Journal of Logical and Algebraic Methods in Programming 109 (2019) 100481

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Graph transformation through graph surfing in reaction systems

Hans-Jörg Kreowski [a,*], Grzegorz Rozenberg [b,c]

[a] *University of Bremen, Department of Computer Science, Bibliothekstr. 5, 28359 Bremen, Germany*
[b] *Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, the Netherlands*
[c] *Department of Computer Science, University of Colorado, Boulder, CO 80309-0347, USA*

ARTICLE INFO

Article history:
Received 30 November 2018
Received in revised form 29 April 2019
Accepted 8 August 2019
Available online 13 August 2019

Keywords:
Reaction systems
Graph-based reaction systems
Graph surfing
Graph transformation
Territorial graph surfing systems

ABSTRACT

In this paper, we introduce graph-based reaction systems as a generalization of set-based reaction systems, a novel and well-investigated model of interactive computation. Graph-based reaction systems allow us to introduce a new methodology for graph transformation, which is not based on the traditional “cut, add, and paste” approach, but rather on moving within a “universe” graph B (surfing on B) from a subgraph of B to a subgraph of B, creating subgraph trajectories within B. We illustrate this approach by small case studies: approximating the Sierpinski triangle, simulating finite automata, implementing two shortest-paths algorithms, and simulating cellular automata. Finally, we introduce the notion of territorial graph surfing systems taking a more “global” look at graph-based reaction systems.

1. Introduction

The goal of this paper is to introduce a novel framework for graph transformation. It results from extending set-based reaction systems to graph-based reaction systems. We introduce the main notions and illustrate the framework by considering four case studies: an approximation of the Sierpinski triangle, a simulation of finite automata, two variants of parallel shortest-path algorithms, and a simulation of cellular automata.

The concept of reaction systems was introduced about ten years ago (see [9]) and has been intensely studied since then (see, e.g., [3,5–8,10,11,14,23,24]). It was inspired by the functioning of living cells and the original motivation was to provide a formal framework for modeling of biochemical processes taking place in the living cell. It turned out to be a novel and actively investigated paradigm of interactive computation interesting also for modeling of information processing beyond biochemistry. The original notion of reaction systems and their interactive processes is purely set-theoretic. In this paper, we enhance the framework by a graph-based level of description so that graph related problems and models can be handled in a natural way.

A graph-based reaction system consists of a finite background graph and a set of reactions. A reaction has three components: a reactant graph, an inhibitor which is a pair consisting of a set of nodes and a set of edges, and a product graph. Reactant graph and product graph are subgraphs of the background graph while the inhibitor is an ordered pair consisting of a subset of the set of nodes and a subset of the set of edges of the background graph. Reactions specify basic transformations of states which are subgraphs of the background graph. A reaction is enabled by a state if the reactant graph is a

[*] Corresponding author.
E-mail addresses: kreo@informatik.uni-bremen.de (H.-J. Kreowski), g.rozenberg@liacs.leidenuniv.nl (G. Rozenberg).

https://doi.org/10.1016/j.jlamp.2019.100481
2352-2208/© 2019 Elsevier Inc. All rights reserved.

subgraph of the state and no nodes or edges of the inhibitor are present in the state. The dynamics of a reaction system is defined by discrete interactive processes. In one step of a process, all reactions enabled by the current state are applied simultaneously and the union of their product graphs forms the successor state. This means that the processing is deterministic and parallel. Since the successor state T' of a current state T is the union of the product graphs of all reactions enabled by T, a node or an edge of T is sustained, i.e., it is also present in T', only if it is produced by one of the enabled reactions. In this sense, each consecutive graph produced by an interactive process is a "new" graph.

Most approaches to rule-based graph transformation consider abstract graphs meaning that the graph resulting from a rule application is uniquely constructed up to isomorphism. In our framework, reactions are applied to subgraphs of a concrete background graph, and they yield concrete subgraphs. Therefore, the processing of graphs in graph-based reaction systems may be seen as a kind of surfing in analogy to the surfing on the Internet by following links, where our "websites" are subgraphs and our "links" are (given by) reactions.

In this paper we demonstrate that such graph surfing by graph-based reaction systems can be used for modeling of various kinds of information processing, where the underlying data structures are either graphs or they can be represented by graphs in a natural way. To this aim, we present four small case studies, which illustrate the potential of graph-based reaction systems to specify diverse sorts of information processing as well as to support the corresponding correctness proofs and semantic analysis.

The paper is organized as follows. After the preliminaries recalling the basic notions and notations concerning graphs in Section 2, Section 3 presents the notion of graph-based reaction systems and their interactive processes. In Section 4 the well-known Sierpinski triangle is approximated by means of graph-based reaction systems. In Section 5 finite automata are simulated by reaction systems operating on the state graphs of the automata. The computational potential of graph-based reaction systems is illustrated in Section 6, where two parallel shortest-path algorithms are considered. In Section 7 the well-known computational model of cellular automata is simulated by graph-based reaction systems. In Section 8 we introduce territorial graph surfing systems, which result from taking a more "global" look at graph-based reaction systems. The discussion in Section 9 concludes the paper.

This paper is a revised and extended version of the paper *Graph Surfing by Reaction Systems* from the proceedings of the International Conference on Graph Transformation 2018 (see [19]).

2. Preliminaries

In this section, the basic notions and notations concerning graphs (to be used in this paper) are recalled.

A (simple, directed, and edge-labeled) *graph* is a system $G = (V, \Sigma, E)$, where V is a finite set of *nodes*, Σ is a finite set of *edge labels*, and $E \subseteq V \times V \times \Sigma$ is a set of *edges*.

If G is such that V is empty, then G is called *empty graph* (*over* Σ), denoted by $\emptyset_\Sigma$.

For an edge $e = (v, v', x)$, v is called the *source* of e, v' the *target* of e, and x the *label* of e. We also say that e is an x*-edge*, and if $v = v'$, then e is an x*-loop* or simply a *loop*.

The components V, Σ, and E of G are also denoted by V_G, Σ_G, and E_G, respectively. The class of all graphs is denoted by $\mathcal{G}$. We reserve a special label $*$ to indicate that an edge labeled by $*$ can be considered as unlabeled. This label is used only for this purpose, and in order to simplify graphic representations, it is omitted in drawings: the lack of the label of an edge e means that e is labeled by $*$.

For graphs G and H such that $\Sigma_H = \Sigma_G$, H is a *subgraph of* G if $V_H \subseteq V_G$, and $E_H \subseteq E_G$. We use $Sub(G)$ to denote the set of subgraphs of G. The inclusion, the union, and the intersection of subgraphs are defined component-wise. Thus, for subgraphs $H, H' \in Sub(G)$ with $\Sigma = \Sigma_H = \Sigma_{H'}$, $H \cup H' = (V_H \cup V_{H'}, \Sigma, E_H \cup E_{H'})$ and $H \cap H' = (V_H \cap V_{H'}, \Sigma, E_H \cap E_{H'})$. Obviously, the union and the intersection of subgraphs of G are subgraphs of G.

For a graph $G = (V, \Sigma, E)$, an ordered pair (X, Y) such that $X \subseteq V$ and $Y \subseteq E$ is called a *selector* of G. Note that X and Y are "independent" of each other: X may contain nodes that are neither sources nor targets of edges in Y, and Y may contain edges such that some of their sources and/or targets are not in X.

The inclusion, union, and intersection of selectors are defined component-wise, i.e., for selectors $P = (X, Y)$ and $P' = (X', Y')$, $P \subseteq P'$ if $X \subseteq X'$ and $Y \subseteq Y'$, $P \cup P' = (X \cup X', Y \cup Y')$, and $P \cap P' = (X \cap X', Y \cap Y')$, respectively.

Subgraphs and selectors of G are closely related to each other.

A selector $P = (X, Y)$ of G induces a subgraph of G, denoted by $ind(P)$, by adding all sources and targets of edges of Y to X, i.e., $ind(P) = (X \cup \{v, v' \mid (v, v', x) \in Y \text{ for some } x \in \Sigma\}, \Sigma, Y)$. This subgraph induction preserves inclusion, union, and intersection, i.e., for all selectors P and P' of G, $P \subseteq P'$ implies $ind(P) \in Sub(ind(P'))$, $ind(P \cup P') = ind(P) \cup ind(P')$, and $ind(P \cap P') = ind(P) \cap ind(P')$.

On the other hand, for $H \in Sub(G)$, the *selector* of H is $U(H) = (V_H, E_H)$. The operation U (of getting $U(H)$ from H) is called *extraction*. Obviously, extraction preserves inclusion, union and intersection, i.e., for all $H, H' \in Sub(G)$, if $H \in Sub(H')$, then $U(H) \subseteq U(H')$, $U(H \cup H') = U(H) \cup U(H')$, and $U(H \cap H') = U(H) \cap U(H')$. It is easy to see that the induced subgraph of the extraction of a subgraph yields the subgraph back, i.e., for all $H \in Sub(G)$, $ind(U(H)) = H$. The other way around, the selector of a subgraph induced by some selector P contains P, i.e., $P \subseteq U(ind(P))$. As a consequence, we get for each selector P of G and for each $H \in Sub(G)$ that $P \subseteq U(H)$ if and only if $ind(P) \in Sub(H)$.

Some special kinds of subgraphs are used in this paper.

A node v of a graph G induces two subgraphs of G: $In(v) = (\{v\} \cup \{v' \mid (v', v, x) \in E_G, v \neq v', x \in \Sigma_G\}, \Sigma_G, \{(v', v, x) \in E_G \mid v' \in V_G, v \neq v', x \in \Sigma_G\})$, and $Out(v) = (\{v\} \cup \{v' \mid (v, v', x) \in E_G, v \neq v', x \in \Sigma_G\}, \Sigma_G, \{(v, v', x) \in E_G \mid v' \in V_G, v \neq v', x \in \Sigma_G\})$. The graphs $In(v)$ and $Out(v)$ are called the *in-neighborhood graph* and the *out-neighborhood graph* of v, respectively. The set of nodes of $In(v)$ consists of v and its *in-neighbors*, and its set of edges consists of the *incoming edges* of v. Dually, the set of nodes of $Out(v)$ consists of v and its *out-neighbors*, and its set of edges consists of the *outgoing edges* of v. Moreover, a subgraph consisting of a single node v and a single edge (v, v, x) for some $x \in \Sigma$ is denoted by $loop(v, x)$.

Finally, we use $\mathbb{N}$ to denote the set of natural numbers, $\mathbb{N}^+$ to denote the set of positive integers, and for each set (alphabet) Σ, we use Σ^* to denote the set of sequences (words) over Σ including the empty word λ. For each $w \in \Sigma^*$, $|w|$ denotes the length of w.

3. Reaction systems on graphs

In this section, the basic notions and notations of reaction systems on graphs are introduced. While the original notion of reaction systems is purely set-theoretic, in this paper it is carried over from sets and subsets to graphs and subgraphs. Let us first briefly recall some notions concerning set-based reaction systems.

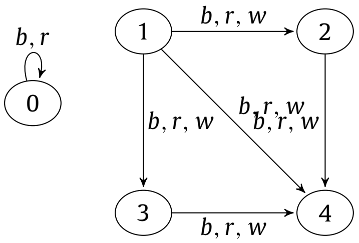
A *set-based reaction system* $\mathcal{A} = (S, A)$ consists of a finite *background set* S and a finite set of *reactions* A each of which is of the form $b = (X, Y, Z)$, where X, Y, Z are non-empty subsets of S such that $X \cap Y = \emptyset$. The components X, Y, Z are called the sets of *reactants*, *inhibitors*, and *products* of b, respectively. *States* (of $\mathcal{A}$) are subsets of the background set S. We say that b is *enabled* by a state T if $X \subseteq T$ and $Y \cap T = \emptyset$. The *result of b on T* equals Z if b is enabled on T, and equals $\emptyset$ otherwise. For a set of reactions A', the *result of A' by T* is the union of the results of reactions in A' which are enabled by T.

In graph-based reaction systems, the background set is replaced by a background graph, and the states and the components of reactions are replaced by subgraphs with one exception. The inhibitor of a reaction is an ordered pair consisting of a set of forbidden nodes and a set of forbidden edges, where, in general, some of the sources and targets of the forbidden edges do not have to belong to the set of forbidden nodes. Therefore, we define inhibitors as selectors, which leads to the following notion of a (graph-based) reaction.

Definition 1 *(Reaction).* Let B be a non-empty graph. A *reaction* over B is a triple $b = (R, I, P)$, where R and P are non-empty subgraphs of B and I is a selector of B such that $I \cap U(R) = (\emptyset, \emptyset)$.

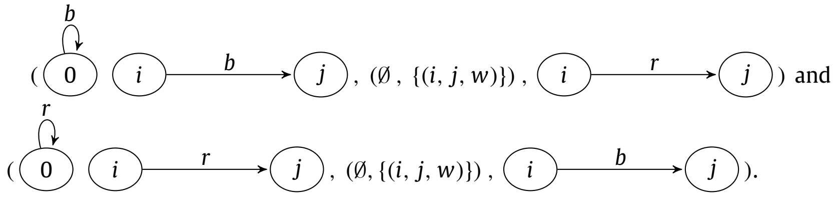
We say that R is the *reactant graph of b*, I is the *inhibitor of b*, and P is the *product graph of b*. Also, b is called *uninhibited* if $I = (\emptyset, \emptyset)$. We use the notations R_b, I_b, and P_b to denote R, I, and P, respectively.

