Transition Graphs of Reversible Reaction Systems[⋆]

Attila Bagossy and György Vaszil

Department of Computer Science, Faculty of Informatics
University of Debrecen, Kassai út 26, 4028 Debrecen, Hungary
bagossy.attila@inf.unideb.hu
vaszil.gyorgy@inf.unideb.hu

Abstract. We study the transition graphs, and thus, the possible computational paths of reaction systems which are reversible according to different notions of reversibility. We show that systems which are reversible in the sense of our earlier work produce very simple types of transition graphs. A somewhat more complicated, but still quite simple class of transition graphs is obtained if we consider so called initialized reversible systems. Finally we introduce the notion of reversibility with lookbehind, and show that systems which are reversible in this sense produce the same transition graphs (and thus, the same computations) as the state transition diagrams of reversible finite transition systems.

1 Introduction

Reaction systems, introduced by Ehrenfeucht and Rozenberg in [6], aim to capture biochemical processes occurring inside living cells. The main intuition behind this model of computation is the interplay between facilitation and inhibition as these mechanisms define which reactions can take place, and thus how computations proceed. A reaction system is a set of reactions, each reaction is represented by a triple of finite sets: the reactants, the inhibitors, and the results. In each step, the system produces resulting elements according to the set of reactants and the set of reactions that are not inhibited. This core idea is further complemented by the model's two distinctive characteristics. In contrast to multiset-based frameworks, reaction systems present a qualitative approach in which if an element (or reactant) is present, then it is assumed to be available in the necessary amount. As a consequence of this principle, reactions may freely use the same resource and will not interfere with each other. The second characteristic is the concept of no permanency which means that if there is no reaction sustaining a particular element, then the element will vanish. Building on these principles, reaction systems perform computation in so-called interactive processes that combine the result of reactions with input from the enclosing environment.

Since its inception, the model of reaction systems received vast research interest thanks to its unique properties and easy-to-extend nature. Research topics, for example, include the study of the state transition function defined by a particular system (see [5] among others), the introduction of time (see [11]), or modules (see [8]). For a more comprehensive enumeration, the reader is referred to [4, 7]

[⋆] The work of Gy. Vaszil was supported by the National Research, Development and Innovation Fund of Hungary through project no. K 120558, financed under the K 16 funding scheme.

In this paper we are going to study the transition graphs of reversible reaction systems. Transition graphs were introduced in [9] to represent the global dynamics of these systems. Such a graph is a directed graph where each vertex is a state of the system (represented as a set of elements present in the system at a given step of the computation) and directed edges from a vertex lead to the vertices representing the new states of the system which can be reached after all reactions enabled at the origin, with possible additions from the outside environment, are performed. The notion of reversibility was also studied in this framework. Reversible processes which do not depend on input from the external environment were considered in [1], and a somewhat more general notion of reversibility when certain kinds of inputs are allowed was proposed in [3].

Starting with our previously established definitions regarding reversible reaction systems in [3], we are going to study and compare the transition (or behavior) graphs corresponding to different reversible system definitions. Instead of examining the properties of individual interactive processes (that can be thought of as computational paths), we take a graph that describes every possible process in a given system and study how varying the underlying definitions (especially the different possible notions of reversibility) affects the graphs. By exploring these graphs, we infer the computational properties of the various definitions of reversible systems. We will consider the following variations.

1. Reversible systems as defined in our previous work [3]. We will consider two variants: First, we will use our notion of reversibility combined with the original definitions given in [6] where the initial state of the system is defined by the first environmental input, then we will use a slight modification which allows for arbitrary elements in the initial states of interactive processes.
2. Systems which are reversible with lookbehind. This is a modified notion of reversibility allowing interactive processes to examine not only the current result set but the previous environmental input as well.

The rest of the paper is organized as follows. In Section 2 we provide a brief introduction to the fundamental notions of reaction systems and the notion of reversibility as introduced in [3]. Then in Section 3 and Section 4 we present the transition graphs for the above mentioned systems as well as the comparison between them. Finally, Section 5 closes the paper with some conclusions.

2 Preliminaries

In this section, we first briefly introduce the essential concepts of reaction systems. We refer the reader to [4, 6] for a more comprehensive description. Concerning reversibility, we only cover the most important definitions and requirements, see [3] for detailed results and proofs. For more information on transition graphs, refer to [11].

Reaction systems model biochemical reactions by the interplay of facilitation and inhibition. Reactions are defined over a finite set of entities (usually denoted by S) and every *reaction* a is a triplet of three finite sets $a = (R_a, I_a, P_a)$ (where each of these sets are subsets of S). The sets R_a, I_a and P_a contain the *reactants*, *inhibitors* and *products* of the reaction, respectively, and they satisfy the following constraints: First, the set of reactants and the set of inhibitors are disjoint ($R_a \cap I_a = \emptyset$), otherwise the reaction would never be applicable, as we will see later. Second, the set of reactants and the set of products are non-empty ($R_a \neq \emptyset$ and $P_a \neq \emptyset$). It is usually also assumed that the set of inhibitors is nonempty, but for the sake of being as general as possible, we will drop this additional assumption here. The set of all reactions over S is denoted by $\mathrm{rac}(S)$.

