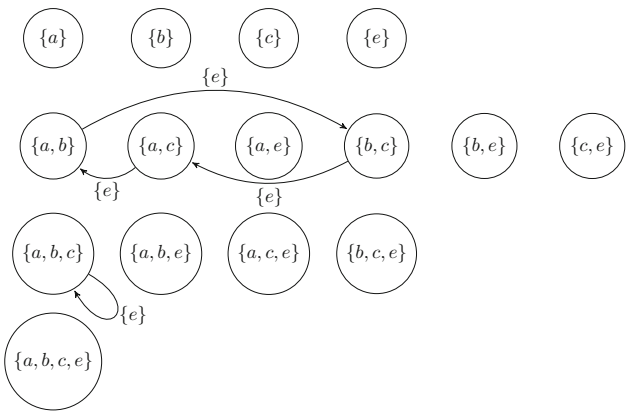

Fig. 1 The transition graph of $\mathcal{A}$

Because of the fourth condition of Definition 6, there is at most one transition leading to every state, then $Y = W_i$.

It remains to prove that $en(\hat{A}, W_i) = \hat{E}_{i-1}$. Because of the second condition of Definition 6, inverse reactions to all reactions from E_{i-1} are enabled. Suppose that there exists a reaction $\hat{b} \in \hat{A}$ such that $\hat{b} \notin \hat{E}_{i-1}$ and $\hat{b} \in en(\hat{A}, W_i)$. This means that the reaction $b \in A$ is not enabled by W_{i-1}, because the state contains some object which is an inhibitor of a reaction. But, we have already proven that this cannot happen because of the missing reactant in W_{i-1} ($\delta'_{\hat{\mathcal{A}}}(W_i) = W_{i-1}$).

$W_{i-1} \cap I_b \neq \emptyset \wedge b \in prod((A, W_i)) \wedge \bigcup_{a \in prod(A, W_i)} I_a \cap \bigcup_{a \in prod(A, W_i)} R_a = \emptyset$ (second condition of Definition 6). Subsequently, any inhibitor s in I_b is not in a set of reactants of any reaction $en(A, W_{i-1})$, it is not processed by any reaction from $en(A, W_{i-1})$, and it disappears from the state after one step. Then, there exists a state $X = W_{i-1} - \{s\}$ such that $en(A, X) = en(A, W_{i-1})$ and $\delta'_{\mathcal{A}}(X) = \delta'_{\mathcal{A}}(W_{i-1}) = W_i$. But the simplified transition mapping is injective and, as a result, $W_{i-1} \cap I_b = \emptyset$. □

Theorem 3 *Let $\mathcal{A} = (A, S)$ be a reversible reaction system and let $\hat{\mathcal{A}} = (\hat{A}, S)$ be a reaction system inverse to $\mathcal{A}$. Then $\hat{\mathcal{A}}$ is reversible, too.*

Proof Because $\hat{\mathcal{A}}$ is the inverse reaction system to $\mathcal{A}$, for the reaction $\hat{a}$, the following equalities hold:

(a) $P_{\hat{a}} = R_a$
(b) $R_{\hat{a}} = P_a$
(c) $I_{\hat{a}} = I_a$

Let us focus to the four conditions of Definition 6 that must hold for the reversible reaction systems.

1. By definition, the set of products and the set of inhibitors of every reaction must be disjoint. This condition holds: see equalities a) and c) above.
2. By the second condition of Definition 6, the following statement should be true. For every state $X \in 2^S$ such that $prod(A, X) \neq \emptyset$

$$\bigcup_{a \in prod(A,X)} I_a \cap \bigcup_{a \in prod(A,X)} R_a = \emptyset$$

holds. If $prod(A, X) = \{a \in A \mid P_a \subset X\}$ and b) holds, then $\{\hat{a} \in \hat{A} \mid R_{\hat{a}} \subset X\} = reac(\hat{A}, X)$. Therefore,

$$\bigcup_{\hat{a} \in reac(\hat{A},X)} I_{\hat{a}} \cap \bigcup_{\hat{a} \in reac(\hat{A},X)} P_{\hat{a}} = \emptyset$$

holds too.