Example 1. As an illustrating example, consider the graph $B_{xmpl} = (V_{xmpl}, \Sigma_{xmpl}, E_{xmpl})$ with $V_{xmpl} = \{0, 1, 2, 3, 4\}$, $\Sigma_{xmpl} = \{b, r, w\}$, and $E_{xmpl} = \{(i, j, c) \mid i, j \in \{1, 2, 3, 4\}, i < j, (i, j) \neq (2, 3), c \in \{b, r, w\}\} \cup \{(0, 0, c) \mid c \in \{b, r\}\}$. Hence B_{xmpl} is of the following form:



where an arrow with a list of labels represents parallel edges with the given labels in the list.

The set of reactions A_{xmpl} consists of the following two reactions over B_{xmpl} for each pair of nodes $i, j \in \{1, 2, 3, 4\}$ with $i < j$, $(i, j) \neq (2, 3)$:



b
(0 i b j , (∅ , {(i, j, w)}) , i r j) and
r
(0 i r j , (∅, {(i, j, w)}) , i b j).

Thus a reaction of the first type replaces a b-edge by an r-edge provided that there is a b-loop on the node 0 and no parallel w-edge. A reaction of the second type replaces an r-edge by a b-edge provided that there is an r-loop on the node 0 and no parallel w-edge.

Next we formalize the application of a reaction and of a set of reactions to a state, which now is a graph (a subgraph of the "universe" graph B).

Definition 2 *(Enabled reaction, result).* Let B be a non-empty graph over the set of labels Σ, and let T be a subgraph of B.

1. A reaction $b = (R, I, P)$ over B is *enabled* by T, denoted by $en_b(T)$, if $R \in Sub(T)$ and $I \cap U(T) = (\emptyset, \emptyset)$.
2. The *result* of a reaction b on T, denoted by $res_b(T)$, is defined by $res_b(T) = P_b$ if $en_b(T)$ and $res_b(T) = \emptyset_\Sigma$ otherwise.
3. The *result* of a set of reactions A over B on T, denoted by $res_A(T)$, is defined by: $res_A(T) = \bigcup_{b \in A} res_b(T)$.

Thus, for a given B, a reaction b over B induces the function $res_b : Sub(B) \to Sub(B)$ such that, for each $T \in Sub(B)$, $res_b(T)$ is defined as above. Similarly, a set of reactions A over B induces the function $res_A : Sub(B) \to Sub(B)$ such that, for each $T \in Sub(B)$, $res_A(T)$ is defined as above. In particular, if A is a singleton set, $A = \{b\}$, then $res_A = res_b$.

Since, for $T \in Sub(B)$ and $b \in A$ such that b is not enabled on T, $res_b(T) = \emptyset_\Sigma$, one can redefine res_A as follows: for each $T \in Sub(B)$, $res_A(T) = \bigcup\{res_b(T) \mid b \in A \text{ and } en_b(T)\}$. Clearly, $res_A(T)$ is a subgraph of B.

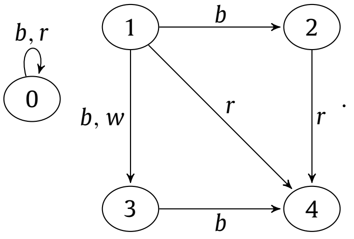
We are ready now to define the notion of a graph-based reaction system.

Definition 3 *(Graph-based reaction system).* A *graph-based reaction system* is a pair $\mathcal{A} = (B, A)$, where B is a finite non-empty graph, called the *background graph* of $\mathcal{A}$, and A is a set of reactions over B.

States of $\mathcal{A}$ are the subgraphs of B. Also, $\mathcal{A}$ induces the function $res_\mathcal{A} : Sub(B) \to Sub(B)$, called the *result function of* $\mathcal{A}$, defined by: for each state $T \in Sub(B)$, $res_\mathcal{A}(T) = res_A(T)$.

Example 2. Together, the graph B_{xmpl} and the set of reactions A_{xmpl} specify a graph-based reaction system $\mathcal{A}_{xmpl} = (B_{xmpl}, A_{xmpl})$. Let T_0 be the following state graph of $\mathcal{A}_{xmpl}$

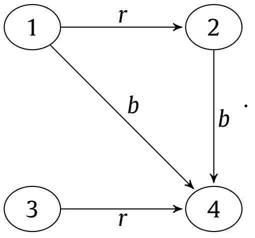
$T_0 =$



.

The application of the reactions of $\mathcal{A}_{xmpl}$ yields the following result graph

$res_{\mathcal{A}_{xmpl}}(T_0) =$



.

Thus a graph-based reaction system $\mathcal{A} = (B, A)$ is basically a set of reactions A over B; unless explicitly clear otherwise, we assume that $A \neq \emptyset$. In specifying $\mathcal{A}$, we also specify its background graph B which is a sort of a "universe" of $\mathcal{A}$, as

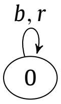
(1) for each reaction $b \in A$ both R_b and P_b are subgraphs of B, and $I_b = (X, Y)$ is such that $X \subseteq V_B$ and $Y \subseteq E_B$, and
(2) B determines the space where all the dynamic processes in $\mathcal{A}$ (defined below) take place.

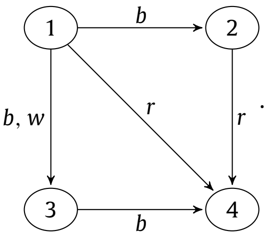
The dynamic processes associated with $\mathcal{A}$, which determine graph transformations specified by $\mathcal{A}$, are defined as follows.

Definition 4 *(Interactive process).* Let $\mathcal{A} = (B, A)$ be a graph-based reaction system.

1. An *interactive process* in $\mathcal{A}$ is an ordered pair $\pi = (\gamma, \delta)$ such that γ, δ are finite sequences of subgraphs of B, $\gamma = C_0, \ldots, C_n$ and $\delta = D_0, \ldots, D_n$, for some $n \in \mathbb{N}^+$, such that $D_i = res_\mathcal{A}(C_{i-1} \cup D_{i-1})$ for $i = 1, \ldots, n$. The sequence γ is the *context sequence of* π, denoted by $cons(\pi)$, the sequence δ is the *result sequence of* π, denoted by $ress(\pi)$, and the sequence $\tau = T_0, \ldots, T_n$ with $T_i = C_i \cup D_i$ for $i = 0, \ldots, n$ is the *state sequence of* π, denoted by $sts(\pi)$.
2. If the context sequence γ is such that $C_i \in Sub(D_i)$ for $i = 0, \ldots, n$, then π is *context-independent*.

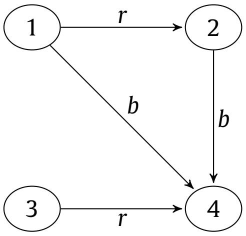
Example 3. Consider the graph-based reaction system $\mathcal{A}_{xmpl} = (B_{xmpl}, A_{xmpl})$ and let π be the interactive process determined by the context sequence $cons(\pi) = C_0, \ldots, C_n$, for some $n \geq 2$, such that

$C_0 = \cdots = C_n =$ 

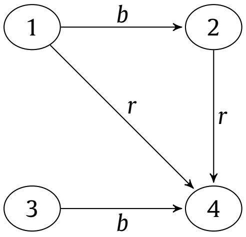
and the initial result graph $D_0 =$ 

.

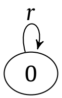
Then the second result graph is

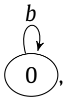
$D_1 = res_{\mathcal{A}_{xmpl}}(C_0 \cup D_0) =$ 

and the third result graph is

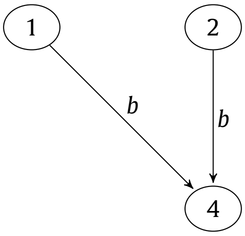
$D_2 = res_{\mathcal{A}_{xmpl}}(C_1 \cup D_1) =$ 

From then on, the consecutive result graphs alternate between D_1 and D_2, hence $ress(\pi) = D_0, D_1, D_2, D_1, D_2, \ldots$.

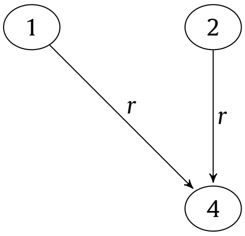
Alternatively, consider the interactive process π' with the context sequence $cons(\pi') = C'_0, \ldots, C'_n$, for some $n \geq 2$, where the context graphs with an even index are 

and the ones with an odd index are 

, while the initial result graph D_0 remains the same. Then the second result graph is

$D'_1 =$ 

and the third result graph is

$D'_2 =$ 

From then on, the consecutive result graphs alternate between D'_1 and D'_2, hence $ress(\pi') = D_0, D'_1, D'_2, D'_1, D'_2, \ldots$.

As the third and final example, consider a context-independent process of a length greater or equal than 2. Then the first result graph is not empty if and only if the initial result graph contains the node 0 with at least one of its two possible loops and there is an edge with the same label as the loop without a parallel w-edge. Hence, since the first result graph does not contain the node 0, the second and all following result graphs must be empty. This illustrates the fact that context graphs that are not subgraphs of their corresponding result graphs can be significant.

Intuitively, one obtains the state sequence of π by an iterative procedure, beginning with the initial state $T_0 = C_0 \cup D_0$. For each $i = 0, \ldots, n-1$, the successor state T_{i+1} of the current state T_i is defined as $res_A(T_i) \cup C_{i+1}$. Note that the context sequence together with the initial result graph D_0 determines π, as they together determine the result sequence δ. Then π is context-independent if the context sequence does nor really contribute to the state sequence: each context graph C_i is a subgraph of the result graph, and so the state sequence is identical to the result sequence. In particular, if each C_i is the empty graph, then we deal with a "pure form" of a context-independent process, referred to as an *empty-context interactive process*.

Since, for each $i = 0, \ldots, n-1$, the successor state T_{i+1} equals $res_A(T_i) \cup C_{i+1}$, a node or an edge of T_i is *sustained*, i.e., also present in T_{i+1}, only if:

- it is either produced by a reaction from A enabled by T_i, or
- it is contributed by the context graph C_{i+1}.

In this sense, each successor state (graph) T_{i+1} of the current state T_i is a "new" graph. If π is context-independent, then T_{i+1} is produced by res_A, i.e., $T_{i+1} = res(T_i)$, and so T_{i+1} is the union of product graphs of all reactions from A which are enabled by T_i.

Since the state sequence $\tau = T_0, \ldots, T_n$ of π is a sequence of subgraphs of B, the consecutive steps of π define a trajectory $T_0, \ldots, T_n$ of subgraphs of B. In other words, following π through its state sequence, we *surf* on B moving from a subgraph of B to a subgraph of B. Therefore, π can be seen as a process of consecutive graph transformations beginning with T_0 and leading to T_n, and so we deal here with graph transformations determined by *graph surfing*.

Note that set-based reaction systems can be seen as a special cases of graph-based reaction systems. For a set-based reaction system $\mathcal{A} = (S, A)$, the corresponding graph-based reaction system $\mathcal{B}(\mathcal{A}) = (B, F)$ is constructed as follows:

- B is the discrete graph with $V_B = S$.
- For each set-based reaction $a = (X, Y, Z)$ from A, let $b_a = (R_X, I_Y, P_Z)$ be the graph-based reaction such that
  - R_X is the discrete graph with X as its set of nodes,
  - $I_Y = (Y, \emptyset)$, and
  - P_Z is the discrete graph with Z as its set of nodes.
- $F = \{b_a \mid a \in A\}$.

The question whether graph-based reaction systems can be transformed into set-based reaction systems in a reasonable way is an open problem. As the number of subgraphs of a graph is not a power of 2 in general while the number of subsets of a set with n elements is 2^n, there cannot be a transformation such that the states are in a one-to-one correspondence. But there may be some kind of a more sophisticated embedding.

Clearly, a graph can be represented by a set: for a given alphabet Σ of edge labels, a graph $G = (V, \Sigma, E)$ can be represented by the set $S_G = V \cup E$. Therefore, graph-based reaction systems can be, in principle, simulated by (transformed into) set-based reaction systems. However, this seems to be quite involved as can be seen as follows.

Let $\mathcal{A} = (B, A)$ be a graph-based reaction system and $\mathcal{B}$ be a set-based reaction system with background set S_B (defined as above). However, not all subsets of S_B qualify as states of $\mathcal{B}$, because not all subsets of S_B represent graphs (e.g., a subset T of S_B may contain a triplet (u, v, x), but either u or v, or even both u and v, may not be included in T). Thus we would have to consider as states of $\mathcal{B}$ only *consistent* subsets of S_B, i.e., the subsets of S_B which represent graphs.

