Journal of Membrane Computing (2020) 2:179–193
https://doi.org/10.1007/s41965-020-00049-9

REGULAR PAPER

Simulating reversible computation with reaction systems

Attila Bagossy[1] · György Vaszil[1]

Received: 14 March 2020 / Accepted: 19 August 2020 / Published online: 8 September 2020

Abstract
Reaction systems are a formal model of computation providing a framework for investigating biochemical reactions inside living cells. We look at the functioning of these systems as a process producing a series of different possible sets of entities representing states which can be changed by the application of reactions, and we study reversibility and its simulation in this framework. Our goal is to establish an Undo-Redo-Do-like semantics of reversibility with environmental control over the direction of the computation following a so-called no-memory approach, that is, without introducing modifications to the model of reaction systems itself. We first establish requirements the systems must satisfy in order to produce processes consisting of states with unique predecessors, then define reversible reaction systems in terms of reversible interactive processes. For such reversible systems, we also construct simulator systems that can traverse between the states of reversible interactive processes back and forth based on the input of a special "rollback" symbol from the environment.

Keywords Reaction Systems · Natural computing · Reversible computing

1 Introduction

Natural computing is a research area concerned with computational models which may be either inspired by some natural phenomenon, or designed to help us better understand natural processes in terms of information processing [16, 29].

Reversible computation is a paradigm extending the standard notion of the forwards-only mode of computation with the ability to be executed also in the reverse direction, such that computations can run backwards as naturally as they can run forwards.[20, 22, 30].

In this paper, we study reversibility and its simulation in the framework of reaction systems, a natural computational model by Ehrenfeucht and Rozenberg [12] aiming to provide a formal framework for investigating the biochemical reactions inside living cells. Computation in this model goes forward by applying reactions to a set of entities (called a state), creating a new set (a new state). The way this model works is also interactive in the sense that each state may also incorporate input, capturing the idea that living cells do not act in isolation but always operate in some environment which may influence their behavior (and thus, the computation). In contrast to previous results, such as those in [5], we aim to investigate how to introduce reversibility in reaction systems without losing its openness (its ability to incorporate input from the enclosing environment), and without making modifications on the model itself.

The rest of the paper is organized as follows. In Sect. 2 we discuss the main paradigms concerning the implementation of reversibility in the context of computational models of our interest. Sect. 3 comprises a concise introduction to the fundamental notions of reaction systems, then in Sect. 4 we present our notion of reversibility and provide necessary and sufficient conditions for reaction systems to be reversible. In Sect. 5 we show how to construct the "reverse" of a set of reactions, and then use these ideas to construct reaction systems which are able to simulate the forward and backward computations of reversible reaction systems. Finally, Sect. 6 ends the paper with some conclusions.

✉ György Vaszil
vaszil.gyorgy@inf.unideb.hu

Attila Bagossy
bagossy.attila@inf.unideb.hu

[1] Department of Computer Science, Faculty of Informatics, University of Debrecen, Kassai út 26, Debrecen 4028, Hungary

2 Paradigms of reversibility

Reversible computation is a field attracting interest from the points of view of several possible applications and much work is also devoted to establish its solid theoretical foundations. For more information on applications of reversibility, see the monograph [26], for a state-of-the-art survey of the area, see the recent collection [30].

A number of theoretical aspects of reversible computing have been studied over the years, see [6] for a survey summarizing recent results concerning categorical foundations of reversibility, foundations of programming languages and term rewriting, various models of sequential and concurrent computations, and addressing some of the challenges posed by quantum computation (which is in part also naturally reversible).

As reaction systems are biochemically inspired computational models, we would also like to mention some of the topics of reversible computation motivated by this area. In most of the cases, biochemical reactions are modeled by distributed systems of concurrent processes and this poses special types of questions with respect to their reversibility. Opposed to sequential processes (like the computations of Turing machines or most types of conventional automata) where the order of the execution of the computational steps can easily be reversed by undoing the last action, the definition of the backward execution of a collection of concurrently executing distributed processes is not straightforward at all, since there is no definite notion of the "last action" which should be undone first. To overcome this difficulty, the concept of causal-consistent reversibility was introduced in [10] as a suitable definition of reversibility for a concurrent scenario, which intuitively says that any action can be undone provided that all its consequences, if any, are undone already. Interestingly, besides the usual "backtracking" type and the more sophisticated causal-consistent type of reversibility, out-of-causal-order reversibility can also be defined, and as it may sound strange, it is important since it might make possible to get to states which cannot be reached by forward execution alone [27].

Another interesting aspect is the controlled vs. uncontrolled nature of reversibility. We speak of external control of reversible computation when some other process is in charge of controlling it by deciding whether it has to go backward or forward [18]. In Section 5 we will follow a similar approach when we present simulations of reversible reaction systems being able to switch between simulating forward or backward computations based on external input from the environment.

For a survey of causal consistent and controlled reversibility, see also [19].

As we already noted, biochemically inspired computational models, even when they are abstract and very much simplified, naturally include some kind of concurrency and parallelism appearing between its different possible computational processes. Suitable examples are membrane systems [24, 25] which deal with multisets of symbols processed in the compartments of hierarchical structure of membranes according to some multiset rewriting rules: some of the symbols are changed in parallel according to the rules associated with their containing regions, while the others remain unchanged (and can be used in the subsequent steps) or get moved to other regions of the membrane structure. Concerning the reversibility of membrane systems, [1, 2] defines it as a form of duality, while in [4] the reversibility of biochemical reactions in parallel rewriting systems are investigated (which can easily represent classes of models such as membrane systems, or Petri nets). In a more recent paper [28], membrane system configurations are enriched with a memory recording the information necessary for reversing steps.

The situation in the case of reaction systems, however, is different. Although, they are motivated by (and in a certain sense can also be used to model) biochemical processes, they represent a qualitative model. As opposed to systems being able to "count" by dealing with entities having multiplicities (as in multisets, for example), reaction systems deal with sets, which distinguishes them from the above mentioned models by (i) a threshold assumption: if a resource is present, it is always present in a sufficient amount necessary for any reaction; (ii) no permanency assumption: if an entity is not produced at a certain step, it will not be available for use at the next step [12]. As we will see later in more detail, such a way of functioning leads to a sequential model, very close in fact to finite transition systems (or finite automata). The concept of reversibility in the context of these types of computational models is rather straightforward. A model is reversible if it is "backward deterministic", that is if each of its computational configurations (or states) has at most one predecessor, or in other words, no state is accessible from two distinct states[1]. As simple as this definition is, it gives rise to different implementational paradigms among the actual models, some of which we will shortly review in the following. Our presentation is based mostly on [22, 26].

As an automaton is reversible if it preserves information so that its computation can be retraced back in time [6], information preserving and reversibility are very closely related concepts. Landauer argued in [17] that any logical operation with information loss necessarily results in heat

[1] A similar approach is also possible in the case of membrane systems by considering deterministic variants, as done in [3, 15], or in [23] where reversible register machines are simulated.

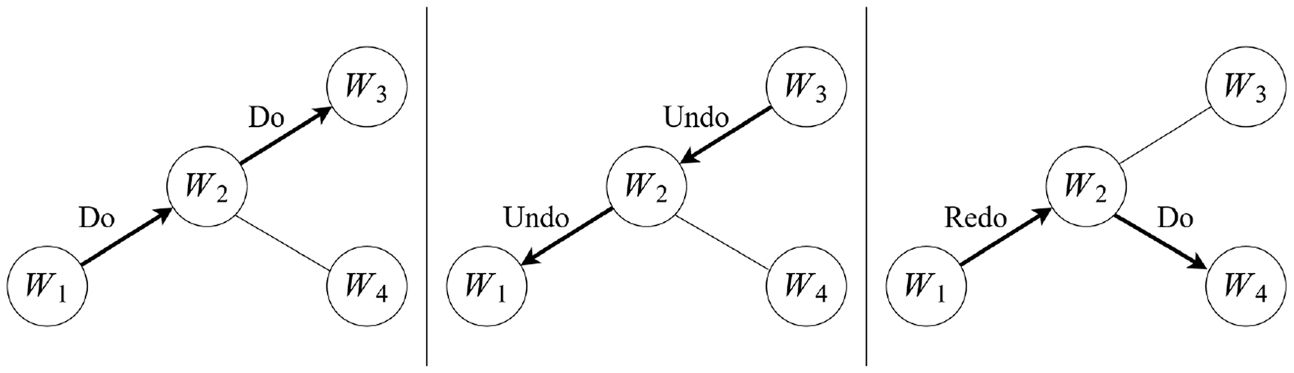


Fig. 1 Altering the flow of computation in the Undo-Redo-Do paradigm

dissipated by the system performing the operation, and as there exist irreversible computations, he also claimed that there are computations where heat generation is inevitable. Inspired by these claims, Bennett created a universal reversible Turing machine, proving that irreversibility is not an inherent property of computation [8]. When constructing his model, he developed the so-called Compute-Copy-Uncompute paradigm, which comprises the following steps: The machine first performs a reversible forward computation, resulting in the desired output. Then, a copy is made from this output. Finally, the reverse execution of the forward steps (which is also reversible) cleans up the effects of the forward computation, leaving the original input and the copy of the corresponding output on the tapes. (See [26] for a more detailed and systematic treatment of this paradigm.) This result spawned extensive research interest in the field of reversible computing since it proved that reversible computing is the tool to overcome the performance constraints of traditional irreversible systems [13]. A more recent and rigorous treatment of reversibility concerning Turing machine computations can also be found in [7].

While the Compute-Copy-Uncompute paradigm fits the power consumption related study of reversibility well, it might be too static for others, since the outcome of the computation is of most importance, as opposed to the actual process of the computation itself. If we place our focus on the processes, however, we can discover another significant implication of reversibility: it allows for exploration and experimentation. Since every configuration has at most one predecessor, we can freely undo any previous computation and proceed by choosing a different computational route.

This very idea serves as the basis of the Undo-Redo-Do paradigm, depicted in Fig. 1. Below, we briefly describe the flow of computation in this paradigm, as discussed in [26].

