Fundamenta Informaticae 154 (2017) 289–306
DOI 10.3233/FI-2017-1567
IOS Press

Verification of Linear-Time Temporal Properties for Reaction Systems with Discrete Concentrations

Artur Męski*
Institute of Computer Science, PAS, Jana Kazimierza 5, 01-248 Warsaw, Poland
Vector Software, Inc., London, UK
meski@ipipan.waw.pl, artur.meski@vectorcast.com

Maciej Koutny
School of Computing Science, Newcastle University, Newcastle upon Tyne, NE1 7RU, UK
maciej.koutny@ncl.ac.uk

Wojciech Penczek
Institute of Computer Science, PAS, Jana Kazimierza 5, 01-248 Warsaw, Poland
University of Natural Sciences and Humanities, ICS, Siedlce, Poland
penczek@ipipan.waw.pl

Abstract. Reaction systems are a formal model for computational processes inspired by the functioning of the living cell. This paper introduces reaction systems with discrete concentrations, which are an extension of reaction systems allowing for quantitative modelling. We demonstrate that although reaction systems with discrete concentrations are semantically equivalent to the original qualitative reaction systems, they provide much more succinct representations in terms of the number of entities being used. We define a variant of Linear Time Temporal Logic interpreted over models of reaction systems with discrete concentrations. We provide its suitable encoding in SMT, together with bounded model checking, and present experimental results demonstrating the scalability of the verification method for reaction systems with discrete concentrations.

Keywords: reaction systems, bounded model checking, linear time temporal logic

*Address for correspondence: Institute of Computer Science, PAS, Jana Kazimierza 5, 01-248 Warsaw, Poland

1. Introduction

In the seminal paper [1] Andrzej Ehrenfeucht and Grzegorz Rozenberg introduced a formal model for processes inspired by the functioning of living cells. The new model of *reaction systems* captured in presumably the simplest way the basic mechanisms responsible for the dynamic behaviour of a living cell. In particular, the key feature of reaction systems is that the functioning of the living cell results from the interactions of biochemical reactions. In turn, these interactions are based on the mechanisms of facilitation and inhibition: the (products of the) reactions may facilitate or inhibit each other. The striking simplicity of the basic model of reaction systems stems also from the fact that they model the reactions, states, and dynamic processes using (tuples of) finite sets. More involved concepts, such as time and probabilities, can be suitably defined using the basic ones.

Reaction system related research topics have so far been motivated by biological issues or by a need to understand computations/processes underlying the dynamic behaviour of reaction systems (see, e.g., [2, 3]). Following their introduction, a number of extensions of reaction systems were studied, e.g., reaction systems with time [4] and quantum and probabilistic reaction systems [5]. Mathematical properties of reaction systems were investigated in, e.g., [6, 7, 8, 9, 10, 11, 12, 13]. Examples of application of reaction systems to modelling of systems include, e.g., [14, 15]. Recently, there has been an increasing interest in verification of reaction systems as described in, e.g., [16, 17, 18].

The basic model of reaction systems is qualitative in the sense that there is no direct representation of the number of molecules involved in biochemical reactions nor the number of molecules present in the current system state. This paper, which extends our conference paper [19], considers reaction systems with discrete concentrations of entities, so defines a model allowing for quantitative modelling. Note that there exist also other approaches that allow for modelling of complex dependencies of concentration levels and their changes, e.g., chemical reaction networks theory based on [20]. However, the formalism of reaction systems is much simpler and the processes of reaction systems depend on interactions with the environment.

Although reaction systems with discrete concentrations are semantically equivalent to the original qualitative reaction systems, they provide much more succinct representations in terms of the number of entities being used, and allow for more efficient verification [19]. Our experimental results show a significant improvement in the execution times in favour of reaction systems with discrete concentrations. Reaction systems with durations introduced in [21] share some similarities with the formalism defined in this paper. However, in contrast to reaction systems with durations, the execution of reaction systems with discrete concentrations does not explicitly depend on a counter which can be implemented using reactions (see the translation into reaction systems presented in [21]).

In this paper we define a variant of Linear-Time Temporal Logic (rsLTL, for short) interpreted over models of reaction systems with discrete concentrations. We provide its suitable encoding in SMT, together with a bounded model checking method, and present experimental results showing the efficiency of verification for reaction systems with discrete concentrations.

The rest of the paper is organised as follows. In the next section, we recall some basic notions related to reaction systems. Then, we define reaction systems with discrete concentrations in Sec. 3, linear-time temporal logic interpreted over such reaction systems (Sec. 4), and bounded model checking via SMT-encoding (Sec. 5) followed by experimental results (Sec. 6). The final section contains some concluding remarks.

2. Preliminaries

A *reaction system* is a pair $rs = (S, A)$, where S is a finite *background* set and A is a set of *reactions* over the background set. Each reaction in A is a triple $b = (R, I, P)$ such that R, I, P are nonempty subsets of S with $R \cap I = \varnothing$. The set of all possible reactions over S is denoted $rac(S)$, and $A \subseteq rac(S)$. The sets R, I, and P are respectively denoted by R_b, I_b, and P_b and called the *reactant*, *inhibitor*, and *product set* of reaction b. A reaction $b \in A$ is *enabled* by $T \subseteq S$, denoted $en_b(T)$, if $R_b \subseteq T$ and $I_b \cap T = \varnothing$. The *result* of b on T is given by $res_b(T) = P_b$ if $en_b(T)$, and by $res_b(T) = \varnothing$ otherwise. Then the *result* of A on T is $res_A(T) = \bigcup\{res_b(T) \mid b \in A\} = \bigcup\{P_b \mid b \in A \text{ and } en_b(T)\}$.

Intuitively, T represents a state of a biochemical system being modelled by listing all present biochemical entities. A reaction b is enabled by T and can take place if all its reactants are present and none of its inhibitors is present in T. The system reaches the next state $T' = res_A(T)$ by executing the reactions enabled in T.

Example 2.1. Let $(S, A) = (\{1, 2, 3, 4\}, \{a_1, a_2, a_3, a_4\})$ be a reaction system, where:

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

In state $T = \{1, 3, 4\}$ reactions a_1, a_3, and a_4 are enabled, while a_2 is not. Hence $res_A(T) = res_{a_1}(T) \cup res_{a_3}(T) \cup res_{a_4}(T) = \{1, 2\} \cup \{1, 2\} \cup \{1\} = \{1, 2\}$. □

Entities in reaction systems are *non-permanent*, i.e., if entity x is present in the successor state T' of a current state T then it must have been produced (sustained) by a reaction enabled by T (thus $x \in res_A(T)$). Also, there are no conflicts between reactions enabled by T. Therefore there is no counting in reaction systems, and so it is a qualitative model. This follows from the level of abstraction adopted for the basic model. However, in the broad framework of reaction systems (see, e.g., [2]) one considers models with aspects of counting.

A reaction system is a finite system in the sense that the size of each state is a priori limited (by the size of the background set), and the state transformations it describes are deterministic since there are no conflicts between enabled reactions. This changes once we decided to take account of the external environment which is necessary to reflect the fact that the living cell is an open system. Such an environment can be represented by a context automaton.

A *context automaton* over a set Ct, is a triple $ca = (Q, q^{init}, R)$, where Q is a finite set of *states*, $q^{init} \in Q$ is the *initial state*, and $R \subseteq Q \times Ct \times Q$ is a *transition relation* labelled with elements of Ct. We assume that for all $q \in Q$ there exists $c \in Ct$ and $q' \in Q$ such that $(q, c, q') \in R$.

A *context restricted reaction system* is a pair $crrs = (rs, ca)$ such that $rs = (S, A)$ is a reaction system, and $ca = (Q, q^{init}, R)$ is a *context automaton* over 2^S. The dynamic behaviour of $crrs$ is then captured by the state sequences of its interactive processes. An *interactive process* in $crrs$ is $\pi = (\zeta, \gamma, \delta)$, where:

- $\zeta = (z_0, z_1, \ldots, z_n)$, $\gamma = (C_0, C_1, \ldots, C_n)$, and $\delta = (D_0, D_1, \ldots, D_n)$
- $z_0, z_1, \ldots, z_n \in Q$ with $z_0 = q^{init}$

- $C_0, C_1, \ldots, C_n, D_0, D_1, \ldots, D_n \subseteq S$ with $D_0 = \varnothing$
- $(z_i, C_i, z_{i+1}) \in R$, for every $i \in \{0, \ldots, n-1\}$
- $D_i = res_A(D_{i-1} \cup C_{i-1})$, for every $i \in \{1, \ldots, n\}$.

Then the *state sequence* of π is $\tau = (W_0, \ldots, W_n) = (C_0 \cup D_0, \ldots, C_n \cup D_n)$. In the above definition it is required that for each context set there exists in the automaton a transition from its current state.

Intuitively, the state sequence of π captures the observed behaviour of $crrs$ by recording the successive states of the evolution of the reaction system rs in the environment represented by the context automaton ca.

3. Reaction systems with discrete concentrations

The enabling of some of biochemical reactions encountered in practical applications depends not only on the availability of the necessary reactants and the absence of inhibitors, but also on their concentration levels. To address this aspect in biochemical modelling, we will now introduce an extension of the basic reaction systems supporting an explicit representation of the discrete concentration levels of entities. The resulting model uses multisets of entities, but otherwise it retains key features of the original framework. The main new idea is that the k-th level of concentration of an entity x is represented by a multiset containing k copies of x.