Even with this choice, the situation is "not stable" from an "operational" point of view as, e.g., the set of consistent subsets is not closed with respect to the difference of sets. Then, in considering the interactive processes of $\mathcal{B}$, one cannot allow all subsets of S_B to be admitted as contexts, but rather one would have to restrict context sequences to consist only of consistent subsets of S_B. Altogether, the simulation of graph-based reaction systems by set-based reaction systems may turn out to be really complex and requiring restrictions not natural from the point of view of the set-based framework.

The choice of selectors (rather than subgraphs) as inhibitors is significant and desirable. Having a subgraph as inhibitor would require that forbidding an edge would imply also forbidding its source and target nodes, which is too restrictive. On the other hand, choosing reactants and products as subgraphs seems to be a good choice. As pointed out in the preliminaries, a selector is included in the selector extracted from a subgraph if and only if its induced subgraph is included in the given subgraph. Therefore, it does not make any difference whether reactants are chosen as subgraphs or selectors. However, if one would define products as selectors, then the union of the products of all enabled reactions would not always be a subgraph so that one would have to define a successor state as the induced subgraph of this union. But this is equal to the union of the induced subgraphs of the involved selectors. Therefore, by using subgraphs to define products we avoid an additional subgraph induction.

The framework of graph-based reaction systems constitutes a novel approach to graph transformation, which differs in number of ways from the approaches studied in a whole spectrum, see, e.g., [22], of other approaches, such as, e.g., the double-pushout approach, the single-pushout approach, hyperedge replacement, node replacement, and the Progres approach.

Although constructions of rule application within different approaches differ from each other, they also share some basic principles. All of them deal with abstract graphs meaning that the results of rule applications are uniquely determined only up to an isomorphism. Their rules consist of left-hand sides and right-hand sides (besides other components in some cases) and a rule is applied to a host graph by matching the left-hand side with a subgraph of the host graph through a graph morphism, cutting off the match or a specific part of it, adding the right-hand side, and then pasting it properly with the remainder of the host graph. In some cases, additional application conditions are employed, such as, e.g., positive and negative context conditions. The semantics is based on sequences of rule applications (usually called derivations).

In contrast, a graph-based reaction system works on concrete graphs and subgraphs. Given a state, a reaction adds its product to the successor state provided that its reactant graph is a subgraph of the host state and its inhibitor (node and edge sets) is absent. This means that a reactant graph specifies a positive context condition and an inhibitor a negative one. But, there is no matching and no extra pasting. Moreover, in the transformation leading to the successor state the whole host state disappears so that the cutting off parts of the current graph (state) is not needed. The semantics is given by interactive processes. In the case of context-independent processes, their state sequences correspond to traditional derivations. But, in the general case, interactive processes describe interactions between a state and a context (which represents the environment).

Another significant difference concerns the state space. While the number of states reachable from an initial state is obviously finite in a graph-based reaction systems as there are only finitely many states, in traditional approaches the number of graphs derivable from an initial graph is, in general, infinite.

Finally, most graph transformation approaches provide some notion of parallelism, where, in general, rule applications can be in conflict with each other, so that they cannot be performed in parallel. In contrast, in graph-based reaction systems, enabled reactions are always independent of each other and take place simultaneously.

The basic concepts of graph-based reaction systems introduced in this section are employed and illustrated in a variety of ways in the four case studies presented in the sequel of the paper. In the modeling of finite automata (Section 5) and cellular automata (Section 7), the reactions are uninhibited. The modeling of finite automata works properly because the relevant interactive processes have context sequences corresponding to the infinitely many input strings. Context-independent processes are considered in the three other case studies. Another interesting aspect of our case studies is their parameterization. For example, a single background graph of a graph-based reaction systems in Section 4 in the context of the Sierpinski triangle depends on a natural number n that specifies the number of nodes $(n^2 + 5n)/2$. This is only a small fraction of the infinite Sierpinski triangle which is fully covered if one considers the family of all respective graph-based reaction systems. Analogously, graph-based reaction systems modeling the shortest path algorithms (Section 6) work for one concrete graph and all its subgraphs. To cover all graphs, one should consider the family of parameterized graph-based reaction systems, for instance, by the complete graphs of any size. Moreover, a cellular automaton has infinitely many cells in general while a modeling graph-based reaction systems considers only a finite number of cells of interest. To cover the whole cellular automaton, one must consider the family of corresponding graph-based reaction systems with monotonously growing sets of cells of interest.

We end this section by summarizing the methodological setup of reaction systems, which are a model of interactive computation. Here the system which interacts with the environment is given by the set of reactions A, the (behavior of) the environment is given by sequences of contexts, and the interaction between the system and the environment is given by interactive processes. This interaction is realized in such a way that the states of the interactive processes result from combining the internal behavior of the system (expressed by the result function) with the behavior of the environment (expressed by the contexts). The behavior of the system itself (determined just by the result function) is essentially finite as each state sequence converts eventually to a loop. However, in the general case, as expressed by the state sequences of interactive processes, the behavior is genuinely infinite.

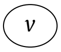
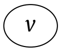
When we deal with the set-based reaction systems, then the states are sets (subsets of the background set), the reactions transform sets into sets, and contexts are sets. In this paper, we deal with graph-based reaction systems, where states are graphs (subgraphs of the background graph), the reactions transform graphs into graphs, and contexts are graphs.

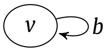
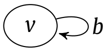
4. Approximating the Sierpinski triangle

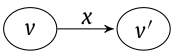
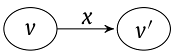
In this section, we demonstrate how to simulate the building principle of Sierpinski triangles (cf., e.g., [21]).

Consider the background graph $B = (V, \{b, S, W\}, E)$ for some finite set V of nodes such that E contains the subset $E(V) = \{(v, v, b) \mid v \in V\}$ of b-loops at each node and each other edge $e \in E \setminus E(V)$ is of the form (v, v', X) with $v \neq v'$ and $X \in \{S, W\}$, i.e., all other edges connect two nodes and are either S-labeled or W-labeled. We also assume that B is *strongly simple* meaning that there is at most one edge between any two nodes. Then we consider the following three kinds of reactions:

1. For each node, each loop, and each non-loop edge, there are the following uninhibited reactions:

   (a) (, $(\emptyset, \emptyset)$, ) for all $v \in V_B$,

   (b) (, $(\emptyset, \emptyset)$, ) for all $v \in V_B$ and $b \in \Sigma_B$, and

   (c) (, $(\emptyset, \emptyset)$, ) for all $(v, v', x) \in E_B$.

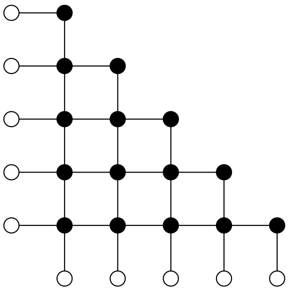


Fig. 1. The background graph TRI(5).

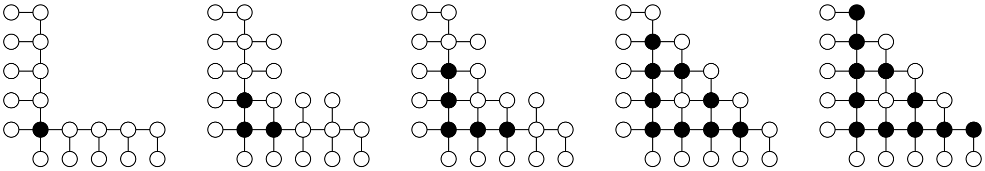
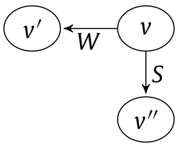
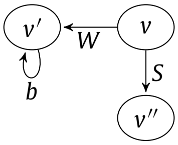
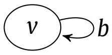


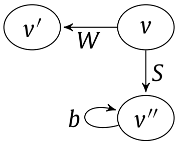
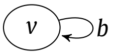
Fig. 2. The state sequence of the context-independent interactive process with the initial state INIT(5).

2. For each "angle" subgraph that consists of three nodes v, v', and v'' together with a W-edge from v to v' and an S-edge from v to v'', there is the following reaction:

( , $(\emptyset, \emptyset)$, $In(v)$).

3. For each angle as above, there are also the following reactions:

( , $(\emptyset, \{(v'', v'', b)\})$, ), and

( , $(\emptyset, \{(v', v', b)\})$, ).

Given a state $T \subseteq B$, the first kind of reactions applies to every node and edge of T that will be sustained ensuring that T is a subgraph of its successor. The second kind of reactions closes the source nodes of angles under their in-neighborhoods. And due to the third kind of reactions, a b-loop is added to the source node if exactly one of the target nodes of the angle has a b-loop. Otherwise, no loop is added.

To see how the interactive processes work, we consider the graph TRI(n) for some $n \in \mathbb{N}^+$ as the background graph B. It has the points of the plane (i, j) with integer coordinates $i, j > 0$ and $i + j \leq n + 1$ as well as the points $(i, 0)$ and $(0, j)$ for $i, j = 1, \ldots, n$ as nodes. Then each node (i, j) with $i, j \geq 1$ is the source of three edges: one edge with label W and target $(i - 1, j)$ (the *Western neighbor*), one edge with label S and target $(i, j - 1)$ (the *Southern neighbor*), and one loop with label b. Accordingly, the node (i, j) is called the *Northern neighbor* of the node $(i, j - 1)$ and the *Eastern neighbor* of $(i - 1, j)$.

For example, TRI(5) is the graph depicted as in Fig. 1, where a node with loop is represented by $\bullet$, a node without loop by $\circ$, and the directions and labels of edges are omitted because they are clear from the geometric situation.

We will consider context-independent interactive processes with the initial state INIT(n) $\in Sub(TRI(n))$ for some $n \in \mathbb{N}^+$, where $V_{INIT(n)} = \{(i, j) |$ either $i = 0, 1$ & $j = 1, \ldots, n$ or $i = 1 \ldots, n$ & $j = 0, 1\}$ and $E_{INIT(n)} = \{((1, j), (0, j), W) \mid j = 1, \ldots, n\} \cup \{((i, 1), (i, 0), S) \mid i = 1, \ldots, n\} \cup \{(1, 1, b)\}$.

For example, beginning with INIT(5), one gets the state sequence depicted in Fig. 2.

From the fifth state on, the state sequence becomes stationary.

More generally, one can see that the context-independent interactive process that begins in INIT(n) for some $n \geq 2$ (with the background graph TRI(n)) behaves analogously. In the first step, the Northern neighbor of the node $(1, 1)$ and its Eastern neighbor get b-loops. Moreover, the in-neighborhoods of the nodes $(1, 2), \ldots, (1, n))$ and $(2, 1), \ldots, (n, 1)$ are

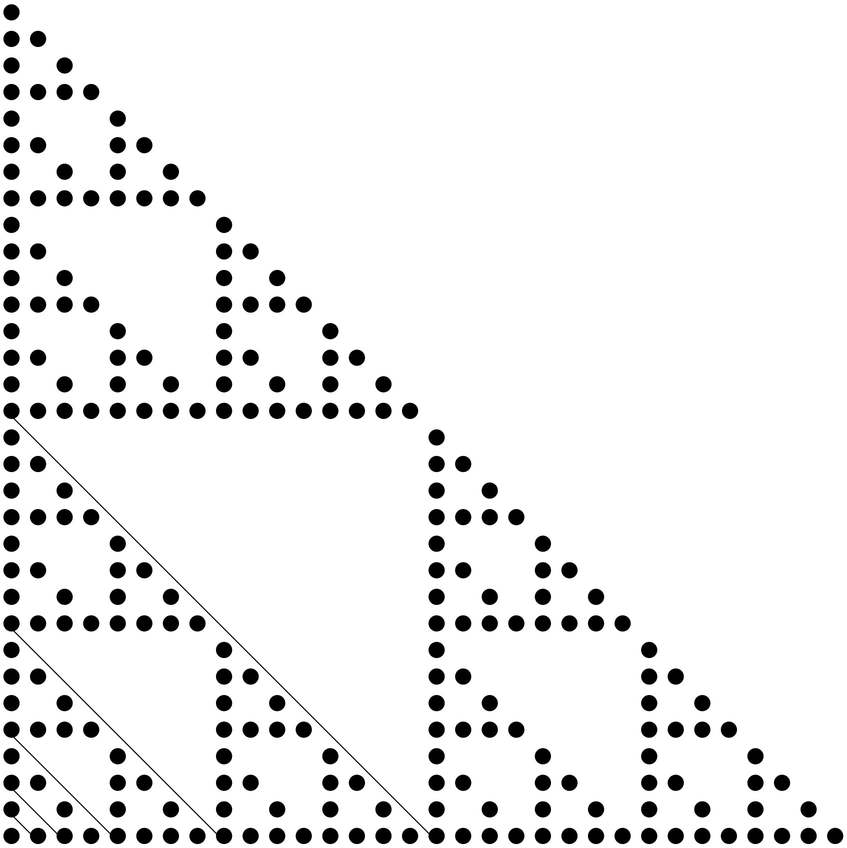


Fig. 3. The states STABLE(2^m) for $m = 0, \ldots, 5$.

added. The node $(2, 2)$ for $n \geq 3$ is connected to its Western and Southern neighbors so that the corresponding reaction of the second type is enabled. Each other new node (for $n \geq 4$) misses either a Western or a Southern neighbor so that it cannot be the source node of the reactions of the second type meaning that these reactions are not enabled. In step k, for some k with $1 \leq k < n$, only nodes on the diagonal connecting the nodes $(1, k+1)$ and $(k+1, 1)$, i.e., the nodes (i, j) with $i + j = k + 2$, can get new b-loops so that from the step n onwards all steps keep the reached state invariant. Therefore, it is called STABLE(n).