– The *Do* operation corresponds to normal forward computation.
– At any state, we can choose to *Undo* our previous step, essentially reversing the execution, taking us back to the single predecessor of the current state.
– Later, if we wish to recover our prior computation (thus, visiting the same states as before), we can perform a *Redo*.
– Instead of recovering previous actions, if we want to experiment by taking a different route, we can dismiss any undone steps yet to be redone, and continue with a *Do* operation.

A similar approach was taken in [14] where reversible nondeterministic automata were investigated, and although in a different context, the idea of exploration and experimentation was also discussed in [11].

When considering reversibility in the case of (sequential) models with an emphasis on interaction with some external environment (such as reaction systems), the implementation of a paradigm like Undo-Redo-Do seems to be more suitable than, for example, Compute-Copy-Uncompute since it is well-aligned with the dynamic and exploratory characteristics of these models. Moreover, as we already mentioned, the process-focused nature of the paradigm (in contrast to the result oriented focus of Compute-Copy-Uncompute) also motivates its use in the following investigations.

3 Preliminaries

In what follows, we are going to briefly introduce the most important notions and notations concerning reaction systems. Our presentation is based on [9].

Reaction systems model biochemical reactions occurring inside a living cell. The intuition behind the model stems from the idea of facilitation and inhibition.

A *reaction* a is a triplet of three finite sets $a = (R_a, I_a, P_a)$. The set R_a contains the *reactants*, I_a contains the *inhibitors*, while P_a consists of the *products*. The set of reactants and the set of inhibitors must be disjoint ($R_a \cap I_a = \emptyset$) and the set of products must not be empty ($P_a \neq \emptyset$).

If $R_a, I_a, P_a \subseteq S$ for some reaction $a = (R_a, I_a, P_a)$ and finite set S, then a is a reaction over S. The set of all reactions over S is denoted by $\mathrm{rac}(S)$.

Intuitively, we can say that a reaction takes place if all of its reactants and none of its inhibitors are present. In such a case the reactants react, and the product set is created, just like in the case of biochemical reactions.

Remark 1 In what follows, if a is a reaction, then we will denote its components as R_a, I_a and P_a without explicitly writing out the complete triplet form $a = (R_a, I_a, P_a)$.

We also note that we use the symbols $\subseteq$ and $\subset$ to denote set inclusion and set inclusion in the strict sense (that is, when equality is excluded), respectively.

Given a set of entities (that can be arbitrary symbols) S and a reaction $a \in \mathrm{rac}(S)$, we can always tell whether a can take place or not. For a finite set $T \subseteq S$, the reaction a is *enabled by T*, if $R_a \subseteq T$ and $I_a \cap T = \emptyset$. The *result of a on T*, denoted by $\mathrm{res}_a(T)$ is defined as $\mathrm{res}_a(T) = P_a$ if a is enabled by T, or $\mathrm{res}_a(T) = \emptyset$ if a is not enabled by T.

In the previous definitions, we only considered a single reaction. It is quite rare, however, that a single reaction can capture the behavior of a complex process. Consequently, the concepts above should be generalized to multiple reactions.

Let A be a finite set of reactions over a set of entities S, and let $T \subseteq S$ be a finite set. Then $\mathrm{en}_A(T)$ denotes the *set of all reactions in A enabled by T*, thus $\mathrm{en}_A(T) = \{a \in A \mid a \text{ is enabled by } T\}$, and the *result of A on T*, denoted by $\mathrm{res}_A(T)$, is defined as $\mathrm{res}_A(T) = \bigcup_{a \in A} \mathrm{res}_a(T)$.

The previous definition reflects two vital aspects of the model. Reactions can be enabled and applied even if their reactant sets overlap. Hence, there are no conflicts: each enabled reaction is allowed to produce its result. The results are also non-conflicting, which means that even if two or more reactions produce the same entity (because of intersections in their product sets), there will be no "multiple occurrences" in the result set. This qualitative nature comes from the fact that reaction systems use sets instead of multisets (these can be viewed as the quantitative counterparts of sets). Thus, in this model, an entity (which might be representing some chemical or biological resource) is either present in an amount that is sufficient or it is missing altogether.

In the following sections, we are going to work with finite sets of reactions. Therefore, we now introduce a shorthand notation for the reactants and products of these sets.

Notation 1 Let A be a finite set of reactions. Then, we denote by R_A and P_A the union of the reactant sets and product sets, respectively: $R_A = \cup_{a \in A} R_a$, $P_A = \cup_{a \in A} P_a$.

We can further generalize the concept of enabled reactions and results, thus formalizing every possible enabled subset of reactions and the results of those.

Notation 2 Let A be a finite set of reactions over the finite set S. We denote by $\mathrm{EN}_A(S)$ the set that contains the sets of reactions which can be enabled simultaneously, that is, every $E \subseteq A$ for which there exists a subset of S enabling every reaction in E. Formally

$$\mathrm{EN}_A(S) = \{E \subseteq A \mid \text{ there exists } S' \subseteq S, \text{ such that } \mathrm{en}_A(S') = E\}.$$

In other words, $\mathrm{EN}_A(S)$ contains the sets of reactions where the members of each set are simultaneously enabled for some subset S' of S.

Further, we denote by $\mathrm{RES}_A(S)$ the set that contains the results of applying every set of reactions in $\mathrm{EN}_A(S)$ to the appropriate subsets of entities, or formally

$$\mathrm{RES}_A(S) = \{\mathrm{res}_E(S') \mid S' \subseteq S, E \subseteq A, \text{ such that } \mathrm{en}_A(S') = E\}.$$

Example 1 Let us consider the set of reactions $A = \{a, b, c\}$ over $S = \{1, 2, 3\}$ where

$$a = (\{1\}, \emptyset, \{2\}),\ b = (\{2\}, \emptyset, \{3\}),\ c = (\{3\}, \{1\}, \{1\}).$$

Here, we have

$$\mathrm{EN}_A(S) = \{\ \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}\ \},$$

since there is no set of elements such that a and c is enabled together (as $R_a \cap I_c \neq \emptyset$).

The elements of $\mathrm{RES}_A(S)$ are the product sets produced by the reactions in the sets of $\mathrm{EN}_A(S)$ applied to appropriate subsets of S, that is,

$$\mathrm{RES}_A(S) = \{\ \{2\}, \{3\}, \{1\}, \{2, 3\}, \{1, 3\}\ \}.$$

Having established the most important notions, we can now recall the definition of a *reaction system*, which is an ordered pair $\mathscr{A} = (S, A)$ such that S is a finite set (called the background set) and $A \subseteq \mathrm{rac}(S)$ (called the set of reactions).

Remark 2 Given a reaction systems $\mathscr{A} = (S, A)$, we assume, without loss of generality, that A does not contain different reactions having the same set of reactants and the same set of inhibitors. (If $a, a' \in A$ are reactions with $a = (R, I, P_1)$ and $a' = (R, I, P_2)$, then they are always enabled and applied simultaneously, so they can be replaced by a single reaction $(R, I, P_1 \cup P_2)$ having the same effect.)

Let $\mathscr{A} = (S, A)$ be a reaction system and let $n \geq 0$ be an integer. An *interactive process in* $\mathscr{A}$ is a pair $\pi = (\gamma, \delta)$ of finite sequences, such that

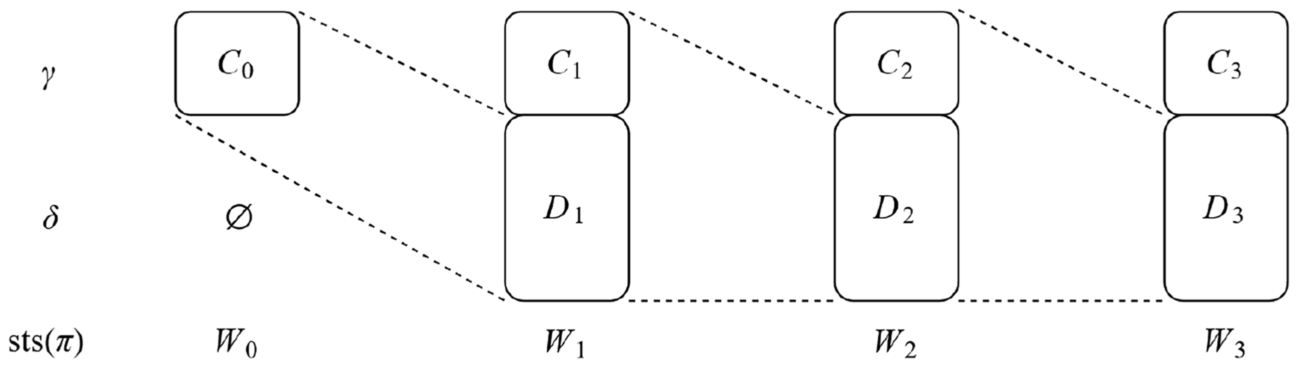


Fig. 2 The first four states of an interactive process $\pi = (\gamma, \delta)$ in some reaction system

- γ is the *context sequence* of π, defined as $\gamma = C_0, C_1, \dots C_n$, where $C_i \subseteq S$ for all $0 \le i \le n-1$, and $C_n = \emptyset$,
- δ is the *result sequence* of π, defined as $\delta = D_0, D_1, \dots D_n$, where $D_0 = \emptyset$ and $D_i = \text{res}_A(D_{i-1} \cup C_{i-1})$ for all $1 \le i \le n$,
- $\text{sts}(\pi)$ is the *state sequence* of π, defined as $\text{sts}(\pi) = W_0, W_1, \dots W_n$, where $W_i = C_i \cup D_i$ for all $0 \le i \le n$.

The first few states of an interactive process are visualized in Fig. 2. Note that even though the figure shows the context set C_i and the result set D_i of the same state as non-overlapping rectangles, they do not have to be disjoint.

The way interactive processes are defined has two significant consequences. An interactive process is finite since the prior choice of n determines its length. Thus, we can think of an interactive process as a finite state sequence. In this sequence, new states are created rather than transformed from the previous states. This leads us to the concept of no permanency: The next state is created from the products of the enabled reactions and the context set. Therefore, if an entity in the current state is not produced by any reaction and is not present in the next context set, then it will disappear. The intuition behind this idea comes from the realm of biochemistry where entities are produced and sustained by active, energy-consuming processes (not to be confused with our formal interactive processes). Thus, if there is no such process to sustain a given entity, then it will vanish. More information on the motivation behind this concept (and on reaction systems in general) can be found in [12].

