PRACE IPI PAN ICS PAS REPORTS

Artur Męski,
Wojciech Penczek,
Grzegorz Rozenberg

Model Checking Temporal Properties of Reaction Systems

Nr 1028

INSTYTUT PODSTAW INFORMATYKI POLSKIEJ AKADEMII NAUK
INSTITUTE OF COMPUTER SCIENCE POLISH ACADEMY OF SCIENCES
ul. Jana Kazimierza 5, 01-248 Warszawa http://www.ipipan.waw.pl

Artur Męski,
Wojciech Penczek,
Grzegorz Rozenberg

Model Checking Temporal Properties of Reaction Systems

Nr 1028

Warszawa, April 2014

Abstract

This paper defines a temporal logic for reaction systems (RSTL). The logic is interpreted over the models for the context restricted reaction systems that generalize standard reaction systems by controlling context sequences. Moreover, a translation from the context restricted reaction systems into boolean functions is defined in order to be used for a symbolic model checking for RSTL over these systems. Finally, model checking for RSTL is proved to be PSPACE-complete.

Keywords: reaction systems, model checking, temporal logic

Streszczenie

Weryfikacja temporalnych własności systemów reakcyjnych

Praca wprowadza logikę temporalną dla systemów reakcyjnych (RSTL), która jest interpretowana w modelach dla systemów reakcyjnych z ograniczeniami kontekstów. Systemy te uogólniają standardowe systemy reakcyjne przez wprowadzenie ograniczeń kontrolujących dopuszczalne konteksty. Ponadto, przedstawiono translację z systemów reakcyjnych z ograniczeniami kontekstów do formuł boolowskich, która umożliwia symboliczną weryfikację modelową dla tych systemów oraz RSTL. Wykazano również, że problem weryfikacji modelowej dla RSTL jest problemem PSPACE-zupełnym.

Słowa kluczowe: systemy reakcyjne, weryfikacja modelowa, logika temporalna

1 Introduction

Model checking [4] is a method that allows for checking whether or not the system in question satisfies a given formula specifying either a desired or an undesired property of that system. Typically, the verified properties are expressed using some modal logic formalism. This method is fully automatic and to be used in practice it does not require expert knowledge of verification techniques.

The basic idea behind the original motivation of reaction systems as models of processes inspired by the functioning of the living cell (see, e.g., [7] and [6]) is that this functioning is based on the interactions of individual biochemical reactions. Moreover, these interactions are driven by two mechanisms: facilitation/acceleration and inhibition/retardation.

This paper introduces a logic for specifying properties of reaction systems, together with a method for verifying these properties. This is the first paper providing a verification method for reaction systems.

Because the processes of reaction systems are guided by the context sequences (which model an interaction with the environment), to enable the verification we introduce a generalisation of reaction systems which allows to specify context entities generating all the context sequences for the processes of the given reaction system. Moreover, we describe an encoding of the model for reaction systems into boolean formulae that can be used for the symbolic model checking approach. We also provide some complexity results for the problem of model checking reaction systems.

The paper is organised as follows. In Section 2 we recall the basic notions of reaction systems, while in Section 3 we define two generalisations of reaction systems. Two basic illustrative examples are given in Section 4. In Section 5 we define the syntax of the logic for reaction systems together with a model used to define the semantics of the logic, based on which, in Section 6 we define a verification method for reaction systems and prove its complexity. In Section 7 we present the encoding for the model defined in Section 5 – this encoding can be used for symbolic model checking of reaction systems. The last section of the paper provides concluding remarks.

2 Reaction Systems

Reaction systems (see, e.g., [2], [7], and [6]) are a formal model for processes instigated by the functioning of living cell. Research topics in this research area are motivated either by biological issues or by a need to understand computations/processes underlying the dynamic behaviour of reaction systems. By now reaction systems became an interesting and novel model of computation.

In this section we recall some basic notions of reaction systems that are used in this paper. First of all, we recall the notion of a reaction.

Definition 2.1. A *reaction* is a triplet $b = (R, I, P)$ such that R, I, P are finite nonempty sets with $R \cap I = \emptyset$.

The sets R, I, P are called the *reactant set of* b, the *inhibitor set of* b, and *the product set of* b, respectively – they are also denoted by P_b, I_b, and P_b, respectively. If $R, I, P \subseteq Z$ for a finite set Z, then we say that b is a *reaction in* Z. We use $rac(Z)$ to denote the set of all reactions in Z.

The above formal notion of a reaction corresponds closely to the basic intuition behind a biochemical reaction. Such a reaction will take place if all of its reactants are present and none of its inhibitors is present, and if it takes place it produces its set of products.

Definition 2.2. Let Z be a finite set, and let $T \subseteq Z$.

1. We say that $b \in rac(Z)$ is *enabled* by T, denoted $en_b(T)$, if $R_b \subseteq T$, and $I_b \cap T = \emptyset$. The *result* of b on T, denoted by $res_b(T)$, is defined by: $res_b(T) = P_b$ if $en_b(T)$, and $res_b(T) = \emptyset$ otherwise.

2. For $B \subseteq rac(Z)$, the result of B on T, denoted by $res_B(T)$, is defined by $res_B(T) = \bigcup\{res_b(T) \mid b \in B\}$.

The intuition underlying the above definition is that T formalises a state of a biochemical system under consideration, e.g., the living cell (T consists of all biochemical entities present in the given state). A reaction b is enabled by T (can take place at T) if all the reactants of b are present in T and none of the inhibitors of b is present in T. Then the result of a set of reactions B is *cumulative* – it is the union of results of the individual

reactions from B. Note that $res_B(T) = \bigcup\{res_b(T) \mid b \in B \text{ and } en_b(T)\}$.

We are ready now to define the notion of *reaction system*.

Definition 2.3. A *reaction system*, *rs* for short, is an ordered pair $\mathcal{R} = (S, A)$, where S is a finite set and $A \subseteq rac(S)$.

The set S is the *background set* of $\mathcal{R}$, each subset of S is called a *state* of $\mathcal{R}$, and A is the *set of reactions from* $\mathcal{R}$.

Example 2.4. Consider the set $S = \{1, 2, 3, 4\}$ and the following reactions from $rac(S)$:

- $a_1 = (\{1, 4\}, \{2\}, \{1, 2\})$,
- $a_2 = (\{2\}, \{3\}, \{1, 3, 4\})$,
- $a_3 = (\{1, 3\}, \{2\}, \{1, 2\})$, and
- $a_4 = (\{3\}, \{2\}, \{1\})$.

Then, $\mathcal{R}_1 = (S, \{a_1, a_2, a_3, a_4\})$ is a rs. □

Dynamic processes taking place in reaction systems are formalised through the notion of an interactive process.

Definition 2.5. Let $\mathcal{R} = (S, A)$ be a rs and let $n \geq 1$.

1. An (n-step) *interactive process* of $\mathcal{R}$ is a pair $\pi = (\gamma, \delta)$ of finite sequences of finite sets such that:
   (a) $\gamma = (C_0, C_1, \cdots, C_n)$ and $\delta = (D_0, D_1, \cdots, D_n)$,
   (b) $C_0, C_1, \cdots, C_n \subseteq S$,
   (c) $D_0, D_1, \cdots, D_n \subseteq S$, with $D_0 = \emptyset$, and
   (d) $D_i = res_A(D_{i-1} \cup C_{i-1})$ for all $i \in \{1, \cdots, n\}$.
2. The *state sequence* of π is the sequence $\tau = (W_0, W_1, \cdots, W_n)$ such that $W_i = C_i \cup D_i$ for all $i \in \{0, \ldots, n\}$.