How such reaction systems and their interactive processes are related to the Sierpinski triangle can be best seen if one considers the background graphs TRI(2^m) for $m \in \mathbb{N}$, and the stable states STABLE(2^m). Fig. 3 shows the stable states for $m = 0, \ldots, 5$. To visualize the pictorial pattern, only the nodes with b-loops are drawn. The whole figure shows $STABLE(32)$. $STABLE(16)$, $STABLE(8)$, $STABLE(4)$, $STABLE(2)$, and $STABLE(1)$ can be seen under the diagonals, from right to left.

In general, one can show that the state STABLE(2^{m+1}) for each $m \in \mathbb{N}$ is “self-similar” in the sense that it consists of three copies of STABLE(2^m), where one copy is shifted by 2^m units up, another copy is shifted by 2^m units to the right, and the third copy stays where it is.

The example of the Sierpinski triangle indicates a way to overcome the limitation of the finiteness of the state space for a given background graph. While a single graph-based reaction system $SIER(n)$ for some $n \in \mathbb{N}$ covers only a small fraction of the infinite Sierpinski triangle, the family $\{SIER(2^k)\}_{k \in \mathbb{N}}$ approximates the whole Sierpinski triangle.

The Sierpinski triangle is one of the most famous self-affine fractals. But one encounters many more fractal patterns of a similar kind in the literature like the Hilbert curve, the dragon curve, the Koch tree, the Koch island, the devil’s staircase, etc. (see, e.g., [21]). We believe that these fractals can also be approximated by graph-based reaction systems analogously to the way we approximated the Sierpinski triangle above. Therefore, it may be worthwhile to have a closer look into the relation of graph-based reaction systems and fractal geometry.

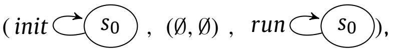
5. Simulating finite automata

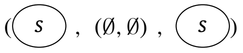
In this section, we deal with interactive processes that are not context-independent by considering a transformation of finite automata into graph-based reaction systems in such a way that the recognition of words by automata is modeled by interactive processes in graph-based reaction systems that run on the state graphs of the automata.

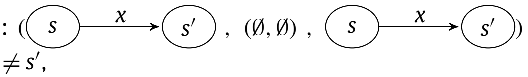
Let $\mathcal{F} = (Q, \Sigma, \mu, s_0, F)$ be a *finite automaton* with the set of *states* Q, the set of *input symbols* Σ, the *state transition relation* $\mu \subseteq Q \times Q \times \Sigma$, the *initial state* $s_0 \in Q$, and the set of *final states* $F \subseteq Q$. Then the corresponding graph-based reaction system $\mathcal{A}(\mathcal{F}) = (B(\mathcal{F}), A(\mathcal{F}))$ is constructed as follows.

The background graph extends the state graph of $\mathcal{F}$ by a *run*-loop at each node and an additional node with an x-loop for each input symbol x. Formally, $B(\mathcal{F}) = (Q \cup \{input\}, \Sigma \cup \{init, fin, run\}, E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5)$ with $E_1 = \mu$, $E_2 = \{(s_0, s_0, init)\}$, $E_3 = \{(s'', s'', fin) \mid s'' \in F\}$, $E_4 = \{(s, s, run) \mid s \in Q\}$, and $E_5 = \{(input, input, x) \mid x \in \Sigma\}$. The subgraph resulting from removing the *run*-loops and the node *input* with its loops is the *state graph* of $\mathcal{F}$, denoted by $gr(\mathcal{F})$, while $gr(\mathcal{F})^-$ denotes the state graph without the *init*-loop. The node *input* with its loops represents the input alphabet. The *run*-loops are used in the interactive processes.

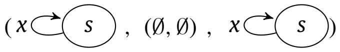
The set of reactions $A(\mathcal{F})$ consists of seven types of uninhibited reactions given below:

1. replacing the *init*-loop by the *run*-loop: 

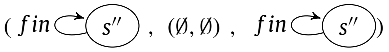
2. sustaining nodes of the state graph:  for all $s \in Q$,

3. sustaining transition edges: 

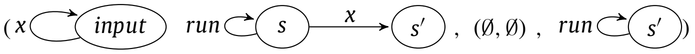
for all $(s, s', x) \in \mu$ with $s \neq s'$,

4. sustaining transition loops: 

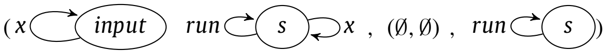
for all $(s, s, x) \in \mu$,

5. sustaining final-state loops: 

for all $s'' \in F$,

6. moving along transition edges due to input symbol:


for all $(s, s', x) \in \mu$ with $s \neq s'$,

7. staying at transition loops due to input symbol:


for all $(s, s, x) \in \mu$.

For each word $w = x_1 \dots x_n$ over Σ, with $n \geq 1$ and $x_i \in \Sigma$ for all $i = 1, \dots, n$, the *context sequence induced by* w is defined by

$$cin(w) = \emptyset_{\Sigma_{B(\mathcal{F})}}, loop(input, x_1), \dots, loop(input, x_n), \emptyset_{\Sigma_{B(\mathcal{F})}}.$$

Moreover, we set $cin(\lambda) = \emptyset_{\Sigma_{B(\mathcal{F})}}, \emptyset_{\Sigma_{B(\mathcal{F})}}$ for the empty word $\lambda \in \Sigma^*$. Note that, for each $w \in \Sigma^*$, $cin(w)$ begins and ends with $\emptyset_{\Sigma_{B(\mathcal{F})}}$ and $|cin(w)| = |w| + 2$. Obviously, this establishes a one-to-one correspondence between Σ^* and $cin(\Sigma^*) = \{cin(w) \mid w \in \Sigma^*\}$.

For a word $w \in \Sigma^*$, we denote by $\pi(w)$ the interactive process in $\mathcal{A}(\mathcal{F})$ such that $cin(w)$ is the context sequence and $gr(\mathcal{F})$ is the initial result of $\pi(w)$. In all steps of $\pi(w)$, the state graph without the *init*-loop $gr(\mathcal{F})^-$ is sustained due to the sustaining reactions of type 2 to 5, i.e., $gr(\mathcal{F})^-$ is a subgraph of all result graphs D_i for $i = 1, \dots, n+1$.

In the first step, the first reaction is enabled replacing the *init*-loop by a *run*-loop. From then on this reaction is not enabled ever again because the *init*-loop is never recreated. In each consecutive step $i+1$ of $\pi(w)$, for $i = 1, \dots, n$, the result graph D_{i+1} may contain some nodes with *run*-loops and it is accompanied by the context graph $loop(input, x_i)$ so that the *run*-reactions are enabled for each x_i-edge, and each x_i-loop with the *run*-nodes as sources. These reactions put *run*-loops at the targets.

In the case of the interactive process $\pi(\lambda)$ that has $cin(\lambda) = \emptyset_{\Sigma_{B(\mathcal{F})}}, \emptyset_{\Sigma_{B(\mathcal{F})}}$ as its context sequence and D_0, D_1 as its result sequence with $D_0 = gr(\mathcal{F})$, one gets $D_1 = gr(\mathcal{F})^- \cup loop(s_0, run)$, as only the first reaction is enabled by $gr(\mathcal{F})$ and its application replaces the *init*-loop by the *run*-loop.

To illustrate how these interactive processes look like, let us consider the automaton $\mathcal{F}_{xy^*z^*x}$ with the state graph given by the upper right graph D_0 in Fig. 4 and the context sequence $cin(xyzx)$.

The resulting interactive process is given in Fig. 4.

We will prove now that recognition processes in $\mathcal{F}$ correspond to specific interactive processes in $\mathcal{A}(\mathcal{F})$. This is stated in the following lemma.

Lemma 1. *Let $\mathcal{F} = (Q, \Sigma, \mu, s_0, F)$ be a finite automaton. Let $w \in \Sigma^*$ be such that $w = uv$, for some $u, v \in \Sigma^*$ with $|u| = i$. Let $D(u)$ be the $(i+1)$-th result graph of the interactive process $\pi(w)$. Then $D(u) = gr(\mathcal{F})^- \cup \bigcup_{s \in \mu^*(s_0, u)} loop(s, run)$.*

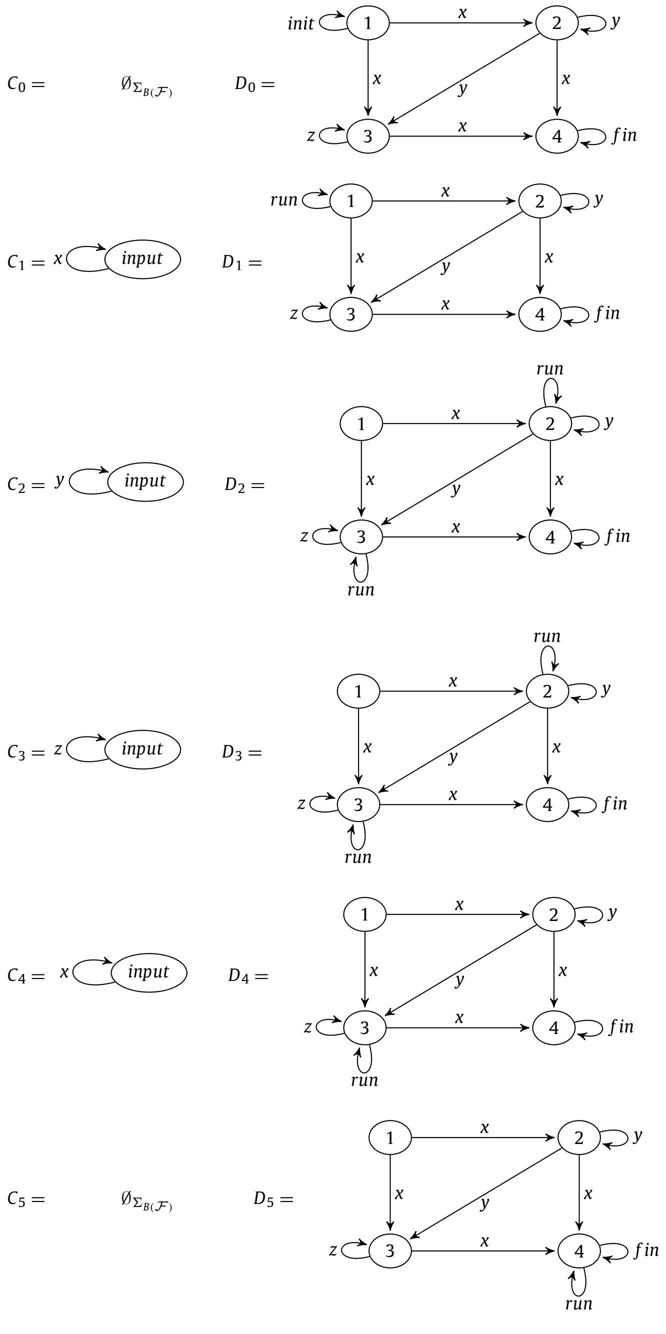


Fig. 4. An interactive process in $\mathcal{A}(\mathcal{F}_{xy^*z^*x})$.

Proof. The proof is by induction on i.

Induction base: Let $i = 0$, i.e., $u = \lambda$. Then $D(\lambda)$ is the result of the first step of $\pi(\lambda)$. Therefore, indeed, $D(\lambda) = gr(\mathcal{F})^- \cup loop(s_0, run)$. This proves the statement for $i = 0$ because $\mu^*(s_0, \lambda) = \{s_0\}$ by definition.

Induction step: Consider $w = uxv$ with $|u| = i$ and $|ux| = i + 1$. By the induction hypothesis, one gets $D(u) = gr(\mathcal{F})^- \cup \bigcup_{s \in \mu^*(s_0,u)} loop(s, run)$. And the successor result graph is

$$D(ux) = gr(\mathcal{F})^- \cup \bigcup_{s \in \mu^*(s_0,u)} \ \bigcup_{(s,s',x) \in \mu} loop(s', run)$$

as described above. This proves the statement for $i + 1$, because $\mu^*(s_0, ux) = \{s' \mid s \in \mu^*(s_0, u) \ \& \ (s, s', x) \in \mu \text{ for some } s \in Q\}$ by the definition of μ^*. □

This lemma allows one to relate the recognition of words by finite automata to certain interactive processes. Let $\mathcal{F} = (Q, \Sigma, \mu, s_0, F)$ be a finite automaton. For a $w \in \Sigma^*$, the interactive process $\pi(w)$ can be considered as accepting $w \in \Sigma^*$ if the final result graph $D(w)$ of $\pi(w)$ contains the subgraph $loop(s'', run) \cup loop(s'', fin)$ for some $s'' \in F$, meaning that some *run*-loop ends up in a final state. Accordingly, the *accepted language* of $\mathcal{A}(\mathcal{F})$ can be defined as

$$L(\mathcal{A}(\mathcal{F})) = \{w \in \Sigma^* \mid (loop(s'', run) \cup loop(s'', fin)) \in Sub(D(w)) \text{ and } s'' \in F\}.$$

It follows then by Lemma 1 that the recognized language of a finite automaton and the accepted language of the corresponding graph-based reaction system coincide.