In what follows, a *multiset* over a set X is any mapping $\mathbf{b} : X \to \{0, 1, \ldots\}$, and the *empty* multiset $\varnothing_X$ is one which always returns 0. We denote this by $\mathbf{b} \in \mathcal{B}(X)$, where $\mathcal{B}(X)$ is the set of all multisets over X. For a set $\mathbf{B}$ of multisets over X, $\uplus(\mathbf{B})$ is the multiset over X such that $\uplus(\mathbf{B})(x) = \max(\{0\} \cup \{\mathbf{b}(x) \mid \mathbf{b} \in \mathbf{B}\})$, for every $x \in X$. For two multisets, $\mathbf{b}$ and $\mathbf{b}'$, we denote $\mathbf{b} \leq \mathbf{b}'$ if $\mathbf{b}(x) \leq \mathbf{b}'(x)$, for every $x \in X$. The *carrier* of a multiset $\mathbf{b}$ is the set $carr(\mathbf{b}) = \{x \in X \mid \mathbf{b}(x) > 0\}$.

A *reaction system with discrete concentrations* is a pair $rsc = (S, A)$, where S is a finite *background* set and A is a nonempty finite set of *c-reactions* over the background set. Each c-reaction in A is a triple $a = (\mathbf{r}, \mathbf{i}, \mathbf{p})$ such that $\mathbf{r}$, $\mathbf{i}$, $\mathbf{p}$ are multisets over S with $\mathbf{r}(e) < \mathbf{i}(e)$, for every $e \in carr(\mathbf{i})$. The multisets $\mathbf{r}$, $\mathbf{i}$, and $\mathbf{p}$ are respectively denoted by $\mathbf{r}_a$, $\mathbf{i}_a$, and $\mathbf{p}_a$ and called the *reactant*, *inhibitor*, and *product concentration levels* of c-reaction a. We would like to stress that an entity e is an inhibitor of a whenever $e \in carr(\mathbf{i}_a)$.

A c-reaction $a \in A$ is *enabled* by $\mathbf{t} \in \mathcal{B}(S)$, denoted $en_a(\mathbf{t})$, if $\mathbf{r}_a \leq \mathbf{t}$ and $\mathbf{t}(e) < \mathbf{i}_a(e)$, for every $e \in carr(\mathbf{i}_a)$. The *result* of a on $\mathbf{t}$ is given by $res_a(\mathbf{t}) = \mathbf{p}_a$ if $en_a(\mathbf{t})$, and by $res_a(\mathbf{t}) = \varnothing_S$ otherwise. Then the *result* of A on $\mathbf{t}$ is $res_A(\mathbf{t}) = \uplus\{res_a(\mathbf{t}) \mid a \in A\} = \uplus\{\mathbf{p}_a \mid a \in A \text{ and } en_a(\mathbf{t})\}$.

In the above, $\mathbf{t}$ is a *state* of a biochemical system being modelled such that, for each entity $e \in S$, $\mathbf{t}(e)$ is the *concentration level* of e (e.g., $\mathbf{t}(e) = 0$ indicates that e is not present in the current state, and $\mathbf{t}(e) = 1$ indicates that e is present at its lowest concentration level). A c-reaction a is enabled by $\mathbf{t}$ and can take place if the current concentration levels of all its reactants are at least as high as those specified by $\mathbf{r}_a$, and the current concentration levels of all its inhibitors (i.e., entities in the carrier of $\mathbf{i}_a$) are below the thresholds specified by $\mathbf{i}_a$.

A *context restricted reaction system with discrete concentrations* is a pair $crrsc = (rsc, ca)$ such that $rsc = (S, A)$ is a reaction system with discrete concentrations, and $ca = (Q, q^{init}, R)$ is a *context*

automaton over $\mathcal{B}(S)$. The dynamic behaviour of $crrsc$ is then captured by the state sequences of its interactive processes. An *interactive process* in $crrsc$ is $\pi = (\zeta, \gamma, \delta)$, where:

- $\zeta = (z_0, z_1, \ldots, z_n)$, $\gamma = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n)$, and $\delta = (\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n)$
- $z_0, z_1, \ldots, z_n \in Q$ with $z_0 = q^{init}$
- $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n, \mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n \in \mathcal{B}(S)$ with $\mathbf{d}_0 = \varnothing_{\mathcal{B}(S)}$
- $(z_i, \mathbf{c}_i, z_{i+1}) \in R$, for every $i \in \{0, \ldots, n-1\}$
- $\mathbf{d}_i = res_A(\mathbb{M}\{\mathbf{d}_{i-1}, \mathbf{c}_{i-1}\})$, for every $i \in \{1, \ldots, n\}$.

Then the *state sequence* of π is $\tau = (\mathbf{w}_0, \ldots, \mathbf{w}_n) = (\mathbb{M}\{\mathbf{c}_0, \mathbf{d}_0\}, \ldots, \mathbb{M}\{\mathbf{c}_n, \mathbf{d}_n\})$.

A context restricted reaction system with discrete concentrations $crrsc = (rsc, ca)$ is a finite state system since it comprises finitely many c-reactions and finitely many multisets labelling the arcs of its context automaton. More precisely, let $\#_{crrsc}(e)$ be the maximum integer assigned to $e \in S$ in all the multisets of entities occurring in both rsc and ca. Then, $\mathbf{w}(e) \leq \#_{crrsc}(e)$, for all $e \in S$ and all states occurring in the state sequences of the interactive processes in $crrsc$. (Note that this bound can be improved by ignoring the reactant and inhibitor multisets in c-reactions.) Moreover, the behaviour of $crrsc$ can be simulated by a suitable context restricted reaction system.

To construct such a system, for every $\mathbf{t} \in \mathcal{B}(S)$, we define two sets of entities, $\Gamma(\mathbf{t}) = \{e.i \mid e \in S \wedge \mathbf{t}(e) = i > 0\}$ and $\Gamma_{all}(\mathbf{t}) = \{e.i \mid e \in S \wedge 1 \leq i \leq \mathbf{t}(e)\}$. The $e.i$'s will be entities of the system we are going to construct. Note that $\Gamma_{all}(\mathbf{t})$ is a *downward-closed* set in the sense that if $e.i \in \Gamma_{all}(\mathbf{t})$ and $i > 1$, then $e.1, \ldots, e.(i-1) \in \Gamma_{all}(\mathbf{t})$. In fact, Γ_{all} is a bijection from $\mathcal{B}(S)$ to all the downward-closed sets, and its inverse Γ_{all}^{-1} is given by $\Gamma_{all}^{-1}(Z)(e) = \max\{\{0\} \cup \{i \mid e.i \in Z\}$, for every $e \in S$. In what follows, Γ_{all} and Γ_{all}^{-1} will be applied component-wise to sequences of respectively multisets and downward-closed sets. For such $crrsc$, we define the corresponding context restricted reaction system as $\Theta(crrsc) = (rs, ca) = ((S', A'), (Q, q^{init}, R'))$, where: $S' = \{e.i \mid e \in S \text{ and } 1 \leq i \leq \#_{crrsc}(e)\}$, $A' = \{(\Gamma(\mathbf{r}), \Gamma(\mathbf{i}), \Gamma_{all}(\mathbf{p})) \mid (\mathbf{r}, \mathbf{i}, \mathbf{p}) \in A\}$, and $R' = \{(z, \Gamma_{all}(\mathbf{c}), z') \mid (z, \mathbf{c}, z') \in R\}$. It is straightforward to see that $\Theta(crrsc)$ is well-defined.

As to the complexity of the translation, the number of reactions, states and arrows remains the same. Moreover, the representations of reactions and inhibitors are of the same order. What changes is the size of the background set, in the worst case by the factor $\max\{\#_{crrsc}(e) \mid e \in S\}$ as well as the representations of products and contexts (again by the same factor).

We will now investigate a very close correspondence between $\Theta(crrsc)$ and $crrsc$. First, we observe that, by the definitions of A' and R', all sets of entities occurring in the interactive processes of $\Theta(crrsc)$ are downward-closed. Then we obtain that all interactive processes of $crrsc$ can be simulated by $\Theta(crrsc)$.

Theorem 3.1. If $\pi = (\zeta, \gamma, \delta)$ is an interactive process in $crrsc$, then $\pi' = (\zeta, \Gamma_{all}(\gamma), \Gamma_{all}(\delta))$ is an interactive process in $\Theta(crrsc)$.

Proof:
It suffices to show for $\mathbf{w}$ in the state sequence of π, $\Gamma_{all}(res_A(\mathbf{w})) = res_{A'}(\Gamma_{all}(\mathbf{w}))$. Suppose $a = (\mathbf{r}, \mathbf{i}, \mathbf{p}) \in A$ and $a' = (\Gamma(\mathbf{r}), \Gamma(\mathbf{i}), \Gamma_{all}(\mathbf{p})) \in A'$. We first observe that a is enabled in $\mathbf{w}$ (i.e., $\mathbf{r} \leq \mathbf{w}$ and $\mathbf{w}(e) < \mathbf{i}(e)$, for all $e \in carr(\mathbf{i})$) iff a' is enabled in $\Gamma_{all}(\mathbf{w})$ (i.e., $\Gamma(\mathbf{r}) \subseteq \Gamma_{all}(\mathbf{w})$ and $\Gamma(\mathbf{i}) \cap \Gamma_{all}(\mathbf{w}) = \varnothing$). Moreover, it is easy to check that $\Gamma_{all}(res_a(\mathbf{w})) = res_{a'}(\Gamma_{all}(\mathbf{w}))$. □

Moreover, all interactive processes of $\Theta(crrsc)$ simulate those of $crrsc$.

Theorem 3.2. If $\pi = (\zeta, \gamma, \delta)$ is an interactive process in $\Theta(crrsc)$, then $\pi' = (\zeta, \Gamma_{all}^{-1}(\gamma), \Gamma_{all}^{-1}(\delta))$ is an interactive process in $crrsc$.

The proof of Theorem 3.2 is similar to the proof of Theorem 3.1. We have therefore obtained a one-to-one correspondence between the interactive processes of $\Theta(crrsc)$ and $crrsc$.

Remark 3.3. From the point of view of enabling c-reactions, not all concentration levels are important and, consequently, they do not need to be represented in the states of $\Theta(crrsc)$. To achieve the desired effect, all one needs to do is re-define Γ_{all}, in the following way: $\Gamma'_{all}(\mathbf{t}) = \Gamma(\mathbf{t}) \cup (\Gamma_{all}(\mathbf{t}) \cap \bigcup_{a \in A} \Gamma(\mathbf{r}_a) \cup \Gamma(\mathbf{i}_a))$.