4 Reversible reactions systems

We start by presenting a possible application of the concept of reversibility for the special case of reaction systems. Based on the discussions in Sect. 2, our aim is to interpret the notion of backward determinism in this framework. In reaction systems state sequences correspond to interactive processes, so we start by defining how the concept of unique predecessors can be applied to the states of interactive processes.

Definition 1 Let $\mathscr{A} = (S, A)$ be a reaction system and $\pi = (\gamma, \delta)$ be an interactive process in $\mathscr{A}$, such that $\gamma = C_0, C_1, \dots C_n$ and $\text{sts}(\pi) = W_0, W_1, \dots W_n$.

A state W_i, $1 \le i \le n$, has *multiple predecessors* if there exists $W'_{i-1}, C'_{i-1} \subseteq S$ such that $W'_{i-1} \neq W_{i-1}$, but $\text{res}_A(W'_{i-1}) \cup C'_{i-1} = W_i$. If there is no such W'_{i-1}, then W_i has a *unique predecessor*.

The interactive process π is *reversible*, if every state W_i, $1 \le i \le n$, has a unique predecessor.

In the following, we will discuss the conditions that are necessary for obtaining reversible interactive processes. It is clear that if different reactions produce the same result, then we can arrive at the same state from different predecessor states. Moreover, since a state is the union of a context set and a result set, the property of having a unique predecessor does not only depend on the reactions and the results of the reactions, but also on the context sets which are added in each step of the interactive process. Even if every result set was unique, identical states could be created with well-chosen contexts. In order for an interactive process to be reversible, there must be at most one way to produce each context-result union, so the unique predecessor property of Definition 1 depends not only on the reactions of the system, but also on the context sets of the process. Since this dependency is rather involved, we are going to follow a step by step approach and introduce lemmas for describing the different requirements related to the reversibility of interactive processes.

First, let us consider the following. According to the general notion of reversibility, every state must have a unique predecessor. Yet, when a given state does not enable any reactions, then the process continues with an empty result. We can then augment this empty result with a non-empty context, essentially restarting the process. Reversing execution from such an empty or restarting state would be equivalent to obtaining "something from nothing" which does not fit the concept of reversibility. Therefore, in what follows, we will only consider processes in which this restart does not occur.

Definition 2 Let $\mathscr{A}$ be a reaction system and $\pi = (\gamma, \delta)$ be an interactive process in $\mathscr{A}$ such that $\delta = D_0, D_1, \dots D_n$. The interactive process π is *non-restarting* if $D_i \neq \emptyset, 1 \leq i \leq n$. If the opposite holds, then π is *restarting*.

Remark 3 If $\pi = (\gamma, \delta)$ is an interactive process with $\delta = D_0, D_1, \dots D_n$ and $\mathrm{sts}(\pi) = W_0, \dots, W_n$ in some $\mathscr{A} = (S, A)$, then (since the product sets of the reactions are nonempty) $D_i = \emptyset$ is only possible, if $\mathrm{en}_A(W_{i-1}) = \emptyset$ for some $1 \leq i \leq n$. Thus, π is non-restarting, if and only if, $\mathrm{en}_A(W_i) \neq \emptyset$ for all $i,\ 0 \leq i \leq n-1$.

We already noted that reversibility implies the uniqueness of result sets. This comes from the fact that regardless of the context, if there are multiple ways to produce the same set of entities, then it is not possible to recover the predecessor.

Example 2 Let $S = \{0, 1, 2, 3, 4\}$ be a set of entities and $A = \{a, b, c\}$ be a set of reactions where

$$a = (\{0\}, \emptyset, \{3\}),\ b = (\{0, 1\}, \emptyset, \{4\}),\ c = (\{2\}, \emptyset, \{3, 4\}).$$

If we consider the set $W = \{3, 4\}$, we can see that there are multiple subsets of reactions in A producing W, for example $\{a, b\}$ and $\{c\}$. As a consequence, just by looking at W, we are unable to determine which reactions produced it. Equivalently, if W was a state in some interactive process, then this process would not be reversible, since we could not recover the predecessor of W.

Deriving from the above example, we can impose the following requirement on the set of reactions. If we take every possible subset of reactions in which each reaction is enabled, then no two subsets should produce the same result. As the following lemma states, if the same result set is produced by two different enabled reaction sets, then there exists a state with multiple predecessors.

Lemma 1 *Let $\mathscr{A} = (S, A)$ be a reaction system. If there exist $E_1, E_2 \in \mathrm{EN}_A(S)$ with $E_1 \neq E_2$, such that $P_{E_1} = P_{E_2}$, then there exists a state W in some interactive process in $\mathscr{A}$, such that W has multiple predecessors.*

Proof Let $\mathscr{A} = (S, A)$ be a reaction system. Assume, that there exists $E_1, E_2 \in \mathrm{EN}_A(S)$, $E_1 \neq E_2$, such that $P_{E_1} = P_{E_2}$. Because $E_1, E_2 \in \mathrm{EN}_A(S)$, we have $T_1, T_2 \subseteq S$ satisfying $\mathrm{en}_A(T_1) = E_1$ and $\mathrm{en}_A(T_2) = E_2$. In addition, since $E_1 \neq E_2$, we also have $T_1 \neq T_2$. This means that there exist $T_1, T_2 \subseteq S$ such that $T_1 \neq T_2$ and $\mathrm{res}_A(T_1) = \mathrm{res}_A(T_2)$, since $P_{E_1} = P_{E_2}$.

Thus, given some context set $C \subseteq S$, if $\mathrm{res}_A(T_1) \cup C = W$, then we also have $\mathrm{res}_A(T_2) \cup C = W$ with $T_1 \neq T_2$. That being so, if W is a state in some interactive process, then W has multiple predecessors.

To see that such an interactive process always exists, consider $\pi = W_0, W_1, \dots W_n$, $n \geq 0$, with $W_0 = C_0$, where $C_0 = T_1$ (or $C_0 = T_2$) is the initial context set. Then, since $W_1 = \mathrm{res}_A(C_0) \cup C_1$, the state W_1 in the interactive process π is a state with multiple predecessors. □

Now, one might be tempted to conclude, that the assumption of Lemma 1 provides a sufficient condition for reversibility in the case of context-independent interactive processes (those with empty contexts except for C_0). Since there is a single reaction set producing every result set and the context is always empty (hence, $W_i = D_i$ for all $i \geq 1$), every state should have a unique predecessor. However, because of the *no permanency* assumption in reaction systems, this is not necessarily the case. According to the no permanency assumption, entities not sustained by at least one reaction will disappear, which can result in states with multiple predecessors even in context-independent interactive processes.

Example 3 Let $S = \{0, 1, 2\}$ be a set of entities and $A = \{a, b\}$ be a set of reactions where

$$a = (\{0\}, \emptyset, \{1\}),\ b = (\{0, 1\}, \emptyset, \{2\}),$$

so Lemma 1 is not applicable to this system.

Consider the sets $W_1 = \{0, 1, 2\}$ and $W_2 = \{0, 1\}$. If we apply the reactions in A to these sets, we get $\mathrm{res}_A(W_1) = \mathrm{res}_A(W_2) = \{1, 2\}$. Thus, if $W = \{1, 2\}$ is a state in some context-independent interactive process, then W has multiple predecessors.

In the above example, the non-uniqueness of the predecessor was caused by a vanishing entity (the entity 2), one that was not a reactant of any of the enabled reactions. Given this observation, we might conjecture that the presence of entities not contained by the reactant sets of any of the reactions should imply the existence of states with multiple predecessors. The following example shows that this is not necessarily so.

Example 4 Let $S = \{0, 1\}$ be a set of entities and $A = \{a, b\}$ be a set of reactions where

$$a = (\{0\}, \emptyset, \{0\}),\ b = (\{0\}, \{1\}, \{1\}).$$

Consider the set $W_1 = \{0\}$. If we apply the reactions in A to W_1, we get $\mathrm{res}_A(W_1) = \{0, 1\}$. Given this result set, we can deduce that the applied reactions were a and b, and can restore the original set W_1.

Now, let us take the set $W_2 = \{0, 1\}$. Since the entity 1 is not a reactant of the reactions, it will disappear when any of the above reactions are applied. In this case, however, we can only apply a since 1 is an inhibitor of b, so we get $\mathrm{res}_A(W_2) = \{0\}$. Even though the element 1 has vanished because it was not sustained by any of the reactions, we can deduce its presence in the predecessor set of $\{0\}$ because it inhibited the reaction b.

As the above example demonstrates, facilitation (being a reactant) is not the only thing that leaves a trace. Since inhibitors also affect the result of reaction application, we might be able to recover them from the result set. Thus, the presence of an entity which is not a reactant of any applied reaction in some state of an interactive process does not necessarily imply the existence of multiple predecessors. With this in mind, we can reformulate our previous observation: The "problematic states" are those, which contain entities whose presence or absence does not affect the set of enabled reactions. This is expressed in the following statement.

Lemma 2 *Let $\mathscr{A} = (S, A)$ be a reaction system. If there exits a set $T \subseteq S$, $\mathrm{en}_A(T) \neq \emptyset$, and an entity $e \in T$, such that*

$$\mathrm{en}_A(T \setminus \{e\}) = \mathrm{en}_A(T),$$

then the there is a state W in some interactive process in $\mathscr{A}$, such that W has multiple predecessors.

Proof Assume, that there exist $T_1 \subseteq S$, $e \in T_1$, such that $\mathrm{en}_A(T_1 \setminus \{e\}) = \mathrm{en}_A(T_1)$, and let $T_2 = T_1 \setminus \{e\}$. This means, that $\mathrm{res}_A(T_1) = \mathrm{res}_A(T_2)$, so if $W = \mathrm{res}_A(T_1) \cup C$ is a state of an interactive process for some context set $C \subseteq S$, then $T_1 \neq T_2$, but both T_1 and T_2 are predecessors of W.

To see that such an interactive process always exists, consider $\pi = W_0, W_1, \ldots W_n$, $n \geq 0$, with $W_0 = C_0$, where $C_0 = T_1$ (or $C_0 = T_2$) is the initial context set. Then, since $W_1 = \mathrm{res}_A(C_0) \cup C_1$, the state W_1 in the interactive process π is a state with multiple predecessors. □

The lemma above implies that whenever the same set of reactions is enabled by two or more different sets of entities, then there is a state with multiple predecessors in the corresponding interactive processes.

