Reaction Mining for Reaction Systems

Artur Męski[1,2(✉)], Maciej Koutny[3], and Wojciech Penczek[1,4]

[1] Institute of Computer Science, PAS, Jana Kazimierza 5, 01-248 Warsaw, Poland
{meski,penczek}@ipipan.waw.pl

[2] Vector GB Limited, London WC2N 4JF, UK
artur.meski@vector.com

[3] School of Computing, Newcastle University, Newcastle upon Tyne NE1 7RU, UK
maciej.koutny@ncl.ac.uk

[4] Faculty of Science, Institute of Computer Science, Siedlce University, 3-Maja 54, 08-110 Siedlce, Poland

Abstract. Reaction systems are a formal model for specifying and analysing computational processes in which reactions operate on sets of entities (molecules), providing a framework for dealing with qualitative aspects of biochemical systems. This paper is concerned with reaction systems in which entities can have discrete concentrations and reactions operate on multisets of entities, providing a succinct framework for dealing with quantitative aspects of systems. This is facilitated by a dedicated linear-time temporal logic which allows one to express and verify a wide range of behavioural system properties.

In practical applications, a reaction system with discrete concentrations may only be partially specified, and effective calculation of the missing details would provide an attractive design approach. To develop such an approach, this paper introduces reaction systems with parameters representing the unknown parts of the reactions. The main result is a method which attempts to replace these parameters in such a way that the resulting reaction system operating in a given external environment satisfies a given temporal logic formula. We provide a suitable encoding of parametric reaction systems in SMT, and outline a synthesis procedure based on bounded model checking for solving the synthesis problem. We also provide preliminary experimental results demonstrating the feasibility of the new synthesis method.

The seminal paper [11] introduced a fundamental *reaction systems* model for computational processes inspired by the functioning of a living cell. The model can capture in a very simple way the basic mechanisms underpinning the dynamic behaviour of a living cell. A key feature of reaction systems is that the latter results from the interactions of biochemical reactions based on the mechanisms of facilitation and inhibition, i.e., the products of reactions may facilitate or inhibit each other. The basic model of reaction systems represents the reactions, states, and dynamic processes using (tuples of) finite sets, and so it directly captures the qualitative aspects of systems. Having said that, more involved concepts can be introduced using the basic ones.

S. Stepney and S. Verlan (Eds.): UCNC 2018, LNCS 10867, pp. 131–144, 2018.
https://doi.org/10.1007/978-3-319-92435-9_10

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., [9,10]). A number of extensions were also introduced, e.g., reaction systems with time [12], reaction systems with durations [5], and quantum and probabilistic reaction systems [16]. Mathematical properties of reaction systems were investigated in, e.g., [1,7,8,13–15,23–26].

Examples of applications of reaction systems to modelling of systems include, e.g., [4,6]. Verification of reaction systems was discussed in, e.g., [2,3,20]. The papers [19,22] introduced reaction systems with discrete concentrations of entities and reactions operating on multisets of entities, resulting in a model allowing direct quantitative modelling. Although there exist other approaches that support modelling of complex dependencies of concentration levels and their changes, e.g., chemical reaction networks theory based on [17], reaction systems provide much simpler framework and the processes of reaction systems take into account interactions with the external environment. Discrete concentrations can be simulated in the original qualitative reaction systems, but reaction systems with discrete concentrations provide much more succinct representations in terms of the number of entities being used, and allow for more efficient verification [19]. The properties being verified are expressed in rsLTL which is a version of the linear-time temporal logic defined specifically for reaction systems.

In practical applications, a reaction system with discrete concentrations may have only partially specified reactions, and a reaction mining i.e., an effective filling in the missing details would provide an attractive design approach. To develop such an approach, this paper introduces reaction systems with parameters representing the unknown parts of the reactions. The main result is a methodology which attempts to replace these parameters in such a way that the resulting reaction system satisfies a given *rsLTL formula* when operating in a given external *environment*. Intuitively, such a formula might correspond to a number of observations (runs) of the behaviour of a partially specified system. Moreover, the environment is specified using a *context automaton* which represents the influence of the bigger system in which the reaction system with discrete concentrations operates. We provide a suitable encoding of parametric reaction systems in SMT, and propose a synthesis procedure based on bounded model checking for solving the synthesis problem. We also provide preliminary successful experimental results demonstrating the scalability of the new synthesis method. The paper is organised in the following way. In the next section, we recall the basic notations and definitions used by reaction systems with discrete concentrations. Section 2 introduces parametric reaction systems, and the following section defines SMT-based encoding of such systems. Section 4 discusses experimental evaluation of the synthesis approach introduced in this paper, and Sect. 5 draws some concluding remarks.

1 Preliminaries

A *multiset* over a set X is a mapping $\mathbf{b} : X \to \{0, 1, \dots\}$, and its *carrier* is $carr(\mathbf{b}) = \{x \in X \mid \mathbf{b}(x) > 0\}$. The *empty* multiset $\varnothing_X$ is one with the empty carrier. $\mathcal{B}(X)$ denotes the set of all multisets over X. For a finite $\mathbf{B} \subset \mathcal{B}(X)$, $\bigwedge\!\!\!\bigwedge(\mathbf{B})$ is the multiset over X such that $\bigwedge\!\!\!\bigwedge(\mathbf{B})(x) = \max(\{0\} \cup \{\mathbf{b}(x) \mid \mathbf{b} \in \mathbf{B}\})$, for every $x \in X$. For $\mathbf{b}, \mathbf{b}' \in \mathcal{B}(X)$, $\mathbf{b} \leq \mathbf{b}'$ if $\mathbf{b}(x) \leq \mathbf{b}'(x)$, for every $x \in X$.

We use $x \mapsto i$ for denoting the multiplicity of x in multisets; e.g., $\{x \mapsto 1, y \mapsto 2\}$ is a multiset with one copy of x, two copies of y, and nothing else. If the multiplicity of an entity is 1, we may also simply omit the value, e.g., $\{x, y \mapsto 2\}$.