3. For the third conditions of Definition 6 the situation is similar to the previous one. For every state $X \in 2^S$ such that $reac(A,X) \neq \emptyset$

$$\bigcup_{a \in reac(A,X)} I_a \cap \bigcup_{a \in reac(A,X)} P_a = \emptyset$$

should hold. If $reac(A,X) = \{a \in A \mid R_a \subset X\}$ and a), then $\{\hat{a} \in \hat{A} \mid P_{\hat{a}} \subset X\} = prod(\hat{A},X)$. Therefore,

$$\bigcup_{\hat{a} \in prod(\hat{A},X)} I_{\hat{a}} \cap \bigcup_{\hat{a} \in prod(\hat{A},X)} R_{\hat{a}} = \emptyset$$

holds, too.

4. The fourth condition of Definition 6 is met as well. Because of the determinism of every interactive process of $\mathcal{A}$, every state of transition graph has an in-degree at most one. For every edge $e = (X,Y)$ of the transition graph $G_{\mathcal{A}}$, there is one edge $\hat{e} = (Y,X)$ of the transition graph $G_{\hat{\mathcal{A}}}$ of the inverse reaction system and vice versa. Subsequently, every node of the transition graph $G_{\hat{\mathcal{A}}}$ has an out-degree at most one.

□

Next, we introduce the notion of a reaction system with a restricted input and the notion of a partially reversible reaction system.

A reaction system with a restricted input is a triplet $\mathcal{A}_{In} = (A,S,In)$ such as $\mathcal{A} = (A,S)$ is a reaction system, and $In \subseteq 2^S$ is a finite non-empty set of input sets.

In an interactive process, every $C_i, i \geq 0$ is an element of the set In. If a reaction system works without the environment influence, then the In contains all possible inputs C_0 for an initialization of an interactive process.

If the $\mathcal{A}_{In}$ satisfies conditions 1.–4. of Definition 6, then the reaction system is called partially reversible. Let the $\mathcal{A}_{In}$ be a partially reversible reaction system and $G_{\mathcal{A}_{In}}$ be its transition graph. A set of the vertices of $G_{\mathcal{A}_{In}}$ contains such states that are connected with the states from the In, and the following holds: If there is a state $X \in V$ with an in-degree equal to zero, then $X \in In$. Every node of the $G_{\mathcal{A}_{In}}$ with an in-degree at least one can be in the input set of the inverse reaction system $\hat{\mathcal{A}}$.

Remark 1 As we mentioned in the Introduction, in Ciencialová et al. (2020), we proved that for any reaction system, the simulating P colony can be constructed. Based on the previous considerations, it can be shown that for any reversible sP colony without the input influence and working in the forgetting mode, the simulating reaction system can be constructed. Notice that the reversible sP colony is deterministic and the in-degree of its transition graph is 1. We sketch the idea of the construction. Suppose that Π is a reversible sP colony without the input influence that works in the forgetting mode. Let V be a set of objects of Π and let it have n agents of capacity k. Then any configuration $c = (w_1, \ldots, w_n, V_E)$ of Π can be represented by the $(n+1)$-tuple $(a_{w_1,1}, \ldots, a_{w_n,n}, a_{V_E})$, where the $a_{w_j,j}$ is an element of the background set S of $\mathcal{A}$ representing that the state of the jth agent is w_j, and a_{V_E} denotes that the state of the environment is V_E. Since the number of all possible configurations is finite, we can easily define such an alphabet S that codes the components of the configurations in a unique manner. Then, a transition of Π from the configuration c to c', where $c = (a_{w_1,1}, \ldots, a_{w_n,n}, a_{V_E})$ and $c' = (a_{w'_1,1}, \ldots, a_{w'_n,n}, a_{V'_E})$ can be considered as a reaction $(\{a_{w_1,1}, \ldots, a_{w_n,n}, a_{V_E}\}, \{X\}, \{a_{w'_1,1}1, \ldots, a_{w'_n,n}, a_{V'_E}\})$, where X is a special symbol used as an inhibitor. Since Π is deterministic, c' is unique; consequently, the coded versions of transitions form a reaction set, and the evolution in $\mathcal{A}$ correspond to a computation in Π.