Note that syntactically crrs are a subclass of crrsc, such that all the concentration levels in crrsc are limited to the value of at most one, that is, for any $\mathbf{t} \in \mathcal{B}(S)$ and for any $e \in carr(\mathbf{t})$ we have $\mathbf{t}(e) = 1$.

When dealing with concentration levels we often need to perform incrementation and decrementation operations. For this we need an additional notation: in the remainder of this paper we use the notation $e \mapsto i$ to indicate the multiplicity of an entity e in a multiset of entities, e.g., $\{e \mapsto 1, f \mapsto 2\}$ is a multiset with one copy of e, two copies of f, and nothing else.

4. Linear-time temporal logic for reaction systems

In this section we demonstrate how linear-time temporal logic can be used to express properties of reaction systems. Firstly, for the convenience of specifying multisets over a given set S we introduce the following grammar of *multiset expressions*: $\mathfrak{a} ::= true \mid e \sim c \mid e \sim e \mid \neg\mathfrak{a} \mid \mathfrak{a} \vee \mathfrak{a}$, where $\sim \in \{<, \leq, =, \geq, >\}$, $e \in S$, and $c \in \mathbb{N}$. The set of all the multiset expressions over S is denoted by $BE(S)$. Let $\mathbf{b}$ be a multiset over S. The fact that $\mathfrak{a}$ holds in $\mathbf{b}$ is denoted by $\mathbf{b} \models_b \mathfrak{a}$, where the relation $\models_b$ is defined recursively as follows:

$$\begin{array}{llll} \mathbf{b} \models_b true & \text{iff} & \text{for any } \mathbf{b} \in \mathcal{B}(S), & \mathbf{b} \models_b e_1 \sim c \quad \text{iff} \quad \mathbf{b}(e_1) \sim c, \\ \mathbf{b} \models_b e_1 \sim e_2 & \text{iff} & \mathbf{b}(e_1) \sim \mathbf{b}(e_2), & \mathbf{b} \models_b \neg\mathfrak{a} \quad \text{iff} \quad \mathbf{b} \not\models_b \mathfrak{a}, \\ \mathbf{b} \models_b \mathfrak{a}_1 \vee \mathfrak{a}_2 & \text{iff} & \mathbf{b} \models_b \mathfrak{a}_1 \text{ or } \mathbf{b} \models_b \mathfrak{a}_2. & \end{array}$$

Next, we derive the conjunction operator: $\mathfrak{a}_1 \wedge \mathfrak{a}_2 \stackrel{def}{\equiv} \neg(\neg\mathfrak{a}_1 \vee \neg\mathfrak{a}_2)$. Notice that for $\sim$ the entire set of relations is not required since we can use the logical operators to obtain the same expressiveness with a minimal set of those operators.

The language of *Reaction Systems Linear-Time Temporal Logic* (rsLTL, for short) is defined by the following grammar: $\phi ::= \mathfrak{a} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathbf{X}_\mathfrak{a}\phi \mid \phi \mathbf{U}_\mathfrak{a} \phi \mid \phi \mathbf{R}_\mathfrak{a} \phi$, where $\mathfrak{a} \in BE(S)$.

The temporal operators are used to express requirements imposed on a path. The $\mathbf{X}\phi$ operator means 'in the next state ϕ holds'. The $\phi_1 \mathbf{U} \phi_2$ operator uses two properties and means 'ϕ_2 holds eventually, and ϕ_1 must hold at every preceding state'. The $\phi_1 \mathbf{R} \phi_2$ operator means 'ϕ_2 holds up to and including the first state where ϕ_1 holds'.

For a given $crrsc$ we define its model, which is then used to formally define the semantics of the introduced operators.

Definition 4.1. Let $crrsc = (rsc, ca)$, where $rsc = (S, A)$ is a reaction system with discrete concentrations, and $ca = (Q, q_{init}, R)$ is a *context automaton* over $\mathcal{B}(S)$. Then, the *model* for $crrsc$ is a quadruple $\mathcal{M} = (\mathbb{W}, \mathtt{w}_0, \longrightarrow, L)$, where:

1. $\mathbb{W} = \mathcal{B}(S) \times Q$ is the set of states,
2. $\mathtt{w}^{init} = (\varnothing, q^{init})$ is the initial state,
3. $\longrightarrow \subseteq \mathbb{W} \times \mathcal{B}(S) \times \mathbb{W}$ is the transition relation such that for all $\mathbf{w}, \mathbf{w}', \alpha \in \mathcal{B}(S)$, $q, q' \in Q$: $\big((\mathbf{w}, q), \alpha, (\mathbf{w}', q')\big) \in \longrightarrow$ iff: $(q, \alpha, q') \in R$ and $\mathbf{w}' = res_A(\mathbb{M}\{\mathbf{w}, \alpha\})$. Each element $(\mathtt{w}, \alpha, \mathtt{w}') \in \longrightarrow$ is denoted $\mathtt{w} \xrightarrow{\alpha} \mathtt{w}'$.

The paths in rsLTL contain additional elements which represent context multisets. The paths are defined as sequences of states interleaved with actions, i.e., the context multisets.

Definition 4.2. A *path* is an infinite sequence $\sigma = (\mathtt{w}_0, \alpha_0, \mathtt{w}_1, \alpha_1, \dots)$ of states and actions such that: $\mathtt{w}_i \xrightarrow{\alpha_i} \mathtt{w}_{i+1}$ and $\alpha_i \in \mathcal{B}(S)$ for $i \geq 0$.

Let σ be a path. For each $i \geq 0$, the i-th state $\mathtt{w}_i$ of the path σ is denoted by $\sigma_s(i)$, and the i-th action α_i of the path σ is denoted by $\sigma_a(i)$. Let $\sigma_s(i) = (\mathbf{w}_i, q_i)$ for each $i \geq 0$. Then, with $\sigma_b(i)$ and $\sigma_{ca}(i)$ we denote $\mathbf{w}_i$ and q_i, respectively. Let $i \geq 0$, then by σ^i we denote the suffix of σ such that $\sigma^i = (\sigma_s(i), \sigma_a(i), \sigma_s(i+1), \sigma_a(i+1), \dots)$, i.e., $\sigma^i_s(j) = \sigma_s(j+i)$ and $\sigma^i_a(j) = \sigma_a(j+i)$ for each $j \geq 0$. By $\Pi_{\mathcal{M}}$ we denote the set of all the paths of the model $\mathcal{M}$, whereas by $\Pi_{\mathcal{M}}(\mathtt{w})$ we denote the set of all the paths that start in $\mathtt{w} \in \mathbb{W}$, that is, $\Pi_{\mathcal{M}}(\mathtt{w}) = \{\sigma \in \Pi \mid \sigma_s(0) = \mathtt{w}\}$.

Definition 4.3. Let $\mathcal{M} = (\mathbb{W}, \mathtt{w}^{init}, \longrightarrow, L)$ be a model and $\sigma \in \Pi_{\mathcal{M}}$ be a path of $\mathcal{M}$. The fact that ϕ holds over σ is denoted by $\mathcal{M}, \sigma \models \phi$ (or $\sigma \models \phi$ if $\mathcal{M}$ is implicitly understood), where the relation $\models$ is defined recursively as follows:

$$
\begin{array}{lll}
\sigma \models \mathfrak{a} & \text{iff} & \sigma_b(0) \models_b \mathfrak{a}, \\
\sigma \models \phi_1 \vee \phi_2 & \text{iff} & \sigma \models \phi_1 \text{ or } \sigma \models \phi_2, \\
\sigma \models \phi_1 \wedge \phi_2 & \text{iff} & \sigma \models \phi_1 \text{ and } \sigma \models \phi_2, \\
\sigma \models \mathbf{X}_{\mathfrak{a}} \phi_1 & \text{iff} & \sigma_a(0) \models_b \mathfrak{a} \text{ and } \sigma^1 \models \phi_1, \\
\sigma \models \phi_1 \mathbf{U}_{\mathfrak{a}} \phi_2 & \text{iff} & (\exists j \geq 0)\big(\sigma^j \models \phi_2 \text{ and } (\forall 0 \leq l < j)(\sigma^l \models \phi_1 \text{ and } \sigma_a(l) \models_b \mathfrak{a})\big), \\
\sigma \models \phi_1 \mathbf{R}_{\mathfrak{a}} \phi_2 & \text{iff} & (\forall j \geq 0)\big(\sigma^j \models \phi_2 \text{ or } (\exists 0 \leq l < j)(\sigma^l \models \phi_1 \text{ and } (\forall 0 \leq m < l)(\sigma_a(m) \models_b \mathfrak{a}))\big).
\end{array}
$$

Next, we derive the following operators: $\mathfrak{a} \Rightarrow \phi \stackrel{def}{\equiv} \neg\mathfrak{a} \vee \phi$, $\mathbf{G}_{\mathfrak{a}}\phi \stackrel{def}{\equiv} false\mathbf{R}_{\mathfrak{a}}\phi$, $\mathbf{F}_{\mathfrak{a}}\phi \stackrel{def}{\equiv} true\mathbf{U}_{\mathfrak{a}}\phi$. Moreover, we assume $\mathfrak{a} = true$ when $\mathfrak{a}$ is unspecified for any of the rsLTL operators, e.g., $\mathbf{F}\phi$ is the same as $\mathbf{F}_{true}\phi$.

An rsLTL formula holds in a model iff it holds in all the paths starting in its initial state, i.e., $\mathcal{M} \models \phi$ iff $\sigma \models \phi$ for all $\sigma \in \Pi_{\mathcal{M}}(\mathtt{w}^{init})$. Additionally, a formula may also hold existentially in a model, i.e., $\mathcal{M} \models_{\exists} \phi$ iff there exists $\sigma \in \Pi_{\mathcal{M}}(\mathtt{w}^{init})$ s.t. $\sigma \models \phi$.