The syntax of *multiset expressions* $BE(X)$ is defined by the following grammar: $\mathfrak{a} ::= \mathit{true} \mid e \sim c \mid e \sim e \mid \neg\mathfrak{a} \mid \mathfrak{a} \vee \mathfrak{a}$, where $\sim \in \{<, \leq, =, \geq, >\}$, $e \in X$, $c \in \mathbb{N}$. Then $\mathbf{b} \models_b \mathfrak{a}$ means that $\mathfrak{a}$ holds for $\mathbf{b} \in \mathcal{B}(X)$ assuming that:

$$\begin{array}{lll}
\mathbf{b} \models_b \mathit{true} & \text{for every } \mathbf{b} \in \mathcal{B}(X), & \\
\mathbf{b} \models_b e \sim c & \text{iff} & \mathbf{b}(e) \sim c, \\
\mathbf{b} \models_b e \sim e' & \text{iff} & \mathbf{b}(e) \sim \mathbf{b}(e'), \\
\mathbf{b} \models_b \neg\mathfrak{a} & \text{iff} & \mathbf{b} \not\models_b \mathfrak{a}, \\
\mathbf{b} \models_b \mathfrak{a} \vee \mathfrak{a}' & \text{iff} & \mathbf{b} \models_b \mathfrak{a} \text{ or } \mathbf{b} \models_b \mathfrak{a}'.
\end{array}$$

Reaction Systems with Discrete Concentrations. The enabling of biochemical reactions may depend not only on the availability of reactants and the absence of inhibitors, but also on their concentration levels. We will now recall an extension of the basic reaction systems with explicit representation of the discrete concentration levels of entities (the k-th level of concentration of x is represented by a multiset containing k copies of x). The model uses multisets rather than sets of entities, but otherwise retains key features of the original framework.

A *reaction system with discrete concentrations* (rsc) is a pair $rsc = (S, A)$, where S is a finite *background* set (comprising *entities*) and A is a nonempty finite set of *reactions* over the background set. Each reaction is a triple $a = (\mathbf{r}, \mathbf{i}, \mathbf{p})$ such that $\mathbf{r}$, $\mathbf{i}$, $\mathbf{p}$ are nonempty multisets over S with $\mathbf{r}(e) < \mathbf{i}(e)$, for every $e \in \mathit{carr}(\mathbf{i})$. 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 reaction a. An entity e is an inhibitor of a whenever $e \in \mathit{carr}(\mathbf{i}_a)$.

A 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 $e \in \mathit{carr}(\mathbf{i}_a)$. The *result* of a on $\mathbf{t}$ is given by $\mathit{res}_a(\mathbf{t}) = \mathbf{p}_a$ if $en_a(\mathbf{t})$, and by $\mathit{res}_a(\mathbf{t}) = \varnothing_S$ otherwise. Then the *result* of A on $\mathbf{t}$ is $\mathit{res}_A(\mathbf{t}) = \bigwedge\!\!\!\bigwedge\{\mathit{res}_a(\mathbf{t}) \mid a \in A\}$.

Intuitively, $\mathbf{t}$ is a *state* of a biochemical system being modelled, and $\mathbf{t}(e)$ is the *concentration level* of each entity e (e.g., $\mathbf{t}(e) = 0$ indicates that e is not present in the current state while $\mathbf{t}(e) = 1$ indicates that e is present at its lowest concentration level). A 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 $\mathbf{i}_a$.

The above gives the behaviour of an rsc as a closed system. To define its operation as an open system, we need a suitable representation of the environment. A *context automaton* over a background set S 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 $R \subseteq Q \times \mathcal{B}(S) \times Q$ is the *transition relation*. We assume that, for every $q \in Q$, there exist $\mathbf{c} \in \mathcal{B}(S)$ and $q' \in Q$ such that $(q, \mathbf{c}, q') \in R$. We also denote $(q, \mathbf{c}, q') \in R$ by $q \xrightarrow{\mathbf{c}} q'$.

A *context restricted reaction system with discrete concentrations* (crrsc) [22] is a pair $crrsc = (rsc, ca)$ such that $rsc = (S, A)$ is an rsc, and $ca = (Q, q^{init}, R)$ is a context automaton over S. The dynamic behaviour of $crrsc$ is captured by the state sequences of its interactive processes, where an *interactive process* in $crrsc$ is a triple $\pi = (\zeta, \gamma, \delta)$ such that:

– $\zeta = (z_0, z_1, \ldots, z_n)$, $\gamma = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n)$, 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 $\mathbf{d}_0 = \varnothing_S$
– $(z_i, \mathbf{c}_i, z_{i+1}) \in R$, for every $i \in \{0, \ldots, n-1\}$
– $\mathbf{d}_i = res_A(\bigwedge\!\!\!\bigwedge\{\mathbf{d}_{i-1}, \mathbf{c}_{i-1}\})$, for every $i \in \{1, \ldots, n\}$.

The sequence γ is the context sequence of π and δ is the *result* sequence, while ζ is simply the sequence of states of ca. The *state sequence* of π is $\tau = (\mathbf{w}_0, \ldots, \mathbf{w}_n) = (\bigwedge\!\!\!\bigwedge\{\mathbf{c}_0, \mathbf{d}_0\}, \ldots, \bigwedge\!\!\!\bigwedge\{\mathbf{c}_n, \mathbf{d}_n\})$.

Reaction Systems Linear-Time Temporal Logic. Here we recall the logic (rsLTL) introduced in [22], which captures requirements imposed on paths of crrsc.

The *model* of a crrsc given by $crrsc = (rsc, ca)$ with $rsc = (S, A)$ and $ca = (Q, q_{init}, R)$ is the triple $\mathcal{M}(crrsc) = (\mathbb{W}, \mathtt{w}^{init}, \longrightarrow, L)$, where: $\mathbb{W} = \mathcal{B}(S) \times Q$ is the set of states; $\mathtt{w}^{init} = (\varnothing, q^{init})$ is the initial state; and $\longrightarrow \subseteq \mathbb{W} \times \mathcal{B}(S) \times \mathbb{W}$ is the transition relation such that, for all $\mathbf{w}, \mathbf{w}', \alpha \in \mathcal{B}(S)$ and $q, q' \in Q$, we have $((\mathbf{w}, q), \alpha, (\mathbf{w}', q')) \in \longrightarrow$ if $(q, \alpha, q') \in R$ and $\mathbf{w}' = res_A(\bigwedge\!\!\!\bigwedge\{\mathbf{w}, \alpha\})$.

We write $\mathcal{M}$ instead of $\mathcal{M}(crrsc)$ when $crrsc$ is understood. We also denote $(\mathtt{w}, \alpha, \mathtt{w}') \in \longrightarrow$ by $\mathtt{w} \xrightarrow{\alpha} \mathtt{w}'$.

A *path* of $\mathcal{M}$ is an infinite sequence $\sigma = (\mathtt{w}_0, \alpha_0, \mathtt{w}_1, \alpha_1, \ldots)$ of states and actions (context multisets) such that $\mathtt{w}_i \xrightarrow{\alpha_i} \mathtt{w}_{i+1}$, for every $i \geq 0$. For every $i \geq 0$, we denote $\sigma_s(i) = \mathtt{w}_i = (\sigma_b(i), \sigma_{ca}(i))$ and $\sigma_a(i) = \alpha_i$. Moreover, $\sigma^i = (\mathtt{w}_i, \alpha_i, \mathtt{w}_{i+1}, \alpha_{i+1}, \ldots)$ is a suffix of σ. By $\Pi_{\mathcal{M}}(\mathtt{w})$ we denote the set of all the paths that start in $\mathtt{w} \in \mathbb{W}$ and $\Pi_{\mathcal{M}} = \bigcup_{\mathtt{w} \in \mathbb{W}} \Pi_{\mathcal{M}}(\mathtt{w})$ is the set of all paths of $\mathcal{M}$.