5 Reversible reaction systems and models of computation

In the following section, we provide two examples where partially reversible reaction systems are used for simulating well-known models as logic gates, circuits, and Turing machines.

5.1 Reaction systems, reversible logic gates, and circuits

The Fredkin gate is one of the reversible logic gates. It was proposed by Fredkin and Toffoli in Fredkin and Toffoli (1982) as a logical gate that realizes the logical function $f_F : \{0,1\}^3 \to \{0,1\}^3$ such that $f_F : \{0,1\}^3 \to \{0,1\}^3$ such that $f_F : (c,p,q) \to (c, (c \wedge p) \vee (\neg c \wedge q), (c \wedge q) \vee (\neg c \wedge p))$. Further information can be found for instance in Morita (2017). First, we provide the basic notions we will use in this particular section.

A pictorial representation and operations of the Fredkin gate are demonstrated by Figs. 2 and 3. The truth table of the logical function f_F of the Fredkin gate is presented in Table 1.

Connecting reversible logic gates, a logic circuit can be composed. We define a reversible combinatorial logic circuit that realizes an injective logical function.

Definition 7 Let S be a finite set of reversible logic gates. A reversible combinatorial logic circuit F over S is

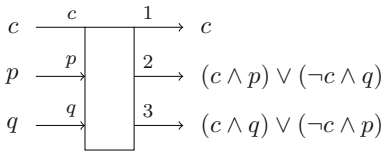

Fig. 2 A pictorial representation of the Fredkin gate

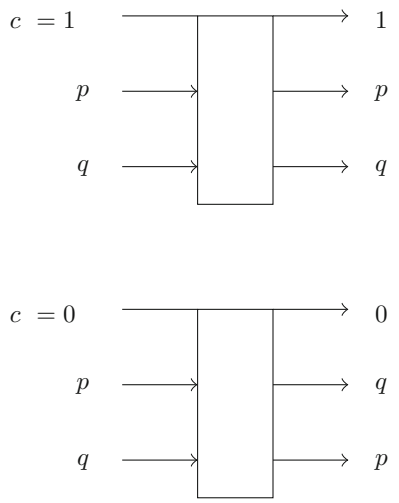

Fig. 3 Operations of the Fredkin gate

Table 1 The truth table of the logical function f_F of the Fredkin gate

c	p	q	x	y	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1

a system composed of a finite number of copies of reversible logic gates taken from S, which are connected under the following constraints.

1. Each output of a gate can be connected to at most one input of some other gate, i.e., a fan-out of the output is inhibited.
2. Two or more outputs should not be connected to one input port of some other gate, i.e., the merging of outputs is inhibited.
3. The circuit should not contain a closed path, i.e., no feedback loop is allowed.

We show that we can effectively construct a partially reversible reaction system simulating the functioning of the Fredkin gate. The background set of the reaction system is formed from elements denoting the truth value (0 or 1) with an index of the input or the output of the Fredkin gate (see Fig. 2). The set of reactions contains such reactions that ensure a transformation of the input values into the output ones.

Let $\mathcal{A}_{In} = (S, A, In)$ where

- $S = \{X_z \mid X \in \{0,1\} \wedge z \in \{c,p,q,1,2,3\}\} \cup \{I\}$ and $\{(\{0_c, X_p, Y_q\}, \{I\}, \{0_1, Y_2, X_3\}),$
- $A = (\{1_c, X_p, Y_q\}, \{I\}, \{1_1, X_2, Y_3\}) \mid X, Y \in \{0,1\}\}$
- $In = \{\{X_c, Y_p, Z_q\} \mid X, Y, Z \in \{0,1\}\}$

Since we consider only valid inputs from the set In for the construction of the transition graph, the reaction system $\mathcal{A}_{In}$ satisfies all conditions to be a partially reversible reaction system and, subsequently, we can construct an inverse reaction system $\hat{\mathcal{A}}_{In'}$.