The sequence γ is the *context sequence of* π and the sequence δ is the *result sequence of* π. The sequence $(C_1, C_2, \cdots, C_n)$ is the *proper context sequence of* π, and the set C_0 is the *initial state of* π. If $C_i \subseteq D_i$ for all $i \in \{1, \ldots, n\}$, then we say that π (and τ) are *context-independent*. Clearly, the context sequence γ of an interactive process $\pi = (\gamma, \delta)$ determines π because the result sequence δ is obtained from γ by reactions of $\mathcal{R}$ (through res_A). The proper context sequence of an interactive process reflects the fact that the behaviour of the living cell is influenced by its context/environment.

Note that:

1. The non-permanency of entities carries over to processes in the obvious way: an entity x from a current state W_i is not sustained in the successor state W_{i+1} unless x is produced by a reaction enabled by W_i ($x \in D_{i+1}$) or x is thrown in by the context ($x \in C_{i+1}$).

2. There is no restriction on the context sequence: any sequence of subsets of the background set S can be a context sequence of an interactive process.

3 Controlling Context Sequences

We will consider now a basic method to control/restrict proper context sequences, just by restricting the set of entities that can occur in them. This leads to the following definition.

Definition 3.1. A *context restricted reaction system*, *crrs* for short, is a triple $\mathcal{U} = (S, A, \mathcal{E})$ where:

1. S is the (finite) background set,

2. $A \subseteq rac(S)$ is the set of reactions,

3. $\mathcal{E} \subseteq S$ is the set of *context entities*.

Our next step to control/restrict interactive processes of reaction systems is to allow only some states to be initial states. This yields the following definition.

Definition 3.2. Let $\mathcal{U} = (S, A, \mathcal{E})$ be a crrs. An *initialised context restricted reaction system*, *icrrs* for short, is a pair $\mathcal{I} = (\mathcal{U}, S_0)$, where $S_0 \subseteq 2^S$ is the set of *initial states* such that $S_0 \neq \emptyset$.

We need to modify now the notion of an interactive process for crrs and icrrs so that it reflects the role of the corresponding restrictions on proper context sets and initial states.

Definition 3.3. Let $\mathcal{U} = (S, A, \mathcal{E})$ be a crrs and let $n \geq 0$ be an integer. An (n-step) *interactive process* in $\mathcal{U}$ is a pair $\pi = (\gamma, \delta)$ of finite sequences of finite sets such that:

1. $\gamma = (C_0, C_1, \cdots, C_n)$ and $\delta = (D_0, D_1, \cdots, D_n)$,
2. $C_0 \subseteq S$, $C_1, \cdots, C_n \subseteq \mathcal{E}$,
3. $D_0, D_1, \cdots, D_n \subseteq S$, $D_0 = \emptyset$, and
4. $D_i = res_A(D_{i-1} \cup C_{i-1})$ for all $i \in \{1, \cdots, n\}$.

For an icrrs $\mathcal{I} = ((S, A, \mathcal{E}), S_0)$ the above definition of an interactive process is augmented by adding the condition $C_0 \in S_0$.

Here is a simple example of an initialised crrs.

Example 3.4. Consider the icrrs $\mathcal{I}_1 = \big((S, A, \{4\}), \{\{1, 4\}\}\big)$, where (S, A) is the rs $\mathcal{R}_1$ from Example 2.4. Let $\gamma = (\{1, 4\}, \emptyset, \{4\}, \{4\})$ and $\delta = (\emptyset, \{1, 2\}, \{1, 3, 4\}, \{1\})$. Then, $\pi = (\gamma, \delta)$ is a 3-step interactive process of $\mathcal{R}_1$. It is also an interactive process of $\mathcal{I}_1$, because all proper context sets are subsets of $\{4\}$ and the initial context set equals $\{1, 4\}$. However π is not an interactive process of the crrs $(S, A, \{1, 3\})$, because the context set $\{4\}$ is not a subset of $\{1, 3\}$. □

4 Basic Examples

Example 4.1. We consider here an implementation by a rs of a small generic regulatory system $\mathcal{G}$ discussed in [7] (Section 3).

The regulatory system contains three (abstract) genes x, y, z expressing proteins X, Y, Z, respectively, protein U, and protein complex Q

formed by X and U. The expression of X by x is inhibited by Y and Z, the expression of Z by z is inhibited by X, and expression of Y by y is inhibited by the protein complex Q.

The background set $S = \{x, \widehat{x}, X, y, \widehat{y}, Y, z, \widehat{z}, Z, Q, U, h\}$, where $\widehat{x}$, $\widehat{y}$, and $\widehat{z}$ denote RNA polymerase sitting on the promoter of genes x, y, and z, respectively. Here h is a "dummy entity" to be used as an inhibitor whenever we do not specify other inhibitors for a reaction (this is a simplification of a specification of a desired rs which is made for didactic reasons).

Finally, the set of reactions consists of four subsets (A_x, A_y, A_z, and A_Q) defined as follows:

- $A_x = \{(\{x\}, \{h\}, \{x\}), (\{x\}, \{Y, Z\}, \{\widehat{x}\}), (\{x, \widehat{x}\}, \{h\}, \{X\})\}$,
- $A_y = \{(\{y\}, \{h\}, \{y\}), (\{y\}, \{Q\}, \{\widehat{y}\}), (\{y, \widehat{y}\}, \{h\}, \{Y\})\}$,
- $A_z = \{(\{z\}, \{h\}, \{z\}), (\{z\}, \{X\}, \{\widehat{z}\}), (\{z, \widehat{z}\}, \{h\}, \{Z\})\}$, and
- $A_Q = \{(\{U, X\}, \{h\}, \{Q\})\}$.

The intuition behind the reactions above corresponds closely to the biological actions taking place in a genetic regulatory system. For example:

- $(\{x\}, \{h\}, \{x\})$ says that if gene x is present and functional in a current state, then x will be present and functional in the successor state, unless something "bad" (inhibition) happens – since $\mathcal{G}$ does not specify such inhibitions, this inhibition is expressed by the dummy inhibitor h (e.g., h can represent a very high level of radiation).

- $(\{x\}, \{Y, Z\}, \{\widehat{x}\})$ says that if gene x is present and functional in current state, then RNA polymerase will sit on its promoter field meaning that $\widehat{x}$ will be present in the successor state (thus $\widehat{x}$ represents RNA polymerase sitting on the promoter field of gene x). This will happen providing that neither protein Y nor protein Z is present in the current state.

- $(\{x, \widehat{x}\}, \{h\}, \{X\})$ says that if gene x is present and functional in the current state and RNA polymerase sits on its promoter field, then eventually protein X will be expressed. This can be inhibited by a whole set of reasons which are not relevant for our story here, and so (again) we set here the dummy inhibitor h.

- $(\{U, X\}, \{h\}, \{Q\})$ says that if both proteins X and U are present in the current state, then protein complex Q will be present in the successor state, unless inhibited by something (not specified in $\mathcal{G}$) formalised by the dummy h.

Now, let us assume that we want to look into the processes starting from the states that already contain x and y. Then, the icrrs for this model could be defined as

$$\mathcal{I}_{ge} = \big((S, A, \{U\}), \big\{\{x, y\}\big\}\big),$$

where: $A = A_x \cup A_y \cup A_z \cup A_Q$. □

Example 4.2. Now we use an implementation by a rs of an n-bit cyclic binary counter given in [2] (Section 4.1) to provide an icrrs implementing such a counter. A current value of the counter can be increased by one or decreased by one depending on the controller's request. If no such request is present in a current state, then the value of the counter remains the same.
The background set of the rs implementing this counter is $S_n = \{p_0, \ldots, p_{n-1}, inc, dec\}$.