The syntax of rsLTL is given 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 $\mathfrak{a} \in BE(S)$. Intuitively, $\mathbf{X}_{\mathfrak{a}}\phi$ means 'following an action satisfying $\mathfrak{a}$, ϕ holds in the next state', $\phi \mathbf{U}_{\mathfrak{a}}\phi'$ means 'ϕ' holds eventually, and ϕ must hold at every preceding state, following only actions satisfying $\mathfrak{a}$', and $\phi \mathbf{R}_{\mathfrak{a}}\phi'$ means 'following only actions satisfying $\mathfrak{a}$, ϕ' holds up to and including the first state where ϕ holds'.

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

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

Moreover, $\mathfrak{a} \Rightarrow \phi$ stands for $\neg\mathfrak{a} \vee \phi$, $\mathbf{G}_{\mathfrak{a}}\phi$ for $false\mathbf{R}_{\mathfrak{a}}\phi$, $\mathbf{F}_{\mathfrak{a}}\phi$ for $true\mathbf{U}_{\mathfrak{a}}\phi$, and $\mathbf{F}\phi$ is the same as $\mathbf{F}_{true}\phi$, for every rsLTL operator $\mathbf{F}$. Thus, $\phi\mathbf{U}\phi'$ means that 'ϕ' holds eventually, and ϕ must hold at every preceding state', and $\phi\mathbf{R}\phi'$ means that 'ϕ' holds up to and including the first state where ϕ holds'.

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

*Bounded Semantics for rs*LTL. We use the bounded model checking approach which requires us to specify when a given formula holds while considering only a finite number of states and actions of the prefix of the path being considered.

A path $\sigma = (\mathtt{w}_0, \alpha_0, \mathtt{w}_1, \alpha_1, \dots)$ is a (k,l)-*loop* (or k-*loop*) if there exist $k \geq l > 0$ such that $\sigma = (\mathtt{w}_0, \alpha_0, \dots, \alpha_{l-2}, \mathtt{w}_{l-1})(\alpha_l, \mathtt{w}_{l+1}, \alpha_{l+1}, \dots, \alpha_{k-1}, \mathtt{w}_k)^{\omega}$ and $\mathtt{w}_{l-1} = \mathtt{w}_k$. 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. The fact that a formula ϕ holds in a path σ with a bound $k \in \mathbb{N}$ is denoted by $\sigma \models^k \phi$ and defined as follows:

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

$\sigma \models_{nl} \mathfrak{a}$	if	$\sigma_b(0) \models_b \mathfrak{a}$
$\sigma \models_{nl} \phi \wedge \phi'$	if	$\sigma \models_{nl} \phi$ and $\sigma \models_{nl} \phi'$
$\sigma \models_{nl} \phi \vee \phi'$	if	$\sigma \models_{nl} \phi$ or $\sigma \models_{nl} \phi'$
$\sigma \models_{nl} \mathbf{X}_{\mathfrak{a}}\phi$	if	$k > 0$, $\sigma_a(0) \models_b \mathfrak{a}$, and $\sigma^1 \models_{nl} \phi$
$\sigma \models_{nl} \phi\mathbf{U}_{\mathfrak{a}}\phi'$	if	$(\exists 0 \leq j \leq k)(\sigma^j \models_{nl} \phi'$ and $(\forall 0 \leq l < j)(\sigma^l \models_{nl} \phi$ and $\sigma_a(l) \models_b \mathfrak{a}))$
$\sigma \models_{nl} \phi\mathbf{R}_{\mathfrak{a}}\phi'$	if	$(\exists 0 \leq j \leq k)(\sigma^j \models_{nl} \phi$ and $((\forall 0 \leq l \leq j)(\sigma^l \models_{nl} \phi')$ and $(\forall 0 \leq l < j)(\sigma_a(l) \models_b \mathfrak{a})))$

For a bound $k \in \mathbb{N}$ and a crrsc model $\mathcal{M}$, $\mathcal{M} \models^k_{\exists} \phi$ if there exists $\sigma \in \Pi_{\mathcal{M}}(\mathtt{w}^{init})$ such that $\sigma \models^k \phi$. The *bounded model checking problem* for rsLTL is the decision problem of checking if $\mathcal{M} \models^k_{\exists} \phi$, for a given $k \in \mathbb{N}$ and $\mathcal{M}$.

Theorem 1 **([22]).** *Let ϕ be an rs*LTL *formula and $\mathcal{M}$ be a crrsc model. Then, $\mathcal{M} \models_{\exists} \phi$ if and only if there exists $k \in \mathbb{N}$ such that $\mathcal{M} \models^k_{\exists} \phi$.*

2 Parametric Reaction Systems

We introduce parametric reaction systems which allow for defining also incomplete reactions by using parameters in place of reactant, inhibitor, and product sets.

A *parametric reaction system* (prs) is a triple $prs = (S, P, A)$, where S is a finite *background* set, P is a finite set of elements called *parameters*, and A is a nonempty finite set of *parametric reactions* over the background set. Each parametric reaction in A is a triple $a = (\mathfrak{r}, \mathfrak{i}, \mathfrak{p})$ such that $\mathfrak{r}, \mathfrak{i}, \mathfrak{p} \in \mathcal{B}(S) \cup P$.

The elements $\mathfrak{r}$, $\mathfrak{i}$, and $\mathfrak{p}$ are respectively denoted by $\mathfrak{r}_a$, $\mathfrak{i}_a$, and $\mathfrak{p}_a$ and called the *reactants*, *inhibitors*, and *products* of parametric reaction a. A *parameter valuation* of *prs* is a function $\mathtt{v} : P \cup \mathcal{B}(S) \to \mathcal{B}(S)$ such that $\mathtt{v}(\mathfrak{b}) = \mathfrak{b}$ if $\mathfrak{b} \in \mathcal{B}(S)$. We also write $\mathfrak{b}^{\leftarrow \mathtt{v}}$ for $\mathtt{v}(\mathfrak{b})$. The set of all the parameter valuations for *prs* is denoted by $\mathtt{PV}_{prs}$. Let $\mathtt{v} \in \mathtt{PV}_{prs}$. For $X \subseteq A$ we define $X^{\leftarrow \mathtt{v}} = \{(a_{\mathfrak{r}}^{\leftarrow \mathtt{v}}, a_{\mathfrak{i}}^{\leftarrow \mathtt{v}}, a_{\mathfrak{p}}^{\leftarrow \mathtt{v}}) \mid a \in X\}$. Then, by $prs^{\leftarrow \mathtt{v}}$ we denote the structure $(S, A^{\leftarrow \mathtt{v}})$ where all the parameters in A are substituted according to the parameter valuation $\mathtt{v}$. We say that $\mathtt{v} \in \mathtt{PV}_{prs}$ is a *valid parameter valuation* if $prs^{\leftarrow \mathtt{v}}$ yields an rsc.