Given an rsLTL formula and a model $\mathcal{M}$, the decision problem of checking if $\mathcal{M} \models \phi$, is the model checking problem for rsLTL.

Example 4.4. We consider an abstract system $rsc = (\{x, y, h, m\}, \{a_1, a_2, \ldots, a_6\})$, where:

$$a_1 = (\{y \mapsto 1, x \mapsto 1\}, \{y \mapsto 2, h \mapsto 1\}, \{y \mapsto 2\}), \quad a_2 = (\{y \mapsto 2, x \mapsto 2\}, \{y \mapsto 3, h \mapsto 1\}, \{y \mapsto 3\}),$$
$$a_3 = (\{y \mapsto 3, h \mapsto 1\}, \{y \mapsto 4, h \mapsto 2\}, \{y \mapsto 4\}), \quad a_4 = (\{y \mapsto 4, h \mapsto 1\}, \{h \mapsto 2\}, \{y \mapsto 3\}),$$
$$a_5 = (\{y \mapsto 4, x \mapsto 2\}, \{h \mapsto 1\}, \{y \mapsto 2\}), \quad a_6 = (\{m \mapsto 1\}, \{y \mapsto 3\}, \{m \mapsto 1\}).$$

We define a context automaton $ca = (\{0, 1\}, 0, \{r_1, r_2, \ldots, r_7\})$, where:

$$r_1 = (0, \{x \mapsto 1, y \mapsto 1, m \mapsto 1\}, 1), \quad r_2 = (1, \{x \mapsto 1\}, 1), \quad r_3 = (1, \{x \mapsto 2\}, 1),$$
$$r_4 = (1, \{x \mapsto 1, h \mapsto 1\}, 1), \quad r_5 = (1, \{x \mapsto 2, h \mapsto 1\}, 1), \quad r_6 = (1, \{h \mapsto 1\}, 1),$$
$$r_7 = (1, \{h \mapsto 2\}, 1).$$

Finally, we define $crrsc = (rsc, ca)$. Intuitively, the system produces the entities y and m at different concentration levels. If there is y present with the concentration level of exactly one unit, the entity x is provided, and h is not present, then the concentration level of y is increased by one unit, i.e., $y \mapsto 2$ is produced. In the next step, the concentration of y is increased further, but the concentration of x is required to be at the level of two units. The level of y increases from three to four units when the level of h is exactly one unit. Then, if h is being continuously provided at the level of exactly one unit the concentration of y oscillates between four and three units. When y is present at the concentration level of at least four units, x is at the concentration level of two units, and h is not provided, the level of y drops to two units. Additionally, when m is provided, it is sustained at the level of one unit unless y reaches the level of three units.

Let $\mathcal{M}$ be the model for crrsc. We formulate the following rsLTL properties interpreted in $\mathcal{M}$:

$$\phi_1 = \mathbf{G}_{x>0}((y = 2) \Rightarrow \mathbf{X}_{x>1}(y \geq 3)), \quad \phi_2 = \mathbf{F}_{x>0}((y = 2) \wedge \mathbf{X}_{x>1}(y < 3)),$$
$$\phi_3 = \mathbf{X}((y = 3)\mathbf{R}(m \geq 1)).$$

The formula ϕ_1 states that, globally, when $x > 0$ is supplied in the context (by the context automaton), and if $y = 2$ then in the next state $y \geq 3$, if $x > 1$ is supplied in the context. This property holds existentially in the model, i.e., $\mathcal{M} \models_\exists \phi_1$. However, it does not hold universally, i.e., $\mathcal{M} \not\models \phi$. This follows from the fact that $\phi_2 \equiv \neg\phi_1$ and $\mathcal{M} \models_\exists \phi_2$, that is, ϕ_2 expresses the property for a counterexample of ϕ_1.

The property described by ϕ_3 holds in a path where: when $y = 3$ holds, it releases $m \geq 1$ from holding, otherwise $m \geq 1$ is required to hold. The release property is required to hold after one step, i.e., by using the $\mathbf{X}$ operator we skip the first step of the path where $\sigma_b(0) = \varnothing$.

4.1. Bounded semantics

Motivated by various successful applications of bounded model checking to practical problems such as software verification [22], in this paper we focus on the bounded model checking approach defined for finite prefixes of paths. This approach requires us to specify when a given formula holds while considering only a finite number of states and actions that belong to the prefix of the considered path.

Definition 4.5. A path $\sigma = (\mathtt{w}_0, \alpha_0, \mathtt{w}_1, \alpha_1, \ldots)$ is a (k, l)-*loop* (or k-*loop*) if there exist $k \geq l > 0$ such that $\mathtt{w}_{l-1} = \mathtt{w}_k$ and $\sigma = (\mathtt{w}_0, \alpha_0, \ldots, \alpha_{l-2}, \mathtt{w}_{l-1})(\alpha_l, \mathtt{w}_{l+1}, \alpha_{l+1}, \ldots, \alpha_{k-1}, \mathtt{w}_k)^\omega$.

The bounded semantics for rsLTL is defined for finite path prefixes. We define a satisfiability relation that for a given path considers its first k states and $k-1$ actions only.

Definition 4.6. The fact that a formula ϕ holds in a path σ with bound $k \in \mathbb{N}$ is denoted by $\sigma \models^k \phi$. Then, $\sigma \models^k \phi$ if and only if:

- σ is a (k,l)-loop for some $0 < l \leq k$ and $\sigma \models \phi$, or
- $\sigma \models_{nl} \phi$, where:

$$
\begin{array}{lll}
\sigma \models_{nl} \mathfrak{a} & \text{iff} & \sigma_s(0) \models_b \mathfrak{a}, \\
\sigma \models_{nl} \phi_1 \wedge \phi_2 & \text{iff} & \sigma \models_{nl} \phi_1 \text{ and } \sigma \models_{nl} \phi_2, \\
\sigma \models_{nl} \phi_1 \vee \phi_2 & \text{iff} & \sigma \models_{nl} \phi_1 \text{ or } \sigma \models_{nl} \phi_2, \\
\sigma \models_{nl} \mathbf{X}_{\mathfrak{a}} \phi & \text{iff} & k > 0, \sigma_a(0) \models_b \mathfrak{a}, \text{ and } \sigma^1 \models_{nl} \phi, \\
\sigma \models_{nl} \phi_1 \mathbf{U}_{\mathfrak{a}} \phi_2 & \text{iff} & (\exists 0 \leq j \leq k)\big(\sigma^j \models_{nl} \phi_2 \text{ and } (\forall 0 \leq l < j)(\sigma^l \models_{nl} \phi_1 \text{ and } \sigma_a(l) \models_b \mathfrak{a})\big) \\
\sigma \models_{nl} \phi_1 \mathbf{R}_{\mathfrak{a}} \phi_2 & \text{iff} & (\exists 0 \leq j \leq k)\big(\sigma^j \models_{nl} \phi_1 \text{ and } ((\forall 0 \leq l \leq j)(\sigma^l \models_{nl} \phi_2) \\
 & & \quad \text{and } (\forall 0 \leq l < j)(\sigma_a(l) \models_b \mathfrak{a}))\big)
\end{array}
$$

Lemma 4.7. Let $k \in \mathbb{N}$, ϕ be an rsLTL formula, and σ be a path. Then, $\sigma \models^k \phi$ implies $\sigma \models \phi$.

Lemma 4.8. Let ϕ be an rsLTL formula and $\mathcal{M}$ be a model. Then, $\mathcal{M} \models \phi$ implies that there exists $k \in \mathbb{N}$ such that $\mathcal{M} \models^k \phi$.

The proofs for these lemmas are similar to the ones for LTL [23]. The only difference in these proofs is related to the augmented temporal operators which impose additional restrictions on the considered path by using multiset expressions.

For a bound $k \in \mathbb{N}$ we define the relation $\models^k_\exists$ for models as follows: $\mathcal{M} \models^k_\exists \phi$ iff there exists $\sigma \in \Pi_{\mathcal{M}}(\mathtt{w}^{init})$ s.t. $\sigma \models^k \phi$. The *bounded model checking problem* for rsLTL is defined as the decision problem of checking if $\mathcal{M} \models^k_\exists \phi$ for a given bound $k \in \mathbb{N}$.

Based on Lemma 4.7 and 4.8 we state the following theorem:

Theorem 4.9. Let ϕ be an rsLTL formula and $\mathcal{M}$ be a model. Then, $\mathcal{M} \models_\exists \phi$ iff there exists $k \in \mathbb{N}$ such that $\mathcal{M} \models^k_\exists \phi$.

5. SMT-based encoding

In this section we provide a translation of the bounded model checking problem for rsLTL into the satisfiability modulo theory (SMT) [22] with the integer arithmetic theory. The SMT problem is a generalisation of the Boolean satisfiability problem, where some functions and predicate symbols have interpretations from the underlying theory.

Let $crrsc = ((S, A), (Q, q^{init}, R))$ and $\mathcal{M}$ be the model for $crrsc$. For an integer $k \geq 0$ we aim to encode all the paths of the model $\mathcal{M}$ that are bounded with k. The entities of S are denoted by $e_1, \ldots, e_m$, where $m = |S|$. We introduce the following sets of positive integer variables used in the encoding: $\mathbf{P} = \bigcup_{i=0}^{k} \{\mathtt{p}_{i,1}, \ldots, \mathtt{p}_{i,m}\}$, $\mathbf{P}^{\mathcal{E}} = \bigcup_{i=0}^{k} \{\mathtt{p}^{\mathcal{E}}_{i,1}, \ldots, \mathtt{p}^{\mathcal{E}}_{i,m}\}$, and $\mathbf{Q} = \{\mathtt{q}_0, \ldots, \mathtt{q}_k\}$. Let σ be a path of $\mathcal{M}$. Then, $\overline{\mathtt{p}}_i = (\mathtt{p}_{i,1}, \ldots, \mathtt{p}_{i,m})$ and $\overline{\mathtt{p}}^{\mathcal{E}}_i = (\mathtt{p}^{\mathcal{E}}_{i,1}, \ldots, \mathtt{p}^{\mathcal{E}}_{i,m})$ encode the state $\sigma_s(i)$, i.e.,

$\sigma_a(i)$ and $\sigma_b(i)$, respectively. The action $\sigma_a(i)$ is encoded with $\overline{\mathtt{p}}_i^{\mathcal{E}} = (\mathtt{p}_{i,1}^{\mathcal{E}}, \ldots, \mathtt{p}_{i,m}^{\mathcal{E}})$. With $\overline{\mathtt{p}}_i[j]$ and $\overline{\mathtt{p}}_i^{\mathcal{E}}[j]$ we denote, respectively, $\mathtt{p}_{i,j}$ and $\mathtt{p}_{i,j}^{\mathcal{E}}$.