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 $a = (R_a, I_a, P_a)$.

Based on the core idea of the model, a reaction is applicable (or enabled) if all of its reactants and none of its inhibitors are present. Applying a reaction creates its products. These intuitions are formalized as follows.

Given a set of arbitrary symbols (or entities) S and a reaction $a \in \mathrm{rac}(S)$, a is *enabled by* $W \subseteq S$ if $R_a \subseteq W$ and $I_a \cap W = \emptyset$. The *result of* a *on* W, denoted by $\mathrm{res}_a(W)$, is defined as $\mathrm{res}_a(W) = P_a$ if a is enabled by W, or $\mathrm{res}_a(W) = \emptyset$ if a is not enabled by W.

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

With these definitions in mind, we can now see how the characteristics mentioned in the Introduction are implemented. Reactions with overlapping reactant sets do not interfere as each one is allowed to create its products if enabled. This non-interefering nature also applies to the products. Even if the same entity is produced by multiple reactions, still there will be a single occurrence in the result set as reaction systems are defined over sets instead of multisets.

Prior to defining reaction systems and how they perform computation, we introduce further shorthand notations to ease our work with reactants and products of finite sets of reactions.

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: $R_A = \cup_{a \in A} R_a$ and $P_A = \cup_{a \in A} P_a$.

If S is a finite set such that $A \subseteq \mathrm{rac}(S)$, then $\mathrm{EN}_A(S)$ contains the sets of reactions where the members of each set can be applied together for some subset of S. Formally

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

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 are enabled together (as $R_a \cap I_c \neq \emptyset$), that is, $\mathrm{en}_A(\{1\}) = \mathrm{en}_A(\{1, 3\}) = \{a\}$, $\mathrm{en}_A(\{1, 2\}) = \mathrm{en}_A(\{1, 2, 3\}) = \{a, b\}$, and we also have $\mathrm{en}_A(\{2\}) = \{b\}$, $\mathrm{en}_A(\{3\}) = \{c\}$, $\mathrm{en}_A(\{2, 3\}) = \{b, c\}$ in addition.

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

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

With the essential notions defined for reactions, we now recall the definition of a *reaction system*, which is an ordered pair $\mathscr{A} = (S, A)$. The background set S is a finite set of entities while $A \subseteq \mathrm{rac}(S)$ is the set of reactions.

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

- γ is the *context sequence* of π, defined as $\gamma = C_0, C_1, \ldots C_n$, where $C_i \subseteq S$ for all $0 \leq i \leq n$, and
- δ is the *result sequence* of π, defined as $\delta = D_0, D_1, \ldots D_n$, where $D_0 = \emptyset$ and $D_i = \mathrm{res}_A(D_{i-1} \cup C_{i-1})$ for all $1 \leq i \leq n$.

We also define $\mathrm{sts}(\pi)$ as the *state sequence* of π by

- $\mathrm{sts}(\pi) = W_0, W_1, \ldots W_n$, where $W_i = C_i \cup D_i$ for all $0 \leq i \leq n$.

The above notion of an interactive process is visualized in Figure 1. Note that the context and results sets are not required to be disjoint, although the figure shows them as non-overlapping rectangles.

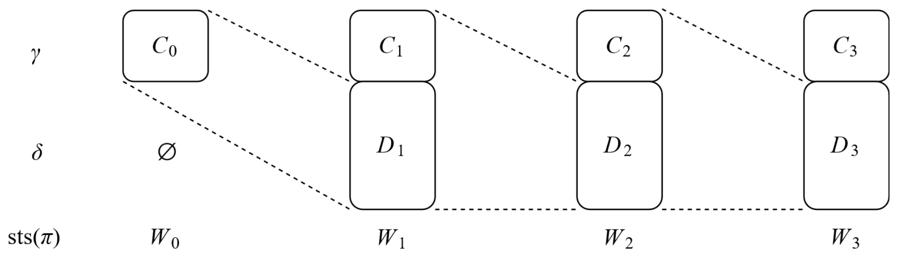

Fig. 1. An interactive process π in a reaction system.

As a consequence of this definition, every interactive process is finite with predetermined length. Therefore an interactive process can be thought of as a finite sequence of states (the union of contexts and results). The idea of input from the surrounding environment is formalized by the context sequence. A process in which every context set is empty is said to be *context-independent*.

Interactive processes also encompass the concept of no permanency. Every new state consists of the products of the previous state and the environmental input. Hence, if an entity is not produced by any enabled reaction and is not present in the context set, then it will disappear. This idea steams from abstract biochemistry where an entity must be sustained by some active process. In the absence of such a process, the entity will vanish. See [4, 6] for motivations and more details.

Now we present a possible notion of reversibility for reaction systems which we introduced and investigated in [3]. Generally, a sequential model of computation (such as the model of reaction systems) might be considered reversible if it is "backward deterministic", that is, if no "configuration" is accessible from two different configurations (in a way which is "undistinguishable"), or in other words, every configuration has a *unique predecessor*. The point is, that given any "state" of the system, we should be able to determine the preceding computational step. Reaction systems perform computations via interactive processes, thus we should interpret the concept of configuration (or "state") and the concept of unique predecessor for such interactive processes.