The entities $\{p_0, \ldots, p_{n-1}\}$ allow to provide a set representation of numbers from the range $\{0, \ldots, 2^{n-1}\}$ represented in the positional binary notation: entity p_i represents the enabled bit corresponding to the value of 2^i, for $i \in \{0, \ldots, n-1\}$. For example, with $n = 5$, the value 01011 (i.e., 11 in base 10) is represented by the set $\{p_0, p_1, p_3\}$. Thus, for each $W \subseteq S_n$, the value represented by W equals $\sum_{p_i \in W'} 2^i$, where $W' = W \setminus \{inc, dec\}$.

The entities *inc* and *dec* are used to represent the instructions to increase by one (successor), and to decrease by one (predecessor) the value of the current state of the counter. If one attempts to increase

and decrease the counter value at the same time, then the value of the counter is reset to zero.

Then we need the following reactions:

- Retention:
  - For all $j \in \{0, \ldots, n-1\}$: $a_j = (\{p_j\}, \{dec, inc\}, \{p_j\})$.
- Increment:
  - $b_0 = (\{inc\}, \{dec, p_0\}, \{p_0\})$,
  - For all $j \in \{1, \ldots, n-1\}$:
$$b_j = (\{inc, p_0, \ldots, p_{j-1}\}, \{dec, p_j\}, \{p_j\}),$$
  - For all $j, k \in \{0, \ldots, n-1\}$, $j < k$:
$$c_{j,k} = (\{inc, p_k\}, \{dec, p_j\}, \{p_k\}).$$
- Decrement:
  - For all $j \in \{0, \ldots, n-1\}$:
$$d_j = (\{dec\}, \{inc, p_0, \ldots, p_j\}, \{p_j\}),$$
  - For all $j, k \in \{0, \ldots, n-1\}$, $j < k$:
$$e_{j,k} = (\{dec, p_j, p_k\}, \{inc\}, \{p_k\}).$$

Let then:

$$B_n = \{a_j : 0 \leq j \leq n\} \cup \{b_j : 0 \leq j \leq n\} \cup \{d_j : 0 \leq j \leq n\} \\ \cup \{c_{j,k} : 0 \leq j < k < n\} \cup \{e_{j,k} : 0 \leq j < k < n\}.$$

Finally the desired icrrs is defined as $\mathcal{I}^n_{bc} = ((S_n, B_n, \mathcal{E}), \{\emptyset\})$, where $\mathcal{E} = \{inc, dec\}$. Thus, with this implementation of an n-bit cyclic binary counter by an icrrs, the only allowed initial state (for interactive processes in $\mathcal{I}^n_{bc}$) is the empty set – it represents the state of the corresponding counter where all bits are set to 0 and no controller request (*inc* or *dec*) is present. □

5 Logic for Reaction Systems

Our aim is to describe properties of reaction systems by using a branching time logic and, later on, to verify these properties by means of a model checking technique. In this section we introduce the syntax and semantics of a logic for reaction systems.

5.1 Syntax and Semantics

Let $\mathcal{I} = ((S, A, \mathcal{E}), S_0)$ be an icrrs, and $\mathcal{PV}$ be a nonempty set of propositional variables. The language of *Reaction Systems Temporal Logic* (RSTL, for short) is defined by the following grammar:

$$\phi := \wp \mid \neg\phi \mid \phi \vee \phi \mid \mathbf{E}_\Psi \mathbf{X}\phi \mid \mathbf{E}_\Psi \mathbf{G}\phi \mid \mathbf{E}_\Psi[\phi \mathbf{U} \phi],$$

where $\wp \in \mathcal{PV}$ and $\emptyset \neq \Psi \subseteq 2^{\mathcal{E}}$.

The operators of RSTL are composed of the path quantifier $\mathbf{E}_\Psi$ and temporal operators ($\mathbf{X}, \mathbf{G}, \mathbf{U}$). The path quantifier $\mathbf{E}_\Psi$ means 'there exists a path over Ψ': the argument Ψ restricts the set of the considered paths by describing the set of actions allowed along the paths. The temporal operators are used to express requirements imposed on the paths selected by the path quantifiers. The $\mathbf{X}\phi$ operator means 'in the next state ϕ holds', $\mathbf{G}\phi$ means 'in each state of the path (globally) ϕ holds'. The $\phi\mathbf{U}\psi$ operator uses two properties and means 'ψ holds eventually, and ϕ must hold at every preceding state'.

Our logic resembles action-restricted CTL of [10], but we use families of sets of entities instead of actions to restrict the path quantifier.

Next, we define the models for RSTL that are used for interpreting the RSTL formulae.

Definition 5.1. Let $\mathcal{I} = ((S, A, \mathcal{E}), S_0)$ be an icrrs. Then, the model for $\mathcal{I}$ over $\mathcal{PV} = S$ is defined as $\mathcal{M}(\mathcal{I}) = (\mathbb{W}, \mathsf{W}_0, \longrightarrow, L)$ where:

1. $\mathbb{W} = 2^S$ is the set of the *states*,
2. $\mathsf{W}_0 = \{res_A(\alpha) \mid \alpha \in S_0\} \subseteq \mathbb{W}$ is the set of the *initial states*,
3. $\longrightarrow \subseteq \mathbb{W} \times 2^{\mathcal{E}} \times \mathbb{W}$ is the *transition relation* such that for all $w, w' \in \mathbb{W}$, $\alpha \in 2^{\mathcal{E}}$: $(w, \alpha, w') \in \longrightarrow$ iff $w' = res_A(w \cup \alpha)$,

4. $L : \mathbb{W} \to 2^{\mathcal{PV}}$ is a *valuation function* such that $L(w) = w$ for all $w \in \mathbb{W}$.

For the sequel of this paper we fix $\mathcal{I} = ((S, A, \mathcal{E}), S_0)$ and its model $\mathcal{M}_\mathcal{I} = (\mathbb{W}, \mathsf{W}_0, \longrightarrow, L)$. Each element $(w, \alpha, w') \in \longrightarrow$ is denoted by $w \xrightarrow{\alpha} w'$.

The following lemma follows immediately from Definition 5.1. It states that the transition relation is serial, i.e., every state of the model has a successor.

Lemma 5.2. *For each $w \in \mathbb{W}$ there exists $\alpha \in 2^\mathcal{E}$ and $w' \in \mathbb{W}$ such that $w \xrightarrow{\alpha} w'$.*

The formulae of RSTL are interpreted in each state, but they express properties of the *paths* initialised in a given state. In CTL [5], the paths are defined as sequences of states. However, the paths in RSTL contain additional elements which represent context sets. Thus, the paths for RSTL are defined as sequences of states, interleaved with *actions*, that is, subsets of the set $\mathcal{E}$.

Definition 5.3. A *path* over $\Psi \subseteq 2^\mathcal{E}$ is an infinite sequence $\sigma = (w_0, \alpha_0, w_1, \alpha_1, \dots)$ of states and actions such that: $w_i \xrightarrow{\alpha_i} w_{i+1}$ and $\alpha_i \in \Psi$ for $i \geq 0$.

For each $i \geq 0$, the i-th state of the path σ is denoted by $\sigma_s(i)$, and the i-th action of the path σ is denoted by $\sigma_a(i)$. By $\Pi_\Psi(w)$ we denote the set of all the paths over Ψ that start in $w \in \mathbb{W}$, that is, $\Pi_\Psi(w) = \{\sigma \mid \sigma_s(0) = w\}$.

Let $w, w' \in \mathbb{W}$ and $\Psi \subseteq \mathcal{E}$. We say that w' is a Ψ-successor of w (denoted by $w \longrightarrow_\Psi w'$) iff there exists $\alpha \in \Psi$ such that $w \xrightarrow{\alpha} w'$.

Next, we introduce the notion of a *reachable state* and, later on, we present an example of the reachable part of a model.