We define the following functions that map background set entities to the corresponding variables of the encoding: for all $0 \leq i \leq k$ we define $\mathtt{t}_i : S \to \mathbf{P}_i$ and $\mathtt{t}_i^{\mathcal{E}} : S \to \mathbf{P}_i^{\mathcal{E}}$ such that $\mathtt{t}_i(e_j) = \mathtt{p}_{i,j}$, $\mathtt{t}_i^{\mathcal{E}}(e_j) = \mathtt{p}_{i,j}^{\mathcal{E}}$ for all $1 \leq j \leq m$. The function $\mathtt{e} : Q \to \{0, \ldots, |Q| - 1\}$ maps states of the context automaton to the corresponding natural values used in the encoding. The set of the reactions that produce $e \in S$ is defined as $Prod(e) = \{a \in A \mid \mathbf{p}_a(e) > 0\}$.

To define the SMT encoding of the paths we need auxiliary functions that correspond to elements of the encoding.

Initial state: $\mathsf{Init}(\overline{\mathtt{p}}_i, \mathtt{q}_i) = \bigwedge_{e \in S}(\mathtt{t}_i(e) = 0) \wedge (\mathtt{q}_0 = \mathtt{e}(q^{init}))$ encodes the initial state of the model, where all the concentration levels are set to zero, and the context automaton is in its initial state.

Context: $\mathsf{Ct}_{\mathbf{c}_i}(\overline{\mathtt{p}}_i^{\mathcal{E}}) = \bigwedge_{e \in S}(\mathtt{t}_i^{\mathcal{E}}(e) = \mathbf{c}_i(e))$ encodes a multiset $\mathbf{c}_i \in \mathcal{B}(S)$ of context entities.

Enabledness: $\mathsf{En}_a\big(\overline{\mathtt{p}}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}\big) = \bigwedge_{e \in S}(\mathtt{t}_i(e) \geq \mathbf{r}_a(e) \vee \mathtt{t}_i^{\mathcal{E}}(e) \geq \mathbf{r}_a(e)) \quad \wedge$
$\bigwedge_{e \in S}(\mathtt{t}_i(e) < \mathbf{i}_a(e) \wedge \mathtt{t}_i^{\mathcal{E}}(e) < \mathbf{i}(e))$ encodes the enabledness of a reaction a.

Entity concentration: Let f_1, f_2, f_3 be expressions over $\mathbf{P} \cup \mathbf{P}^{\mathcal{E}}$, then we introduce the *if-then-else* operator: $f_1 \to f_2 \mid f_3 = (f_1 \wedge f_2) \vee (\neg f_1 \wedge f_3)$. Let $e \in S$, then $Prod^{sorted}(e) = (a_1, a_2, \ldots, a_w)$ is an ordered list of the reactions producing e, where $w = |Prod(e)|$ and $\mathbf{p}_{a_j} \leq \mathbf{p}_{a_{j+1}}$ for all $1 \leq j < w$. The produced concentration level for entity e, reaction a_j, and $1 \leq j \leq w$, is encoded as: $\mathsf{C}_e^j\big(\overline{\mathtt{p}}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}, \overline{\mathtt{p}}_{i+1}\big) = \mathsf{En}_{a_j}\big(\overline{\mathtt{p}}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}\big) \to \mathtt{t}_{i+1}(e) = \mathbf{p}_{a_j} \mid \mathsf{C}_e^{j+1}\big(\overline{\mathtt{p}}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}, \overline{\mathtt{p}}_{i+1}\big)$ if $j < w$, and $\mathsf{En}_{a_j}\big(\overline{\mathtt{p}}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}\big) \wedge \mathtt{t}_{i+1}(e) = \mathbf{p}_{a_j}$ if $j = w$. Finally, we define the complete entity concentration encoding for all the reactions. If $w = 0$, then $\mathsf{C}_e\big(\overline{\mathtt{p}}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}, \overline{\mathtt{p}}_{i+1}\big) = (\mathtt{t}_{i+1}(e) = 0)$, otherwise $\mathsf{C}_e(\overline{\mathtt{p}}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}, \overline{\mathtt{p}}_{i+1}) = \mathsf{C}_e^1(\overline{\mathtt{p}}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}, \overline{\mathtt{p}}_{i+1}) \vee ((\bigwedge_{a \in Prod(e)} \neg \mathsf{En}_a(\overline{\mathtt{p}}_i, \overline{\mathtt{p}}_i^{\mathcal{E}})) \wedge \mathtt{t}_{i+1}(e) = 0)$.

Transitions of context automaton: The encoding of the transition relation of the context automaton is a disjunction of the encodings for each transition:

$$\mathsf{Tr}_{ca}(\mathtt{q}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}, \mathtt{q}_{i+1}) = \bigvee_{(q, \mathbf{c}, q') \in R} (\mathtt{q}_i = \mathtt{e}(q) \wedge \mathsf{Ct}_{\mathbf{c}}(\overline{\mathtt{p}}_i^{\mathcal{E}}) \wedge \mathtt{q}_{i+1} = \mathtt{e}_{i+1}(q')).$$

Transition relation: We build a conjunction of the produced concentration levels for all entities and the transition relation for the context automaton to encode the transition relation of the model:

$$\mathsf{Tr}_{rsc}(\overline{\mathtt{p}}_i, \mathtt{q}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}, \overline{\mathtt{p}}_{i+1}, \mathtt{q}_{i+1}) = (\bigwedge_{e \in S} \mathsf{C}_e(\overline{\mathtt{p}}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}, \overline{\mathtt{p}}_{i+1})) \wedge \mathsf{Tr}_{ca}(\mathtt{q}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}, \mathtt{q}_{i+1}).$$

Paths: To encode the paths of $\mathcal{M}$ that are bounded with k we unroll the transition relation up to k and combine it with the encoding of the initial state of the model:

$$\mathsf{Paths}^k = \mathsf{Init}(\overline{\mathtt{p}}_0, \mathtt{q}_0) \wedge \bigwedge_{i=0}^{k-1} \mathsf{Tr}_{rsc}(\overline{\mathtt{p}}_i, \mathtt{q}_i, \overline{\mathtt{p}}_i^{\mathcal{E}}, \overline{\mathtt{p}}_{i+1}, \mathtt{q}_{i+1}).$$

Next, we present a translation of the rsLTL formulae into an SMT encoding. The rsLTL encoding is based on the fixed point encoding for LTL presented in [24].

In rsLTL, in place of propositional variables appearing in standard LTL formulae, we use multiset expressions. Let $\mathfrak{a}$ be a multiset expression, $enc_i^b(\mathfrak{a})$ and $enc_i^{ct}(\mathfrak{a})$ denote the encoding of $\mathfrak{a}$ using the variables of, respectively, $\overline{\mathtt{p}}_i$ and $\overline{\mathtt{p}}_i^{\mathcal{E}}$. The former refers to states of rsc, while the latter refers to actions. Since we are defining a translation into SMT those encodings are defined in a straightforward way. To deal with (k,l)-loops we introduce an integer variable $\mathtt{L}$. When $\mathtt{L} = l$ holds for a path then the path is a (k,l)-loop: $\mathsf{Loops}^k(\mathtt{L}) = \neg(\mathtt{L}=0) \wedge \bigwedge_{i=1}^{k} \big((\mathtt{L}=i) \Rightarrow \mathsf{E}(\overline{\mathtt{p}}_{i-1}, \mathtt{q}_{i-1}, \overline{\mathtt{p}}_k, \mathtt{q}_k)\big)$, where E encodes the equivalence of two states of the model: $\mathsf{E}(\overline{\mathtt{p}}_i, \mathtt{q}_i, \overline{\mathtt{p}}_j, \mathtt{q}_j) = (\mathtt{q}_i = \mathtt{q}_j) \wedge \bigwedge_{c=1}^{m}(\overline{\mathtt{p}}_i[c] = \overline{\mathtt{p}}_j[c])$. The encoding of an rsLTL formula ϕ at the position $i \in \{0,\ldots,k\}$ is defined as $[\![\phi]\!]_i^k$. Firstly, we introduce the encoding for propositional formulae:

$[\![\phi]\!]_i^k$	$0 \le i \le k$
$[\![\mathfrak{a}]\!]_i^k$	$enc_i^b(\mathfrak{a})$
$[\![\phi_1 \wedge \phi_2]\!]_i^k$	$[\![\phi_1]\!]_i^k \wedge [\![\phi_2]\!]_i^k$
$[\![\phi_1 \vee \phi_2]\!]_i^k$	$[\![\phi_1]\!]_i^k \vee [\![\phi_2]\!]_i^k$

Next, we define the encoding for temporal formulae. The translations of the until and release operators are based on the fixed point encoding for CTL [25]. The encoding introduces an auxiliary translation $\langle\!\langle\phi\rangle\!\rangle_i^k$ which corresponds to computing fixed point approximations.