In [3] we have followed the natural idea to identify the configurations we are interested in (from the point of view of reversibility) with the states of interactive processes as defined above to be the union of the result sets and the corresponding context sets.

Definition 1. ([3]) *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* $\mathrm{sts}(\pi) = W_0, W_1, \dots W_n$.

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

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

After defining reversibility for individual interactive processes, we would like to continue defining reversible reaction systems as ones with reversible interactive processes only. However, if no state is allowed to have multiple predecessors, then this constraint applies to the empty state as well.

Since reaction systems are finite both in terms of reactions and entities, regardless of which state we begin with, we eventually run out of enabled reactions or start to loop. Thus, if every interactive process is required to be reversible, then the empty state must have a single, unique predecessor. As this requirement would greatly restrict reversible reaction systems, it seems reasonable to not consider every interactive process in a reaction system when defining reversibility.

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*.

Definition 2. ([3]) *A reaction system $\mathscr{A}$ is* reversible*, if every non-restarting interactive process in $\mathscr{A}$ is reversible.*

Before we continue, we would like to present the requirements this definition poses on reversible interactive processes. As simple as this notion of a reversible process is, it has great consequences on the possible reactions and context entities of the enclosing system. For a formal description of the necessary and sufficient conditions reaction systems need to fulfill in order to be reversible, see Theorem 1 of [3]. Here we present the basic ideas without proofs in a less formal manner.

In the following discussion, we assume a finite background set S and a finite set of reactions $A \subseteq \mathrm{rac}(S)$. In order for a reaction system to be reversible, it must fulfill the following conditions.

1. If $E_1, E_2 \subseteq A$ are different sets of reactions in $\mathrm{EN}_A(S)$, they must produce different result sets, or in other terms, $E_1, E_2 \in \mathrm{EN}_A(S)$ with $E_1 \neq E_2$ implies $P_{E_1} \neq P_{E_2}$. Clearly, if we had two different sets of reactions that produce the same entities when applied, then the state formed by these entities would have multiple predecessors.

2. If we take any two distinct subsets of S, then the reactions enabled by these sets must be different as well. In other words, if $T_1, T_2 \subseteq S$ with $T_1 \neq T_2$ and $\mathrm{en}_A(T_1) \neq \emptyset$, then $\mathrm{en}_A(T_1) \neq \mathrm{en}_A(T_2)$ must also hold. To see this, consider that if two subsets would enable the same set of reactions, then the result set of these reactions would be a state having at least two predecessors (the very subsets T_1, T_2 we took from S).
3. Finally, consider the result sets of the reaction sets in $\mathrm{EN}_A(S)$. If we are able to transform one such result set R_1 to another, say R_2, by augmenting it with entities which can also appear in the context (as part of a context set of some interactive process), then the state $W = R_2$ will have multiple predecessors. To see this, let $D = R_2$, $C = \emptyset$ and $D' = R_1$, $C' = R_2 \setminus R_1$, then consider the state $W = D \cup C = D' \cup C'$. If $R_2 = D = \mathrm{res}_{E_2}(W')$ and $R_1 = D' = \mathrm{res}_{E_1}(W'')$, then W has at least two predecessors, W' and W''. (We can obtain W from W' by applying the reactions in E_2 and adding the context C, or we can obtain W also from W'' by applying the reactions in E_1 and adding the context set C'.) To formalize this condition, we "refine" the background set as $S = \Sigma_c \cup \Sigma_p$, the union of (not necessarily disjoint) alphabets of symbols where $\Sigma_c \subseteq S$ contains those entities which can appear as environmental input in the context sets of interactive processes, and $\Sigma_p \subseteq S$ containing those which can appear as products of reactions. (This is similar to so called context-restricted reaction systems studied in [11].) Using this notation, we can formalize the above idea by requiring that having $R_1, R_2 \in \mathrm{RES}_A(S)$ with $R_1 \neq R_2$ should also imply that $R_1 \setminus \Sigma_c \neq R_2 \setminus \Sigma_c$.

3 Transition Graphs of Reversible Systems

When introducing reversibility into a particular model of computation, the question naturally arises, how this affects the computational properties of the model. In the case of reaction systems, interactive processes form the only device of computation, thus when examining the higher-level computational properties of a specific system, we should start by looking at the contained processes. Since any system may only contain a finite number of possible states, so-called transition graphs offer a concise way of depicting every possible interactive process a particular system may enclose. In this section, we introduce the definition of transition graphs and then explore the graphs generated by the reversible systems of Section 2.

In what follows, we first introduce reachable result sets, that will eventually form the vertices of the transition graphs. Assuming the standard definition of interactive processes (see Section 2) D_0 must be empty, hence there might be result sets that cannot occur in any interactive process. By considering reachable result sets only, we exclude these sets from the transition graphs.