Let $\hat{\mathcal{A}}_{In'} = (S, \hat{A}, In')$ where

- $S = \{X_z \mid X \in \{0,1\} \wedge z \in \{c,p,q,1,2,3\}\} \cup \{I\}$ and $\{(\{0_1, X_2, Y_3\}, \{I\}, \{0_c, Y_p, X_q\});$
- $\hat{A} = (\{1_1, X_2, Y_3\}, \{I\}, \{1_c, X_p, Y_q\}) \mid X, Y \in \{0,1\}\}$
- $In' = \{\{X_1, Y_2, Z_3\} \mid X, Y, Z \in \{0,1\}\}$

To design the circuit, we have to distinguish individual connections between logic gates. We use the labels of inputs and outputs of particular gates in such a way that the input of some gate has the same label as the output of its connected gate. The labeling of the connections is possible because of the conditions set in Definition 7, i.e., in the definition of the reversible combinatorial logic circuit.

The last problem to solve is that all three inputs must arrive at the gate at the same time. If they do not arrive at the same moment, the reaction system has no reaction to process them; as a result, they are erased. To prevent these outputs from being erased, we add the following reactions to the reaction system:

Let I_1, I_2, I_3 be the inputs of the Fredkin gate. Let
$(\{I_1, I_2\}, \{I_3\}, \{I_1, I_2\}),\ (\{I_2, I_3\}, \{I_1\}, \{I_2, I_3\}),$
$(\{I_1, I_3\},\ \{I_2\},\ \{I_1, I_3\}), (\{I_1\},\ \{I_2, I_3\},\ \{I_1\}),$
$(\{I_2\},\ \{I_1, I_3\},\ \{I_2\}), (\{I_3\},\ \{I_1, I_2\},\ \{I_3\})$

However, these reactions do not meet the second condition for reversible reaction system (see Definition 6). A solution for the timing problem mentioned above can be obtained by modifying the circuit. This can be done by adding delay units to each input that arrives "early". Each delay unit provides the output with the same value as the input after n time units ($n \geq 1$).

Let U_j be the output of the Fredkin gate that has to be delivered to another gate as I_k after n steps of evolution. We add the following reactions to the set A:

$(\{U_j\}, \{I\}, \{U_j^{n-1}\})$
$(\{U_j^{n-2}\}, \{I\}, \{U_j^{n-3}\}), \ldots,$
$(\{U_j^1\}, \{I\}, \{I_k\})$

In the case that the outputs can appear in the environment in different steps, we add several reactions for

simulating delay units as well. There is no reaction to the case that all outputs are present in the environment, and the interactive process terminates in this state.

For all the valid inputs, there is no state $M \subseteq S$ with $M - lhs(en(A,M)) \neq \emptyset$ and $en(A,M) \neq \emptyset$. Because of the construction of the reactions (reactants, inhibitors, and products), the system is reversible.

5.2 Reaction systems and deterministic reversible Turing machines

In the following part of the paper, we show how the computations by deterministic reversible Turing machines can be simulated by the reaction systems. For more detailed information on the reversible Turing machines, see Morita (2017).

A one-tape Turing machine (a TM, for short) is defined by a sixtuple $T = (Q, \Sigma, q_0, F, B, \delta)$ where the Q is a non-empty finite set of states, Σ is a non-empty finite set of tape symbols, q_0 is an initial state ($q_0 \in Q$), F is a set of final states ($F \subseteq Q$), and B is a special blank symbol ($B \in \Sigma$). The sixth item δ is a move relation, which is a subset of $(Q \times \Sigma \times \Sigma \times Q) \cup (Q \times \{\varepsilon\} \times \{-1, 0, +1\} \times Q)$. The symbol ε means that T does not read the tape symbol. The symbols -1, 0, and $+1$ are shift directions of the head, which are used for the "left-shift", "zero-shift", and "right-shift", respectively.

Each element of the δ is a quadruple of the form $(p, s, s', q) \in (Q \times \Sigma \times \Sigma \times Q)$, or $(p, \varepsilon, d, q) \in (Q \times \{\varepsilon\} \times \{-1, 0, +1\} \times Q)$. The quadruple (p, s, s', q) is called a read-write quadruple or a read-write rule. It means that if T reads the symbol s in the state p, then T writes s', and enters the state q. The quadruple (p, ε, d, q) is called a shift quadruple or a shift rule. This means that if T is in the state p, then it shifts the head to the direction d and enters the state q. Each state $q_f \in F$ is a halting state, i.e., there is no quadruple of the form (q_f, x, y, q) in δ.