$[\![\phi]\!]_i^k$	$0 \le i < k$
$[\![\mathbf{X}_{\mathfrak{a}}\phi_1]\!]_i^k$	$[\![\phi_1]\!]_{i+1}^k \wedge enc_i^{ct}(\mathfrak{a})$
$[\![\phi_1 \mathbf{U}_{\mathfrak{a}} \phi_2]\!]_i^k$	$[\![\phi_2]\!]_i^k \vee \big([\![\phi_1]\!]_i^k \wedge ([\![\phi_1 \mathbf{U}_{\mathfrak{a}}\phi_2]\!]_{i+1}^k \wedge enc_i^{ct}(\mathfrak{a}))\big)$
$[\![\phi_1 \mathbf{R}_{\mathfrak{a}} \phi_2]\!]_i^k$	$[\![\phi_2]\!]_i^k \wedge \big([\![\phi_1]\!]_i^k \vee ([\![\phi_1 \mathbf{R}_{\mathfrak{a}}\phi_2]\!]_{i+1}^k \wedge enc_i^{ct}(\mathfrak{a}))\big)$
	$i = k$
$[\![\mathbf{X}_{\mathfrak{a}}\phi_1]\!]_i^k$	$\bigvee_{j=1}^{k}((\mathtt{L}=j) \wedge [\![\phi_1]\!]_j^k) \wedge enc_i^{ct}(\mathfrak{a})$
$[\![\phi_1 \mathbf{U}_{\mathfrak{a}} \phi_2]\!]_i^k$	$[\![\phi_2]\!]_i^k \vee \big([\![\phi_1]\!]_i^k \wedge (\bigvee_{j=1}^{k}((\mathtt{L}=j) \wedge \langle\!\langle\phi_1 \mathbf{U}_{\mathfrak{a}}\phi_2\rangle\!\rangle_{i+1}^k) \wedge enc_i^{ct}(\mathfrak{a}))\big)$
$[\![\phi_1 \mathbf{R}_{\mathfrak{a}} \phi_2]\!]_i^k$	$[\![\phi_2]\!]_i^k \wedge \big([\![\phi_1]\!]_i^k \vee (\bigvee_{j=1}^{k}((\mathtt{L}=j) \wedge \langle\!\langle\phi_1 \mathbf{R}_{\mathfrak{a}}\phi_2\rangle\!\rangle_{i+1}^k) \wedge enc_i^{ct}(\mathfrak{a}))\big)$
$\langle\!\langle\phi\rangle\!\rangle_i^k$	$0 \le i < k$
$\langle\!\langle\phi_1 \mathbf{U}_{\mathfrak{a}} \phi_2\rangle\!\rangle_i^k$	$[\![\phi_2]\!]_i^k \vee \big([\![\phi_1]\!]_i^k \wedge (\langle\!\langle\phi_1 \mathbf{U}_{\mathfrak{a}}\phi_2\rangle\!\rangle_{i+1}^k \wedge enc_i^{ct}(\mathfrak{a}))\big)$
$\langle\!\langle\phi_1 \mathbf{R}_{\mathfrak{a}} \phi_2\rangle\!\rangle_i^k$	$[\![\phi_2]\!]_i^k \wedge \big([\![\phi_1]\!]_i^k \vee (\langle\!\langle\phi_1 \mathbf{R}_{\mathfrak{a}}\phi_2\rangle\!\rangle_{i+1}^k \wedge enc_i^{ct}(\mathfrak{a}))\big)$
	$i = k$
$\langle\!\langle\phi_1 \mathbf{U}_{\mathfrak{a}} \phi_2\rangle\!\rangle_i^k$	$[\![\phi_2]\!]_i^k$
$\langle\!\langle\phi_1 \mathbf{R}_{\mathfrak{a}} \phi_2\rangle\!\rangle_i^k$	$[\![\phi_2]\!]_i^k$

The cases for $[\![\phi]\!]_i^k$ when $i = k$ are considered separately: additional transitions for $j \in \{1,\ldots,k\}$ are encoded when (k,j)-loop exists, i.e., when $\mathtt{L} = j$ holds. In contrast to the LTL encoding of [24], we require for all the transitions to be constrained by the parameter $\mathfrak{a}$ encoded with $enc_i^{ct}(\mathfrak{a})$.

Finally, the bounded model checking problem for rsLTL is reduced to satisfiability checking, i.e., to verify if $\mathcal{M} \models_{\exists}^k \phi$ we check the satisfiability of the following formula: $\mathsf{Paths}^k \wedge \mathsf{Loops}^k \wedge [\![\phi]\!]_0^k$.

6. Experimental evaluation

In this section we present the results of an experimental evaluation of the translation presented in Section 5. The verification tool was implemented in Python and uses Z3 [26] for SMT-solving. We

implement an incremental approach, i.e., in a single SMT instance we increase the length of the encoded interactive processes by unrolling their encoding until a witness for the verified property is found, instead of creating separate instances for each length tested.

Additionally, we compare the implementation for crrsc with an implementation for crrs by verifying reachability properties of the crrs obtained by applying the translation defined in Section 3 to crrsc. To provide a fair comparison, both the verification approaches were implemented in Python using similar techniques. The implementation for crrs is based on the encoding from Section 5 which is optimised for crrs by using boolean variables instead of integer variables. The translation into SMT for crrs corresponds to the translation for crrsc – it is assumed that all concentration levels are equal to 1 when an entity is present, and equal to 0 otherwise.

The *k-reachability* is defined for a pair $\rho = (\mathbf{x}, \mathbf{y})$ where $\mathbf{x}, \mathbf{y} \in \mathcal{B}(S)$. We say that ρ is k-reachable if there exists an interactive process $\pi = (\zeta, \gamma, \delta)$ in $crrsc$ such that $\delta = (\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_k)$, and $\mathbf{x} \leq \mathbf{d}_k$, $\mathbf{d}_k(e) < \mathbf{y}(e)$, for every $e \in carr(\mathbf{y})$.

Incrementation and decrementation operations. With $\uparrow_e^g$ and $\downarrow_e^g$ we denote the set of reactions encoding the operation of, respectively, incrementation and decrementation of concentration levels of $e \in S$ when $g \in S$ is present with a non-zero concentration. With M_e we denote the maximal allowed value of e. Then $\uparrow_e^g = \{(\{e \mapsto i, g \mapsto 1\}, \emptyset_S, \{e \mapsto i+1\}) \mid 1 \leq i < M_e\}$ and $\downarrow_e^g = \{(\{e \mapsto i, g \mapsto 1\}, \emptyset_S, \{e \mapsto i-1\}) \mid 2 < i \leq M_e\}$.

Permanency. In a similar way we introduce a set of reactions for encoding permanency:

$$\Diamond_e^{\mathbf{i}} = \{(\{e \mapsto i\}, \mathbf{i}, \{e \mapsto i\}) \mid 1 \leq i \leq M_e\}$$

is a set of reactions ensuring permanency of $e \in S$ which can be inhibited by $\mathbf{i} \in \mathcal{B}(S)$.

We exploit the notation to use $\uparrow_e^g$, $\downarrow_e^g$, and $\Diamond_e^{\mathbf{i}}$ in place of regular reactions ignoring that they are in fact sets of reactions. In the implementation for crrsc we introduce an optimisation where these reactions are encoded as *macro-reactions*, that is, as simple operations on integer variables that increment, decrement, or retain the value of the variable encoding concentration of e.

We assume the macro-reactions are allowed only when no ordinary reaction is enabled.

6.1. Eukaryotic heat shock response

Firstly, we test our implementation using the model of the eukaryotic heat shock response (HSR) [19] originally introduced in [14]. HSR is an internal repair mechanism triggered when a cell is subjected to an environmental stressor – increased temperature that is not ideal for its functioning.

A temperature exceeding the ideal temperature causes the proteins ($prot$) of a cell to misfold (mfp), which in turn may cause its malfunctioning. To facilitate refolding of the proteins, heat shock response proteins (hsp) are produced, which are molecular chaperones for the misfolded proteins. The production of hsp is initiated by heat shock factors (hsf) which are, dimerised (hsf_2), and then trimerised (hsf_3). Next, hsf_3 activates hsp production by binding to the heat shock element (hse) which is the promoter-site of the gene encoding the heat shock proteins.

The original model of [14] used $stress$ and $nostress$ entities to distinguish between the presence and absence of the heat shock. In the model of [19] it is assumed that the heat shock appears at (and above) the temperature of $42\,^\circ C$, and this is modelled using the $temp$ entity. All the entities

Table 1. Entities used in the heat shock response model.

entity	description	entity	description
hsp	heat shock protein	$hsf_3{:}hse$	hsf_3 bound with hse
hsf	heat shock factor	$hsp{:}mfp$	hsp bound with mfp
hsf_2	dimerised heat shock factor	$hsp{:}hsf$	complex consisting of hsp and hsf
hsf_3	trimerised heat shock factor	$temp$	temperature value
hse	heat shock element	$cool$	decreases the temperature
mfp	misfolded protein	$heat$	increases the temperature
$prot$	protein		

except $temp$ remain at the concentration level of one unit. We assume that the maximal value of the temperature modelled using the entity $temp$ is 50.