Definition 3. *Let $\mathscr{A} = (S, A)$ be a reversible reaction system with $S = \Sigma_p \cup \Sigma_c$ (where Σ_p and Σ_c are not necessarily disjoint). The result set $D \subseteq \Sigma_p$ is* reachable *if there exists a non-restarting interactive process $\pi = (\gamma, \delta)$ in $\mathscr{A}$ with $\delta = D_0, D_1, \ldots D_n$ such that $D = D_i$ for some $0 \leq i \leq n$. The set of reachable result sets in $\mathscr{A}$ is denoted by $\mathrm{REACH}_{\mathscr{A}}$.*

If D_0 was allowed to be non-empty (thus, the reaction system can initiate its computation from an arbitrary result set), then every result set is reachable (since, we can freely choose the initial result set). By requiring D_0 to be empty, we restrict the possible result sets in interactive processes to the reachable sets of Definition 3. The set of reachable results sets is, in turn, determined by the reactions of the underlying reaction system.

Transition graphs were first introduced in [9] as vertices representing the subsets of the background set (usually denoted as S) connected by directed edges equivalent to the relation of "can be obtained from". Formally, the edge set is defined as $E = \{(W_1, W_2) \mid W_1 \subseteq S,\ \mathrm{res}_A(W_1) \subseteq W_2\}$. Here, we would like to underline the subset relationship between the underlying sets of the connected vertices. This transition graph definition incorporates context sets (environmental input) by considering two states connected, if the result of the former can be augmented with context to form the latter.

As we are exclusively interested in subsets of S that appear in interactive processes, we will modify this notion to include only reachable results sets as vertices. Furthermore, since we wanted to put more emphasis on the role of the input, in our definition of transition graphs, edges are labeled with input sets from the environment. Such a labeled edge is drawn between two vertices if the union of the source vertex and the label produces the destination vertex as a result.

Definition 4. *Let $\mathscr{A} = (S, A)$ be reaction system, with $S = \Sigma_c \cup \Sigma_p$ as above. The* transition graph *of $\mathscr{A}$ is the graph $\mathrm{TG}_{\mathscr{A}} = (V, E)$, where $V = \mathrm{REACH}_{\mathscr{A}}$ is the set of vertices and*

$$E = \{\, (D, C, D') \mid D, D' \in V \text{ and } C \subseteq \Sigma_c \text{ such that } \mathrm{res}_A(D \cup C) = D' \,\}$$

is the set of directed edges with D being the starting vertex, D′ the end vertex and C the label.

Example 2. Let $\mathscr{A}$ be a reaction system in which $\Sigma_p = \{1, 3, 5\}$ is the product alphabet, $\Sigma_c = \{0, 2, 4\}$ is the context alphabet and $A = \{a, b, c\}$ is the set of reactions, where

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

Then, $\mathrm{TG}_{\mathscr{A}}$ consists of the vertices $V = \{\ \emptyset, \{1\}, \{3\}, \{5\}, \{3,5\}\ \}$ and edges

$$E = \{\ (\emptyset, \{0\}, \{1\}),\ (\{1\}, \{2\}, \{3\}),\ (\{1\}, \{4\}, \{5\}),\ (\{1\}, \{2,4\}, \{3,5\})\ \}.$$

With all the necessary notions in place, we now continue by examining the transition graphs of reversible reaction systems. In such systems, each result set may be the result of exactly one other state. As a consequence, for example, loops are forbidden (explained in more detail and proved in 1), which puts a firm constraint on the complexity of the non-restarting interactive processes. Therefore, the transition graphs of these systems are rather simple, they only contain finite computational branches.

Theorem 1. *If $\mathscr{A}$ is a reversible reaction system, then the transition graph of $\mathscr{A}$ is either a single vertex or a directed rooted tree with all the edges pointing away from the root.*

Proof. Let $\mathscr{A} = (S, A)$ be a reversible reaction system with $S = \Sigma_p \cup \Sigma_c$ (where Σ_p and Σ_c are not necessarily disjoint), and let $\mathrm{TG}_{\mathscr{A}}$ be its transition graph.

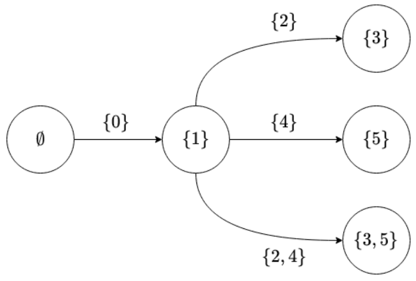

Fig. 2. Transition graph of the reaction system from Example 2.

By definition, $\mathrm{TG}_{\mathscr{A}}$ only includes an edge between two vertices if they are subsequent result sets in some non-restarting interactive process. Thus, the vertex for the empty result set does not have any incoming edges. Since the empty set is the initial result set (D_0) of every non-restarting interactive process in any reaction system, it will always be included in the transition graph.

If there is no context set C such that $\mathrm{en}_A(\emptyset \cup C) \neq \emptyset$ then $\mathrm{TG}_{\mathscr{A}}$ consists of a single vertex: the empty result set.

Now we show that $\mathrm{TG}_{\mathscr{A}}$ is a directed rooted tree if it has multiple vertices. A graph is a directed rooted tree if there is exactly one path between the root vertex and any other vertex. Since every vertex in the transition graph is a result set in some non-restarting interactive process, there must be a path between the root vertex and the vertex representing this set. Thus, we know that at least one path must exist connecting the root vertex with every other vertex.