Definition 5.4. Let $\mathcal{I} = ((S, A, \mathcal{E}), S_0)$ be an icrrs and let $\mathcal{M}_\mathcal{I} = (\mathbb{W}, \mathsf{W}_0, \longrightarrow, L)$ be the model for $\mathcal{I}$. We say that a state $w \in \mathbb{W}$ is *reachable* over $\Psi \subseteq 2^\mathcal{E}$ in $\mathcal{M}_\mathcal{I}$ if there exists $w \in \mathsf{W}_0$ and a path $\sigma \in \Pi_\Psi(w)$ such that $\sigma_s(i) = w$ for some $i \geq 0$.

Example 5.5. Consider the icrrs $\mathcal{I}_1$ from Example 3.4. Then, the reachable part of the model $\mathcal{M}_{\mathcal{I}_1}$ is shown in Figure 1. The states of the model are depicted as the ellipsis.

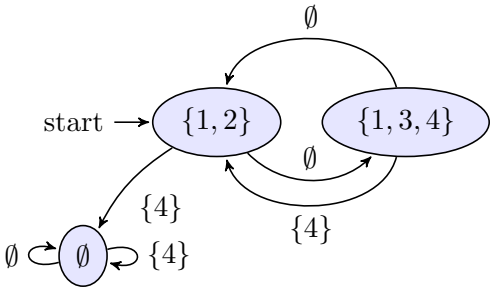

Figure 1: The reachable part of the model for the icrrs $\mathcal{I}_1$ from Example 3.4

Now we are ready to define the semantics of RSTL.

Definition 5.6. Let $\mathcal{M}_{\mathcal{I}} = (\mathbb{W}, \mathsf{W}_0, \longrightarrow, L)$ be a model and $w \in \mathbb{W}$ be a state of $\mathcal{M}_{\mathcal{I}}$. The fact that ϕ holds in the state w of the model $\mathcal{M}_{\mathcal{I}}$ is denoted by $\mathcal{M}_{\mathcal{I}}, w \models \phi$, where the relation $\models$ is defined recursively as follows:

$$
\begin{array}{lll}
\mathcal{M}_{\mathcal{I}}, w \models \wp & \text{iff} & \wp \in L(w) \text{ for } \wp \in \mathcal{PV}, \\
\mathcal{M}_{\mathcal{I}}, w \models \neg\phi & \text{iff} & \mathcal{M}_{\mathcal{I}}, w \not\models \phi, \\
\mathcal{M}_{\mathcal{I}}, w \models \phi \vee \psi & \text{iff} & \mathcal{M}_{\mathcal{I}}, w \models \phi \text{ or } \mathcal{M}_{\mathcal{I}}, w \models \psi, \\
\mathcal{M}_{\mathcal{I}}, w \models \mathbf{E}_\Psi \mathbf{X}\phi & \text{iff} & (\exists \sigma \in \Pi_\Psi(w))\ \mathcal{M}_{\mathcal{I}}, \sigma_s(1) \models \phi, \\
\mathcal{M}_{\mathcal{I}}, w \models \mathbf{E}_\Psi \mathbf{G}\phi & \text{iff} & (\exists \sigma \in \Pi_\Psi(w))(\forall i \geq 0)\big(\mathcal{M}_{\mathcal{I}}, \sigma_s(i) \models \phi\big), \\
\mathcal{M}_{\mathcal{I}}, w \models \mathbf{E}_\Psi[\phi \mathbf{U} \psi] & \text{iff} & (\exists \sigma \in \Pi_\Psi(w))(\exists i \geq 0)\big(\mathcal{M}_{\mathcal{I}}, \sigma_s(i) \models \psi \\
 & & \quad \text{and } (\forall 0 \leq j < i)\ \mathcal{M}_{\mathcal{I}}, \sigma_s(j) \models \phi\big).
\end{array}
$$

We define now derived operators which also introduce the universal path quantifier $\mathbf{A}_\Psi$ meaning 'for all the paths over Ψ':

- $true \stackrel{def}{=} \wp \vee \neg\wp$ for any $\wp \in \mathcal{PV}$,
- $\phi \wedge \psi \stackrel{def}{=} \neg(\neg\phi \vee \neg\psi)$,
- $\phi \Rightarrow \psi \stackrel{def}{=} \neg\phi \vee \psi$,

- $\mathbf{E}_\Psi \mathbf{F}\phi \stackrel{def}{=} \mathbf{E}_\Psi[true\mathbf{U}\phi]$,
- $\mathbf{A}_\Psi \mathbf{F}\phi \stackrel{def}{=} \neg\mathbf{E}_\Psi \mathbf{G}\neg\phi$,
- $\mathbf{A}_\Psi \mathbf{X}\phi \stackrel{def}{=} \neg\mathbf{E}_\Psi \mathbf{X}\neg\phi$,
- $\mathbf{A}_\Psi \mathbf{G}\phi \stackrel{def}{=} \neg\mathbf{E}_\Psi[true\mathbf{U}\neg\phi]$,

Moreover, we assume $\Psi = 2^{\mathcal{E}}$ when the set Ψ is unspecified for any of the RSTL operators, e.g., $\mathbf{EF}\phi \stackrel{def}{=} \mathbf{E}_{2^\mathcal{E}}\mathbf{F}\phi$.

We assume that a formula ϕ is *valid in the model* $\mathcal{M}_\mathcal{I}$ iff $\mathcal{M}_\mathcal{I}, w \models \phi$ for all $w \in \mathsf{W}_0$, that is, the formula ϕ holds in all the initial states. This fact is denoted by $\mathcal{M}_\mathcal{I} \models \phi$.

5.2 Examples of Properties Expressible in RSTL

The following lemma states that there exists a context-independent sequence, if and only if, there exists a sequence with all the proper context sets being empty.

Lemma 5.7. *Let $\mathcal{I} = ((S, A, \mathcal{E}), S_0)$ be an icrrs and $\mathcal{M}_\mathcal{I}$ be the model for $\mathcal{I}$. Then, there exists a path σ such that $\sigma_a(i) \subseteq \sigma_s(i)$ for each $i \geq 0$, if and only if, there exists a path σ' such that $\sigma'_s(i) = \sigma_s(i)$ and $\sigma'_a(i) = \emptyset$ for each $i \geq 0$.*

Proof. First, we assume that there exists a path σ such that $\sigma_a(i) \subseteq \sigma_s(i)$ for each $i \geq 0$. We assume $\sigma'_s(i) = \sigma_s(i)$ for each $i \geq 0$. We need to show that $\sigma'_s(i) \xrightarrow{\emptyset} \sigma'_s(i+1)$ for each $i \geq 0$. From the definition of the transition relation and the fact that $\sigma_a(i) \subseteq \sigma_s(i)$, it follows that $res_A(\sigma_s(i) \cup \sigma_a(i)) = res_A(\sigma_s(i) \cup \emptyset) = res_A(\sigma_s(i))$. From this and the definition of the transition relation, the states of the path σ' may be defined as: $\sigma'_s(i+1) = res_A(\sigma'(i))$, therefore $\sigma'_s(i) \xrightarrow{\emptyset} \sigma'_s(i+1)$, i.e., $\sigma'_a(i) = \emptyset$ for each $i \geq 0$. The converse follows immediately. □

This allows to choose only from the paths with all the proper context sets being empty in order to verify properties over context-independent

sequences. This follows from the fact that in σ and σ' we preserve the order of the states, and the RSTL formulae are interpreted in the states.

The following examples demonstrate how RSTL can be used for expressing properties of icrrs.