Corollary 1 *Let $\mathscr{A} = (S, A)$ be a reaction system. If there exist $T_1, T_2 \subseteq S$, $T_1 \neq T_2$, such that $\mathrm{en}_A(T_1) = \mathrm{en}_A(T_2) \neq \emptyset$, then there exists a state W in some interactive process in $\mathscr{A}$ such that W has multiple predecessors.*

As we briefly noted, when discussing Definition 1, the notion of unique predecessor depends on both the reactions of the containing system and the context sets of the enclosing process. Given an appropriate set of reactions, it might still be possible to construct states with multiple predecessors, even if none of the above lemmas are applicable. To see this, consider the following. When assembling a new interactive process, we can make arbitrary choices regarding the elements of the contexts sets. Consequently, for every pair of distinct result sets, we can always choose an appropriate context set, so that the union of these sets will be equal.

Example 5 Let $S = \{0, 1\}$ be a set of entities and $A = \{a, b, c\}$ be a set of reactions where

$$a = (\{0\}, \{1\}, \{0, 1\}),\ b = (\{1\}, \{0\}, \{0\}),\ c = (\{0, 1\}, \emptyset, \{1\}).$$

None of the previous lemmas are applicable to this system, but it still produces a state with multiple predecessors. Consider the states $W = \{1\}$ and $W' = \{0\}$. As $\mathrm{res}_A(W) = \{0\}$ and $\mathrm{res}_A(W') = \{0, 1\}$, if we have a state $W_i = \{0, 1\}$ of an interactive process for some $i \geq 1$ with $C_i = \{1\}$, then W_i has multiple predecessors: Since $W_i = \mathrm{res}_A(W) \cup C_i = \mathrm{res}_A(W') \cup C_i = \{0, 1\}$, the predecessor of W_i can be any of the states W or W'.

If the context sets can be arbitrary subsets of the background set (even the background set itself can be a context), then regardless of how well-chosen our reactions are, an appropriate context set can turn a state into one with multiple predecessors. Thus, we need to restrict which entities may appear in the contexts, or in other words, there have to be entities which cannot appear in any context set. To this aim, we write the background set S of a reaction system as the union of two not necessarily disjoint sets, the *product alphabet* $\Sigma_p \subseteq S$ (entities that appear in the product sets of the reactions) and the *context alphabet* $\Sigma_c \subset S$ (entities that can appear in the context sets). The model we obtain this way is similar to the one called context restricted reaction systems in [21].

Notice, however, that if restricting the sets of possible contexts, then there might be states which are not "reachable" in the sense that they cannot appear in any interactive process.

Definition 3 Let $\mathscr{A} = (S, A)$ be a reaction system with $S = \Sigma_p \cup \Sigma_c$, that is, the background set being the (not necessarily disjoint) union of Σ_p, the entities that are allowed to appear as products of reactions and Σ_c, the entities that are allowed to appear in the context sets.

A *state* $W \subseteq S$ *is reachable* if there exists an interactive process $W_0, W_1, \ldots, W_n$ in $\mathscr{A}$ with $W = W_i$ for some $0 \le i \le n$, such that $W_i = D_i \cup C_i$ with $D_i \subseteq \Sigma_p$, $C_i \subseteq \Sigma_c$, and $D_0 = C_n = \emptyset$.

The following statement establishes a relationship between the properties of the reaction sets, the context alphabet, and the existence of states with multiple predecessors.

Lemma 3 *Let* $\mathscr{A} = (S, A)$ *be a reaction system with* $S = \Sigma_p \cup \Sigma_c$ *(where* Σ_p *and* Σ_c *are not necessarily disjoint). If there exist* $R_1, R_2 \in \mathrm{RES}_A(S)$ *such that* $R_1 \neq R_2$, $R_1 = \mathrm{res}_A(W)$ *for some state* $W \subseteq S$ *which is reachable in* $\mathscr{A}$*, and*

$$R_1 \setminus \Sigma_c = R_2 \setminus \Sigma_c,$$

then there exists a state with multiple predecessors in some interactive process in $\mathscr{A}$.

Proof Let $\mathscr{A} = (S, A)$ be a reaction system with $S = \Sigma_p \cup \Sigma_c$. Assume that there exist $R_1, R_2 \in \mathrm{RES}_A(S)$ satisfying the conditions of the statement. As R_1 and R_2 are in $\mathrm{RES}_A(S)$, there exist $W, T \subseteq S$ such that $\mathrm{res}_A(W) = R_1$, $\mathrm{res}_A(T) = R_2$, and W is a reachable state in $\mathscr{A}$. (Note that $W \neq T$, since given a fixed set of reactions, different result sets may only be created from different states.) Furthermore, since $R_1 \neq R_2$ but $R_1 \setminus \Sigma_c = R_2 \setminus \Sigma_c$, if we choose the context set C as $C = (R_1 \cap \Sigma_c) \cup (R_2 \cap \Sigma_c)$, then we have

$$R_1 \cup C = R_2 \cup C,$$

so there exist $W, T \subseteq S$, $W \neq T$, and $C \subseteq \Sigma_c$ such that,

$$\mathrm{res}_A(W) \cup C = \mathrm{res}_A(T) \cup C = W'$$

for some state $W' \subseteq S$.

Since W is reachable in $\mathscr{A}$, there is an interactive process π, such that W' is a state in π, and as $W' = \mathrm{res}_A(W) \cup C = \mathrm{res}_A(T) \cup C$ with $W \neq T$, the state W' in π has multiple predecessors. □

To see the condition of the previous statement from a different point of view, we may also formulate it as follows.

Corollary 2 *Let* $\mathscr{A} = (S, A)$ *be a reaction system with* $S = \Sigma_p \cup \Sigma_c$ *(where* Σ_p *and* Σ_c *are not necessarily disjoint). If there exist* $R_1, R_2 \in \mathrm{RES}_A(S)$ *such that* $R_1 \neq R_2$, $R_1 = \mathrm{res}_A(W)$ *for some state* $W \subseteq S$ *which is reachable in* $\mathscr{A}$*, and*

$$(R_1 \cup R_2) \setminus (R_1 \cap R_2) \subseteq \Sigma_c,$$

then there exists a state with multiple predecessors in some interactive process in $\mathscr{A}$.

Since computation in reaction systems is done using interactive processes, we can naturally formalize the definition of reversible reaction systems based on the reversibility of interactive processes.

Definition 4 A *reaction system* $\mathscr{A}$ *is reversible*, if every non-restarting interactive process in $\mathscr{A}$ is reversible.

Based on the lemmas above, we can formulate the necessary and sufficient conditions for the reversibility of a reaction system as follows.

Theorem 1 *Let* $\mathscr{A} = (S, A)$ *be a reaction system with* $S = \Sigma_p \cup \Sigma_c$ *(where* Σ_p *and* Σ_c *are not necessarily disjoint). The system* $\mathscr{A} = (S, A)$ *is reversible, if and only if the following conditions hold.*

(1) *For all* $E_1, E_2 \in EN_A(S)$,

$E_1 \neq E_2$ implies $P_{E_1} \neq P_{E_2}$.

(2) *For all* $T_1, T_2 \subseteq S$, $\mathrm{en}_A(T_1) \neq \emptyset$,

$T_1 \neq T_2$ implies $\mathrm{en}_A(T_1) \neq \mathrm{en}_A(T_2)$.

(3) *For all* $R_1, R_2 \in \mathrm{RES}_A(S)$ *such that* $R_1 = \mathrm{res}_A(W)$ *for some state* $W \subseteq S$ *which is reachable in* $\mathscr{A}$,

$R_1 \neq R_2$ implies $R_1 \setminus \Sigma_c \neq R_2 \setminus \Sigma_c$.

Proof According to the Lemma 1, Corollary 1, and Lemma 3, no reaction system can be reversible if any of the above conditions does not hold.

To also see that the conditions imply the reversibility of a system, let us assume indirectly that there is a reaction system $\mathscr{A} = (S, A)$ (where $S = \Sigma_p \cup \Sigma_c$) which satisfies all three conditions of the theorem, but is not reversible. As $\mathscr{A}$ is not reversible, there is a non-restarting interactive process $\pi = (\gamma, \delta)$ in $\mathscr{A}$ with $\mathrm{sts}(\pi) = W_0, \ldots, W_n$, where $W_i = D_i \cup C_i$ with $D_i \subseteq \Sigma_p$, $C_i \subseteq \Sigma_c$, $0 \le i \le n$, such that there is an $i \ge 1$ for which W_i has multiple predecessors, that is,

$$\mathrm{res}_A(W) \cup C = \mathrm{res}_A(W_{i-1}) \cup C_i = W_i \text{ for some } W \neq W_{i-1}, \tag{1}$$

with $W \subseteq S$, and $C, C_i \subseteq \Sigma_c$.

Since $W \neq W_{i-1}$, we have $\mathrm{en}_A(W) \neq \mathrm{en}_A(W_{i-1})$ according to condition (2), and then $P_{\mathrm{en}_A(W)} \neq P_{\mathrm{en}_A(W_{i-1})}$, that is,

$$\mathrm{res}_A(W) \neq \mathrm{res}_A(W_{i-1})$$

according to condition (1). Since W_{i-1} is reachable in $\mathscr{A}$, condition (3) is applicable, which implies

$$\mathrm{res}_A(W)\backslash\Sigma_c \neq \mathrm{res}_A(W_{i-1})\backslash\Sigma_c.$$

This means that $\mathrm{res}_A(W)$ and $\mathrm{res}_A(W_{i-1})$ differ also in entities that are not in Σ_c, therefore, there is no $C, C' \subseteq \Sigma_c$ such that

$$\mathrm{res}_A(W) \cup C = \mathrm{res}_A(W_{i-1}) \cup C'$$

which contradicts our assumption at (1), and thus completes the proof. □

Example 6 Let $\mathscr{A} = (S, A)$ be the reaction system with $S = \Sigma_c \cup \Sigma_p$, $\Sigma_p = \{1, 3, 5\}$ being the product alphabet, $\Sigma_c = \{0, 2, 4\}$ being the context alphabet, and $A = \{a, b, c\}$ being the set of reactions, where

$$a = (\{0\}, \{1, 2, 3, 4, 5\}, \{1\}),\ b = (\{1, 2\}, \{0\}, \{3\}),$$
$$c = (\{1, 4\}, \{0\}, \{5\}).$$

According to Theorem 1, $\mathscr{A}$ is reversible as it satisfies all three conditions.