A *context-restricted parametric reaction system* (crprs) is a pair $crprs = (prs, ca)$ such that $prs = (S, P, A)$ is a prs and $ca = (Q, q^{init}, R)$ is a context automaton over S. For $\mathtt{v} \in \mathtt{PV}_{prs}$ we define $crprs^{\leftarrow \mathtt{v}} = (prs^{\leftarrow \mathtt{v}}, ca)$.

Example 1. We consider a simple prs for a simplified abstract genetic regulatory system based on [9]. The system contains two (abstract) genes x and y expressing proteins X and Y, respectively, and a protein complex Q formed by X and Y. The background set is defined as $S = \{x, \widehat{x}, X, y, \widehat{y}, Y, h, Q\}$, where $\widehat{x}$ and $\widehat{y}$ denote RNA polymerase attached to the promoter of genes x and y, respectively. Here h is used as an abstract inhibitor. Finally, the set of parametric reactions consists of the following subsets: $A_x = \{(\{x\}, \{h\}, \{x\}), (\{x\}, \{h\}, \{\widehat{x}\}), (\{x, \widehat{x}\}, \{h\}, \{X\})\}$, $A_y = \{(\{y\}, \{h\}, \lambda_1), (\lambda_2, \{h\}, \{\widehat{y}\}), (\{y, \widehat{y}\}, \{h\}, \lambda_3)\}$, $A_Q = \{(\{X, Y\}, \{h\}, \{Q\})\}$. Notice that the reactions of A_y use parameters $\lambda_1, \lambda_2, \lambda_3$ to define expression of the protein Y. Suppose that we investigate the processes starting from the states that already contain x and y. This leads to the following definition of the context automaton: $ca = (\{0, 1\}, 0, R)$, where: $R = \{0 \xrightarrow{\{x,y\}} 1, 1 \xrightarrow{\varnothing} 1, 1 \xrightarrow{\{h\}} 0)\}$. When the context set contains the entity h, ca reverts back to the initial state, while for the empty context set the ca remains in the state 1. Then, crprs is defined as $crprs = ((S, P, A), ca)$, where: $P = \{\lambda_1, \lambda_2, \lambda_3\}$ and $A = A_x \cup A_y \cup A_Q$. $\diamond$

In this paper, our focus is on the synthesis of a parameter valuation given some observations expressed with rsLTL formulae. Let $crprs = (prs, ca)$ be a crprs, and $F = \{\phi_1, \ldots, \phi_n\}$ be a set of rsLTL formulae. The aim of *parameter synthesis* for crprs is to find a valid parameter valuation $\mathtt{v}$ of *crprs* such that $(\mathcal{M}(crprs^{\leftarrow \mathtt{v}}) \models_{\exists} \phi_1) \wedge \cdots \wedge (\mathcal{M}(crprs^{\leftarrow \mathtt{v}}) \models_{\exists} \phi_n)$. Each formula of F corresponds to an interactive process observed in the analysed system via, e.g., experiments or simulations. Therefore, for each such process we expect an individual path in $\mathcal{M}(crprs^{\leftarrow \mathtt{v}})$ and we solve n the model checking problems for rsLTL in one instance. However, the parameter valuation v is shared among all instances, which allows us to calculate v such that all properties of F are satisfied.

Example 2. Let us assume we performed an experiment on the system of Example 1 where protein Y was expressed. We have the following observations related to the expression of the protein Y:

– whenever the current state contains y, then y and $\widehat{y}$ are found in the next state: $\phi_1^c = \mathbf{G}_{\neg h}(y \Rightarrow \mathbf{X}(y \wedge \widehat{y}))$;

- when y and $\widehat{y}$ are present, then Y is finally produced: $\phi_2^c = \mathbf{G}_{\neg h}((y \wedge \widehat{y}) \Rightarrow \mathbf{F}Y)$;
- the entities y, $\widehat{y}$, and Y are eventually produced: $\phi^r = (\mathbf{F}_{\neg h} y) \wedge (\mathbf{F}_{\neg h}\widehat{y}) \wedge (\mathbf{F}_{\neg h} Y)$.

These observations are made assuming h is not provided in the context set. Additionally, we observe that the protein Q is not present in the first three steps of the execution and then, after an arbitrary number of steps it is finally produced: $\phi^d = \neg Q \wedge \mathbf{X}(\neg Q \wedge \mathbf{X}(\neg Q \wedge \mathbf{F}Q))$. The observations are related to a single interactive process (experiment), therefore we constrain the problem using the conjunction of all the observations. Finally, the observations are expressed using the rsLTL formula $\phi = \phi^r \wedge \phi_1^c \wedge \phi_2^c \wedge \phi^d$. Next, we perform parameter synthesis for $F = \{\phi\}$, that is, we obtain a valid parameter valuation $\mathtt{v}$ such that $\mathcal{M}(crprs^{\leftarrow \mathtt{v}}) \models_\exists \phi$. A parameter valuation $\mathtt{v}$ such that $\lambda_1^{\leftarrow \mathtt{v}} = \{y\}$, $\lambda_2^{\leftarrow \mathtt{v}} = \{y\}$, $\lambda_3^{\leftarrow \mathtt{v}} = \{Y\}$ is valid and satisfies the requirements of our observations. $\diamond$

Parameter Constraints. In some cases restricting parameter valuations using only rsLTL formulae may prove to be less efficient than constraining the valuation using specialised constraints for the parameters of a prs.

For $prs = (S, P, A)$ the *parameter constraints* $PC(prs)$ are defined using the following grammar:

$$\mathfrak{c} ::= true \mid \lambda[e] \sim c \mid \lambda[e] \sim \lambda[e] \mid \neg\mathfrak{c} \mid \mathfrak{c} \vee \mathfrak{c},$$

where $\lambda \in P$, $e \in S$, $c \in \mathbb{N}$, and $\sim \in \{<, \leq, =, \geq, >\}$. Intuitively, $\lambda[e]$ can be used to refer to the concentration of $e \in S$ in the multisets corresponding to the valuations of λ.