Example 5.8. For the icrrs $\mathcal{I}_{ge}$ defined in Example 4.1 we can describe the following properties interpreted according to their validity in the model $\mathcal{M}_{\mathcal{I}_{ge}}$, i.e., they must hold in the initial states of the model:

1. It is possible that the protein Q will be finally produced:

$$\mathbf{EF}Q.$$

2. If Q is present, then the polymerase will not land on the gene y (that is, $\widehat{y}$ will not be present in any of the immediate successors):

$$\mathbf{AG}(Q \Rightarrow (\mathbf{AX}\neg\widehat{y})).$$

3. Always, if the gene x is present, the polymerase lands on x (that is, $\widehat{x}$ is present), and the protein U is supplied in the context, then always when we supply U the protein Q is produced:

$$\mathbf{A}_{\{\{U\}\}}\mathbf{G}((x \wedge \widehat{x}) \Rightarrow \mathbf{A}_{\{\{U\}\}}\mathbf{F}Q).$$

4. It is possible that the protein Q will never be produced:

$$\mathbf{EG}(\neg Q).$$

5. If we do not supply U in the context, then Q will never be produced:

$$\mathbf{A}_{\Psi}\mathbf{G}(\neg Q), \text{ where } \Psi = \{\alpha \subseteq \mathcal{E} \mid U \notin \alpha\} = \{\emptyset\}.$$

6. There exists a context-independent sequence (see Lemma 5.7) over which the state where X and Y are present is reachable:

$$\mathbf{E}_{\{\emptyset\}}\mathbf{F}(X \wedge Y).$$

Example 5.9. For the icrrs $\mathcal{I}_{bc}$ from Example 4.2 we can describe the following properties interpreted according to their validity in $\mathcal{M}_{\mathcal{I}_{ge}}$:

1. Always, if the counter is at its minimal value, then it is possible to reach the maximal value by supplying as context sets only *inc* or *dec* entities:

$$\mathbf{AG}((\neg b_0 \wedge \cdots \wedge \neg b_n) \Rightarrow \mathbf{E}_{\{\{inc\},\{dec\}\}}\mathbf{F}(b_0 \wedge \cdots \wedge b_n)).$$

2. Always, if the counter reaches its maximal value, then always in the next step the counter will make a transition to the minimal value when we supply only *inc* entity:

$$\mathbf{AG}((b_0 \wedge \cdots \wedge b_n) \Rightarrow \mathbf{A}_{\{\{inc\}\}}\mathbf{X}(\neg b_0 \wedge \cdots \wedge \neg b_n)).$$

6 Verification of RSTL properties of crrs

In this section we describe a model checking method for verification of the RSTL properties. The method described here leads to a symbolic model checking problem which we define in Section 7.

To be able to verify RSTL properties of a given icrrs $\mathcal{I}$, we need the set of the reachable states of the model $\mathcal{M}_{\mathcal{I}}$. Firstly, we describe an algorithm for computing all the reachable states and, later on, we provide a method for computing the set of states, where a given RSTL formula holds.

For the purpose of computing the set of all the reachable states we need the notion of a fixed point (we use $|W|$ to denote the cardinality of a set W).

Let W be a finite set and $\tau : 2^W \longrightarrow 2^W$ be a *monotone* function, i.e., $X \subseteq Y$ implies $\tau(X) \subseteq \tau(Y)$ for all $X, Y \subseteq W$. Let $\tau^i(X)$ be defined by $\tau^0(X) = X$ and $\tau^{i+1}(X) = \tau(\tau^i(X))$. We say that $X' \subseteq W$ is a *fixed point* of τ if $\tau(X') = X'$. It can be proved that if τ is monotone and W is a finite set, then there exist $m, n \leq |W|$ such that $\tau^m(\emptyset)$ is the least fixed point of τ (denoted by $\mu X.\tau(X)$) and $\tau^n(W)$ is the greatest fixed point of τ (denoted by $\nu X.\tau(X)$).

Let $\mathcal{M}_{\mathcal{I}} = (\mathbb{W}, \mathsf{W}_0, \longrightarrow, L)$ be a model. From Definition 5.1 it follows that any $w \in \mathbb{W}$ may have many different successors (at most $2^{|\mathcal{E}|}$). Thus we define the function that assigns the set of the Ψ-successors to the states in $W \subseteq \mathbb{W}$:

$$\mathsf{post}_{\Psi}(W) = \{w' \in \mathbb{W} \mid (\exists w \in W)\ w \longrightarrow_{\Psi} w')\},$$

where $\Psi \subseteq 2^{\mathcal{E}}$.

The set of all the reachable states over $2^{\mathcal{E}}$ in $\mathcal{M}_{\mathcal{I}}$ is denoted by $Reach(\mathcal{M}_{\mathcal{I}})$. The set $Reach(\mathcal{M}_{\mathcal{I}})$ can be characterised by the following fixed point equation:

$$Reach(\mathcal{M}_{\mathcal{I}}) = \mu X.\big(\mathsf{W}_0 \cup X \cup \mathsf{post}_{2^{\mathcal{E}}}(X)\big).$$

Algorithm 1 The algorithm for computing the set $Reach(\mathcal{M}_{\mathcal{I}})$

1: $X := \mathsf{W}_0$
2: $X_p := \emptyset$
3: **while** $X \neq X_p$ **do**
4: $\quad X_p := X$
5: $\quad X := X \cup \mathsf{post}_{2^{\mathcal{E}}}(X)$
6: **end while**
7: **return** X

Algorithm 1 implements the fixed-point computation of the reachable states for a given model $\mathcal{M}_{\mathcal{I}} = (\mathbb{W}, \mathsf{W}_0, \longrightarrow, L)$. Line 7 of the algorithm returns the set X which is equal to $Reach(\mathcal{M}_{\mathcal{I}})$.

The set of all the reachable states of the model $\mathcal{M}_{\mathcal{I}}$ at which ϕ holds is denoted by $[\![\mathcal{M}_{\mathcal{I}}, \phi]\!]$ or by $[\![\phi]\!]$ if $\mathcal{M}_{\mathcal{I}}$ is implicitly understood. For $W \subseteq Reach(\mathcal{M}_{\mathcal{I}})$ we define a function that assigns the set of the Ψ-predecessors to W:

$$\mathsf{pre}^{\exists}_{\Psi}(W) = \{w \in Reach(\mathcal{M}_{\mathcal{I}}) \mid (\exists w' \in W)\ w \longrightarrow_{\Psi} w')\}.$$

Let ϕ, ψ be some RSTL formulae. We define then the following sets:

- $[\![\neg\phi]\!] \stackrel{def}{=} Reach(\mathcal{M}_{\mathcal{I}}) \setminus [\![\phi]\!]$,

- $[\![\phi \wedge \psi]\!] \stackrel{def}{=} [\![\phi]\!] \cap [\![\psi]\!]$, $[\![\phi \vee \psi]\!] \stackrel{def}{=} [\![\phi]\!] \cup [\![\psi]\!]$,
- $[\![\mathbf{E}_\Psi \mathbf{X}\phi]\!] \stackrel{def}{=} \mathsf{pre}^{\exists}_\Psi([\![\phi]\!])$.

The remaining operators are defined as the following fixed points:

- $[\![\mathbf{E}_\Psi \mathbf{G}\phi]\!] \stackrel{def}{=} \nu X.\big([\![\phi]\!] \cap \mathsf{pre}^{\exists}_\Psi(X)\big)$,
- $[\![\mathbf{E}_\Psi [\phi \mathbf{U} \psi]]\!] \stackrel{def}{=} \mu X.\big([\![\psi]\!] \cup ([\![\phi]\!] \cap \mathsf{pre}^{\exists}_\Psi(X))\big)$.

In the case of $[\![\mathbf{E}_\Psi \mathbf{G}\phi]\!]$ the greatest fixed point computation is involved, which in each iteration removes states that do not have a Ψ-predecessor in which ϕ is satisfied. In the case of $[\![\mathbf{E}_\Psi [\phi \mathbf{U} \psi]]\!]$ the least fixed point computation is involved, such that in each iteration the Ψ-predecessors, in which ϕ is satisfied are added to the set of states in which ψ is satisfied. See Algorithm 2 and 3 for the pseudocode of the procedures implementing the described fixed point computations.