In a reaction system satisfying the conditions of Theorem 1, every non-restarting interactive process is reversible, even in the presence of input from the environment in the form of context sets. Therefore, given any state of some non-restarting interactive process, the predecessor of this state is unique and can be restored. Using this environmental input, however, one can only control the forward computation of the system. In the next section, we are going to extend these results so that the environment can also trigger backward computation.

5 Simulating reversible reaction systems

As discussed in Sect. 2, we believe that the Undo-Redo-Do paradigm fits well to the reversible variants of computational models with environmental interaction, as opposed to other, more static paradigms (such as Compute-Copy-Uncompute). The reversible reaction systems of Theorem 1, however, do not support this kind of controlled reversibility. To enable interactive environmental control over the direction of the computation inside a reaction system, we discuss in this section the interactive simulation of such systems. To this aim, we also need to be able to construct "reverse reactions", reactions which execute the backward computations of reversible reaction systems.

When creating simulating systems, we are going to make extensive use of interactive processes and consequently, finite sequences of sets. To ease notation, here we will introduce some new notations regarding such sequences.

Notation 3 Let $\mathcal{W} = W_0, W_1, \ldots, W_n$ be a finite sequence of $n + 1$ sets. Then

- the *length of* $\mathcal{W}$ is denoted by $|\mathcal{W}|$,
- the *reverse of* $\mathcal{W}$ is denoted by $\overleftharpoondown{\mathcal{W}}$ and is defined as $\overleftharpoondown{\mathcal{W}} = W_n, W_{n-1}, \ldots, W_0$.

For a finite set D, the finite sequence obtained by *subtracting Dfrom each set* in $\mathcal{W}$ is denoted by $\mathcal{W}\backslash D$, that is, $\mathcal{W}\backslash D = W_0\backslash D, W_1\backslash D, \ldots, W_n\backslash D$.

For a finite sequence of sets $\mathcal{M} = M_0, M_1, \ldots, M_m$ where $m \leq n$, we say that $\mathcal{W}$ *contains* $\mathcal{M}$, denoted by $\mathcal{M} \subseteq \mathcal{W}$, if $\mathcal{M}$ is a consecutive subsequence of $\mathcal{W}$, or formally, there exists $0 \leq i \leq n - m$, such that for all $0 \leq j \leq m$ we have $W_{i+j} = M_j$.

For finite sequences of sets $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_n$, $n \geq 0$, the finite sequence $\mathcal{W}$ obtained by *concatenating* these sequences is denoted by $\mathcal{W} = \mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_n$.

The intuition behind the simulator reaction systems is rather simple. If we take some interactive process in the simulator, then state sets of this process can be divided into one or more shorter subsequences. The first subsequence always represents a forward computation of the simulated system. Then, the second subsequence corresponds to a backward computation of the simulated system, undoing some actions previously computed by the first subsequence. Afterwards, another forward subsequence follows which is in turn equivalent to a series of Redo and Do operations. Backward computation does not occur by accident but is controlled by the environment using a special auxiliary symbol ρ. When ρ is present in the context, the simulator undoes its last computation, simulating a backward step of the original simulated system. This intuition is formalized in the following two definitions.

Definition 5 Let $\mathscr{A}$ be a reaction system, $\pi = (\gamma, \delta)$ be an interactive process in $\mathscr{A}$, with $\gamma = C_0, C_1, \ldots, C_n$, $\delta = D_0, D_1, \ldots, D_n$, and let $\rho \in S$ be a special entity in the background set. We say that the interactive process π is a *well-formed simulating interactive process* if the following conditions hold for every $0 \leq i \leq n$:

- If $D_i = \emptyset$ or $D_i \subseteq \Sigma_c$, then $\rho \notin C_i$.
- If $\rho \in C_i$, then $C_i = \{\ \rho\ \}$.

The well-formedness of simulator interactive processes is a necessity, since ρ cannot appear in the context arbitrarily. If a result set is empty or consists solely of entities from the context alphabet of the simulated system, then we have reached an initial state. In this case, backward computation makes no sense, as initial states lack predecessors. Additionally, if ρ is present in the context, then regardless of any other input entity, a simulated backward step will take place. As only entities in the product alphabet of the simulated system influence the backward computation, any unnecessary context entity other than ρ is forbidden.

Definition 6 Let $\mathscr{A} = (S, A)$ and $\mathscr{B} = (S \cup \{\rho\}, B)$ be two reaction systems, with ρ being an auxiliary symbol in the background set of $\mathscr{B}$ (forcing the system to compute a simulated backward step).

The system $\mathscr{B}$ *interactively simulates* the interactive processes of $\mathscr{A}$ if the following conditions hold.

(1) For every interactive process π in $\mathscr{A}$, there is a well formed interactive process σ in $\mathscr{B}$ such that $\mathrm{sts}(\pi) = \mathrm{sts}(\sigma)$.
(2) For every *well-formed simulating interactive process* σ in $\mathscr{B}$, $\mathrm{sts}(\sigma)$ can be written as $\mathrm{sts}(\sigma) = \mathcal{V}_0, \dots, \mathcal{V}_k$, where each $\mathcal{V}_i$ is a finite sequence of sets such that:

- $\mathcal{V}_0 = \mathrm{sts}(\pi)$ for some interactive process π in $\mathscr{A}$.
- If $i = 2m$ for some $m \geq 1$, then there exists an interactive process π in $\mathscr{A}$ such that $\mathcal{V}_i \subseteq \mathrm{sts}(\pi)$.
- If $i = 2m + 1$ for some $m \geq 0$, then $\rho \in W$ for every W in $\mathcal{V}_i$ and there exists an interactive process π in $\mathscr{A}$, such that $\mathcal{V}_i \setminus \{ \rho \} \subseteq \mathrm{sts}(\pi)$.

According to the above definition, the simulating system can compute everything that the original simulated system is capable of computing (because of including all of its interactive processes), and, furthermore, the simulator can traverse back and forth among the states of the interactive processes of the simulated system. Backward computation is initiated by the auxiliary symbol ρ. This notion of back and forth traversal is captured by the appropriate subdivision of the states of the interactive processes.

Now, our goal is to show that the Undo-Redo-Do paradigm of reversibility can be achieved for reversible reaction systems using the above definition of interactive simulation. In what follows, we first show how to construct appropriate simulator systems and then prove that they adhere to the requirements of Definition 6.

Definition 7 Let $\mathscr{A} = (S, A)$ (with $S = \Sigma_p \cup \Sigma_c$) be a reversible reaction system. A reaction system $\mathscr{B}$, called the *interactive undo-redo simulator* of $\mathscr{A}$ is constructed as follows.

Let $\mathscr{B} = (S \cup \{\rho\}, B)$ where $B = \overrightarrow{B} \cup \overleftarrow{B}$ such that

$$\overrightarrow{B} = \{ (R_a, I_a \cup \{ \rho \}, P_a) \mid a \in A \},$$

$$\overleftarrow{B} = \{(P_E \cup \{\rho\},\ \mathrm{CONT}_A(S,E),\ R_E \cup \mathrm{DI}_A(S,E)) \mid E \in EN_A(S)\},$$

where $\mathrm{CONT}_A(S,E)$ is defined as

$$\mathrm{CONT}_A(S,E) = \bigcup_{\substack{F \in \mathrm{EN}_A(S) \\ P_E \subset P_F}} P_F \setminus P_E$$

and $\mathrm{DI}_A(S,E)$ is defined as

$$\mathrm{DI}_A(S,E) = \bigcup_{\substack{F \in \mathrm{EN}_A(S) \\ E \neq F,\ R_F = R_E}} I_F \setminus I_E.$$

The set $\overrightarrow{B}$ consists of reactions implementing forward computational steps of $\mathscr{A}$ (if ρ is not present in the context, the reactions of A can be performed also in the simulating system), while the reactions in $\overleftarrow{B}$ implement the simulated backward computational steps of $\mathscr{A}$.

The intuition behind the constructions of the backward reactions of $\overleftarrow{B}$ can be summarized as follows. For each simultaneously applicable set of reactions $E \in \mathrm{EN}_A(S)$, the presence of the product set P_E of E implies that a backward reaction produces the reactants of E, possibly together with the elements of $\mathrm{DI}_A(S,E)$ in addition. The set $\mathrm{DI}_A(S,E)$ contains those entities which might have also been necessary to make only the reactions of E simultaneously enabled by inhibiting other reactions which could have also been applied. This is how $\mathrm{DI}_A(S,E)$ is constructed: It contains the inhibitors of those simultaneously applicable sets of reactions which have the same set of reactants as E, since these entities must have been present in the predecessor state (otherwise not E, but some other set of reactions would have been applied). There is also a set of inhibitors added to the backward simulating reactions, the set $\mathrm{CONT}_A(S,E)$, which is necessary, because there might be different sets of simultaneously enabled reactions $E_1, E_2 \in \mathrm{EN}_A(S)$, such that $P_{E_1} \subset P_{E_2}$, that is, the product set of E_1 is a subset of the product set of E_2. In this case, performing the reactions of E_1 backwards should only be possible if the elements of $E_2 \setminus E_1$ are not present. (The presence of these entities would indicate that state for which the predecessor should be produced was not the result of applying the reactions of E_1, but the reactions of E_2 instead.)

Example 7 Consider the reversible reaction system $\mathscr{A} = (S, A)$ with $\Sigma_p = \{1, 3, 5, 7\}$ being the product alphabet, $\Sigma_c = \{0, 2, 4\}$ being the context alphabet, and $A = \{a, b, c, d\}$ being the set of reactions

$$a = (\{0\},\ \{1,2,3,4,5\},\ \{1\}), \quad c = (\{1,4\},\ \{0,3,5\},\ \{5\}),$$
$$b = (\{1,2\},\ \{0,5\},\ \{3\}), \quad d = (\{1,2\},\ \{0,3\},\ \{3,7\}).$$

Based on Definition 7, the interactive undo-redo simulator$\mathscr{B} = (S \cup \{\rho\}, B)$ interactively simulating $\mathscr{A}$ is constructed as follows.