Let $\mathtt{v}$ be a parameter valuation of *prs*. The fact that $\mathfrak{c}$ holds in $\mathtt{v}$ is denoted by $\mathtt{v} \models_p \mathfrak{c}$ and defined as follows:

$$\begin{array}{ll}
\mathtt{v} \models_p true & \text{for every } \mathtt{v}, \\
\mathtt{v} \models_p \lambda[e] \sim c & \text{if } \lambda^{\leftarrow \mathtt{v}}(e) \sim c, \\
\mathtt{v} \models_p \lambda_1[e_1] \sim \lambda_2[e_2] & \text{if } \lambda_1^{\leftarrow \mathtt{v}}(e_1) \sim \lambda_2^{\leftarrow \mathtt{v}}(e_2), \\
\mathtt{v} \models_p \neg\mathfrak{c} & \text{if } \mathtt{v} \not\models_p \mathfrak{c}, \\
\mathtt{v} \models_p \mathfrak{c}_1 \vee \mathfrak{c}_2 & \text{if } \mathtt{v} \models_p \mathfrak{c}_1 \text{ or } \mathtt{v} \models_p \mathfrak{c}_2.
\end{array}$$

A *constrained parametric reaction system* (cprs) is a tuple $cprs = (S, P, A, \mathfrak{c})$ such that (S, P, A) is a prs and $\mathfrak{c} \in PC(prs)$. For $\mathtt{v} \in \mathtt{PV}_{prs}$, we then define $cprs^{\leftarrow \mathtt{v}} = prs^{\leftarrow \mathtt{v}}$. A parameter valuation $\mathtt{v} \in \mathtt{PV}_{prs}$ is *valid* in *cprs* if it is valid in *prs* and $\mathtt{v} \models_p \mathfrak{c}$. A *context-restricted cprs* (cr-cprs) is a pair $cr\text{-}cprs = (cprs, ca)$ such that $cprs = (S, P, A, \mathfrak{c})$ is a cprs and ca is a context automaton over S. We also denote $cr\text{-}cprs^{\leftarrow \mathtt{v}} = (cprs^{\leftarrow \mathtt{v}}, ca)$.

The proposed language of parameter constraints allows for specifying constraints on multisets corresponding to parameters and relationships between them, which is demonstrated in the following example.

Example 3. Suppose $\lambda_1, \lambda_2, \lambda_3 \in P$. To constrain $\lambda_1^{\leftarrow \mathtt{v}}$ to be a sub-multiset of $\lambda_2^{\leftarrow \mathtt{v}}$ (i.e., $\lambda_1^{\leftarrow \mathtt{v}} \subseteq \lambda_2^{\leftarrow \mathtt{v}}$, for all $\mathtt{v}$), we define $submset(\lambda_1, \lambda_2) = \bigwedge_{e \in S}(\lambda_1[e] \leq$

$\lambda_2[e]$). To constrain $\lambda_3^{\leftarrow \mathtt{v}}$ to be the intersection of $\lambda_1^{\leftarrow \mathtt{v}}$ and $\lambda_2^{\leftarrow \mathtt{v}}$ (i.e., $\lambda_1^{\leftarrow \mathtt{v}} \cap \lambda_2^{\leftarrow \mathtt{v}} = \lambda_3^{\leftarrow \mathtt{v}}$, for all $\mathtt{v}$), we define $intersect(\lambda_1, \lambda_2, \lambda_3)$ as

$\bigwedge_{e \in S}(((\lambda_1[e] > \lambda_2[e]) \wedge (\lambda_3[e] = \lambda_2[e])) \vee ((\lambda_1[e] \leq \lambda_2[e]) \wedge (\lambda_3[e] = \lambda_1[e])))$.
$\diamond$

The parameter synthesis problem for cr-cprs is defined similarly as for crprs. Let $cr\text{-}cprs = (cprs, ca)$, $F = \{\phi_1, \ldots, \phi_n\}$ be a rsLTL formulae, and $\mathfrak{c}$ be a parameter constraint. The aim is to calculate a valid parameter valuation $\mathtt{v}$ of $cr\text{-}cprs$ such that $(\mathcal{M}(cr\text{-}cprs^{\leftarrow \mathtt{v}}) \models_{\exists} \phi_1) \wedge \cdots \wedge (\mathcal{M}(cr\text{-}cprs^{\leftarrow \mathtt{v}}) \models_{\exists} \phi_n)$. In the next section, we show how this problem can be solved using an incremental approach, which amounts to checking $(\mathcal{M}(cr\text{-}cprs^{\leftarrow \mathtt{v}}) \models_{\exists}^{k} \phi_1) \wedge \cdots \wedge (\mathcal{M}(cr\text{-}cprs^{\leftarrow \mathtt{v}}) \models_{\exists}^{k} \phi_n)$ for $k \geq 0$, by increasing the value of k until a valid parameter valuation is found.

3 SMT-Based Encoding

In this section we provide a translation of the parameter synthesis problem for cr-cprs and rsLTL into the satisfiability modulo theory (SMT) [18] 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 $cr\text{-}cprs = ((S, P, A, \mathfrak{c}), (Q, q^{init}, R))$ and $F = \{\phi_1, \ldots, \phi_n\}$ be a set of rsLTL formulae. Then, we encode the model $\mathcal{M}(cr\text{-}cprs^{\leftarrow \mathtt{v}})$, where $\mathtt{v}$ is a valid parameter valuation of $cr\text{-}cprs$. Let $k \geq 0$ be an integer, then for each $f \in \{1, \ldots, n\}$ we encode any possible path of $\mathcal{M}(cr\text{-}cprs^{\leftarrow \mathtt{v}})$ bounded with k. That is, for each formula ϕ_f we encode a separate bounded path representing its witness. The entities of S are denoted by $e_1, \ldots, e_m$, where $m = |S|$. For each $\phi_f \in F$ and $i \in \{0, \ldots, k\}$ we introduce sets of positive integer variables:

▶ $\mathbf{P}_{f,i} = \{\mathtt{p}_{f,i,1}, \ldots, \mathtt{p}_{f,i,m}\}$, $\mathbf{P}^{\mathcal{E}}_{f,i} = \{\mathtt{p}^{\mathcal{E}}_{f,i,1}, \ldots, \mathtt{p}^{\mathcal{E}}_{f,i,m}\}$, $\mathbf{Q}_f = \{\mathtt{q}_{f,0}, \ldots, \mathtt{q}_{f,k}\}$.

Let $\mathtt{ta} : A \to \{1, \ldots, |A|\}$ be a bijection mapping all the reactions to integers. Then, for each $a \in A$ we also introduce a set of variables encoding products:

▶ $\mathbf{P}^{p}_{f,i,a} = \{\mathtt{p}^{p}_{f,i,\mathtt{ta}(a),1}, \ldots, \mathtt{p}^{p}_{f,i,\mathtt{ta}(a),m}\}$.