Algorithm 2 The algorithm for computing the set $[\![\mathbf{E}_\Psi \mathbf{G}\phi]\!]$

1: $X := Reach(\mathcal{M}_\mathcal{I})$, $X_p := \emptyset$
2: **while** $X \neq X_p$ **do**
3: $\quad X_p := X$
4: $\quad X := \big([\![\phi]\!] \cap \mathsf{pre}^{\exists}_\Psi(X)\big)$
5: **end while**
6: **return** X

Algorithm 3 The algorithm for computing the set $[\![\mathbf{E}_\Psi [\phi \mathbf{U} \psi]]\!]$

1: $X := \emptyset$, $X_p := Reach(\mathcal{M}_\mathcal{I})$
2: **while** $X \neq X_p$ **do**
3: $\quad X_p := X$
4: $\quad X := [\![\psi]\!] \cup ([\![\phi]\!] \cap \mathsf{pre}^{\exists}_\Psi(X))$
5: **end while**
6: **return** X

Lemma 6.1. *Given an icrrs $\mathcal{I} = ((S, A, \mathcal{E}), S_0)$ and an RSTL formula ϕ, the problem of deciding whether $\mathcal{M}_\mathcal{I} \models \phi$ is PSPACE-hard.*

Proof. The proof is by reduction of QSAT[1], which is a known PSPACE-complete problem [11], to the RSTL model checking problem. The construction used for the reduction is similar to the one of [8] for timed automata and TCTL. Let $PV = \{x_1, x_2, \ldots, x_n\}$ be a set of propositional variables, $\beta(x_1, x_2, \ldots, x_n)$ be a boolean formula in 3-CNF , and $\gamma = \mathcal{Q}_1 x_1 \mathcal{Q}_2 x_2 \ldots \mathcal{Q}_n x_n \beta(x_1, x_2, \ldots, x_n)$ be a quantified boolean formula, where $\mathcal{Q}_i = \exists$ if i is odd, $\mathcal{Q}_i = \forall$ if i is even. Then, the QSAT problem consists in deciding if the formula γ is true.

We define an additional set of the negated propositional variables $\overline{PV} = \{\bar{x} \mid x \in PV\}$ and assume that β is represented as the conjunction of m clauses: $\beta = c_1 \wedge c_2 \wedge \cdots \wedge c_m$, where $c_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$ with $l_{i,j} \in (PV \cup \overline{PV})$ for all $0 < i \leq m$, $1 \leq j \leq 3$. Moreover, for each clause c we define the set $vars(c) = \{0 < k \leq n \mid x_k \in PV \text{ is in } c\}$ and the set $\overline{vars}(c) = \{0 < k \leq n \mid \bar{x}_k \in \overline{PV} \text{ is in } c\}$. Next, we define the icrrs which we use for the translation. Let $V = \{p_1, \bar{p}_1, \ldots, p_n, \bar{p}_n\}$ be a set of the entities that represent the propositional variables and their negations, and $C = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_m\}$ be the set of the entities that correspond to the clauses. The entity t is used to indicate that under the considered valuation the formula γ is true. The entity h is used as the inhibitor of the reactions where no inhibitors are needed for the translation to work. This guarantees that the inhibitor set is non-empty. Then, the background set is defined as $S = V \cup C \cup \{t, h\}$. Next, we define the following sets of reactions:

- $P_i = \{(\{p_i\}, \{h\}, \{p_i\}), (\{\bar{p}_i\}, \{h\}, \{\bar{p}_i\})\}$ for $0 < i \leq n$,
- $L_i = \{(\{p_k\}, \{\bar{p}_k\}, \{\hat{c}_i\}) \mid k \in vars(i)\} \cup \{(\{\bar{p}_k\}, \{p_k\}, \{\hat{c}_i\}) \mid k \in \overline{vars}(i)\}$ for $0 < i \leq n$,
- $F = \{(\{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_n\}, \{h\}, \{t\})\}$.

The set P_i contains the reactions responsible for preserving the valuations of the variables along the execution sequences. The reactions of L_i produce entities that indicate whether a single clause is satisfied, whereas the

[1]Quantified SAT (QSAT) is a problem which consists in checking whether a quantified boolean formula is in the language of TQBF (true quantified boolean formulae)

reaction of F produces the entity indicating that all the clauses are satisfied. The set of all the reactions of the icrrs is defined as $A = \bigcup_{i=1}^{n} P_i \cup \bigcup_{i=1}^{n} L_i \cup F$. As the context entities $\mathcal{E}$ we use the entities corresponding to the propositional variables, i.e., $\mathcal{E} = \{p_1, \bar{p}_1, \ldots, p_n, \bar{p}_n\}$. We assume that the set of the initial states contains only the empty set, i.e., $S_0 = \{\emptyset\}$. Then, the icrrs for the described QSAT problem and the formula γ is defined as $\mathcal{I}_{\mathbf{QSAT}} = ((S, A, \mathcal{E}), S_0)$. We define the following RSTL formulae for all $1 \leq i \leq n$:

$$\phi_i = \begin{cases} \mathbf{A}_{\left\{\{p_i\},\{\bar{p}_i\}\right\}} \mathbf{X}\phi_{i+1} \text{ if } \mathcal{Q}_i = \forall, \\ \mathbf{E}_{\left\{\{p_i\},\{\bar{p}_i\}\right\}} \mathbf{X}\phi_{i+1} \text{ if } \mathcal{Q}_i = \exists. \end{cases}$$

$$\phi_{n+1} = \mathbf{E}_{\{\emptyset\}} \mathbf{X} t$$

The formula ϕ_1 consists of n nested next-state operators that restrict the choice of entities either to p_i or $\bar{p}_i$ (no contradictions are allowed) at each level $0 < i \leq n$. For the level $n+1$ in the formula, we check if there exists a successor state in which the entity t exists, indicating that γ is true. The assumed set $\mathcal{E}$ allows us to generate all the valuations that need to be considered. Then, we restrict these valuations using the RSTL formula according to the QSAT quantification.

Then, it is easy to see that γ is true if and only if $\mathcal{M}_{\mathcal{I}_{\mathbf{QSAT}}} \models \phi_1$. □

Lemma 6.2. *Given an icrrs $\mathcal{I} = ((S, A, \mathcal{E}), S_0)$ and an RSTL formula ϕ, the problem of deciding whether $\mathcal{M}_{\mathcal{I}} \models \phi$ is in PSPACE.*

Proof. To prove that the problem is in PSPACE we show that there exists a nondeterministic algorithm for deciding whether $\mathcal{M}_{\mathcal{I}} \models \phi$ that requires at most polynomial space in the size of the input, i.e., the formula ϕ and the icrrs $\mathcal{I}$. The proof is similar to the one of [1] for TCTL interpreted over timed graphs.

The algorithm uses the recursive procedure $label(w, \phi)$ which returns *true* iff $\mathcal{M}_{\mathcal{I}}, w \models \phi$, where $w \subseteq S$; otherwise, it returns $false$. The encoding of each state requires space $O(|S|)$ and each successor can be generated in space $O(|S|)$, whereas the overall algorithm requires space

$O\big(|S| \cdot \mathsf{d}(\phi)\big)$, where $\mathsf{d}(\phi)$ is the maximal nesting depth of ϕ and is defined recursively as follows:

- if $\phi = \wp$, where $\wp \in \mathcal{PV}$, then $\mathsf{d}(\phi) = 0$,
- if $\phi \in \{\neg\phi', \mathbf{E}_\Psi \mathbf{X}\phi', \mathbf{E}_\Psi \mathbf{G}\phi'\}$, then $\mathsf{d}(\phi) = \mathsf{d}(\phi') + 1$,
- if $\phi \in \{\phi' \vee \phi'', \mathbf{E}_\Psi[\phi' \mathbf{U} \phi'']\}$, then $\mathsf{d}(\phi) = max\{\mathsf{d}(\phi'), \mathsf{d}(\phi'')\} + 1$.