Let the $\{0,1,2,3,4,5,7\} \cup \{\rho\}$ be the background set and $B = \underline{B} \cup \underline{B}$ be the set of reactions where forward reactions are defined as

$$\overset{\rightharpoonup}{B} = \{\ (\{0\},\ \{1,2,3,4,5,\rho\},\ \{1\}),\ (\{1,2\},\ \{0,5,\rho\},\ \{3\}),$$
$$(\{1,4\},\ \{0,3,5,\rho\},\ \{5\}),\ (\{1,2\},\ \{0,3,\rho\},\ \{3,7\})\ \}.$$

To construct the backward reactions, consider

$$\mathrm{EN}_A(S) = \{\ \{a\}, \{b\}, \{c\}, \{b,d\}, \{b,c,d\}\ \}.$$

Then we compute

$$\mathrm{CONT}_A(S,\{a\}) = \emptyset, \qquad \mathrm{CONT}_A(S,\{b,d\}) = \{5\},$$
$$\mathrm{CONT}_A(S,\{b\}) = \{5,7\}, \quad \mathrm{CONT}_A(S,\{b,c,d\}) = \emptyset,$$
$$\mathrm{CONT}_A(S,\{c\}) = \{3,7\},$$

and

$$\mathrm{DI}_A(S,\{a\}) = \emptyset, \qquad \mathrm{DI}_A(S,\{b,d\}) = \{0,3,5\},$$
$$\mathrm{DI}_A(S,\{b\}) = \{3\}, \quad \mathrm{DI}_A(S,\{b,c,d\}) = \{0,3,5\}.$$
$$\mathrm{DI}_A(S,\{c\}) = \emptyset,$$

Based on these, the set of backward reactions is

$$\overset{\leftharpoonup}{B} = \{\ (\{1,\rho\},\ \emptyset,\ \{0\}), (\{3,\rho\},\ \{5,7\},\ \{1,2,3\}),$$
$$(\{5,\rho\},\ \{3,7\},\ \{1,4\}), (\{3,7,\rho\},\ \{5\},\ \{1,2,5\})\ \}.$$

Just as in the case of reversible reaction systems, we are going to take a step-by-step approach to prove that interactive undo-redo simulatorsystems constructed using Definition 7 are indeed interactive simulators in the sense of Definition 6. Taking an arbitrary well-formed simulating interactive process σ in a simulator system, we subdivide the state sequence of σ into smaller subsequences and for each subsequence of interest, we prove that it satisfies the appropriate condition of Definition 6.

Lemma 4 *Let $\mathscr{A}$ be a reversible reaction system, let $\mathscr{B}$ be the interactive undo-redo simulatorof $\mathscr{A}$, and and let σ be a well-formed simulating interactive process in $\mathscr{B}$.*

Then, the state sequence of σ can be written as $\mathrm{sts}(\sigma) = \mathcal{V}_0\mathcal{V}_1$, *where $\mathcal{V}_0$ is a finite sequence, such that* $\mathcal{V}_0 \backslash \{\rho\} = \mathrm{sts}(\pi)$ *for some interactive process π in $\mathscr{A}$, that is, $\mathcal{V}_0 \backslash \{\rho\}$ is the state sequence of a forward computation of the simulated system $\mathscr{A}$.*

Proof Let $\mathscr{A} = (S, A)$ $(S = \Sigma_p \cup \Sigma_c)$ be a reversible reaction system, let $\mathscr{B}$ be the interactive undo-redo simulatorconstructed from $\mathscr{A}$ using Definition 7, and let σ be a well-formed simulating interactive process in $\mathscr{B}$ with $\mathrm{sts}(\sigma) = W_0, W_1, \ldots, W_n$.

Since σ is a well-formed simulating interactive process, there exists $0 \le m_0 \le n$ such that $\rho \notin C_i$, $0 \le i \le m_0$. This means that σ starts with a finite sequence of states such that the length of this sequence is greater than or equal to one, and none of the sets in the sequence includes ρ.

Now we prove by induction on the value of m_0 that, given the previous condition, there exists an interactive process π in $\mathscr{A}$ such that $\mathrm{sts}(\pi) = W_0, W_1, \ldots, W_{m_0}$ when $m_0 = n$, or $\rho \in W_{m_0+1}$ and $\mathrm{sts}(\pi) = W_0, W_1, \ldots, W_{m_0}, W_{m_0+1} \backslash \{\ \rho\ \}$ when $m_0 < n$. Thus, ρ might be present in the $(m_0 + 1)$st context, nevertheless, with ρ not taken into consideration, the state can still be part of some interactive process in $\mathscr{A}$.

Given the well-formedness of σ, we can be sure that $\rho \notin C_0$ i.e. $C_0 \subseteq \Sigma_c$. Hence, for the initial state $W_0 = C_0$ of σ there exists some interactive process π in $\mathscr{A}$ sharing the very same initial state. Thus, for $m_0 = 0$, the previous statement is true.

Assuming that the statement holds for an arbitrary $m_0 = k$, we now show that it also holds for $m_0 = k + 1$. In this case, we know that there exists some interactive process π in $\mathscr{A}$ such that $\mathrm{sts}(\pi) = W_0, W_1, \ldots, W_k$. This implies that $W_k \subseteq S$. Let us now enumerate the reactions applicable to this set. As $\rho \notin W_k$, one can only apply reactions from $\underline{B}$, since ρ takes the role of a reactant in every reaction of B. Note, that B contains the very same reactions as A with a small difference: the inhibitor sets were augmented with ρ. In our current case however, this role of ρ is irrelevant, since $\rho \notin W_k$. Consequently, we have that $\mathrm{res}_A(W_k) = \mathrm{res}_B(W_k) = D_{k+1}$. Thus, there exists some interactive process $\pi = (\gamma_\pi, \delta_\pi)$ in $\mathscr{A}$ such that $\delta_\pi = D_1, D_2, \ldots D_k, D_{k+1}$.

Let us now consider the context. As $\rho \notin C_{k+1}$ we know that $C_{k+1} \subseteq S$. Since $D_{k+1} \subseteq S$ and $C_{k+1} \subseteq S$ we have that $W_{k+1} \subseteq S$ and, in turn, there exists some interactive process π in $\mathscr{A}$ such that $\mathrm{sts}(\pi) = W_0, W_1, \ldots, W_{k+1}$.

We now continue by considering $m_0 = k + 1 < n$, meaning that the next state starts a new, simulated backward computation. In this case $C_{m_0+1} = \{\ \rho\ \}$. On the other hand, since we proved, that there exists some interactive process in $\mathscr{A}$ such that the state sequence of the process is equal to the sequence $W_0, W_1, \ldots, W_{m_0}$, then it also holds, that extending this sequence with $W_{m_0+1} = \mathrm{res}_A(W_{m_0}) \cup \{\ \rho\ \}$, there exists some interactive process in $\mathscr{A}$ such that the state sequence of the process is equal to $W_0, W_1, \ldots, W_{m_0}, W_{m_0+1} \backslash \{\ \rho\ \}$.

This means, that the statement holds for $m_0 = k + 1$ which, in turn, renders our initial statement true. The argument above also implies that every interactive process in $\mathscr{A}$ is an interactive process in $\mathscr{B}$ as well, since both the C_i

context sets and the length k of the initial forward computing sequence can be chosen arbitrarily. □

Now we continue by showing that after the initial state sequence simulating a forward computation, the well-formed simulating interactive processes might continue with the simulation of backward computations.

Lemma 5 *Let $\mathscr{A}$ be a reversible reaction system, let $\mathscr{B}$ be the interactive undo-redo simulatorof $\mathscr{A}$, and let σ be a well-formed simulating interactive process in $\mathscr{B}$.*

Then, the state sequence of σ can be written as $\mathrm{sts}(\sigma) = \mathcal{V}_0\mathcal{V}_1$, where $\mathcal{V}_0\backslash\{\rho\}$ is the state sequence of a forward computation of the simulated system $\mathscr{A}$, and $\mathcal{V}_1$ can be written as $\mathcal{V}_1 = \mathcal{V}_2\mathcal{V}_3$ where $\mathcal{V}_2$ is a finite sequence such that there exists some interactive process π in $\mathscr{A}$ for which $\overleftarrow{\mathcal{V}_2}\backslash\{\rho\} \subseteq \mathrm{sts}(\pi)$. Thus, $\mathcal{V}_2\backslash\{\rho\}$ is a state sequence of a backward computation of the simulated system $\mathscr{A}$.

Proof Let $\mathscr{A} = (S, A)$ $(S = \Sigma_p \cup \Sigma_c)$ be a reversible reaction system, let $\mathscr{B}$ be the interactive undo-redo simulatorconstructed from $\mathscr{A}$ using Definition 7, and let σ be a well-formed simulating interactive process in $\mathscr{B}$ with $\mathrm{sts}(\sigma) = W_0, W_1, \dots, W_n = \mathcal{V}_0\mathcal{V}_1$, where $\mathcal{V}_0\backslash\{\rho\}$ is the state sequence of a forward computation of the simulated system $\mathscr{A}$. Since $\mathcal{V}_0$ is covered by Lemma 4, we know that there exists $m_0 < n$ such that the finite sequence $W_0, W_1, \dots W_{m_0}$ is the state sequence of some interactive process π in $\mathscr{A}$. In this case, for some $m_0 < m_1 \leq n$, the well-formedness of σ implies that $\rho \in C_i$ for $m_0 + 1 \leq i \leq m_1$, and if $m_1 < n$, then $\rho \notin C_{m_1+1}$.

Now we prove by induction on the value of m_1 that there exists some interactive process π in $\mathscr{A}$ for which the $\mathcal{V}_1 = W_{m_0+1}, W_{m_0+2}, \dots, W_{m_1}$ sequence is such that $\overleftarrow{\mathcal{V}_1}\backslash\{\rho\} \subseteq \mathrm{sts}(\pi)$ holds.