Let $\sigma.f$ be a path of $\mathcal{M}(cr\text{-}cprs^{\leftarrow \mathtt{v}})$. Then

▶ $\overline{\mathbf{p}}_{f,i} = (\mathtt{p}_{f,i,1}, \ldots, \mathtt{p}_{f,i,m})$ and $\overline{\mathbf{p}}^{\mathcal{E}}_{f,i} = (\mathtt{p}^{\mathcal{E}}_{f,i,1}, \ldots, \mathtt{p}^{\mathcal{E}}_{f,i,m})$

are used to encode $(\sigma.f)_b(i)$ and $(\sigma.f)_a(i)$, respectively. With $\overline{\mathbf{p}}_{f,i}[j]$ and $\overline{\mathbf{p}}^{\mathcal{E}}_{f,i}[j]$ we denote, respectively, $\mathtt{p}_{f,i,j}$ and $\mathtt{p}^{\mathcal{E}}_{f,i,j}$. If $i \geq 1$, we define, for all $a \in A$:

▶ $\overline{\mathbf{p}}^{p}_{f,i} = (\mathtt{p}^{p}_{f,i,1,1}, \ldots, \mathtt{p}^{p}_{f,i,1,m}, \ldots, \mathtt{p}^{p}_{f,i,|A|,1}, \ldots, \mathtt{p}^{p}_{f,i,|A|,m})$.

The following functions map the background set entities to the corresponding variables of the encoding: for all $i \in \{0, \ldots, k\}$ we define $\mathtt{t}_{f,i} : S \to \mathbf{P}_{f,i}$ and $\mathtt{t}^{\mathcal{E}}_{f,i} : S \to \mathbf{P}^{\mathcal{E}}_{f,i}$ such that

▶ $\mathtt{t}_{f,i}(e_j) = \mathtt{p}_{f,i,j}$ and $\mathtt{t}^{\mathcal{E}}_{f,i}(e_j) = \mathtt{p}^{\mathcal{E}}_{f,i,j}$ for all $j \in \{1, \dots, m\}$.

For all $i \in \{0, \dots, k\}$ and $a \in A$ we define $\mathtt{t}^{p}_{f,i,a} : S \to \mathbf{P}^{p}_{f,i,a}$ such that:

▶ $\mathtt{t}^{p}_{f,i,a}(e_j) = \mathtt{p}^{p}_{f,i,\mathtt{ta}(a),j}$ for all $j \in \{1, \dots, m\}$.

The bijection $\mathtt{e} : Q \to \{1, \dots, |Q|\}$ maps states of the context automaton to the values used in the encoding. Let $\mathtt{tp} : P \to \{1, \dots, |P|\}$ be a bijection mapping all the parameters to their corresponding integers. Then we introduce the tuple of parameters:

▶ $\overline{\mathbf{p}}^{par} = (\mathtt{p}^{par}_{1,1}, \dots, \mathtt{p}^{par}_{1,m}, \dots, \mathtt{p}^{par}_{|P|,1}, \dots, \mathtt{p}^{par}_{P,m})$.

For each parameter $\lambda \in P$ we define

▶ $\mathbf{P}^{par}_{\lambda} = \{\mathtt{p}^{par}_{\mathtt{tp}(\lambda),1}, \dots, \mathtt{p}^{par}_{\mathtt{tp}(\lambda),m}\}$

and $\mathtt{pm}_{\lambda} : S \to \mathbf{P}^{par}_{\lambda}$ such that $\mathtt{pm}_{\lambda}(e_j) = \mathtt{p}^{par}_{\mathtt{tp}(\lambda),j}$. Let $a \in A$ and $\mathfrak{s} \in \{\mathfrak{r}_a, \mathfrak{i}_a, \mathfrak{p}_a\}$. Then, $re^{\mathfrak{s}}(e_j)$ denotes $\mathtt{pm}_{\mathfrak{s}}(e_j)$ if $\mathfrak{s} \in P$, and $\mathfrak{s}(e_j)$ otherwise. To define the SMT encoding of the paths we need auxiliary functions that correspond to elements of the encoding.

Initial state: To encode the initial state of the model for $\phi_f \in F$ we define

▶ $\mathsf{Init}(\overline{\mathbf{p}}_{f,i}, \mathtt{q}_{f,i}) = (\bigwedge_{e \in S} \mathtt{t}_{f,i}(e) = 0) \wedge \mathtt{q}_{f,i} = \mathtt{e}(q^{init})$,

where all the concentration levels are set to zero, and the context automaton is in its initial state.

Context: To encode a multiset $\mathbf{c} \in \mathcal{B}(S)$ of context entities we define:

▶ $\mathsf{Ct}_{\mathbf{c}}(\overline{\mathbf{p}}^{\mathcal{E}}_{f,i}) = \bigwedge_{e \in S} \mathtt{t}^{\mathcal{E}}_{f,i}(e) = \mathbf{c}(e)$

Parameter correctness: With $\mathsf{PC}(\overline{\mathbf{p}}^{par})$ we encode the parameter constraints, require that the concentration levels of the reactants are always lower than the concentration levels of the inhibitors, and ensure that all the multisets corresponding to the parameters are non-empty, i.e., for each parameter at least one entity must have positive concentration level:

$$\begin{aligned}\blacktriangleright\ \mathsf{PC}(\overline{\mathbf{p}}^{par}) = enc^{par}(\mathfrak{c}) \wedge (\textstyle\bigwedge_{a \in A} \bigwedge_{e \in S} re^{\mathfrak{i}_a}(e) > 0 \\ \Rightarrow (re^{\mathfrak{r}_a}(e) < re^{\mathfrak{i}_a}(e))) \wedge (\textstyle\bigwedge_{\lambda \in P} \bigvee_{e \in S} \mathtt{pm}_{\lambda}(e) > 0)\end{aligned}$$

where $enc^{par}(\mathfrak{c})$ is the encoding of $\mathfrak{c}$ which follows directly from the semantics of parameter constraints.