Theorem 1. *Let $\mathcal{F}$ be a finite automaton and $\mathcal{A}(\mathcal{F})$ the corresponding graph-based reaction system. Then $L(\mathcal{F}) = L(\mathcal{A}(\mathcal{F}))$.*

Proof. Let $\mathcal{F} = (Q, \Sigma, \mu, s_0, F)$ be a finite automaton. For $w \in \Sigma^*$, $w \in L(\mathcal{F})$ if and only if $\mu^*(s_0, w) \cap F \neq \emptyset$. By Lemma 1, $D(w)$ contains the subgraph $loop(s'', run)$ for every $s'' \in \mu^*(s_0, w)$. Therefore, $w \in L(\mathcal{A}(\mathcal{F}))$ if and only if some node $s'' \in \mu^*(s_0, w)$ carries a fin-loop, i.e., $s'' \in F$. Therefore, $w \in L(\mathcal{F})$ if and only if $w \in L(\mathcal{A}(\mathcal{F}))$. Consequently, the theorem holds. □

6. Computing shortest paths

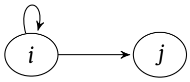
Graph-based reaction systems can be used to check graph properties and to compute functions on graphs. In this section, we present two families of graph-based reaction systems that compute the lengths of shortest paths in different ways.

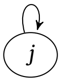
6.1. Parallel traversal

The first family $TRAV(n) = (B_{TRAV(n)}, A_{TRAV(n)})$ for $n \in \mathbb{N}^+$ of graph-based reaction systems computes the lengths of shortest paths from the given initial node to all other nodes of a given graph, which are reachable by paths from the initial node, by a parallel traversal of all paths starting in the initial node.

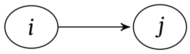
The background graph is the complete unlabeled directed graph with n nodes $B_{TRAV(n)} = ([n], \{*\}, [n] \times [n] \times \{*\})$ with the set of nodes $[n] = \{1, \ldots, n\}$ and all pairs of nodes as unlabeled edges, i.e., according to our convention, $*$ is the label of all edges.

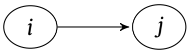
The set $A_{TRAV(n)}$ of reactions consists of two reactions for each pair $(i, j) \in [n] \times [n]$ of nodes with $i \neq j$:

1. (

, $(\emptyset, \{(j, j, *)\})$, 

), and

2. (

, $(\emptyset, \{(j, j, *)\})$, 

).

The first reaction moves a loop from the source of an edge to its target, provided that there is no loop on the target node. The second reaction sustains an edge together with its source and its target provided that the target carries no loop. Thus edges incoming to nodes with loops are not sustained. Loops are never sustained. A node is sustained only if it is the source or the target of a sustained edge. These reactions are a good illustration of the usefulness of using selectors (rather than subgraphs) as inhibitors.

What happens in context-independent interactive processes of this reaction system, is illustrated by the example in Fig. 5, where the illustrated process is represented by its state sequence. It starts with a state that is equipped with a single loop attached to the initial node 2. In the successor state resulting from the first step of the interactive process, the nodes 3, 4, and 5 have loops. Then, after the second step the nodes 6 and 7 have loops, after the third step, the nodes 8 and 9 have loops, and after the fourth step, the node 10 has a loop. In other words, the number of steps needed for a node to get a loop corresponds to the length of the shortest path from the initial node 2 to this node.

In general, one can prove the following result.

Theorem 2. *Let $\tau = T_0, \ldots, T_m$ for some $m \in \mathbb{N}^+$ be the state sequence of a context-independent interactive process in the reaction system $TRAV(n)$, for some $n \in \mathbb{N}^+$, where T_0 is a state with a single loop attached to node i_0. Then, for each $k = 1, \ldots, n$, node i of $B_{TRAV(n)}$ has a loop in state T_k for some k if and only if the length of a shortest path from i_0 to i equals k.*

Proof. By induction on k.

Induction base for $k = 0$: The initial node i_0 is the only node with a loop in T_0, and the empty path (the path of length 0), from i_0 to i_0 is the shortest path of length 0.

Induction hypothesis: The statement holds for all $l \leq k - 1$ for some $k \geq 1$.

Induction step: Let i be a node of T_k with a loop. Then it got its loop from a node j in T_{k-1} with a loop by applying a reaction of type 1 that moves the loop from j to i along the edge $(j, i, *)$. By induction hypothesis, a shortest path from i_0 to j has length $k - 1$. Extending this path by the edge (j, i), one gets a path from i_0 to i of length k. This must be a shortest

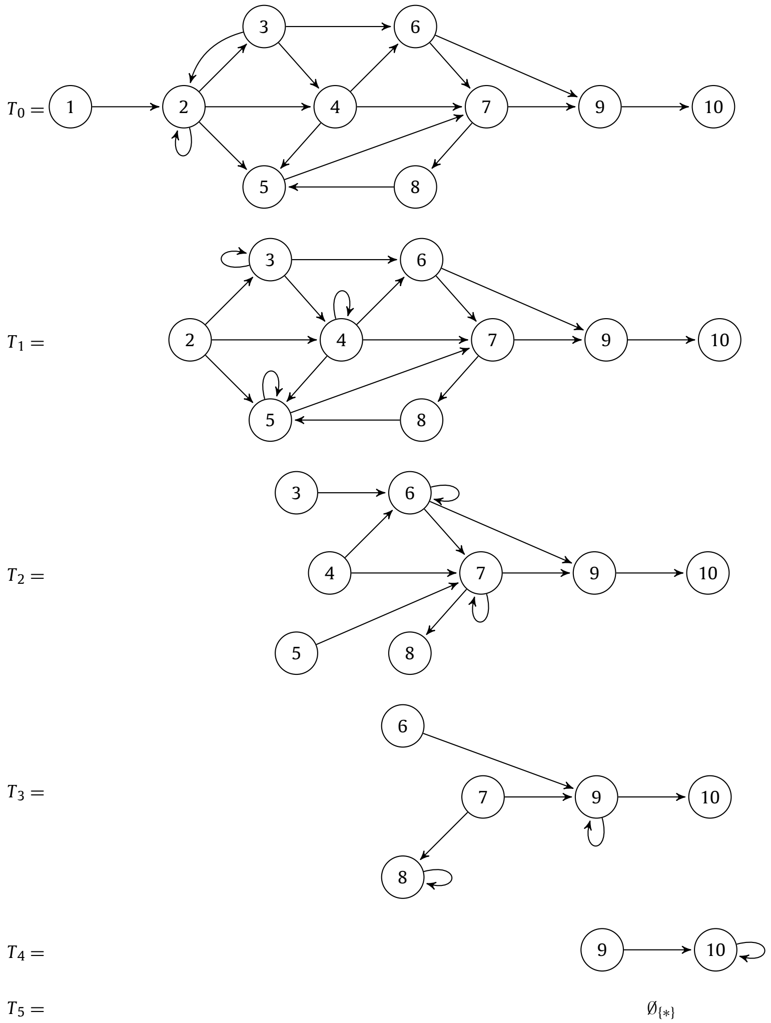


Fig. 5. The state sequence of a context-independent interactive process of *TRAV*(*10*).

path because otherwise i would acquire a loop earlier in the process (by induction hypothesis). But, due to the inhibitors of the reactions, no node gets a loop twice because none of the incoming edges of a node with loop is present in the successor state and no edges which are not loops are created. This completes the proof. □

6.2. Parallel summation

The second family $\mathrm{SUM}(n) = (B_{\mathrm{SUM}(n)}, A_{\mathrm{SUM}(n)})$, for $n \in \mathbb{N}^+$, of graph-based reaction systems computes the lengths of shortest paths from each node to each other node of a given graph by an iterated parallel summation.

The background graph is the complete loop-free directed graph with n nodes where there are n edges from each node to each other node that are labeled by $1, \ldots, n-1$ and $*$ respectively, i.e., $B_{\mathrm{SUM}(n)} = ([n], [n-1] \cup \{*\}, \{(i, j, x) \mid i, j \in [n], i \neq j, x \in [n-1] \cup \{*\}\})$.

The set $A_{\mathrm{SUM}(n)}$ consists of reactions of the following three types:

1. Relabeling unlabeled edges by 1:

   $(\ \textcircled{i} \longrightarrow \textcircled{j} \ , \ (\emptyset, \emptyset) \ , \ \textcircled{i} \xrightarrow{1} \textcircled{j} \ )$ for all $i, j \in [n], i \neq j$.

2. Sustaining all nodes and all edges labeled by labels from $[n-1]$:

   $(\ \textcircled{i} \ , \ (\emptyset, \emptyset) \ , \ \textcircled{i} \ )$ for all $i \in [n]$ and

   $(\ \textcircled{i} \xrightarrow{x} \textcircled{j} \ , \ (\emptyset, \emptyset) \ , \ \textcircled{i} \xrightarrow{x} \textcircled{j} \ )$

   for all $i, j \in [n]$ and $x \in [n-1]$.

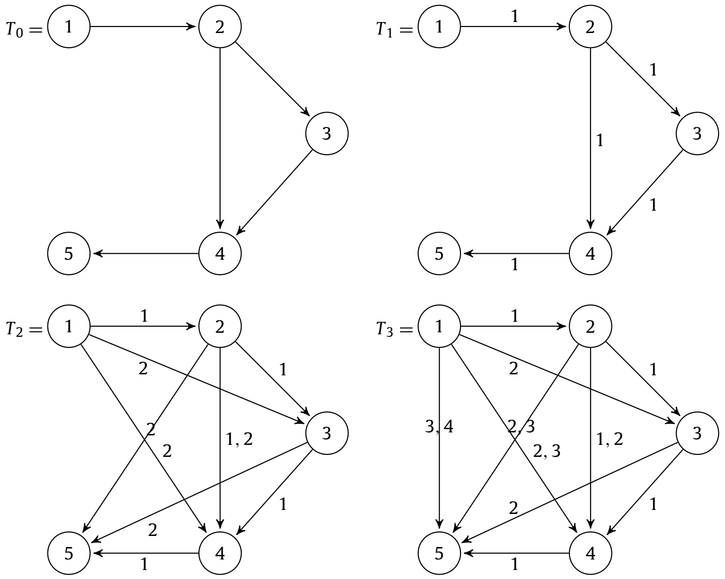


Fig. 6. The state sequence of a context-independent interactive process of SUM(5).

3. Summing up along consecutive edges:

$$\left(\ (i) \xrightarrow{x} (j) \xrightarrow{y} (k) \ , \ (\emptyset, \emptyset) \ , \ (i) \xrightarrow{x+y} (k) \ \right)$$

for all $i, j, k \in [n]$ and $x, y \in [n-1]$ with $x + y \leq n - 1$.

For an unlabeled edge, there is a reaction of the first type that can be applied to it, replacing it by a 1-edge, The second type of reactions sustains all nodes and all edges that are not unlabeled (i.e., not labeled by $*$). And the third type of reactions produces an edge with label $x + y$ whenever there are consecutive edges with labels x and y, respectively, as long as the sum of their labels is smaller than n.

What happens in context-independent interactive processes in these reaction systems, is illustrated by the example in Fig. 6, where the illustrated process is represented by its state sequence.

In T_2 and T_3, an edge with two labels represents two parallel edges with the respective labels. The state sequence starts with an unlabeled graph. In the successor state of the first step of the interactive process, the edges are relabeled by 1 by the reactions of the first type. In the second step, 2-edges are added connecting start and end nodes of consecutive 1-edges by the reactions of the third type. In the third step, 3-edges and 4-edges are added connecting start and end nodes of consecutive edges, where at least one edge is labeled by 2. In principle, this interactive process could go on, but it would become stationary as the labels are "bounded by" 4. It is easily seen that there is an l-edge from node i to node j in the final state T_3 if and only if there is a path of length $l \leq n - 1$ from i to j in the initial state T_0. The length of a shortest path from i to j (if there is one at all) is the smallest label of the edges from i to j in the final state T_3.

If the state sequence $\tau = T_0, \ldots, T_m$, for some $m \in \mathbb{N}^+$, of an arbitrary context-independent interactive process begins with an unlabeled graph, then the reactions of the first type the reactants of which contain edges which occur in T_0 are enabled, so that all edges of T_0 get the label 1. Moreover, all isolated nodes in T_0 are sustained. From then on, all nodes and edges are sustained by the reactions of the second type, as unlabeled edges are never produced. In addition, a new edge from node i to node k with label z is produced if there is a corresponding enabled reaction of the third type, This holds if, for some node j, there is an x-edge from i to j and a y-edge from j to k with $z = x + y$, as proved by the following lemma.