Since $\mathscr{A}$ is reversible, every state has a unique predecessor. Consequently, every result set has a unique predecessor. As the vertices in the transition graph represent result sets and edges represent predecessor/successor relations, this means that every vertex other than the root has exactly one incoming edge. Therefore, there must be at most one path going from the root vertex to every other vertex. Because both the lower and the upper bound are equal to one, we have that there is a single path from the root vertex to any other vertex. Thus, if $\mathrm{TG}_{\mathscr{A}}$ has more than one vertex, it's a directed rooted tree with all edges pointing away from the root. □

As a consequence of the above theorem, non-restarting interactive processes in reversible systems (that are essentially computations) are just paths in a finite tree. Since this is a rather strict limitation, we might start experimenting with small relaxations in the underlying definitions to give rise to more complex graphs (those with vertices having in-degree greater than one or even containing cycles).

Such modification of particular interest is concerned with the definition of interactive processes. In the standard setting, D_0 (the initial result set) is empty for every interactive process. If the context sets can incorporate arbitrary entities from the background set, this does not pose any constraint on the initial state. On the other hand, in the case when the context and the product alphabets are different (as in the case of reversible systems), the product and the context alphabet can be disjoint, some results sets may not even be reachable at all. Similar ideas motivated the introduction of so called initialized context-restricted reaction systems in [11] where nonempty initial product sets D_0 are also allowed. Now, let us examine how non-empty D_0 sets affect the transition graphs of reversible reaction systems. Following [11], we call our model *initialized* reversible reaction systems.

Theorem 2. *If $\mathscr{A}$ is an initialized reversible reaction system, then every component of the transition graph of $\mathscr{A}$ is either*

- *a single vertex,*
- *a directed tree with edges pointing away from the root, or*
- *a component with exactly one directed cycle, such that each vertex of the cycle can also be the root of a tree with edges pointing away from the cycle.*

Proof. Let $\mathscr{A} = (S, A)$ be an initialized reversible reaction system with $S = \Sigma_p \cup \Sigma_c$ (where Σ_p and Σ_c are not necessarily disjoint) and with interactive processes that might start with a non-empty D_0 set.

Since our definition for the transition graph is the same as in Theorem 1, the reversibility of $\mathscr{A}$ results in a maximum of one for the in-degree of every vertex.

Now, let us consider the components of the transition graph. Given a result set $D \subseteq \Sigma_p$, if there is no $W \subseteq S$ such that $\text{res}_A(W) = D$ (in any of the non-restarting interactive processes of $\mathscr{A}$), then the in-degree of the vertex corresponding to D is equal to 0. In this case, this vertex is either a component in itself or the root of a directed rooted tree. The former holds, if no result set can be reached from D in any of the non-restarting interactive processes (thus, the out-degree of the vertex is zero), while the latter is proved in the proof of Theorem 1.

With the first two cases (single vertex and tree) covered, let us examine components with exactly one cycle. We already know, that vertices with in-degree equal to zero either form single-vertex components or root trees. Therefore, we only need to consider components in which the in-degree of every vertex is equal to one (as we previously proved that no vertex has in-degree greater than one). In this case, the component must include at least one cycle, otherwise there could be vertices with no incoming edges. However, while a component can include a single cycle when the in-degree of every vertex is one, multiple cycles are not possible. Single cycle components can take the form of a “branching ring”, where the component includes a ring (or cycle) at its core and each member of this ring can additionally be the root of a tree branching out. On the other hand, multiple cycles can only be realized by including at least one vertex in each cycle with an edge coming from the cycle itself and an edge coming from a vertex outside of the appropriate cycle. As such configurations are forbidden for transition graphs of reversible systems, all remaining components must form a branching ring. □

Example 3. Let $\mathscr{A} = (S, A)$ be an initialized reversible reaction system with $S = \Sigma_p \cup \Sigma_c$ where $\Sigma_p = \{1, 3, 5, 7, 9, 11, 13, 15\}$ is the product alphabet, $\Sigma_c = \{0, 2, 4, 6, 8, 10, 12\}$ is the context alphabet and $A = \{a, b, c, d, e, f, g\}$ is the set of reactions, where

$$a = (\{0,1\},\ \{6\},\ \{3\}), \qquad b = (\{2,3\},\ \emptyset,\ \{5\}), \qquad c = (\{4,5\},\ \emptyset,\ \{1\}),$$
$$d = (\{1,6\},\ \{0\},\ \{7\}), \qquad e = (\{7,8\},\ \{10\},\ \{9\}), \qquad f = (\{7,10\},\ \{8\},\ \{11\}),$$
$$g = (\{12,13\},\ \emptyset,\ \{15\}).$$

The transition graph $\mathrm{TG}_{\mathscr{A}}$ consists of three components: a branching ring, a tree, and a single vertex, as shown in Figure 3.