Let $T = (Q, \Sigma, q_0, F, B, \delta)$ be a TM. T is called deterministic, if for any pair of distinct quadruples (p_1, x_1, y_1, q_1) and (p_2, x_2, y_2, q_2) in δ, the following conditions are met:

$$(p_1 = p_2) \Rightarrow (x_1 \neq \varepsilon \wedge x_2 \neq \varepsilon \wedge x_1 \neq x_2)$$

This means that for any pair of distinct rules, if their present states are the same, the rules are both read-write rules and the symbols read by them are different.

We call the T a reversible TM (RTM), if any pair of distinct quadruples (p_1, x_1, y_1, q_1) and (p_2, x_2, y_2, q_2) in δ satisfies the following:

$$(q_1 = q_2) \Rightarrow (x_1 \neq \varepsilon \wedge x_2 \neq \varepsilon \wedge y_1 \neq y_2).$$

That is, for any pair of distinct rules, if their next states are the same, then they are both read-write rules, and the written symbols are different. This is called the reversibility condition for Turing machines (given in the sixtuple form).

Theorem 4 *For every linear bounded deterministic reversible Turing machine T and input ω, there exists a partially reversible reaction system $\mathcal{A}_{In}$ which simulates the computation of the T on the ω.*

Proof If a deterministic reversible Turing machine is linear bounded, then there is a linear function such that for every input word of a length n, the space needed to process the word by the Turing machine is bounded by this linear function.

At this point, we provide the component the partially reversible reaction system $\mathcal{A}_{In}$. We start with defining the background set, S.

Let x be a $max(\mathcal{O}(n))$ for an input word of the length n. Let us label all positions on the tape from 1 to x. We use these labels as indices of symbols in an alphabet, i.e., the a_i means that the symbol a is on the i-th place of the tape. Because the simulation of one step of the Turing machine will take two steps done by the reaction system, we need to distinguish between them, and therefore we use the symbol a_i' in every second step. The position of the reading-writing head is marked as a part of a symbol of a working set of a reaction system. If the head reads the i-th symbol of the tape, then there is the symbol $\overline{a}_i$ in a working set. Let

$$\{a_i, a_i', \overline{a}_i \mid i \in \{1, \ldots, x\} \wedge a \in \Sigma\} \subseteq S.$$

If the head of the TM performs a move, then we need to remove a bar from one symbol and add it to another one. A generated symbol $\overline{a}$ can be done because the special symbols M_a or W_a are present. Let

$$\{M_{a_i}, W_{a_i} \mid i \in \{1, \ldots, x\} \wedge a \in \Sigma\} \subseteq S.$$

Finally, we add a set of symbols denoting the states of the TM to S, let

$$\{q, q' \mid q \in Q\} \subseteq S.$$

We continue with defining the set of reactions, A. The reactions simulate one step of the computation of the TM in two steps. If the element of δ of the TM is in the form (p, a, b, q) where $a, b \in \Sigma$, then there occurs this reaction

$$(\{p, \overline{a}_i\}, \{I\}, \{q', W_{b_i}\})$$

in the reaction set A. To ensure that no symbol will disappear, we also add other reactions rewriting all tape symbols into their prime versions:

$$(\{p, \overline{a}_i, c_j\}, \{I\}, \{q', W_{b_i}, c'_j\}),$$
$$1 \leq i, j \leq x, j \neq i, a, b, c \in \Sigma$$

If $(p, \varepsilon, d, q) \in \delta$, then for all $a, b \in \Sigma$, there are these reactions

$$(\{p, \overline{a}_i, b_y\}, \{I\}, \{q, a'_i, M_{b_y}\})$$
$$(\{p, \overline{a}_i, c_j\}, \{I\}, \{q', M_{b_y}, c'_j\}),$$
$$1 \leq i, j \leq x, j \neq i, a, b, c \in \Sigma$$

in the reaction set A, where $y = i + d$ and $i \in \{1, \ldots n-1\}$ for $d = +1$, $i \in \{2, \ldots n\}$ for $d = -1$ and $i \in \{1, \ldots n\}$ for $d = 0$.