Lemma 2. *Let $T_0 \in Sub(B_{SUM(n)})$ be an unlabeled graph and $\tau = T_0, \ldots, T_m$, for some $m \in \mathbb{N}^+$, be the state sequence of a context-independent interactive process in SUM(n) for some $n \in \mathbb{N}^+$. Then the following holds for each $l = 1, \ldots, m$: $(i, k, z) \in E_{T_l}$ if and only if there is a path from i to k of length z in T_0 such that $1 \leq z \leq min(n - 1, 2^{l-1})$.*

Moreover, $T_l = T_{l+1}$ for all l such that $n - 1 \leq 2^{l-1}$.

Proof. The statement is proved by induction on l.

Induction base for $l = 1$: All edges of T_1 have label 1. For $n = 1$, there is neither an edge nor a path of length 1. For $n \geq 2$, $1 \leq z \leq min(n - 1, 2^{l-1}) = min(n - 1, 1) = 1$ implies that all paths are of length $z = 1$ and coincide with the edges.

Induction hypothesis: The statement holds for $l \geq 1$.

Induction step for $l+1$: Let $(i,k,z) \in T_{l+1}$. Then
(1) either $(i,k,z) \in T_l$ such that there is a path from i to k of length z with $1 \leq z \leq 2^{l-1} \leq 2^l$ by induction hypothesis, or
(2) (i,k,z) is produced by a reaction of the third type meaning that there are $(i,j,x), (j,k,y) \in E_{T_l}$ for some node j with $1 \leq x, y \leq min(n-1, 2^{l-1})$ and $x+y=z$. By induction hypothesis, there are paths from i to j of length x and from j to k of length y respectively. The sequential composition yields paths from i to k of length $x+y=z$ and $1 \leq x+y \leq min(n-1, 2^{l-1}+2^{l-1}=2^l)$.

Conversely, let p be a path from i to k of length z with $1 \leq z \leq min(n-1, 2^l)$ in T_0. Then either $z \leq 2^{l-1}$ such that there is a z-edge from i to k due to the induction hypothesis, or $z > 2^{l-1}$. Then p can be sequentially decomposed into a path p_1 from i to some intermediate node j of length 2^{l-1} and a path p_2 from j to k of length $z-2^{l-1} \leq 2^{l-1}$. Using the induction hypothesis, one gets a 2^{l-1}-edge from i to j and a $(z-2^{l-1})$-edge from j to k. Moreover, $1 \leq 2^{l-1}+z-2^{l-1} = z \leq min(n-1, 2^l)$. Therefore, the corresponding reaction of the third type is enabled and produces an edge from i to k with the label z in T_{l+1}. This completes the proof. □

Based on the lemma, one can prove a shortest-paths result. To formulate it, the following notation is used. Let $T \in Sub(B_{\mathrm{SUM}(n)})$ be a graph without unlabeled edges. Then the partial function $min_T : [n] \times [n] \to [n-1]$ yields, for each pair of nodes, the smallest label of the edges from the first node to the second node, i.e., $min(i,j) = min\{z \mid (i,j,z) \in E_T\}$ for all $i, j \in [n]$ with $i \neq j$, where $min(\emptyset)$ is undefined. The theorem states that the lengths of shortest paths are computed by certain context-independent processes with a number of steps that is logarithmic in the number of nodes of the initial graph.

Theorem 3. *Let $T_0 \in Sub(B_{SUM(n)})$ be an unlabeled graph and $\tau = T_0, \ldots, T_m$ for the smallest integer m with $n-1 \leq 2^{m-1}$ be the state sequence of a context-independent interactive process in SUM(n). Then the following holds for all nodes $i, k \in [n]$ with $i \neq k$ and defined $min_{T_m}(i,k)$: there is a shortest path from i to k of length z in T_0 if and only if $z = min_{T_m}(i,k)$.*

Proof. If there is a shortest path from i to k of length z, then, by Lemma 2, there is a z-edge from i to k in T_m. If we assume that there is an x-edge from i to k in T_m with $x < z$, then, by Lemma 2, there would have to be a path from i to k in T_m with length x, which contradicts the choice of z. Conversely, there is a $min_{T_m}(i,k)$-edge from i to k in T_m if $min_{T_m}(i,k)$ is defined. Then, by Lemma 2, there is a path of this length from i to k. This must be a shortest path because, by the definition of $min_{T_m}(i,k)$, no path from i to k can be shorter. □

7. Simulating cellular automata

The framework of cellular automata provides a quite old paradigm of computation with massive parallelism. The notion was introduced by von Neumann (see [20]) and has been intensively studied since then (see, e.g., [18]). In this section, we show how computations in cellular automata can be simulated by interactive processes of graph-based reaction systems.

7.1. Cellular automata

A *cellular automaton* is a system $\mathcal{C} = (COL, w, k, \mu, CELL, \mathcal{N})$, where COL is a finite set of *colors*, $w \in COL$ is a special *default color*, $k \in \mathbb{N}^+$ is the *neighborhood size*, $\mu : COL^{k+1} \to COL$ is a *transition function* subject to the condition $\mu(w, \ldots, w) = w$ for the default color w, $CELL$ is a set of *cells*, and $\mathcal{N} = \{N_i \mid i = 1, \ldots, k\}$ is the *neighborhood specification*, where each $N_i : CELL \to CELL$, for $i = 1, \ldots, k$, is the i-th *neighborhood function*. For each $v \in CELL$, $(N_1(v), \ldots, N_k(v))$ is the *neighborhood* of v subject to the condition that $N_i(v) \neq N_j(v)$ for all $i, j \in \{1, \ldots, k\}$ such that $i \neq j$.

Thus $\mathcal{C}$ consists of a set of cells $CELL$, that may be infinite, such that each cell v has k neighbor cells $N_1(v), \ldots, N_k(v)$. Moreover, there is a finite set of colors COL and a transition function μ that maps each selection of $k+1$ colors $c_0, \ldots, c_k$ to the color $\mu(c_0, \ldots, c_k)$. In the definition of a configuration given below, each cell gets a color in such a way that only a finite number of cells are not colored by the default color w. The triple (COL, COL^k, μ) may be interpreted as a finite transition system with the state set COL, the input alphabet COL^k, and the deterministic transition function μ. We speak about colors rather than states to avoid any confusion with the states of reaction systems. For each cell, the neighborhood specification $\mathcal{N}$ determines its k neighbors ordered from 1 to k.

A *configuration* of $\mathcal{C}$ is a function $\alpha : CELL \to COL$ such that the set of *active cells* $act(\alpha) = \{v \in CELL \mid \alpha(v) \neq w\}$ is finite. Although $CELL$ may be infinite, each configuration is finitely specified by its set of active cells.

Given a configuration α, one gets a uniquely determined *successor configuration* $\alpha' : CELL \to COL$ defined by: for each $v \in CELL$,

$$\alpha'(v) = \mu(\alpha(v), \alpha(N_1(v)), \ldots, \alpha(N_k(v))).$$

To get the new color of a cell, the transition function is applied to the current color of the cell and the colors of the neighbors. Note that, because a w-cell can only become active if some of its neighbors are active, for each configuration α, $act(\alpha')$ is finite, so that α' is a configuration. Such a transition from α to its successor α' is denoted by $\alpha \to \alpha'$.

Given configurations α and β, a *computation (in $\mathcal{C}$) from α to β of length $n \in \mathbb{N}^+$* is a sequence of configurations $\alpha_0, \alpha_1, \dots, \alpha_n$ such that $\alpha = \alpha_0$, $\beta = \alpha_n$, and $\alpha_0 \to \dots \to \alpha_n$. We write $\alpha \to^+ \beta$ if there is a computation from α to β of length n, for some $n \in \mathbb{N}^+$.

7.2. Related graph-based reaction systems

Cellular automata give rise to a special type of graph-based reaction systems that simulate their computational behavior. This requires a choice of proper background graphs and of sets of reactions as well as resolving the problem of potential infinity of the sets of cells.

Let $\mathcal{C} = (COL, w, k, \mu, CELL, \mathcal{N})$ be a cellular automaton.

For each finite subset $Z \subseteq CELL$, the *cells of interest*, a graph-based reaction system $\mathcal{A}(\mathcal{C}, Z) = (B(\mathcal{C}, Z), A(\mathcal{C}, Z))$ is defined as follows.

The background graph $B(\mathcal{C}, Z) = (V, \Sigma, E)$ is defined by

$$V = Z \cup \{N_i(v) \mid v \in Z, i = 1, \dots, k\},$$

$$\Sigma = COL \cup \{1, \dots, k\}, \text{ and}$$

$$E = \{(v, N_i(v), i) \mid v \in Z, i = 1, \dots, k\} \cup \{(v, v, c) \mid v \in Z, c \in COL\} \cup \{(v, v, w) \mid v \in V \setminus Z\}.$$

The set of reactions $\mathcal{A}(\mathcal{C}, Z)$ consists of the following uninhibited reactions (recall that $Out(v)$ denotes the out-neighborhood of v, see the formal definition at the end of Section 2):

$(Out(v) \cup loop(v, c_0) \cup \bigcup_{i=1}^{k} loop(N_i(v), c_i) \quad , \quad (\emptyset, \emptyset) \quad , \quad loop(v, \mu(c_0, c_1, \dots, c_k)))$
for all $v \in Z$ and $c_i \in COL$, $i = 0, \dots, k$,

$(Out(v) \quad , \quad (\emptyset, \emptyset) \quad , \quad Out(v))$ for all $v \in Z$, and

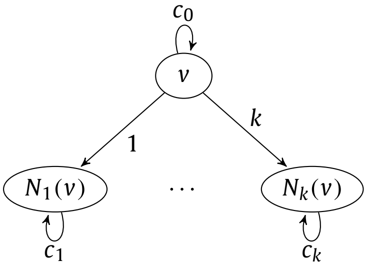
$(loop(v, w) \quad , \quad (\emptyset, \emptyset) \quad , \quad loop(v, w))$ for all $v \in V \setminus Z$.

Here is the intuition behind the formal definition of $\mathcal{A}(\mathcal{C}, Z)$.

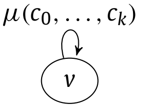
The set of nodes of the background graph consists of all cells of interest and all their neighbors. The edges that are not loops connect the cells of interest with their neighbors (directed from each cell of interest to its neighbors), where the labels reflect the order of the neighborhood nodes. Moreover, there are loops at each cell of interest, one for each color, while the cells that are not cells of interest carry a w-loop each.

There are three kinds of reactions, all uninhibited:

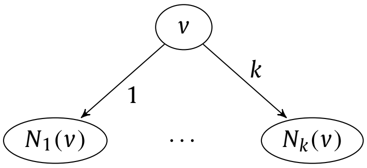
1. For each cell of interest $v \in Z$ and each selection $(c_0, c_1, \dots, c_k)$ of labels of loops on v and all its neighbors (where the label of each cell not in Z is w), there is exactly one uninhibited reaction such that its reactant graph is of the form:



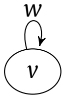
and its product graph is of the form:



2. For each cell of interest $v \in Z$, there is exactly one uninhibited reaction such that both its reactant graph and its product graph are of the form:



3. For each cell $v \in V \setminus Z$, there is exactly one uninhibited reaction such that both its reactant graph and its product graph are of the form:



To see how the reactions work, we consider well-formed states. A state $T \in Sub(B(\mathcal{C}, Z))$ is *well-formed* if T contains all nodes and all out-neighborhood edges of $B(\mathcal{C}, Z)$, and also one loop at each node. Such a state is determined by a function $lab : V_{B(\mathcal{C},Z))} \to COL$ that defines the labels of the loops at the nodes. We use T_{lab} to denote this state.

A step of an interactive process in $\mathcal{A}(\mathcal{C}, Z)$ applied to T_{lab} yields a well-formed state $T_{lab'}$. Each node $v \in Z$ and all of its out-neighbors have single loops, so that the reaction of the first kind with the fitting loop labels applies to the out-neighborhood of $v \in Z$, sustains v and produces a loop at v. The second kind of reactions sustains the out-neighborhood of v. The third kind of reactions sustains the w-loops of nodes that are not cells of interest. Altogether, the resulting state is well-formed with respect to the labeling function $lab' \colon V_{B(\mathcal{C},Z)} \to COL$ such that $lab'(v) = \mu(lab(v), lab(N_1(v), \dots, lab(N_k(v))$ for $v \in Z$ and $lab'(v) = w$ otherwise.

This consideration allows to relate the context-independent interactive processes in $\mathcal{A}(\mathcal{C}, Z)$ on well-formed states with the computations in the modeled cellular automaton. The formulation of this result makes use of the fact that each labeling function $lab : V_{B(\mathcal{C},Z)} \to COL$ can be extended to a configuration $\alpha(lab) : CELL \to COL$ defined by $\alpha(lab)(v) = lab(v)$ for $v \in V_{B(\mathcal{C},Z)}$ and $\alpha(lab)(v) = w$ otherwise.