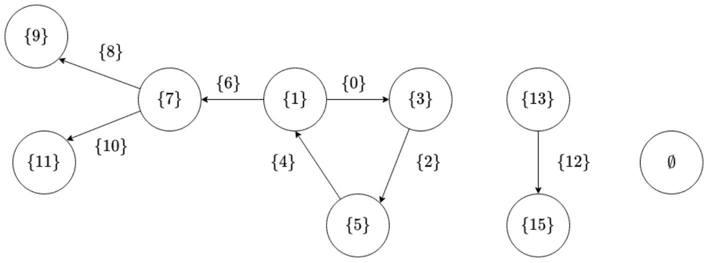

Fig. 3. Transition graph of the reaction system from Example 3.

As stated by Theorem 2, with a slight modification in the definition of interactive processes, we can achieve more involved transition graphs: those with multiple components and even a cycle per component. This allows for the occurrence of computations with loops, increasing the intuitive computational power of the model. Nevertheless, examined solely from an informal perspective, even with this change, we are unable to achieve the computational power of reversible finite automata, for example. The lack of higher in-degrees and multiple cycles places a severe constraint on the set of possible non-restarting interactive processes.

4 Reaction Systems which are Reversible with Lookbehind

As we have discussed in Section 2, we would like to look at the notion of reversibility as a kind of backward determinism, that is, given any "state" or configuration of the system, we should be able to determine the preceding computational step. In the previous section, following our earlier work in [3], we have interpreted the concept of configurations (or "states") as the state sets (that is, as the unions of product sets and context sets) of interactive processes. In this section we follow a different approach, one that is similar to how reversibility of (finite) automata is usually defined: The "state" of the machine is interpreted as the internal state of the finite control together with additional information regarding the position of the reading head on the input tape and the contents of the corresponding tape cell. See [2] and[12] for more details, or [10] for some more recent work regarding reversibility of finite automata. In such an interpretation, the predecessor configurations in a computation should be unique with respect to the internal state of the finite control and the input symbol that the reading head has just left behind, that is, the symbol that was read from the input tape in the previous computational step.

In this section we introduce the notion of reversibility with lookbehind by interpreting the "state" (or configuration) of reaction systems (in a similar manner as described above) as the current result set obtained in the previous computational step, together with the context set that was input in the same (that is, the previous) computational step. Thus, predecessor states of interactive processes should be unique with respect to the current product set, and the context set which was used to obtain the product, that is, the context set corresponding to the previous state of the interactive process.

In other words, our notion of reversibility for reaction systems which we recalled in Section 2 is based on the concept of unique predecessors. If a combination of reaction results and context entities (together forming what is called a state) has exactly one way to be produced, then it is said to have a unique predecessor. Hence, this definition focuses on the current result set and context set when determining which state occurred previously.

In this section, we introduce reversibility with lookbehind by taking a different approach to the definition of unique predecessors. Inspired by finite state automata, which are considered backward deterministic (and thus, reversible) if the current internal state and the previously consumed input symbol uniquely determine the previous internal state, reaction systems which are reversible with lookbehind can inspect both the current state and the previous context set. As a consequence, multiple state sets can produce the very same result sets (without loosing the property of being reversible) given they include distinct context sets.

Definition 5. *Let $\mathscr{A} = (S, A)$ be a reaction system and $\pi = (\gamma, \delta)$ be an interactive process in $\mathscr{A}$, such that $\gamma = C_0, C_1, \ldots C_n$, $\delta = D_0, D_1, \ldots D_n$ and* $\mathrm{sts}(\pi) = W_0, W_1, \ldots W_n$.

A state W_i, $1 \leq i \leq n$, has multiple predecessors with lookbehind *if there exist $D \subseteq S$ such that $D \neq D_{i-1}$, but $\mathrm{res}_A(D \cup C_{i-1}) = D_i$. If there is no such D, then W_i has a* unique predecessor with lookbehind.

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

Now, we can define reversible systems using the above definition.

Definition 6. *A reaction system $\mathscr{A}$ is* reversible with lookbehind *if every non-restarting interactive process in $\mathscr{A}$ is reversible with lookbehind.*

Regarding transition graphs, an immediate consequence of the new definition of reversibility is the possibility of in-degrees higher than one. As shown in Figure 4, despite the two incoming edges of the vertex $\{4\}$, when reversing the previous computation, we can now decide which one to take based on the preceding context (the label of the edge).

Continuing our previous discussion we now compare the state diagrams and the transition graphs of reversible finite transition systems and reversible reaction systems with lookbehind, respectively. A finite transition system is usually denoted as a triplet $F = (Q, \Sigma, \delta)$, where Q is the set of states, Σ is the input alphabet and δ is the state transition function, mapping the current state and an input symbol to a result state. Finite

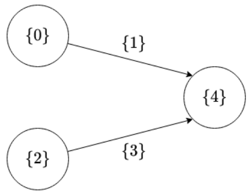

Fig. 4. Transition graph configuration that is not permitted for ordinary reversible systems, but is allowed in the case of systems which are reversible with lookbehind.

transition systems differ from finite state automata by not having distinguished start and final states. Although the following results can be stated for finite state automata as well, since interactive processes in reaction systems lack the concept of final (or accepting) state, reaction systems seem to be more closely related to transition systems. (This is further supported by [4] where a method is presented to convert finite transitions systems to reaction systems.)