Table 2. Reactions of the heat shock response model (curly brackets are omitted).

reactants	inhibitors	products
$hsf \mapsto 1$	$hsp \mapsto 1$	$hsf_3 \mapsto 1$
$hsf \mapsto 1, hsp \mapsto 1, mfp \mapsto 1$	$\varnothing_S$	$hsf_3 \mapsto 1$
$hsf_3 \mapsto 1$	$hsp \mapsto 1, hse \mapsto 1$	$hsf \mapsto 1$
$hsp \mapsto 1, hsf_3 \mapsto 1, mfp \mapsto 1$	$hse \mapsto 1$	$hsf \mapsto 1$
$hsf_3 \mapsto 1, hse \mapsto 1$	$hsp \mapsto 1$	$hsf_3{:}hse \mapsto 1$
$hsp \mapsto 1, hsf_3 \mapsto 1, mfp \mapsto 1, hse \mapsto 1$	$\varnothing_S$	$hsf_3{:}hse \mapsto 1$
$hse \mapsto 1$	$hsf_3 \mapsto 1$	$hse \mapsto 1$
$hsp \mapsto 1, hsf_3 \mapsto 1, hse \mapsto 1$	$mfp \mapsto 1$	$hse \mapsto 1$
$hsf_3{:}hse \mapsto 1$	$hsp \mapsto 1$	$hsp \mapsto 1, hsf_3{:}hse \mapsto 1$
$hsp \mapsto 1, mfp \mapsto 1, hsf_3{:}hse \mapsto 1$	$\varnothing_S$	$hsp \mapsto 1, hsf_3{:}hse \mapsto 1$
$hsf \mapsto 1, hsp \mapsto 1$	$mfp \mapsto 1$	$hsp{:}hsf \mapsto 1$
$hsp{:}hsf \mapsto 1, temp \mapsto 42$	$\varnothing_S$	$hsf \mapsto 1, hsp \mapsto 1$
$hsp{:}hsf \mapsto 1$	$temp \mapsto 42$	$hsp{:}hsf \mapsto 1$
$hsp \mapsto 1, hsf_3 \mapsto 1$	$mfp \mapsto 1$	$hsp{:}hsf \mapsto 1$
$hsp \mapsto 1, hsf_3{:}hse \mapsto 1$	$mfp \mapsto 1$	$hse \mapsto 1, hsp{:}hsf \mapsto 1$
$temp \mapsto 42, prot \mapsto 1$	$\varnothing_S$	$mfp \mapsto 1, prot \mapsto 1$
$prot \mapsto 1$	$temp \mapsto 42$	$prot \mapsto 1$
$hsp \mapsto 1, mfp \mapsto 1$	$\varnothing_S$	$hsp{:}mfp \mapsto 1$
$mfp \mapsto 1$	$hsp \mapsto 1$	$mfp \mapsto 1$
$hsp{:}mfp \mapsto 1$	$\varnothing_S$	$hsp \mapsto 1, prot \mapsto 1$

The background set S for the rsc modelling HSR consists of the entities in Table 1. The set A_{ord} comprises the reactions in Table 2. We also define the set of reactions dealing with temperature

$A_{temp} = \uparrow_{temp}^{heat} \cup \downarrow_{temp}^{cool} \cup \diamond_{temp}^{\mathbf{i}}$, where $\mathbf{i} = \{heat \mapsto 1, cool \mapsto 1\}$. By defining the set $\mathbf{i}$ in this way we ensure that the result of changing the temperature will not be overridden by the permanency.

The rsc for HSR is defined as $rsc_{\text{HSR}} = (S, A_{ord} \cup A_{temp})$.

To define a crrrs for rsc_{HSR} we use the context automaton $ca_{\text{HSR}} = (Q, q_0, R)$ where $Q = \{0, 1\}$, $q_0 = 0$ and $R = \{(0, \{hsf \mapsto 1, prot \mapsto 1, hse \mapsto 1, temp \mapsto 35\}, 1), (1, \{cool \mapsto 1\}, 1), (1, \{heat \mapsto 1\}, 1), (1, \emptyset_S, 1)\}$. Then, the crrsc for rsc_{HSR} is defined as $crrsc_{\text{HSR}} = (rsc_{\text{HSR}}, ca_{\text{HSR}})$. The context set specified in ca_{HSR} for the transition from 0 (the initial state) corresponds to the initial context set used in [14] as the minimal set of entities needed in HSR, together with the $temp$ entity indicating a temperature that does not cause the heat shock.

First, we test the efficiency of our implementation by verifying the reachability of the following results of $crrsc_{\text{HSR}}$: $\rho_1 = (\mathbf{x}_1, \mathbf{y}_1)$ where $\mathbf{x}_1 = \{hsp{:}hsf \mapsto 1, hse \mapsto 1, prot \mapsto 1\}$, $\mathbf{y}_1 = \{temp \mapsto 42\}$ and $\rho_2 = (\mathbf{x}_2, \mathbf{y}_2)$ where $\mathbf{x}_2 = \{mfp \mapsto 1\}$, $\mathbf{y}_2 = \emptyset_S$. Reachability of ρ_1 proves that it is possible to enter the state where HSR may become stable, while reachability of ρ_2 proves that it is possible for the proteins to eventually misfold. The k-reachability for ρ_1 is proved for $k = 4$, while ρ_2 for

Table 3. Results for the verification of reachability properties of HSR

	ρ_1		ρ_2	
	time [s]	memory [MB]	time [s]	memory [MB]
crrs	17.32	25.08	38.78	28.38
crrsc	**0.35**	**24.87**	**0.93**	**24.99**
improvement	49.48×	1.01×	41.69×	1.13×

$k = 9$. There is no noticeable improvement in memory consumption for the verification of crrsc over crrs. However, there is a significant difference in the execution times in favour of crrsc, e.g., for ρ_1 the verification for crrsc is 49.48 times faster. The verification results[1] for the reachability properties are summarised in Table 3.

Table 4. rsLTL formulae for HSR with the verification performance

	Formula	k	time [s]	memory [MB]
ϕ_1	$\mathbf{XF}_{heat>0}(temp > 42)$	9	1.01	30.04
ϕ_2	$\mathbf{XG}_{heat>0}((temp > 42) \Rightarrow \mathbf{F}(mfp > 0))$	21	3.18	34.77

The verification results for rsLTL formulae are presented in Table 4. The verification of the formula ϕ_2 requires more resources than ϕ_1, since the result is found for a larger value of k and the verified property contains more temporal operators, resulting in a larger encoding.

[1]The experimental results were obtained using a system equipped with 3.7GHz Intel Xeon E5 processor and 12GB of memory, running Mac OS X 10.12.3.

6.2. Scalable chain

As the next benchmark we use the *scalable chain* (SC) model [19]. The background set for the system is defined as $S = \{e_1, e_2, \ldots, e_m, inc, dec\}$. Intuitively, the system executes reactions incrementing concentration levels of m entities, each up to a maximal concentration level c. For $i < m$, when the maximal concentration level of e_i is reached, then the entity e_{i+1} is produced.

The inc and dec entities cause, respectively, incrementation or decrementation of concentration levels. We define the following sets of reactions: $\mathcal{P} = \big\{(\{e_i \mapsto c\}, \emptyset_S, \{e_{i+1} \mapsto 1)\}) \mid 1 \leq i < m\big\}$, $\mathcal{O} = \big\{ \uparrow_{e_i}^{inc}, \downarrow_{e_i}^{dec} \mid 1 \leq i \leq m\big\}$, $\mathcal{F} = \big\{(\{e_m \mapsto c\}, \{dec \mapsto 1\}, \{e_m \mapsto c\})\big\}$. The reactions of $\mathcal{P}$ implement the production of the subsequent entities, while their concentration levels are changed by the reactions of $\mathcal{O}$. The reaction of $\mathcal{F}$ ensures persistency of the "final" entity e_m when it reaches the concentration of c, unless dec is present. The rsc for the scalable chain system is defined as $rsc_{\text{SC}} = (S, \mathcal{P} \cup \mathcal{O} \cup \mathcal{F})$. Next, we define the context automaton $ca_{\text{SC}} = (Q, q_0, R)$ where $Q = \{0, 1\}$, $q_0 = 0$, and the set R consists of the following transitions: $(0, \{e_1 \mapsto 1, inc \mapsto 1\}, 1)$, $(1, \{inc \mapsto 1\}, 1)$, $(1, \{dec \mapsto 1\}, 1)$. Finally, we define $crrsc_{\text{SC}} = (rsc_{\text{SC}}, ca_{\text{SC}})$.

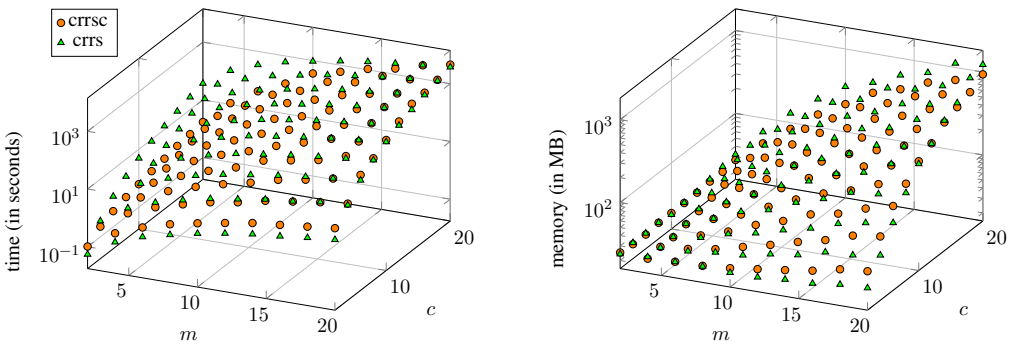

Figure 1. Verification results for the reachability property of SC

The verified reachability property of the scalable chain system is proved for $k = m \cdot c - 1$. The property expresses the reachability of the maximal concentration level of the entity e_m. The time and memory consumption results are presented in Fig. 1.

In most cases there is an observable advantage of the implementation for crrsc when the value of c is relatively large compared to m, e.g., for $m = 8$ and $c = 20$ the results for crrsc are 5.6 times better. For $m = 10$ and $c = 14$ the verification of crrs proved to be 1.6 times more efficient as it only consumed 1334 seconds, compared to 2155 seconds for crrsc. However, for $m = 20$ and $c = 16$ crrs was only 1.2 times better. We attribute this inconsequence to the heuristics of the SMT-solver used. The crrsc implementation appears to be more memory-efficient when dealing with larger concentration level values. It appears that when the verified system is highly-dependent on a large domain of concentration levels, then the crrsc will most likely be more suitable.