In the second step of the simulation, the reactions that are enabled rewrite the symbols with primes into the symbols without primes, and one special symbol corresponding to the position of the reading-writing head into bar symbol. Let these reactions be given as follows:

$$\begin{aligned} A_W = \quad & \{(\{q', a'_i, W_{b_y}\}, \{I\}, \{q, a_i, \overline{b}_y\}) \mid \\ & i, y \in \{1, \ldots, x\} \wedge i \neq y \ \wedge a, b \in \Sigma\} \cup \\ & \cup\{(\{q', a'_y, W_{b_y}\}, \{I\}, \{q, \overline{b}_y\}) \mid \\ & y \in \{1, \ldots, x\} \ \wedge a, b \in \Sigma\} \subseteq A \\ A_M = \quad & \{(\{q', a'_i, M_{b_y}\}, \{I\}, \{q, a_i, \overline{b}_y\}) \mid \\ & i, y \in \{1, \ldots, x\} \wedge i \neq y \ \wedge a, b \in \Sigma\} \cup \\ & \cup\{(\{q', b'_y, M_{b_y}\}, \{I\}, \{q, \overline{b}_y\}) \mid \\ & y \in \{1, \ldots, x\} \ \wedge b \in \Sigma\} \subseteq A \end{aligned}$$

The computation of the reaction system $\mathcal{A}_{In}$ starts with

$$C_0 = \{\overline{a}_1, a_2, \ldots, a_n, B_{n+1}, \ldots, B_x, q_0\}$$

where $\omega = a_1 \ldots a_n$, and there is one reaction r having $q_0 \in R_r$ as well as $x-1$ reactions in the form $(\{a_i\}, \{I\}, \{a'_i\}), 2 \leq i \leq x$ in the set $E_0 = en(A, C_0)$. The next state of an interactive process W_1 is formed from prime symbols, i.e., the symbols on the tape and the state of the TM, and there is the symbol W_{a_i} (for the read-write rule) or M_{b_y} (for the shift rule) in the set W_1. Because the TM is deterministic, there is only one such rule that can be executed.

The set W_1 enables the only subset of reactions of the S_W or S_M containing the state of the TM q' and the symbol for the head positioning in the set of reactants. For every symbol on the input tape, there is one reaction. If the position (index) of a symbol is different from the position of the head, then the prime symbol is evolved into a non-prime symbol. If the position is the position of the head, the prime symbol is changed to a bar symbol. In the case of the read-write rule, the symbol on the tape is changed as well. The set W_2 contains symbols with information about their position such that for every position, there is exactly one symbol (one of them has a bar) and a symbol for the current state of the TM.

We have to check whether the constructed reaction system, $\mathcal{A}_{In}$, is partially reversible. The first condition ($P_a \cap I_a = \emptyset$) is met for all reactions. Because the inhibitor I is not used as a reactant or product in any reaction, the second and the third conditions of the definition are also met. The states of the reaction system are of two types. The first type of states corresponds to the configuration of the TM, i.e., the content of the tape with the position of the head is indicated by the bar symbol and the current state of the TM. The second type of states is associated with the configuration of the TM, that is, to every symbol on the tape, there is the prime symbol; it contains the symbol of the next state and a special symbol determining what should be done with the position of the head, and - in the case of the read-write rule - with the currently read symbol. Let c_1 and c_2 be two configurations of the TM having the same state, and let $r_1 = (p_1, x_1, y_1, q)$ and $r_2 = (p_2, x_2, y_2, q)$ be the rules that were used by the TM to enter these configurations. Because of the reversibility of the TM, if $p_1 = p_2$, then the rules cannot be shift rules and they must have $y_1 \neq y_2$ and $x_1 \neq x_2$ (it is a deterministic TM). The state of the TM and the symbol determining what action is done by the execution of a rule ($M_{\text{somewhere}}$ or $W_{\text{something}}$) uniquely identifies the rule that was used in the previous step of the computation. In consequence, it also determines a subset of all reactions that was enabled by the previous state of the TM. Because of the reversibility of the TM, this holds for the pair (a state, a symbol in the position of the head) too. Therefore, there is only one state W_{i-1} such that $res(A, W_{i-1}) = W_i$, and the fourth condition is fulfilled. As a result, the reaction system $\mathcal{A}_{In}$ is partially reversible. The members of the input set for the inverse reaction system correspond to the final configurations of the TM. Hereby, the statement of the theorem holds.□