First, let us prove the statement for $m_1 = m_0 + 1$, when the sequence consists of a single element. In this case, $\rho \in C_{m_1}$ and, implied by the proof of Lemma 4, there exists some interactive process π in $\mathscr{A}$ such that $\mathrm{sts}(\pi) = W_0, W_1, \dots W_{m_0+1}\backslash\{\rho\}$. Accordingly, if $\mathcal{V}_1 = W_{m_1}$, then $\mathcal{V}_1\backslash\{\rho\} \subseteq \mathrm{sts}(\pi)$. Thus, for $m_1 = m_0 + 1$, the statement holds.

Let us now assume, that the statement holds if $m_1 = m_0 + k$, and then prove that it also holds for $m_1 = m_0 + k + 1$. We know that the state sequence $\mathcal{V}_2 = W_{m_0+1}, W_{m_0+2}, \dots W_{m_0+k}$ is such that $\overleftarrow{\mathcal{V}_2}\backslash\{\rho\} \subseteq \mathrm{sts}(\pi)$ for some interactive process π in $\mathscr{A}$, furthermore, $C_{m_0+k} = \{\rho\}$ and $C_{m_1} = \{\rho\}$.

Building on the previous facts, let us continue by defining W_{m_1}. Since $\rho \in W_{m_0+k}$, only reactions in B can be applied to W_{m_0+k} (as the presence of ρ forbids the application of reactions in B). $\mathscr{A}$ is reversible, therefore, no two reaction sets $E_1, E_2 \in \mathrm{EN}_A(S)$ produces the same result set. Thus, given an arbitrary result set $D = P_E$ for some $E \in \mathrm{EN}_A(S)$, we are able to restore W such that $\mathrm{res}_{\mathscr{A}}(W) = D$. In order to do so, we take every reaction in E and create a new reaction with reactant set equal to $D = P_E$ and product set equal to $W = R_E$. However, care should be taken, as there might be one ore more $F \in \mathrm{EN}_A(S)$ reaction set such that $P_E \subset P_F$. If so, then elements not in P_E must be forbidden by including every element in $P_F\backslash P_E$ in the inhibitor set of the newly created reaction. This is exactly, how $\mathrm{CONT}_A(E)$ is defined. That way, we will not end up falsely applying reactions because of the subset relationships. Our job is not done yet, however, since we must also consider inhibitors. As demonstrated in Example 4, inhibitors might also leave a trace by inhibiting some reactions and thus affecting the result. Hence, given the reactant set R_E, we need to find reaction sets with the same reactants. Then, the inhibitors of the reactions in these sets should also be restored, since their presence denied the application of these reactions. Such inhibitor entities are captured by $\mathrm{DI}_A(S, E)$ in Definition 7, and as a result, we will have a product set $R_E \cup \mathrm{DI}_A(S, E)$.

Turning back to the definition of $\mathscr{B}$, we can see, that reactions in B are constructed using this very method. What remains is, to find the reaction in B that can be applied to W_{m_0+k}. Such a reaction always exists and is always unique: unique, since $\mathscr{A}$ is reversible, which means that there is only a single way to produce every result set, and exists, since $\overleftarrow{\mathcal{V}_2}\backslash\{\rho\} \subseteq \mathrm{sts}(\pi)$ for some interactive process π in $\mathscr{A}$, which means that the first set in $\overleftarrow{\mathcal{V}_2}\backslash\{\rho\}$, that is, $W_{m_0+k}\backslash\{\rho\}$ was surely produced from some state by applying some set of reactions in A. The only case in which the previous would fail is if $W_{m_0+k}\backslash\{\rho\}$ was the initial state, however, this case is forbidden by the well-formedness of σ.

Concluding the previous reasoning, we have that the only reaction applicable to W_{m_0+k} is a uniquely determined reaction $b \in B$ for which $R_b = W_{m_0+k}\backslash\{\rho\}$ holds. By applying this reaction to $R_b = W_{m_0+k}\backslash\{\rho\}$ we obtain $P_b = D_{m_1}$ such that $\mathrm{res}_A(D_{m_1}) = W_{m_0+k}\backslash\{\rho\}$. Since $C_{m_1} = \{\rho\}$, we have that $W_{m_1} = P_b \cup \{\rho\}$. As a consequence, these states can be written as a sequence $\mathcal{V}_2 = W_{m_0+1}, W_{m_0+2}, \dots, W_{m_0+k}, W_{m_1}$ such that $\overleftarrow{V_2}\backslash\{\rho\} \subseteq \mathrm{sts}(\pi)$ for some interactive process π in $\mathscr{A}$ (as $\mathcal{V}_2$ is essentially an appropriate continuation of $\mathcal{V}_1$ for which we already proved the same). This means, that the statement holds for $m_1 = m_0 + k + 1$ which, in turn, renders our initial statement true. □

Now we combine the previous two statements to obtain the following.

Lemma 6 *Let $\mathscr{A}$ be a reversible reaction system, let $\mathscr{B}$ be the interactive undo-redo simulatorof $\mathscr{A}$, and let σ be a well-formed simulating interactive process in $\mathscr{B}$, such that*

- $\mathrm{sts}(\sigma) = \mathcal{V}_0\mathcal{V}_1$, *where $\mathcal{V}_0\backslash\{\rho\}$ is the state sequence of a forward computation of $\mathscr{A}$, and*
- *$\mathcal{V}_1$ can be written as $\mathcal{V}_1 = \mathcal{V}_2\mathcal{V}_3$, where $\mathcal{V}_2\backslash\{\rho\}$ is a state sequence of a backward computation of $\mathscr{A}$.*

Then, if $\mathcal{V}_3$ is not of zero-length, then subdividing $\mathcal{V}_3$ into smaller subsequences will result in forward and backward computations of the simulated system $\mathscr{A}$, analogous to those covered by Lemmas 4, 5.

Proof Let $\mathscr{A} = (S, A)$ ($S = \Sigma_p \cup \Sigma_c$) be a reversible reaction system, let $\mathscr{B}$ be the interactive undo-redo simulatorconstructed from $\mathscr{A}$ using Definition 7, and let σ be a well-formed simulating interactive process in $\mathscr{B}$ with $\mathrm{sts}(\sigma) = W_0, W_1, \dots, W_n = \mathcal{V}_0\mathcal{V}_1$, where $\mathcal{V}_0\backslash\{\rho\}$ is the state sequence of a forward computation of $\mathscr{A}$, and $\mathcal{V}_1 = \mathcal{V}_2\mathcal{V}_3$, where $\mathcal{V}_2\backslash\{\rho\}$ is a state sequence of a backward computation of $\mathscr{A}$.

Since $\mathcal{V}_2$ is covered by Lemma 5, we know that there exists $m_0 < m_1 < n$ such that the finite sequence $W_{m_1}\backslash\{\rho\}, W_{m_1-1}\backslash\{\rho\}, \dots W_{m_0+1}\backslash\{\rho\}$ is the state sequence of some interactive process π in $\mathscr{A}$. We now show that the proof for subsequent sequences in σ is analogous to the those of Lemmas 4, 5.

Now, because of the well-formedness of σ, we have that $C_{m_1} = \{\rho\}$ and $\rho \notin C_{m_1+1}$. As a consequence of the former, we have that D_{m_1+1} is the result of a simulated backward step, which means that $\mathrm{res}_{\mathscr{A}}(D_{m_1+1}) = W_{m_1}\backslash\{\rho\}$. Also, $W_{m_1+1} = D_{m_1+1} \cup C_{m_1+1}$. As for the sequence $\mathcal{V}_4 = W_{m_0+1}, W_{m_0+2}, \dots W_{m_1}, D_{m_1+1}$, there exists some interactive process π in $\mathscr{A}$ such that $\mathcal{V}_4\backslash\{\rho\} \subseteq \mathrm{sts}(\pi)$, we have two possibilities regarding D_{m_1+1}:

- $D_{m_1+1} \subseteq \Sigma_c$, thus D_{m_1+1} is equal to a restarting or initial state of π, or
- there exists some state W in π such that $D_{m_1+1}\backslash\Sigma_c = \mathrm{res}_A(W)$.

Regardless, the sequence $\mathcal{V}_5 = D_{m_1+1}$ is contained within the sequence of states of π. Since D_{m_1+1} can only be extended with elements from Σ_c to form W_{m_1+1} then there is going to be an interactive process τ in $\mathscr{A}$ such that the sequence of states of τ contains W_{m_1+1}. If we now apply the induction for forward computations (from Lemma 4), then we can say that given the subsequent contexts do not contain ρ, then the newly started sequence of states is going to adhere Definition 6 (i.e. there exists some interactive process in $\mathscr{A}$ with an appropriate sequence of states containing this sequence).

Now, if some C_j context set contains ρ, then we can apply the same induction as in Lemma 5, that is, starting from a set which is included within the state sequence of some interactive process in $\mathscr{A}$, we can see that only valid backward steps can be simulated.

Therefore, we have shown, that any further subsequence of σ is either a proper forward or backward computation of the simulated system $\mathscr{A}$. □

Combining the three previous lemmas, we can state the following.

Theorem 2 *For every reversible reaction system $\mathscr{A}$, the interactive undo-redo simulatorconstructed from $\mathscr{A}$ using Definition 7 interactively simulates the interactive processes of $\mathscr{A}$.*

Proof Let $\mathscr{A} = (S, A)$ ($S = \Sigma_p \cup \Sigma_c$) be a reversible reaction system, let $\mathscr{B}$ be the interactive undo-redo simulatorconstructed from $\mathscr{A}$ using Definition 7, and let σ be a well-formed simulating interactive process in $\mathscr{B}$

Given the previous assumptions, Lemmas 4, 5 and 6 can be applied. In these lemmas, by subdividing the state sequence of σ, we showed that σ satisfies the requirements of condition (2) in Definition 6. Since our choice of σ (apart from the well-formedness) is arbitrary, this also means that every well-formed interactive process in $\mathscr{B}$ satisfies the requirements of condition (2). When considering the first subsequence of σ in Lemma 4, we also showed, that well-formed interactive processes in $\mathscr{B}$ may start with any interactive process in $\mathscr{A}$. Thus, for every interactive process in $\mathscr{A}$ we have an interactive process in $\mathscr{B}$ with the very same state sequence. Consequently $\mathscr{B}$ satisfies condition (1) in Definition 6.