In what follows, when constructing transition graphs, we assume the definition of interactive processes in which D_0 (or, the initial result set) is empty. This means that every interactive process must start from the empty result set which makes it easier to create reaction systems from transition systems, as it allows for more control over the reachable result sets. Example 4 explores this idea in greater detail.

Example 4. Let F be a reversible finite transition system for which we want to construct a corresponding reaction system. Starting with the background set, we can create an entity for each input symbol of F as well as for each state of F. The entities created from the input symbols comprise the input alphabet Σ_c, while the entities representing the states belong to the product alphabet Σ_p. Now, we have that the result sets of the reaction system (D_i) represent the actual state of the underlying transition system, and the context sets correspond to the received input.

If the initial result set D_0 was allowed to be non-empty, then any entity of the product alphabet could be present in this set, even multiple entities. Since each entity represents a distinct state of the underlying transition system, the presence of multiple entities would mean that the transition system is in multiple states at once. As this is not permitted, one should require D_0 to be empty, since that way, the facilitation and inhibition aspects of the reaction can prevent such cases.

Theorem 3. *For every reversible finite transition system, there is a reaction system which is reversible with lookbehind and has the same states and transitions (apart from a starting state and its corresponding transitions).*

Proof. Let $F = (Q, \Sigma, \delta)$ be a reversible finite transition system. Then, the state diagram of F, $\mathrm{SD}_F = (V_F, E_F)$ is a directed graph, where $V_F = Q$ is the set of vertices and $E_F = \{(v, l, w) \mid \delta(v, l) = w\}$ is the set of directed, labeled edges.

Now, let us construct a reaction system $\mathscr{A} = (S, A)$ from F. Initially, we choose the background set S to be the union of two disjoint sets ($S = \Sigma_p \cup \Sigma_c$): the product alphabet corresponds to the states of the transition system (thus, $\Sigma_p = Q$), while the input alphabet is equivalent to the input alphabet of F (thus, $\Sigma_c = \Sigma$). We can define a set of reactions using the transition function of F:

$$\{(\{q, i\}, \Sigma_c \setminus \{i\}, \{r\}) \mid \delta(q, i) = r, \text{ for } q, r \in Q, \ i \in \Sigma\}.$$

Since the initial result set of an interactive process must be the empty set, we will now have that the transition graph of $\mathscr{A}$ is a single vertex representing the empty set. Therefore, we will introduce a new entity, α_q for each state $q \in Q$ and an appropriate reaction that lead from the empty set to the result set representing this state q of F, respectively. This way we can overcome the fact that finite transition systems do not have a designated initial state in contrast with reaction systems.

With this in mind, let us redefine the background and the reaction set of $\mathscr{A}$. The background set S is now the union of the following two sets: $\Sigma_p = Q$ and $\Sigma_c = \Sigma \cup \{\alpha_q \mid q \in Q\}$. The reaction set A is then defined as follows:

$$\begin{aligned} A = &\{(\{q, i\}, \Sigma_c \setminus \{i\}, \{r\}) \mid \delta(q, i) = r \text{ for } q, r \in Q, \ i \in \Sigma\} \cup \\ &\{(\{\alpha_q\}, \Sigma_c \setminus \{\alpha_q\}, \{q\}) \mid q \in Q\}. \end{aligned}$$

The transition graph of $\mathscr{A}$ is defined as a directed graph based on the result sets and inputs of the non-restarting interactive processes in the system. Because of the definition of the transition graph, given any vertex q in the state diagram of F, we have a vertex in $\mathrm{TG}_{\mathscr{A}}$ corresponding to $D = \{q\}$. Additionally, for every edge (v, l, w) in the state diagram of F, we have an appropriate edge pointing from the vertex $D_1 = \{v\}$ to the vertex $D_2 = \mathrm{res}_{\mathscr{A}}(\{v, l\}) = \{w\}$ (because of the definition of A).

Consequently, apart from the vertex representing the empty state and its outgoing edges, the transition graph of $\mathscr{A}$ and the state diagram of F are isomorphic.

What is left to prove is that the states in the non-restarting interactive processes of $\mathscr{A}$ have unique predecessors with lookbehind (making $\mathscr{A}$ reversible with lookbehind). Since F is a reversible transition system, there is no vertex in its state diagram that has more than one incoming edge with the same label. As a consequence, each vertex in the transition graph of $\mathscr{A}$ satisfies the same property. Combining this fact with the disjointness of the product and input alphabets, we have that no state can be reached with the same input (or label, in the transition graph) from two different result sets. Which is precisely the definition of having a unique predecessor with lookbehind. □

Theorem 4. *For every reaction system which is reversible with lookbehind, there is a reversible finite transition system with the same states and transitions.*

Proof. Let $\mathscr{A} = (S, A)$ be a reversible lookbehind reaction system with $S = \Sigma_p \cup \Sigma_c$ (where Σ_p and Σ_c are not necessarily disjoint). Also, let $\mathrm{TG}_{\mathscr{A}} = (V_{\mathscr{A}}, E_{\mathscr{A}})$ be the transition graph of $\mathscr{A}$.