To test the performance of our rsLTL implementation we use a fixed maximal concentration level $c = 2$ and verify the properties presented in Table 5. The time and memory consumption results are

Table 5. rsLTL formulae for Scalable Chain

	Formula	k
ϕ_1	$\mathbf{F}_{inc>0}(e_m = c)$	$2 \cdot m - 1$
ϕ_2	$\phi_2 = \phi_2^1$, $\phi_2^i = \mathbf{F}_{inc>0}((e_i = c) \wedge \phi_2^i)$ for $i \in \{1, \ldots, m-1\}$, where $\phi_2^m = \mathbf{F}_{inc>0}((e_m = c))$	$2 \cdot m - 1$
ϕ_3	$\mathbf{G}((e_1 = 1) \Rightarrow \mathbf{F}_{inc>0}(e_m = c))$	$2 \cdot m$
ϕ_4	$\mathbf{F}_{inc>0}(e_1 = c)$	1
ϕ_5	$\mathbf{X}((e_1 > 0)\mathbf{R}_{inc>0}(e_2 > 0))$	2

presented in Fig. 2. The properties expressed with ϕ_1, ϕ_2, and ϕ_3 are proved for variable values of k that depend on the scaling parameter m. Verifying the formula ϕ_2 requires the most resources since it contains multiple nested operators which also result in multiple levels of recursion when computing the translation. Our implementation proved to be the most efficient for ϕ_4 and ϕ_5. This is mostly due to the very low and constant value of k. This means that only a very small portion of the model needs to be traversed to prove these properties.

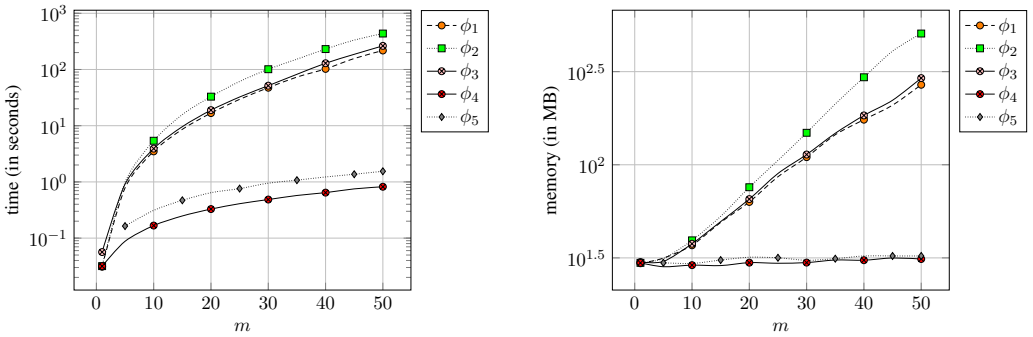

Figure 2. Verification results for the rsLTL properties of SC

7. Concluding remarks

This paper introduced reaction systems with discrete concentrations, which support quantitative modelling. Although the formalism is not more expressive than the standard reaction systems, our experimental results demonstrate that expressing concentration levels in an explicit way allows for some improvements in the efficiency of verification, and opens up possibilities for introducing different optimisations. Since we applied bounded model checking, the class of properties the method is able to verify is limited: it is unable to disprove existential properties (or alternatively, to prove universal properties). Therefore, in our future work we plan to provide a complete method for the verification of rsLTL, possibly by establishing a method for computing the diameter (completeness threshold) [27] of the model. The reachability problem for context-restricted reaction systems (with discrete concentrations) is NP-hard [19]. Therefore the computational complexity of the model checking problem for

reaction systems brings limitations to the practical applicability of the method. However, our experimental results demonstrate that the presented method scales well when verifying properties of large models. In our future work we plan also to extend this approach to provide a comprehensive framework for verifying quantitative properties of reaction systems, as well as provide a logical characterisation for the introduced logic and a complete computational complexity analysis for the verification problems that it introduces.

References

[1] Ehrenfeucht A, Rozenberg G. Reaction Systems. Fundamenta Informaticae, 2007;75(1-4):263–280. URL http://content.iospress.com/articles/fundamenta-informaticae/fi75-1-4-15.

[2] Ehrenfeucht A, Kleijn J, Koutny M, Rozenberg G. Reaction Systems: A Natural Computing Approach to the Functioning of Living Cells. A Computable Universe, Understanding and Exploring Nature as Computation, 2012. pp. 189–208. doi:10.1142/9789814374309_0010.

[3] Ehrenfeucht A, Kleijn J, Koutny M, Rozenberg G. Evolving reaction systems. Theoretical Computer Science, 2017;682:79–99. doi:10.1016/j.tcs.2016.12.031.

[4] Ehrenfeucht A, Rozenberg G. Introducing time in reaction systems. Theoretical Computer Science, 2009;410(4-5):310–322. doi:10.1016/j.tcs.2008.09.043.

[5] Hirvensalo M. On probabilistic and quantum reaction systems. Theoretical Computer Science, 2012;429:134–143. doi:10.1016/j.tcs.2011.12.032.

[6] Alhazov A, Aman B, Freund R, Ivanov S. Simulating R Systems by P Systems. In: Membrane Computing, 17th International Conference, CMC 2016, Milan, Italy, July 25-29, 2016. 2016 pp. 51–66. doi:10.1007/978-3-319-54072-6_4.

[7] Formenti E, Manzoni L, Porreca AE. Cycles and Global Attractors of Reaction Systems. In: Descriptional Complexity of Formal Systems - 16th International Workshop, DCFS 2014, LNCS. 2014 pp. 114–125. doi:10.1007/978-3-319-09704-6_11.

[8] Formenti E, Manzoni L, Porreca AE. Fixed Points and Attractors of Reaction Systems. In: Language, Life, Limits - 10th Conference on Computability in Europe, CiE 2014, volume 8493 of LNCS. Springer, 2014 pp. 194–203. doi:10.1007/978-3-319-08019-2_20.

[9] Formenti E, Manzoni L, Porreca AE. On the complexity of occurrence and convergence problems in reaction systems. Natural Computing, 2014. pp. 1–7. doi:10.1007/s11047-014-9456-3.

[10] Salomaa A. Functions and sequences generated by reaction systems. Theoretical Computer Science, 2012;466:87–96. doi:10.1016/j.tcs.2012.07.022.

[11] Salomaa A. On State Sequences Defined by Reaction Systems. In: Logic and Program Semantics. 2012 pp. 271–282. doi:10.1007/978-3-642-29485-3_17.

[12] Salomaa A. Functional Constructions between reaction Systems and Propositional Logic. Int. J. of Foundations of Computer Science, 2013;24(1):147–160. doi:10.1142/S0129054113500044.

[13] Salomaa A. Minimal and almost minimal reaction systems. Natural Computing, 2013;12(3):369–376. doi:10.1007/s11047-013-9372-y.

[14] Azimi S, Iancu B, Petre I. Reaction System Models for the Heat Shock Response. Fundamenta Informaticae, 2014;131(3-4):299–312. doi:10.3233/FI-2014-1016.

[15] Corolli L, Maj C, Marini F, Besozzi D, Mauri G. An excursion in reaction systems: From computer science to biology. Theoretical Computer Science, 2012;454:95–108. doi:10.1016/j.tcs.2012.04.003.

[16] Azimi S, Gratie C, Ivanov S, Manzoni L, Petre I, Porreca AE. Complexity of Model Checking for Reaction Systems. Theoretical Computer Science, 2016;623:103–113. doi:10.1016/ j.tcs.2015.11.040.

[17] Azimi S, Gratie C, Ivanov S, Petre I. Dependency Graphs and Mass Conservation in Reaction Systems. Theoretical Computer Science, 2015;598:23–39. doi:10.1016/ j.tcs.2015.02.014.

[18] Męski A, Penczek W, Rozenberg G. Model Checking Temporal Properties of Reaction Systems. Information Sciences, 2015;313:22–42. doi:10.1016/j.ins.2015.03.048.

[19] Męski A, Koutny M, Penczek W. Towards Quantitative Verification of Reaction Systems. In: Unconventional Computation and Natural Computation: 15th International Conference, UCNC 2016, Manchester, UK, July 11-15, 2016, Proceedings. 2016 pp. 142–154. doi:10.1007/978-3-319-41312-9_12.

[20] Horn F, Jackson R. General mass action kinetics. Archive for Rational Mechanics and Analysis, 1972;47(2):81–116. doi:10.1007/BF00251225.

[21] Brijder R, Ehrenfeucht A, Rozenberg G. Reaction Systems with Duration. In: Computation, Cooperation, and Life - Essays Dedicated to Gheorghe Paun on the Occasion of His 60th Birthday, volume 6610 of LNCS. Springer, 2011 pp. 191–202. doi:10.1007/978-3-642-20000-7_16.

[22] Kroening D, Strichman O. Decision Procedures - An Algorithmic Point of View, Second Edition. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2016. doi:10.1007/978-3-662-50497-0.

[23] Biere A, Cimatti A, Clarke EM, Zhu Y. Symbolic Model Checking Without BDDs. In: Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems, TACAS '99. Springer-Verlag, 1999 pp. 193–207. doi:10.1007/3-540-49059-0_14.

[24] Biere A, Heljanko K, Junttila TA, Latvala T, Schuppan V. Linear Encodings of Bounded LTL Model Checking. Logical Methods in Computer Science, 2006; 2(5). doi:10.2168/LMCS-2(5:5)2006.

[25] Clarke E, Grumberg O, Peled D. Model Checking. MIT Press, 1999. ISBN 978-0-262-03270-4.

[26] de Moura L, Bjørner N. Z3: An Efficient SMT Solver. In: Proceedings of the 14th International Conference on Tools and Algorithms for Construction and Analysis of Systems, TACAS 2008. Springer-Verlang, 2008 pp. 337–340. doi:10.1007/978-3-540-78800-3_24.

[27] Clarke EM, Kroening D, Ouaknine J, Strichman O. Completeness and Complexity of Bounded Model Checking. In: Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Proceedings. 2004 pp. 85–96. doi:10.1007/978-3-540-24622-0_9.