This implies that $\mathscr{B}$ interactively simulates the interactive processes of $\mathscr{A}$. □

Based on the above proof, we can construct an interactive simulator system (the interactive undo-redo simulator) for every reversible reaction system. The method of the construction is rather straightforward: First, we include a new symbol ρ in the original background set and we create forward reactions by including ρ in every inhibitor set. Then we assemble the backward reactions by enumerating the reaction sets which can ever become enabled (the elements of $\mathrm{EN}_A(S)$) and follow the construction of Definition 7.

The simulators obtained that way may freely perform the forward and the backward computations of the original system, allowing for an Undo-Redo-Do-like semantics of reversibility where the environment has to control over the direction of the computation.

6 Conclusion

In this paper, we investigated the reversibility of reaction systems. This area was studied before by Aman and Ciobanu [5], however, in their work, they did not consider open interactive processes (processes with non-empty context sets) and they extended the model with features implementing a memory to remember vanished entities. We took a different approach by working with the model as-is (thus, adding no extensions) and allowing the use of non-empty context sets. We first identified the requirements which are necessary and sufficient for a reaction system to be reversible, then by showing how to build reverse reactions, we constructed simulator reaction systems which are able to simulate reversible reaction systems with their (forwards-only) computations in both directions, backwards and forwards.

As reversible reaction systems according to our notion of reversibility seem to be rather restricted, to study and determine the class of the possible computations which they can perform would definitely be of interest for further research. Since we expect that the class of possible computations are similarly restricted, some kind of relaxation of the unique-predecessor concept to arrive to a different, less restrictive notion of reversibility would also be of interest. One approach that we would like to explore in the future is a kind of "lookback" reversibility which is similar to the usual notion of backward determinism in automata. An automaton is backward deterministic (and thus, reversible) if the current state and the previously consumed input symbol together uniquely determine the previous state of the machine. A similar notion could be defined for interactive processes by noticing that the context sets play a similar role as the input symbols of automata, they are consumed by the actual step of the computation. According to the proposed reversibility concept, an interactive process would be reversible if the current state and the context set used in the previous step (to arrive to the current state) together uniquely determine the previous state of the process.

Acknowledgements This research was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project is supported by the European Union, co-financed by the European Social Fund. Gy. Vaszil was also supported by Project No. K 120558 of the National Research, Development and Innovation Fund of Hungary, financed under the K 16 funding scheme. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions on a previous version of the paper.

Funding Open access funding provided by University of Debrecen.

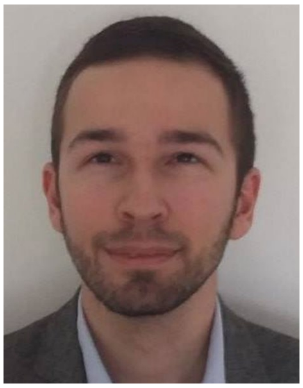
Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Agrigoroaiei, O., & Ciobanu, G. (2008) Dual P systems. In: D.W. Corne, P. Frisco, G. Paun, G. Rozenberg, A. Salomaa (eds.) Membrane Computing - 9th International Workshop, WMC 2008, Edinburgh, UK, July 28–31, 2008, Revised Selected and Invited Papers, *Lecture Notes in Computer Science*, vol. 5391, pp. 95–107. Springer. https://doi.org/10.1007/978-3-540-95885-7_7.
2. Agrigoroaiei, O., & Ciobanu, G. (2010). Reversing computation in membrane systems. *The Journal of Logical and Algebraic Methods in Programming*, *79*(3–5), 278–288. https://doi.org/10.1016/j.jlap.2010.03.003.
3. Alhazov, A., Freund, R., & Morita, K. (2012). Sequential and maximally parallel multiset rewriting: Reversibility and determinism. *Search Results*, *11*(1), 95–106. https://doi.org/10.1007/s11047-011-9267-8.
4. Aman, B., & Ciobanu, G. (2017). Reversibility in parallel rewriting systems. *Journal of Universal Computer Science*, *23*(7), 692–703.
5. Aman, B., & Ciobanu, G. (2018). Controlled reversibility in reaction systems. In M. Gheorghe, G. Rozenberg, A. Salomaa, & C. Zandron (Eds.), *Membrane computing* (pp. 40–53). Cham: Springer International Publishing.
6. Aman, B., Ciobanu, G., Glück, R., Kaarsgaard, R., Kari, J., Kutrib, M., et al. (2020). *Foundations of reversible computation* (pp. 1–40)., Lecture notes in computer science Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-47361-7_1.
7. Axelsen, H. B., & Glück, R. (2016). On reversible turing machines and their function universality. *Acta Informatica*, *53*(5), 509–543. https://doi.org/10.1007/s00236-015-0253-y.
8. Bennett, C. H. (1973). Logical reversibility of computation. *IBM Journal of Research and Development*, *17*(6), 525–532.
9. Brijder, R., Ehrenfeucht, A., Main, M., & Rozenberg, G. (2011). A tour of reaction systems. *International Journal of Foundations of Computer Science*, *22*, 1499–1517. https://doi.org/10.1142/S0129054111008842.
10. Danos, V., & Krivine, J. (2004) Reversible Communicating Systems. In: P. Gardner, N. Yoshida (eds.) CONCUR 2004 - Concurrency Theory, Lecture Notes in Computer Science, pp. 292–307. Springer, Berlin, Heidelberg . https://doi.org/10.1007/978-3-540-28644-8_19.

11. Danos, V., & Krivine, J. (2005) Transactions in RCCS. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings, *Lecture Notes in Computer Science*, vol. 3653, pp. 398–412. Springer . https://doi.org/10.1007/11539452_31.
12. Ehrenfeucht, A., & Rozenberg, G. (2007). Reaction systems. *Fundamenta Informaticae*, *75*(1–4), 263–280.
13. Frank, M.P. (2005) Introduction to reversible computing: Motivation, progress, and challenges. In: Proceedings of the 2nd Conference on Computing Frontiers, CF ’05, p. 385–390. Association for Computing Machinery, New York, NY, USA . https://doi.org/10.1145/1062261.1062324.
14. Holzer, M., & Kutrib, M. (2017) Reversible nondeterministic finite automata. In: I. Phillips, H. Rahaman (eds.) Reversible Computation, pp. 35–51. Springer International Publishing, Cham . https://doi.org/10.1007/978-3-319-59936-6_3.
15. Ibarra, O. H. (2011). On strong reversibility in P systems and related problems. *International Journal of Foundations of Computer Science*, *22*(1), 7–14. https://doi.org/10.1142/S0129054111007782.
16. Kari, L., & Rozenberg, G. (2008). The many facets of natural computing. *Communications of the ACM*, *51*, 72–83. https://doi.org/10.1145/1400181.1400200.
17. Landauer, R. (1961). Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, *5*(3), 183–191.
18. Lanese, I., Mezzina, C.A., & Stefani, J.B. (2013) Controlled Reversibility and Compensations. In: R. Glück, T. Yokoyama (eds.) Reversible Computation, Lecture Notes in Computer Science, pp. 233–240. Springer, Berlin, Heidelberg . https://doi.org/10.1007/978-3-642-36315-3_19.
19. Lanese, I., Mezzina, C. A., & Tiezzi, F. (2014). Causal-consistent reversibility. *Bulletin-European Association for Theoretical Computer Science*, *114*, p. 17.
20. Lanese, I., & Rawski, M. (eds.) (2020) Reversible Computation: 12th International Conference, RC 2020, Oslo, Norway, July 9-10, 2020, Proceedings. Programming and Software Engineering. Springer International Publishing . https://doi.org/10.1007/978-3-030-52482-1.
21. Meski, A., Penczek, W., & Rozenberg, G. (2015). Model checking temporal properties of reaction systems. *The Journal of Information Science*, *313*, 22–42. https://doi.org/10.1016/j.ins.2015.03.048.
22. Morita, K. (2017). *Theory of Reversible Computing*. Springer, Japan,. https://doi.org/10.1007/978-4-431-56606-9.
23. Nishida, T.Y. (2009). Reversible p systems with symport/antiport rules. In: G. Paun, M. Pérez-Jiménez, A. Riscos-Núñez (eds.) Proceedings of the 10th Workshop on Membrane Computing, WMC 10, pp. 452–460.
24. Paun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences*, *61*(1), 108–143. https://doi.org/10.1006/jcss.1999.1693.
25. Paun, G., Rozenberg, G., & Salomaa, A. (Eds.). (2010). *The Oxford handbook of membrane computing*. Oxford: Oxford University Press Inc.
26. Perumalla, K.S. (2013). Introduction to Reversible Computing. Chapman & Hall/CRC. Computational Science Series. Boca Raton: CRC Press.
27. Phillips, I., Ulidowski, I., & Yuen, S.(2013). A Reversible Process Calculus and the Modelling of the ERK Signalling Pathway. In: R. Glück, T. Yokoyama (eds.) Reversible Computation, Lecture Notes in Computer Science, pp. 218–232. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-36315-3_18.
28. Pinna, M.G. (2018). Reversing steps in membrane systems computations. In: M. Gheorghe, G. Rozenberg, A. Salomaa, C. Zandron (eds.) Membrane Computing, Lecture Notes in Computer Science, pp. 245–261. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-73359-3_16.
29. Rozenberg, G., Bäck, T., & Kok, J. N. (2012). *Handbook of natural computing*. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-92910-9.
30. Ulidowski, I., Lanese, I., Schultz, U.P., & Ferreira, C. (eds.): (2020). Reversible Computation: Extending Horizons of Computing - Selected Results of the COST Action IC1405, *Lecture Notes in Computer Science*, vol. 12070. Springer. https://doi.org/10.1007/978-3-030-47361-7.



Attila Bagossy graduated in computer science from the Faculty of Informatics of the University of Debrecen in 2019. He started his PhD in the same year at the Doctoral School of Informatics. His current research interests include unconventional models of computation, reversible computation, and WebAssembly.

György Vaszil obtained his PhD in 2001 at the Eötvös Loránd University of Budapest. Since 2015, he is full professor at the Faculty of Informatics of the University of Debrecen where he is the head of the Department of Computer Science. His research interests include the theory of formal languages and automata, unconventional or nature motivated computational models, such as bio-inspired models like membrane systems. He has published more than 140 papers in international journals, conferences, or workshops.