Parametric reaction: The parametric reactions $a \in A$ are encoded with

$$\begin{aligned}\blacktriangleright\ \mathsf{Rct}_a(\overline{\mathbf{p}}_{f,i}, \overline{\mathbf{p}}^{\mathcal{E}}_{f,i}, \overline{\mathbf{p}}^{p}_{f,i+1}, \overline{\mathbf{p}}^{par}) = \textstyle\bigwedge_{e \in S}((\mathtt{t}_{f,i}(e) \geq re^{\mathfrak{r}_a}(e) \vee \mathtt{t}^{\mathcal{E}}_{f,i}(e) \geq re^{\mathfrak{r}_a}(e)) \\ \wedge(\mathtt{t}_{f,i}(e) < re^{\mathfrak{i}_a}(e) \wedge \mathtt{t}^{\mathcal{E}}_{f,i}(e) < re^{\mathfrak{i}_a}(e)) \wedge (\mathtt{t}^{p}_{f,a,i+1}(e) = re^{\mathfrak{p}_a}(e))).\end{aligned}$$

The encoding for parametric reactions specifies the required concentration levels for $a \in A$ to be enabled, as well as encodes the concentration levels for the produced entities. The encoding for the produced entities uses the variables specific to the encoded reaction.

Transitions of cprs: Then, we encode the local state changes of *cprs* with

▶ $\mathsf{Tr}_{cprs}(\overline{\mathbf{p}}_{f,i}, \overline{\mathbf{p}}^{\mathcal{E}}_{f,i}, \overline{\mathbf{p}}^{p}_{f,i+1}, \overline{\mathbf{p}}_{f,i+1}, \overline{\mathbf{p}}^{par}) = (\bigwedge_{a \in A} \mathsf{Rct}_a(\overline{\mathbf{p}}_{f,i}, \overline{\mathbf{p}}^{\mathcal{E}}_{f,i}, \overline{\mathbf{p}}^{p}_{f,i+1}, \overline{\mathbf{p}}^{par})) \wedge (\bigwedge_{e \in S} \mathtt{t}_{f,i+1}(e) = max(\{0\} \cup \bigcup_{a \in A} \{\mathtt{t}^{p}_{f,a,i+1}(e)\}))$.

In this function, we encode the concentration levels for all the entities in the successor state using the individual concentration levels encoded for all $a \in A$ in Rct_a.

Transitions of context automaton: The encoding of the transition relation of the context automaton is a disjunction of the encoded transitions:

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

Transition relation: The transition relation of the model for *cr-cprs* is a conjunction of the transition relations for *cprs* and *ca*:

▶ $\mathsf{Tr}_{cr\text{-}cprs}(\overline{\mathbf{p}}_{f,i}, \mathsf{q}_{f,i}, \overline{\mathbf{p}}^{\mathcal{E}}_{f,i}, \overline{\mathbf{p}}^{p}_{f,i+1}, \overline{\mathbf{p}}_{f,i+1}, \overline{\mathbf{p}}^{par}) = \mathsf{Tr}_{cprs}(\overline{\mathbf{p}}_{f,i}, \overline{\mathbf{p}}^{\mathcal{E}}_{f,i}, \overline{\mathbf{p}}^{p}_{f,i+1}, \overline{\mathbf{p}}_{f,i+1}, \overline{\mathbf{p}}^{par}) \wedge \mathsf{Tr}_{ca}(\mathsf{q}_{f,i}, \overline{\mathbf{p}}^{\mathcal{E}}_{f,i}, \mathsf{q}_{f,i+1})$.

Paths: Finally, to encode the paths of $\mathcal{M}(cr\text{-}cprs^{\leftarrow \mathtt{v}})$ 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:

▶ $\mathsf{Paths}^k_f = \mathsf{Init}(\overline{\mathbf{p}}_{f,0}, \mathsf{q}_{f,0}) \wedge (\bigwedge_{i=0}^{k-1} \mathsf{Tr}_{cr\text{-}cprs}(\overline{\mathbf{p}}_{f,i}, \mathsf{q}_{f,i},, \overline{\mathbf{p}}^{\mathcal{E}}_{f,i}, \overline{\mathbf{p}}^{p}_{f,i+1}, \overline{\mathbf{p}}_{f,i+1}, \overline{\mathbf{p}}^{par}))$.

The encoded rsLTL formula ϕ_f at the position $i \in \{0, \ldots, k\}$ is denoted by $[\![\phi_f]\!]^k_i$. To encode the formula $[\![\phi_f]\!]^k_i$ we use our translation presented in Sect. 5 of [22]. However, for each formula $\phi_f \in F$, we use independent sets of variables corresponding to its path, i.e., the variables indexed with f. The encoding Loops^k_f for the loop positions is defined for each formula $\phi_f \in F$.

Calculation of Parameter Valuation. We perform the synthesis of the parameter valuation $\mathtt{v}$ by testing the satisfiability of the formula:

$$\left(\bigwedge_{\phi_f \in F} \mathsf{Paths}^k_f \wedge \mathsf{Loops}^k_f \wedge [\![\phi_f]\!]^k_0 \right) \wedge \mathsf{PC}(\overline{\mathbf{p}}^{par}).$$

Therefore, in the first step we test the satisfiability of the formula and then we extract the valuation of the parameters of P when the formula is satisfiable. That is, for the satisfied formula we obtain its model, i.e., the satisfying valuations of the variables used in the formula. Let $V(\mathtt{p})$ be the valuation of a variable $\mathtt{p}$ used in our encoding. Then, the parameter valuations are defined as follows: $\lambda^{\leftarrow \mathtt{v}}(e) = V(\mathtt{pm}_\lambda(e))$ for each $e \in S$ and $\lambda \in P$.

4 Experimental Evaluation

In this section we present the results of an experimental evaluation of the translation presented in Sect. 3. We test our method on the reaction system model for the mutual exclusion protocol (Mutex) introduced in [20]. The system consists of $n \geq 2$ processes competing for exclusive access to the critical section. The background set of crrsc modelling the mutual exclusion protocol is defined as $S = \bigcup_{i=1}^{n} S_i$, where the set of background entities corresponding to the i-th process is defined as $S_i = \{out_i, req_i, in_i, act_i, lock, done, s\}$, where the entities $lock$, $done$, and s are shared amongst all the processes. The set of reactions is defined as $A = \bigcup_{i=1}^{n} A_i \cup \big\{(\{lock\}, \{done\}, \{lock\})\big\}$, where A_i is the set of reactions associated with the i-th process. The complete description of the system may be found in [20]. The context automaton ca provides the initial context set and provides context sets such that only at most two simultaneously active processes are allowed. We define the crrsc modelling Mutex as $crrsc_M = ((S, A), ca)$.

Next, we assume here that the system is open and we allow for introducing new processes that participate in the communication to gain access to the critical section. Let us assume we are allowed to modify the behaviour of the additional process (here, the n-th process) only by introducing an additional reaction. Such an assumption could be justified by a mechanism that accepts new processes to participate in the protocol only if they contain the reactions of A_i for any $i \in \{1, \dots, n\}$, while the remaining reactions could be performing some computation outside of the critical section.