The proof follows by the induction on the length of the formula ϕ. The cases where ϕ does not contain temporal operators or $\phi = \mathbf{E}_\Psi \mathbf{X}\phi_1$ are straightforward.

The nondeterministic procedure for checking $\phi = \mathbf{E}_\Psi[\phi_1 \mathbf{U} \phi_2]$ in $w \subseteq S$ is outlined in Algorithm 4. It nondeterministically chooses states and actions of a path over Ψ, checks at each step if the state chosen is a successor of the previous state via the action chosen, and if ϕ_1 holds in that state. If not, then an action and a state are selected again. At each step of the procedure, only two states are stored: the current state and its successor. If in the current state ϕ_2 holds, then the algorithm returns *true*.

The procedure for checking $\phi = \mathbf{E}_\Psi \mathbf{G}\phi_1$ in $w \subseteq S$ is outlined in Algorithm 5. Similarily to the previous case, the algorithm guesses a path over Ψ, and nondeterministically chooses a state of that path, for the purpose of detecting a loop. At each step only three states are stored: the current state, its successor, and a state for detecting a loop. The procedure ensures that ϕ_1 holds in the current state and generates its successor. If the state used for detecting a loop has appeared again in the sequence, then the search stops, and the algorithm returns *true*.

The procedure returns false if no sequence for which the procedure returns *true* could be found. To ensure that the procedure terminates, for each sequence guessed the procedure nondeterministically chooses a state $\widehat{w}_r$ of that sequence. The guessing of the sequence stops when the newly guessed state is $\widehat{w}_r$. For simplicity of the presentation, this part of the procedure is not included in Algorithm 4 and Algorithm 5.

To check if $\mathcal{M}_\mathcal{I} \models \phi$, the procedure *label* is executed for all the initial states to check if ϕ holds for all $w \in S_0$. For each execution, the procedure is called recursively for each subformula of ϕ. At a given recursion level the procedure requires only a constant number of variables

Algorithm 4 Procedure for checking $\mathbf{E}_\Psi[\phi_1 \mathbf{U} \phi_2]$ in $w \subseteq S$

1: $\widehat{w} := w$
2: *checking*:
3: **if** $label(\widehat{w}, \phi_2)$ **then**
4: **return** $true$
5: **end if**
6: **if** $\neg label(\widehat{w}, \phi_1)$ **then**
7: **return** $false$
8: **end if**
9: *guessing*:
10: guess $\widehat{w}' \subseteq S$
11: guess $\alpha \in \Psi$
12: **if** $\widehat{w}' \neq res_A(\widehat{w} \cup \alpha) \vee \neg label(\widehat{w}', \phi_1)$ **then**
13: **goto** *guessing*
14: **else**
15: $\widehat{w} := \widehat{w}'$
16: **goto** *checking*
17: **end if**

to be stored. The total space requirement depends on $O(\mathsf{d}(\phi))$ calls of the *label* procedure, where a single call needs space $O(|S|)$. For each call of the *label* procedure, i.e., for each nesting level of ϕ, the *label* procedure is called recursively at most twice, as each operator of RSTL has at most two arguments. Thus, the overall space requirement of the procedure is $O\big(|S| \cdot \mathsf{d}(\phi)\big)$. Therefore, by Savitch's theorem, the deterministic algorithm can be implemented in polynomial space.

□

The following theorem follows directly from Lemma 6.1 and Lemma 6.2:

Theorem 6.3. *The model checking problem for RSTL is PSPACE-complete.*

This RSTL verification method can be performed symbolically using binary decision diagrams (BDDs) [3]. The operations on sets of states,

Algorithm 5 Procedure for checking $\mathbf{E}_\Psi \mathbf{G}\phi_1$ in $w \subseteq S$

1: $\widehat{w} := w$
2: $L := false$
3: **if** $\neg label(\widehat{w}, \phi_1)$ **then**
4: **return** $false$
5: **end if**
6: *guessing*:
7: guess $\widehat{w}' \subseteq S$
8: guess $\alpha \in \Psi$
9: **if** $\widehat{w}' \neq res_A(\widehat{w} \cup \alpha) \vee \neg label(\widehat{w}', \phi_1)$ **then**
10: **goto** *guessing*
11: **end if**
12: **if** $\neg L$ **then**
13: guess $\gamma \in \{true, false\}$
14: **if** γ **then**
15: $\widehat{w}_l := \widehat{w}'$
16: $L := true$
17: **end if**
18: **else**
19: **if** $\widehat{w}' = \widehat{w}_l$ **then**
20: **return** true
21: **end if**
22: **end if**
23: $\widehat{w} := \widehat{w}'$
24: **goto** *guessing*

such as union, intersection, and difference, can be carried out efficiently on BDDs representing boolean functions.

7 Encoding icrrs into boolean functions

In this section we provide an encoding of the context restricted reaction systems into boolean functions in order to be used for symbolic model checking. We start with defining the *symbolic model checking problem*

for RSTL.

Let $\mathcal{I}$ be an icrrs and ϕ be an RSTL formula. The symbolic model checking problem for RSTL consists in deciding whether $\mathcal{M}_{\mathcal{I}} \models \phi$, using boolean formulae for representing the model $\mathcal{M}_{\mathcal{I}}$.

The general idea of the encoding into boolean functions is similar to the one introduced for reaction systems in [6], but it differs in how the encoding of the transitions between states is defined.

Let $\mathcal{I} = ((S, A, \mathcal{E}), S_0)$ be an icrrs and $\mathcal{M}_{\mathcal{I}} = (\mathbb{W}, \mathsf{W}_0, \longrightarrow, L)$ be its model defined over S. We denote the elements of S by $e_1, \ldots, e_n$, where $n = |S|$. We introduce sets of the propositional variables used in the encoding. The states of the model are encoded using the set $\mathrm{P} = \{\mathtt{p}_1, \ldots, \mathtt{p}_n\}$. The actions representing context entities are encoded with the variables $\mathrm{P}^{\mathcal{E}} = \{\mathtt{p}_1^{\mathcal{E}}, \ldots, \mathtt{p}_n^{\mathcal{E}}\}$. To encode the successors in the transition relation, we use the set $\mathrm{P}' = \{\mathtt{p}'_1, \cdots, \mathtt{p}'_n\}$ of the primed variables. The set of reactions producing $e \in S$ is defined as $Prod(e) = \{a \in A \mid e \in P_a\}$. For brevity, we use the following vectors of variables: $\overline{\mathbf{p}} = (\mathtt{p}_1, \ldots, \mathtt{p}_n)$, $\overline{\mathbf{p}}^{\mathcal{E}} = (\mathtt{p}_1, \ldots, \mathtt{p}_n)$, $\overline{\mathbf{p}}' = (\mathtt{p}'_1, \ldots, \mathtt{p}'_n)$. Moreover, we define the following functions: $\mathtt{m} : S \to \mathrm{P}$, $\mathtt{m}' : S \to \mathrm{P}'$, $\mathtt{m}^{\mathcal{E}} : S \to \mathrm{P}^{\mathcal{E}}$, such that $\mathtt{m}(e_i) = \mathtt{p}_i$, $\mathtt{m}'(e_i) = \mathtt{p}'_i$, $\mathtt{m}^{\mathcal{E}}(e_i) = \mathtt{p}_i^{\mathcal{E}}$, for all $1 \leq i \leq n$. The functions map the background set entities to the corresponding variables of the encoding.