Now, let us construct a finite transition system $F = (Q, \Sigma, \delta)$ from $\mathscr{A}$. Using the transition graph of $\mathscr{A}$, we can define the states of the transition system as $Q = V_{\mathscr{A}}$. The input alphabet of F is going to contain the subsets of the input alphabet of $\mathscr{A}$, thus $\Sigma = 2^{\Sigma_c}$. By considering the edges in $\mathrm{TG}_{\mathscr{A}}$, we can define the transition function as

$$\delta(q, i) = r, \text{ if there is an edge in } E_{\mathscr{A}} \text{ from } q \text{ to } r \text{ with label } i.$$

Because of our above definition of F, given any vertex (representing a result set) in the transition graph of $\mathscr{A}$, we will have a corresponding vertex in the state diagram of F. Furthermore, since the edges in the state diagram correspond to the transition function δ, which in turn was defined via the edges of $\mathrm{TG}_{\mathscr{A}}$, we have that each edge in the state diagram of F will map to an edge in $\mathrm{TG}_{\mathscr{A}}$. Thus, we have that the state diagram of F and the transition graph of $\mathscr{A}$ are isomoprhic.

Analogous to the proof of Theorem 3, now we need to show that F is a reversible transition system. Since $\mathscr{A}$ is reversible, no vertex in its transition graph has more than one incoming edge with the same label. As the state diagram of F is isomoprhic to $\mathrm{TG}_{\mathscr{A}}$ and each edge has the same label as its counterpart in $\mathrm{TG}_{\mathscr{A}}$, the same is true for each vertex in the state diagram. Thus, because the edges represent the state transitions induced by δ, we have that F is reversible. □

Based on the previous two theorems, we can state the following.

Proposition 1. *The state transition graphs of reversible finite transition systems correspond to the transition graphs of reaction systems which are reversible with lookbehind, apart from the special initial vertex corresponding to the initial empty result set of the reaction system, and vice-versa, the transition graph of any reaction system which is reversible with lookbehind corresponds to the state transition graph of a reversible finite transition system.*

5 Conclusion

In this paper, we have studied the possible computations of reversible reaction systems by examining their transition graphs. First, we have considered reaction systems which are reversible according to our definition of reversibility given in [3] and concluded that the computational graphs (and so the possible computations) are very simple. Then we examined the same notion of reversibility for initialized reaction systems (see [11]) and obtained transition graphs which are somewhat more complicated, but still quite simple. Finally, motivated by the reversibility of (finite) automata, we introduced the notion of reversibility with lookbehind, which finally is able to produce functioning corresponding to the same types of transition graphs (and thus, the same possible computations) as the state transition diagrams of reversible finite transition systems.

To study other aspects of reaction systems which are reversible with lookbehind is a research topic that we would like to investigate in more detail in the future.

References

1. Aman, B., Ciobanu, G.: Controlled reversibility in reaction systems. In: Gheorghe, M., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) Membrane Computing. pp. 40–53. Springer International Publishing, Cham (2018)
2. Angluin, D.: Inference of reversible languages. J. ACM 29(3), 741–765 (1982)
3. Bagossy, A., Vaszil, G.: Simulating reversible computation with reaction systems. Journal of Membrane Computing (Accepted)

4. Brijder, R., Ehrenfeucht, A., Main, M., Rozenberg, G.: A tour of reaction systems. Int. J. Found. Comput. Sci. 22, 1499–1517 (11 2011)
5. Dennunzio, A., Formenti, E., Manzoni, L., Porreca, A.E.: Reachability in resource-bounded reaction systems. In: International Conference on Language and Automata Theory and Applications. pp. 592–602. Springer (2016)
6. Ehrenfeucht, A., Rozenberg, G.: Reaction systems. Fundam. Inf. 75(1–4), 263–280 (Jan 2007)
7. Ehrenfeucht, A., Kleijn, J., Koutny, M., Rozenberg, G.: Minimal reaction systems. In: Priami, C., Petre, I., de Vink, E. (eds.) Transactions on Computational Systems Biology XIV. pp. 102–122. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
8. Ehrenfeucht, A., Rozenberg, G.: Events and modules in reaction systems. Theor. Comput. Sci. 376, 3–16 (05 2007)
9. Genova, D., Hoogeboom, H.J., Jonoska, N.: A graph isomorphism condition and equivalence of reaction systems. Theoretical Computer Science 701, 109–119 (2017)
10. Holzer, M., Kutrib, M.: Reversible nondeterministic finite automata. In: Phillips, I., Rahaman, H. (eds.) Reversible Computation. pp. 35–51. Springer International Publishing, Cham (2017)
11. Męski, A., Penczek, W., Rozenberg, G.: Model checking temporal properties of reaction systems. Information Sciences pp. – (08 2015)
12. Pin, J.E.: On reversible automata. In: Simon, I. (ed.) Proceedings of the first LATIN conference. pp. 401–416. Lecture Notes in Computer Science 583, Springer (1992), https://hal.archives-ouvertes.fr/hal-00019977