6 Conclusion

In this paper, we introduced the sP colonies without the input influence (during computation) and set the condition of the existence of the inverse sP colony to the SP colony. In the second part of the paper, we focused on the reaction systems that work in a way similar to sP colonies. For reaction systems, we showed that for every reversible reaction system, there exists the inverse reaction system. In the last part of the paper, reaction systems simulating reversible logic gates and reversible Turing machines were presented. In the future work, we plan not only to continue our investigations of the connections between the

sP systems and reaction systems but also to study further properties of sP systems.

Acknowledgements This work is partially supported by the Silesian University in Opava under the Student Funding Scheme, project SGS/11/2019. It was also supported in part by the National Research, Development, and Innovation Office – NKFIH, Hungary, Grant No. K 120558 and and by "Integrált kutatói utánpótlás-képzési program az informatika és számítástudomány diszciplináris területein", EFOP 3.6.3-VEKOP-16-2017-00002, a Project supported by the Hungarian Government and co-funded by the European Social Fund.

Funding Open access funding provided by Eötvös Loránd University.

References

Bagossy A, Vaszil G (2020) Simulating reversible computation with reaction systems. J Membr Comput 2(3):179–193. https://doi.org/10.1007/s41965-020-00049-9

Bagossy A, Vaszil G (2020b) Transition graphs of reversible reaction systems. In: Freund R, Ishdorj T, Rozenberg G, et al (eds) Membrane computing - 21st international conference, CMC 2020, Virtual Event, September 14-18, 2020, Revised Selected Papers, Lecture Notes in Computer Science, vol 12687. Springer, pp 1–16, https://doi.org/10.1007/978-3-030-77102-7_1

Ciencialová L, Csuhaj-Varjú E, Cienciala L et al. (2019) P colonies. J Membrane Comput 1(3):178–197. https://doi.org/10.1007/s41965-019-00019-w

Ciencialová L, Cienciala L, Csuhaj-Varjú E (2020) P colonies and reaction systems. J Membrane Comput 2(4):269–280. https://doi.org/10.1007/s41965-020-00051-1

Csuhaj-Varjú E, Kelemen J, Păun Gh et al. (1994) Grammar systems: a grammatical approach to distribution and cooperation, 1st edn. Gordon and Breach Science Publishers Inc, USA

Diestel R (2005) Graph theory (Graduate Texts in Mathematics). Springer, Berlin, Heidelberg

Ehrenfeucht A, Rozenberg G (2005) Basic notions of reaction systems. In: Calude CS, Calude E, Dinneen MJ (eds) Developments in language theory. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 27–29

Fredkin E, Toffoli T (1982) Conservative logic. Int J Theor Phys 21(3):219–253. https://doi.org/10.1007/BF01857727

Kelemen J, Kelemenová A (1992) A grammar-theoretic treatment of multiagent systems. Cybern Syst 23(6):621–633

Kelemen J, Kelemenová A, Păun Gh (2004) Preview of P colonies: A biochemically inspired computing model. In: Workshop and tutorial proceedings. ninth international conference on the simulation and synthesis of living systems (Alife IX), Boston, Massachusetts, USA, pp 82–86

Kelemenová A (2010) P colonies. In: Păun Gh, Rozenberg G, Salomaa A (eds) The oxford handbook of membrane computing. Oxford University Press, New York, NY, USA, pp 584–593

Morita K (2017) Theory of reversible computing. monographs in theoretical computer science. An EATCS Series, Springer, https://doi.org/10.1007/978-4-431-56606-9

Păun Gh, Rozenberg G, Salomaa A (eds) (2010) The oxford handbook of membrane computing. Oxford University Press Inc, New York, NY, USA

Rozenberg G, Salomaa A (eds) (1997) Handbook of formal languages, Vol. 1: Word, Language, Grammar. Springer-Verlag New York, Inc., New York, NY, USA