Theorem 4. *Let $\mathcal{C} = (COL, w, k, \mu, CELL, \mathcal{N})$ be a cellular automaton and $Z \subseteq CELL$. Let $\mathcal{A}(\mathcal{C}, Z)$ be the reaction system defined by Z. Let $lab_0 : V_{B(\mathcal{C},Z)} \to COL$ be a labeling function and T_{lab_0} the well-formed state given by lab_0. Let $\alpha(lab_0) = \alpha_0 \to \alpha_1 \to \cdots \to \alpha_n$, for $n \in \mathbb{N}^+$, be a computation in $\mathcal{C}$ starting in a configuration that extends lab_0 in such a way that $act(\alpha_i) \subseteq Z$ for all $i = 1, \dots, n$. Let $T_{lab_0} \to \cdots \to T_{lab_n}$ be the state sequence of the corresponding interactive process. Then, $\alpha_i = \alpha(lab_i)$ for each $i = 1, \dots, n$.*

Proof. By induction on n.

We begin with the induction step. Consider $\alpha(lab_0) = \alpha_0 \to \alpha_1 \to \cdots \to \alpha_{n+1}$. Using the induction hypothesis for n, one gets $\alpha_i = \alpha(lab_i)$ for $i = 1, \dots, n$, where the labeling functions are given by the state sequence $T_{lab_0} \to \cdots \to T_{lab_n}$ of some interactive process. By the definition of a process step, $T_{lab_{n+1}}$ is specified by $lab_{n+1} : V_{B(\mathcal{C},Z)} \to COL$ such that the labeling function $lab_{n+1}(v) = \mu(lab_n(v), lab_n(N_1)(v), \dots, lab_n(N_k)(v)))$ for $v \in Z$ and $lab_{n+1}(v) = w$ otherwise. Now one can show that $\alpha_{n+1} = \alpha(lab_{n+1})$ by considering two cases. For $v \in Z$, one obtains the following sequence of equalities using the definition of α_{n+1}, the induction hypothesis, the definition of $\alpha(lab)$ for some lab, the definition of lab_{n+1}, and the definition of $\alpha(lab_{n+1})$, in this order:

$$\begin{aligned}
\alpha_{n+1}(v) &= \mu(\alpha_n(v), \alpha_n(N_1(v)), \dots, \alpha_n(N_k(v))) \\
&= \mu(\alpha(lab_n)(v), \alpha(lab_n)(N_1(v)), \dots, \alpha(lab_n)(N_k(v))) \\
&= \mu(lab_n(v), lab_n(N_1(v)), \dots, lab_n(N_k(v))) \\
&= lab_{n+1}(v) \\
&= \alpha(lab_{n+1})(v).
\end{aligned}$$

Finally, one can show the equality for $v \in CELL \setminus Z$. As $act(\alpha_{n+1}) \subseteq Z$, we get $\alpha_{n+1}(v) = w$ for all $v \in CELL \setminus Z$. The same holds for $\alpha(lab_{n+1})$, because $\alpha(lab_{n+1})(v) = w$ for $v \in CELL \setminus V_{B(\mathcal{C},Z)}$ by definition and $\alpha(lab_{n+1})(v) = lab_{n+1}(v)$ for $v \in V_{B(\mathcal{C},Z)}$. But $lab_{n+1}(v) = w$ for $v \in V_{B(\mathcal{C},Z)} \setminus Z$ by the definitions of $\alpha(lab_{n+1})$ and lab_{n+1}.

To prove the induction base for $n = 1$, the argumentation of the induction step can be repeated for α_1 and $\alpha(lab_1)$ in the way it was done for α_{n+1} and $\alpha(lab_{n+1})$, using the fact that, by definition, $\alpha_0 = \alpha(lab_0)$.

This completes the proof. □

If a computation of a cellular automaton reaches a configuration in which a cell that is not of interest becomes active, then the related graph-based reaction system cannot produce the color of this cell because only cells of interest can be recolored. But if one considers an arbitrary computation of finite length, then the set of active cells along the computation is finite so that the graph-based reaction system with the set of active cells as the set of cells of interest, or some larger set, simulates the given computation.

8. Territorial graph surfing systems

In this section we introduce the notion of a *territorial graph surfing system* which results from taking a more “global” look at graph-based reaction systems.

Let $\mathcal{A} = (B, A)$ be a graph-based reaction system. A reaction $b \in A$ naturally induces the subgraph B_b of B defined through the inhibitor $I_b = (X_b, Y_b)$ as follows: $B_b = (V, \Sigma, E)$, where $V = V_B \setminus X_b$, $\Sigma = \Sigma_B$, and $E = (E_B \setminus Y_b) \cap (V \times V \times \Sigma)$. Thus B_b results from B by removing all nodes and edges specified by I_b and all edges attached to nodes in X_b.

For a state T of $\mathcal{A}$, b is enabled by T if two conditions are satisfied:

- T must "be located" within B_b, i.e., $T \in Sub(B_b)$.
- The fact that $T \in Sub(B_b)$ is not sufficient for b to be enabled by T. It is also required that T is located within B_b in a specific way, viz., T must "cover" the reactant graph R_b of b, i.e., $R_b \in Sub(T)$. In other words, R_b is a sort of a "hub" that must be covered by T to make sure that b is enabled by T.

The above reasoning leads naturally to the notion of a "territory" of $\mathcal{A}$ (defined formally below) as a subgraph G of B with a designated hub (which is a subgraph H of G).

We observe that, given the territory $M_b = (B_b, R_b)$ of $\mathcal{A}$ determined by a reaction b (as described above), using b to define M_b may be just one specific way of defining M_b. One can in fact equivalently (as far as a graph surfing is concerned) define $\mathcal{A}$ through a background graph and a finite number of territories of $\mathcal{A}$, where a destination function assigns a subgraph of B (corresponding to the product P_b of b) to each territory of $\mathcal{A}$.

In this way, one gets a more general, *global* approach to defining surfing on graphs. One can consider then reactions as a "local" method for defining territories, where "local" refers here to the *explicit* listing of nodes and edges to be removed from B.

With this intuition in mind, we proceed now to formally define the approach to surfing on graphs by "territorial graph surfing systems". We begin by defining three basic graph-theoretic notions.

Definition 5. A *territory* is an ordered pair $M = (G, H)$ such that G and H are graphs with $H \in Sub(G)$.

We say that G is the *graph of* M, denoted by $gr(M)$, and H is the *hub of* M, denoted by $hub(M)$.

Definition 6. Let M be a territory and F be a graph. Then F is *anchored in* M, denoted by $F\, anch\, M$, if $F \in Sub(gr(M))$ and $hub(M) \in Sub(F)$.

Thus a subgraph F of $gr(M)$ is anchored in M if it contains the hub of M, i.e., $hub(M) \in Sub(F)$.

Definition 7. Let M be a territory and B be a graph. Then M is a *territory of* B if $gr(M) \in Sub(B)$.

We are ready now to define the main notion of this section.

Definition 8. A *territorial graph surfing system* is a 3-tuple $\mathcal{T} = (B, \mathcal{M}, \eta)$, where:

- B is a finite non-empty graph,
- $\mathcal{M}$ is a finite set of territories of B, and
- η is a function, $\eta : \mathcal{M} \to Sub(B)$.

We call B the *background graph of* $\mathcal{T}$, $\mathcal{M}$ the *set of territories of* $\mathcal{T}$, η the *destination function of* $\mathcal{T}$, and, for each $M \in \mathcal{M}$, we call $\eta(M)$ the *M-destination of* $\mathcal{T}$. Subgraphs of B are called *states of* $\mathcal{T}$. Also, we use the notations $B_{\mathcal{T}}$, $\mathcal{M}_{\mathcal{T}}$, and $\eta_{\mathcal{T}}$ for B, $\mathcal{M}$, and η respectively.

The basic dynamics of surfing within a territorial graph surfing system is defined as follows.

Definition 9. Let $\mathcal{T} = (B, \mathcal{M}, \eta)$ be a territorial graph surfing system, and let F be a state of $\mathcal{T}$.

(1) For $M \in \mathcal{M}$, the *M-destination of F in* $\mathcal{T}$, denoted by $des_{M,\mathcal{T}}(F)$, is defined by: $des_{M,\mathcal{T}}(F) = \eta(M)$ if $F\, anch\, M$, and $des_{M,\mathcal{T}}(F) = \emptyset_{\Sigma_B}$ otherwise.
(2) For $\mathcal{P} \subseteq \mathcal{M}$, the *$\mathcal{P}$-destination of* F in $\mathcal{T}$, denoted by $des_{\mathcal{P},\mathcal{T}}(F)$, is defined by: $des_{\mathcal{P},\mathcal{T}}(F) = \bigcup\{des_{M,\mathcal{T}}(F) \,:\, M \in \mathcal{P}\}$.
(3) The *successor of* F *in* $\mathcal{T}$, denoted by $suc_{\mathcal{T}}(F)$, is defined by: $suc_{\mathcal{T}}(F) = des_{\mathcal{M},\mathcal{T}}(F)$.

We have used above a terminology which deviates from the standard terminology of reaction systems (e.g., we use "destination" rather than "product", and "successor" rather than "result") so that it clearly expresses the dynamic behavior of territorial graph surfing systems, viz., graph surfing.

The notion of the successor of F is illustrated in Fig. 7. We assume that $\mathcal{M} = \{M_1, M_2, M_3, M_4\}$ where $M_i = (G_i, H_i)$ for each $i \in \{1, 2, 3, 4\}$. For a state F of $\mathcal{T}$ depicted in Fig. 7, we have:

- $F \in Sub(G_1)$, but $F\, anch\, M_1$ does not hold because $H_1 \notin Sub(F)$,
- $F \in Sub(G_2)$ and $H_2 \in Sub(F)$, hence $F\, anch\, M_2$,
- $F \notin Sub(G_3)$, hence $F\, anch\, M_3$ does not hold,
- $F \in Sub(G_4)$ and $H_4 \in Sub(F)$, hence $F\, anch\, M_4$.

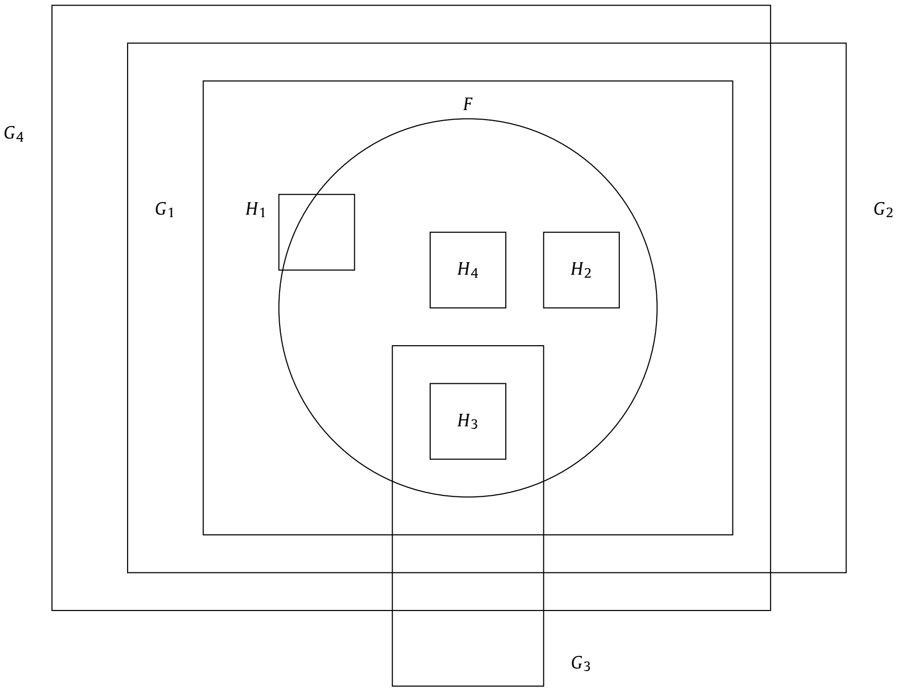


Fig. 7. Example of a state of a territorial graph surfing system.

Therefore

- M_1-destination of F in $\mathcal{T}$ equals $\emptyset_{\Sigma_B}$,
- M_2-destination of F in $\mathcal{T}$ equals $\eta(M_2)$,
- M_3-destination of F in $\mathcal{T}$ equals $\emptyset_{\Sigma_B}$, and
- M_4-destination of F in $\mathcal{T}$ equals $\eta(M_4)$,

Consequently, $suc_{\mathcal{T}}(F) = \eta(M_2) \cup \eta(M_4)$.

The processes of territorial graph surfing systems are defined analogously to processes of (graph-based) reaction systems.

Definition 10 *(Interactive process).* Let $\mathcal{T} = (B, \mathcal{M}, \eta)$ be a territorial graph surfing system.