Single state. A state $w \in \mathbb{W}$ is encoded as the conjunction of all the variables corresponding to the entities that are present in w, and the conjunction of all the negations of the variables that are not present in w:

$$\mathsf{St}_w(\overline{\mathbf{p}}) = \Big(\bigwedge_{e \in w} \mathtt{m}(e) \Big) \wedge \Big(\bigwedge_{e \in (S \setminus w)} \neg \mathtt{m}(e) \Big).$$

Sets of states. A set $W \subseteq \mathbb{W}$ is encoded as the disjunction of all the encoded states that are in $\mathbb{W}$:

$$\mathsf{SetSt}_W(\overline{\mathbf{p}}) = \bigvee_{w \in W} \mathsf{St}_w(\overline{\mathbf{p}}).$$

Context sets. A set $\alpha \subseteq S$ of context entities is encoded as follows:

$$\mathsf{Ct}_\alpha(\overline{\mathbf{p}}^{\mathcal{E}}) = \Big(\bigwedge_{e \in \alpha} \mathtt{m}^{\mathcal{E}}(e) \Big) \wedge \Big(\bigwedge_{e \in (S \setminus \alpha)} \neg \mathtt{m}^{\mathcal{E}}(e) \Big).$$

Sets of context sets. A set $\Psi \subseteq 2^{\mathcal{E}}$ of sets of context entities is encoded as the disjunction of all the encoded sets of Ψ:

$$\mathsf{SetCt}_{\Psi}(\overline{\mathbf{p}}^{\mathcal{E}}) = \bigvee_{\alpha \in \Psi} \mathsf{Ct}_{\alpha}(\overline{\mathbf{p}}^{\mathcal{E}}).$$

Entity production condition. A single entity $e \in S$ can be produced if there exists a reaction $a \in Prod(e)$ that is enabled, that is, all of the variables corresponding to reactants of a (including the variables representing the context) are true and all of the variables corresponding to inhibitors of a (also including the variables representing the context) are false. The formula encoding this is defined as follows:

– if $Prod(e) \neq \emptyset$, then

$$\mathsf{En}_e\big(\overline{\mathbf{p}}, \overline{\mathbf{p}}^{\mathcal{E}}\big) = \bigvee_{a \in Prod(e)} \Big(\bigwedge_{e' \in R_a} (\mathtt{m}(e') \vee \mathtt{m}^{\mathcal{E}}(e')) \wedge \bigwedge_{e' \in I_a} \neg(\mathtt{m}(e') \vee \mathtt{m}^{\mathcal{E}}(e')) \Big);$$

– if $Prod(e) = \emptyset$, then

$$\mathsf{En}_e\big(\overline{\mathbf{p}}, \overline{\mathbf{p}}^{\mathcal{E}}\big) = false.$$

Entity production. The function $\mathsf{Pr}_e\big(\overline{\mathbf{p}}, \overline{\mathbf{p}}^{\mathcal{E}}, \overline{\mathbf{p}}'\big)$ encodes the production of a single entity $e \in S$. When the production of e is enabled, then the variable corresponding to e is required to be true; otherwise, the variable is required to be false.

$$\mathsf{Pr}_e\big(\overline{\mathbf{p}}, \overline{\mathbf{p}}^{\mathcal{E}}, \overline{\mathbf{p}}'\big) = \Big(\mathsf{En}_e\big(\overline{\mathbf{p}}, \overline{\mathbf{p}}^{\mathcal{E}}\big) \wedge \mathtt{m}'(e)\Big) \vee \Big(\neg \mathsf{En}_e\big(\overline{\mathbf{p}}, \overline{\mathbf{p}}^{\mathcal{E}}\big) \wedge \neg \mathtt{m}'(e)\Big).$$

Permitted context sets. The following formula encodes the permitted context sets by blocking the not allowed context entities.

$$\mathsf{PCt}\big(\overline{\mathbf{p}}^{\mathcal{E}}\big) = \bigwedge_{e \in (S \setminus \mathcal{E})} \neg \mathtt{m}^{\mathcal{E}}(e).$$

Transition relation. To encode the transition relation, we define the function that is the conjunction of the entity production encodings for all the background set entities and restricts the allowed context sets.

$$\mathsf{Tr}\big(\overline{\mathbf{p}}, \overline{\mathbf{p}}^{\mathcal{E}}, \overline{\mathbf{p}}'\big) = \bigwedge_{e \in S} \mathsf{Pr}_e\big(\overline{\mathbf{p}}, \overline{\mathbf{p}}^{\mathcal{E}}, \overline{\mathbf{p}}'\big) \wedge \mathsf{PCt}\big(\overline{\mathbf{p}}^{\mathcal{E}}\big).$$

The described encoding method can be used for the symbolic model checking problem for RSTL. In the following theorem we state the complexity of this problem.

Theorem 7.1. *The symbolic model checking problem for RSTL is PSPACE-complete.*

Proof. The theorem follows from the fact that the symbolic model checking for CTL is PSPACE-complete [9] and that the transition relation of an icrrs can be represented (encoded in polynomial time) with a boolean function, as demonstrated above. Moreover, the complexity of the verification algorithm for RSTL does not change with respect to CTL. This follows from the fact that RSTL only restricts the choice of the transitions to be considered. Therefore, in the symbolic model checking algorithm we replace the function computing the predecessors of a given state with the function $\mathsf{pre}^{\exists}_{\Psi}(X)$ which restricts the choice of the predecessors to only those accessible via Ψ (see Section 6). In the symbolic model checking algorithm this requires one more conjunction to be computed at each iteration, therefore the modification introduces only a difference of a constant in the overall complexity. □

8 Conclusions

We have introduced a temporal logic for reaction systems (RSTL) allowing to express properties that depend on the context sequences specified by the context sets in the temporal operators. We have also described a symbolic verification method for RSTL and provided some complexity results for the problems of model checking and symbolic model checking for RSTL.

This paper is the first step towards verification of reaction systems. We are also planning to work on the experimental evaluation of the presented method and a comprehensive tool for the verification of reaction systems.

References

[1] R. Alur, C. Courcoubetis, and D. Dill. Model checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[2] R. Brijder, A. Ehrenfeucht, M. G. Main, and G. Rozenberg. A tour of reaction systems. *Int. J. Found. Comput. Sci.*, 22(7):1499–1517, 2011.

[3] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.

[4] E. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[5] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[6] A. Ehrenfeucht and G. Rozenberg. Reaction systems. *Fundamenta Informaticae*, 75(1-4):263–280, 2007.

[7] A. Ehrenfeucht, J. Kleijn, M. Koutny, and G. Rozenberg. Reaction systems: A natural computing approach to the functioning of living cells. *A Computable Universe, Understanding and Exploring Nature as Computation (H. Zenil, ed.)*, pages 189–208, 2012.

[8] F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking timed automata with one or two clocks. In *CONCUR*, pages 387–401, 2004.

[9] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[10] C. Pecheur and F. Raimondi. Symbolic model checking of logics with actions. In *MoChArt*, pages 113–128, 2006.

[11] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *STOC*, pages 1–9, 1973.

Pracę zgłosił Józef Winkowski

Adres autora: Artur Męski
Instytut Podstaw Informatyki PAN
ul. Jana Kazimierza 5
01-248 Warszawa, Polska

Wojciech Penczek
Instytut Podstaw Informatyki PAN
ul. Jana Kazimierza 5
01-248 Warszawa, Polska

Grzegorz Rozenberg
LIACS, Leiden University
P.O. Box 9512, 2300 RA, The Netherlands

E-mail: meski@ipipan.waw.pl
penczek@ipipan.waw.pl
g.rozenberg@liacs.leidenuniv.nl

Symbol klasyfikacji rzeczowej (CR): F.3.1, I.2.2, I.2.4, I.6.4

Na prawach rękopisu
Printed as manuscript

Nakład 90 egzemplarzy. Papier kserograficzny klasy III. Oddano do druku w kwietniu 2014 r. Wydawnictwo IPI PAN. ISSN 0138-0648