1. An *interactive process* in $\mathcal{T}$ is an ordered pair $\pi = (\gamma, \delta)$ where γ and δ are finite sequences of states of $\mathcal{T}$, $\gamma = C_0, \dots, C_n$ and $\delta = D_0, \dots, D_n$, for some $n \in \mathbb{N}^+$, such that $D_i = suc_{\mathcal{T}}(C_{i-1} \cup D_{i-1})$ for $i = 1, \dots, n$. The sequence γ is the *context sequence of* π, the sequence δ is the *successor sequence of* π, and the sequence $\tau = T_0, \dots, T_n$ with $T_i = C_i \cup D_i$ for $i = 0, \dots, n$ is the *state sequence of* π.
2. If the context sequence γ is such that $C_i \in Sub(D_i)$, for $i = 0, \dots, n$, then π is *context-independent.*

For a graph-based reaction system $\mathcal{A} = (B, A)$, a natural representation of the context-independent behavior is given by the (unlabeled) *context-independent transition graph* of $\mathcal{A}$, where the nodes are the states of $\mathcal{A}$, i.e., the subgraphs of B, and there is an edge from a node W to a node T if $T = res_{\mathcal{A}}(W)$. Such a graph is of the form illustrated in Fig. 8.

An analogous representation of context-independent behavior of set-based reaction systems was extensively studied (see, e.g., [15]).

For a territorial graph surfing system $\mathcal{T} = (B, \mathcal{M}, \eta)$, a primary representation of the context-independent behavior of $\mathcal{T}$ is given the bipartite *territorial destination graph* of $\mathcal{T}$ where the set of nodes is $\mathcal{M} \cup Sub(B)$, each edge is from a territory in $\mathcal{M}$ to a subgraph in $Sub(B)$, there is an edge from $M \in \mathcal{M}$ to $T \in Sub(B)$ if $\eta(M) = T$, and for each $M \in \mathcal{M}$ there is an edge outgoing from M. Such a graph is of the form illustrated in Fig. 9, where rectangles represent territories of $\mathcal{M}$ and circles represent elements of $Sub(B)$.

From this territorial destination graph, one can construct the (unlabeled) *context-independent transition graph* of $\mathcal{T}$, where nodes are states of $\mathcal{T}$ and there is an edge from a node F to a node T if $T = suc_{\mathcal{T}}(F)$, i.e., $T = des_{\mathcal{M},\mathcal{T}}(F)$. Hence, the existence of an edge here from F to T is deduced from the territorial destination graph of $\mathcal{T}$ in the way explained in our discussion of Fig. 7.

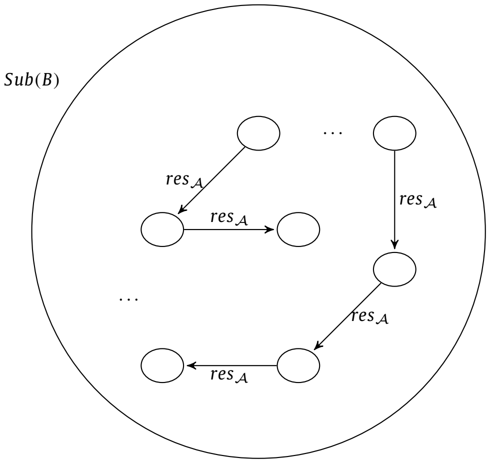


Fig. 8. The context-independent transition graph of a graph-based reaction system.

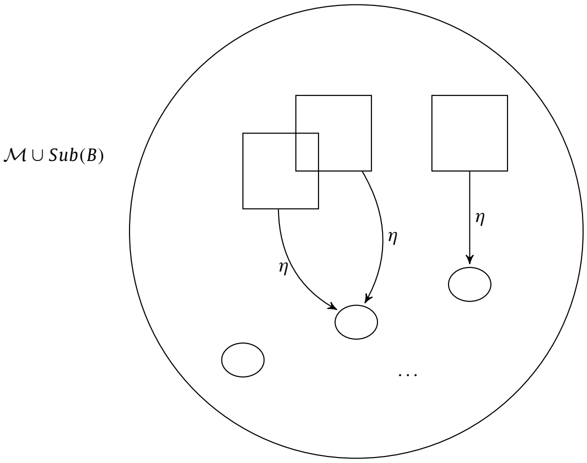


Fig. 9. The territorial destination graph of a territorial graph surfing system.

9. Discussion

We have introduced graph-based reaction systems enhancing in this way the framework of set-based reaction systems so that we can deal with processing (transformation) of graphs. This is illustrated by four small case studies: approximation of Sierpinski triangles, recognition of strings by finite state automata through processing their state graphs, modeling of graph algorithms through two parallel shortest-path algorithms, and modeling of parallel computation through a simulation of cellular automata. Moreover, we have taken a more "global" look at graph-based reaction systems resulting in the notion of territorial graph surfing systems.

Graph-based reaction systems provide a novel approach to graph transformation. Traditional approaches to graph transformation (see, e.g., the handbook volumes [13,22] and individual papers, such as [4,12,17]) follow the "cut, add, and paste" methodology. Given a graph T, one removes some parts of it and pastes some new parts into the remainder of T. The framework of reaction systems employs a novel, very different methodology: no "cut, add, and paste" takes place.

Instead, the whole process of graph transformation takes place within a given "graph universe" specified by the background graph B. Then, given a subgraph T of B, the reactions of a graph-based reaction system together with a possible context graph C (which is also a subgraph of B) determine the subgraph T_1 which is the successor of T. In this way, T is transformed into T_1 and consequently one moves in B from its subgraph T to its subgraph T_1. Iterating this procedure (for a given sequence γ of context graphs), one moves from T_1 to T_2, from T_2 to T_3, etc. creating in B a trajectory of subgraphs of B. Each such trajectory determines a specific way of surfing on B.

The idea of graph transformation by surfing on a given universe graph originates in [11]. The graph universe there is given by a (possibly infinite) well-founded partial-order graph $\mathcal{Z}$ (called zoom structure), and the surfing is directed by some reaction system from a given finite set $\mathcal{F}$ of reaction systems – each reaction systems F in $\mathcal{F}$ explores a specific segment of $\mathcal{Z}$. An inzoom of $\mathcal{Z}$ is a finite reverse walk in $\mathcal{Z}$ (i.e., a walk against the directions of edges in $\mathcal{Z}$). A state of $\mathcal{Z}$ is a finite set of inzooms of $\mathcal{Z}$, and a reaction system $\mathcal{A}$ over $\mathcal{Z}$ transforms states of $\mathcal{Z}$ into states of $\mathcal{Z}$ (the background set of $\mathcal{A}$ is a subset of the set of inzooms of $\mathcal{Z}$). From a graph-theoretic point of view, each reverse walk in $\mathcal{Z}$ represents a finite (one-)path graph and so each state T of $\mathcal{Z}$ represents a subgraph G_T of $\mathcal{Z}$ (which is the union of the path graphs of T). Therefore, moving in $\mathcal{A}$ from state T to state T' corresponds to moving in $\mathcal{Z}$ from its subgraph G_T to its subgraph $G_{T'}$.

From the point of view of graph transformation, graph-based reaction systems generalize the setup above in several ways:

(i) by allowing the background graph to be an arbitrary graph,
(ii) by allowing states as well as reactants and products to be arbitrary subgraphs of the background graph, and
(iii) by allowing inhibition by nodes and edges (rather than only by inzooms).

In this paper, we have introduced a novel framework for graph transformation through (interactive processes of) graph-based reaction systems. There are several “natural” research directions to be pursued in order to develop this framework.

1. Further case studies utilizing this framework should be investigated. For example, graph-based variants of tilings or of fractals such as the approximation of the Sierpinski triangle and the evaluation of reversible circuits on their diagrammatic representations are good candidates (see, e.g., [16,21,1]).
2. Surfing on graphs through graph-based reaction systems brings sequences of graphs (rather than traditional sets of graphs) to the forefront of research. Investigation of sequences of sets “generated” by set-based reaction systems turned out to be very interesting and fruitful (see, e.g., [6,14,23,24]). Clearly, this line of research should be pursued for sequences of graphs defined by graph-based reaction systems.
3. A natural first step towards the understanding of the dynamics of graph-based reaction systems is the investigation of context-independent processes. For example, what sequences of graphs are generated by such processes?
4. An important research theme is “the structure of a system vs. its behavior”. In the case of graph-based reaction systems it may be translated into the influence of the structure of reactions on interactive processes. For example, what kind of processes (sequences of graphs) result from restricting the reactions by requiring that each inhibitor $I = (X, Y)$ is such that $|Y| = 1$ (or dually that $|X| = 1$)? Graph-based reaction system satisfying such restriction(s) are kind of minimal systems.

Finally, we would like to mention that another kind of relationship between graphs and reaction systems is discussed in [2], where one considers networks of reaction systems. Such a network is a graph where reaction systems are placed in the nodes of the graph and then edges between nodes provide communication channels between reaction systems placed in the corresponding nodes.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgement

We are grateful to the anonymous reviewers for their valuable comments that led to various improvements.

References

[1] N. Abdessaied, R. Drechsler, Reversible and Quantum Circuits: Optimization and Complexity Analysis, Springer, 2016.
[2] P. Bottoni, A. Labella, G. Rozenberg, Networks of reaction systems, Int. J. Found. Comput. Sci. (2019), in press.
[3] R. Brijder, A. Ehrenfeucht, M.G. Main, G. Rozenberg, A tour of reaction systems, Int. J. Found. Comput. Sci. 22 (7) (2011) 1499–1517.
[4] F. Drewes, A. Habel, H.-J. Kreowski, Hyperedge replacement graph grammars, in: G. Rozenberg (Ed.), Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations, World Scientific Publishing Co., Singapore, 1997, pp. 95–162, chap. 2.
[5] A. Ehrenfeucht, M.G. Main, G. Rozenberg, Combinatorics of life and death for reaction systems, Int. J. Found. Comput. Sci. 21 (3) (2010) 345–356.
[6] A. Ehrenfeucht, M.G. Main, G. Rozenberg, Functions defined by reaction systems, Int. J. Found. Comput. Sci. 22 (1) (2011) 167–178.
[7] A. Ehrenfeucht, I. Petre, G. Rozenberg, Reaction systems: a model of computation inspired by the functioning of the living cell, in: S. Konstantinidis, N. Moreira, R. Reis, J. Shallit (Eds.), The Role of Theory in Computing, World Scientific Publishing Co., Singapore, 2017, pp. 11–32.
[8] A. Ehrenfeucht, G. Rozenberg, Events and modules in reaction systems, Theor. Comput. Sci. 376 (1–2) (2007) 3–16.
[9] A. Ehrenfeucht, G. Rozenberg, Reaction systems, Fundam. Inform. 75 (1–4) (2007) 263–280.
[10] A. Ehrenfeucht, G. Rozenberg, Introducing time in reaction systems, Theor. Comput. Sci. 410 (4–5) (2009) 310–322.
[11] A. Ehrenfeucht, G. Rozenberg, Zoom structures and reaction systems yield exploration systems, Int. J. Found. Comput. Sci. 25 (4–5) (2014) 275–305.
[12] H. Ehrig, Introduction to the algebraic theory of graph grammars, in: V. Claus, H. Ehrig, G. Rozenberg (Eds.), Graph Grammars and Their Applications to Computer Science and Biology, in: Lecture Notes in Computer Science, vol. 73, 1979, pp. 1–69.
[13] H. Ehrig, H.J. Kreowski, U. Montanari, G. Rozenberg (Eds.), Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 3. Concurrency, Parallelism, and Distribution, World Scientific Publishing Co., Singapore, 1999.
[14] E. Formenti, L. Manzoni, A.E. Porreca, Fixed points and attractors of reaction systems, in: Language, Life, Limits, in: Lecture Notes in Computer Science, vol. 8493, 2014, pp. 194–203.
[15] D. Genova, H.J. Hoogeboom, N. Jonoska, Graph isomorphism condition and equivalence of reaction systems, Theor. Comput. Sci. 701 (1–3) (2017) 109–119.

[16] B. Grünbaum, G.C. Shephard, Tilings and Patterns, W. H. Freeman, 1987.

[17] D. Janssens, G. Rozenberg, Graph grammars with neighbourhood-controlled embedding, Theor. Comput. Sci. 21 (1982) 55–74.

[18] J. Kari, Theory of cellular automata: a survey, Theor. Comput. Sci. 334 (1–3) (2005) 3–33.

[19] H.-J. Kreowski, G. Rosenberg, Graph surfing by reaction systems, in: L. Lambers, J.H. Weber (Eds.), Proc. 11th Intl. Conference on Graph Transformation (ICGT 2018), in: Lecture Notes in Computer Science, vol. 10887, 2018, pp. 45–62.

[20] J. von Neumann, The general and logical theory of automata, in: L. Jeffress (Ed.), Cerebral Mechanisms in Behavior – The Hixon Symposium, John Wiley & Sons, New York, 1951.

[21] H. Peitgen, H. Jürgens, D. Saupe, Chaos and Fractals: New Frontiers of Science, Springer, Berlin, 2004.

[22] G. Rozenberg (Ed.), Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1. Foundations, World Scientific Publishing Co., Singapore, 1997.

[23] A. Salomaa, Functions and sequences generated by reaction systems, Int. J. Found. Comput. Sci. 466 (4–5) (2012) 87–96.

[24] A. Salomaa, On state sequences defined by reaction systems, in: Logic and Program Semantics, in: Lecture Notes in Computer Science, vol. 7230, 2012, pp. 271–282.