Our aim is to violate the property of mutual exclusion by making the first and the n-th process enter their critical sections simultaneously. The additional (malicious) reaction uses the parameters of $P = \{\lambda_r, \lambda_i, \lambda_p\}$. Then, we define the extended model $cr\text{-}cprs_M = ((S, P, A \cup \{(\lambda_r, \lambda_i, \lambda_p)\}, \mathfrak{c}), ca)$, where $\mathfrak{c} = \neg\lambda_p[in_n] \wedge \bigwedge_{\lambda \in P, e \in S \setminus S_n} \neg\lambda[e]$ constrains the additional reaction by requiring that it may produce only entities related to the n-th process and it cannot produce in_n, to avoid trivial solutions. Then, we need to synthesise a parameter valuation $\mathtt{v}$ of $cr\text{-}cprs_M$ which gives the rsLTL property $\phi = \mathbf{F}(in_1 \wedge in_n)$, i.e., $\mathcal{M}(cr\text{-}cprs_M^{\leftarrow \mathtt{v}}) \models_\exists \phi$.

The verification tool was implemented in Python and uses Z3 4.5.0 [21] 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 witnesses for all the verified formulae are found. Then, the corresponding parameter valuation is extracted. The verification results[1] presented in Figs. 1 and 2 compare four approaches: the implementation of the encoding from Sect. 3 ($cr\text{-}cprs$) and its extension ($cr\text{-}cprs_{opt}$) that optimises the obtained parameter valuations by using OptSMT provided with Z3. Then, we also use the same encoding for verification of the rsLTL property ($crrsc$), i.e., we replace all the parameters with the obtained parameter valuations and test the formula ϕ in the same way as in [22]. Next, we compare our results with the ones obtained using the non-parametric method ($crrsc_{np}$) of [22]. The results presented are attained from averaging three executions of the benchmark. Our experimental implementation provides a valuation $\mathtt{v}$ which allows to violate the mutual exclution property, where $\lambda_r^{\leftarrow \mathtt{v}} = \{out_n\}$, $\lambda_i^{\leftarrow \mathtt{v}} = \{s\}$, and

[1] The experimental results were obtained using a system equipped with 3.7 GHz Intel Xeon (E5-1620 v2) processor and 12 GB of memory, running Mac OS X 10.13.2.

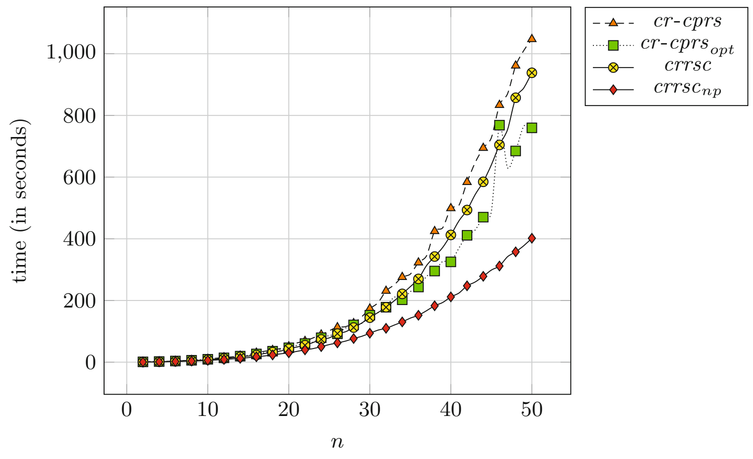

Fig. 1. Synthesis results for Mutex (time)

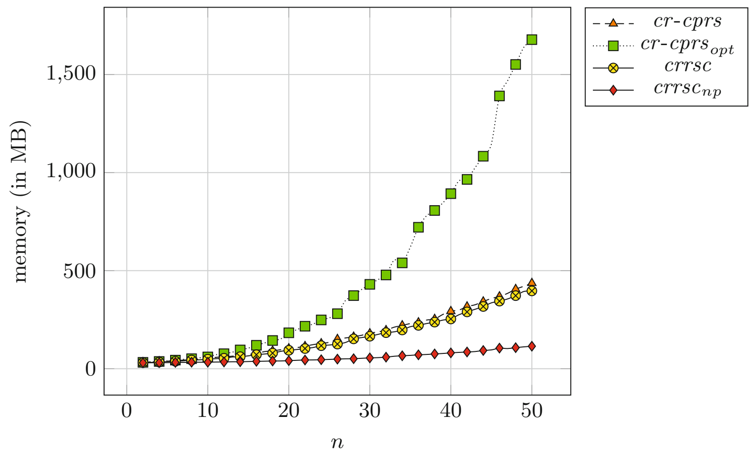

Fig. 2. Synthesis results for Mutex (memory)

$\lambda_p^{\leftarrow \mathrm{v}} = \{req_n, done\}$ for all the tested values $n \geq 2$. This valuation was obtained using $cr\text{-}cprs_{opt}$.

When using $cr\text{-}cprs_{opt}$, the memory consumption increases. However, the method might require less time to calculate the result than *cr-cprs*. The difference in time and memory consumption between the parametric (*cr-cprs*) and the non-parametric (*crrsc*) approach is minor. However, $crrsc_{np}$ is the most efficient of all the approaches tested. This suggests that our parameter synthesis method might possibly be improved by optimising the encoding used. However, this is merely a preliminary experimental evaluation and in the future we are going to test our method on a larger number of systems.

5 Concluding Remarks

We have presented a method for reaction mining which allows for calculating parameter valuations for partially defined reactions of reaction systems. We also demonstrated how the presented method can be used for synthesis of an attack in which we inject an additional instruction represented by a reaction, where we use rsLTL to express the goal of the attack.

Assuming there is a finite set of allowed concentration levels for the parameters, the presented method also allows for enumerating all the possible parameter valuations for fixed-length processes. This can be achieved by adding an additional constraint blocking the parameter valuation obtained in the previous step.

Our method focuses only on existential observations which can be obtained from simulations or experiments performed on the system. However, when we consider some widely accepted laws governing the system under investigation, those should be formulated as universal observations.

Since we use the bounded model checking approach, if no valid parameter valuation exists and no bound on k is assumed, then our method does not terminate.

In our future work we are going to focus on complexity considerations of the parameter synthesis, tackle the problem of universal observations, as well as optimise the SMT encoding.

Acknowledgements. W. Penczek acknowledges the support of the National Centre for Research and Development (NCBR), Poland, under the PolLux project VoteVerif (POL-LUX-IV/1/2016).

References

1. Alhazov, A., Aman, B., Freund, R., Ivanov, S.: Simulating R systems by P systems. In: Leporati, A., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) CMC 2016. LNCS, vol. 10105, pp. 51–66. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54072-6_4
2. Azimi, S., Gratie, C., Ivanov, S., Manzoni, L., Petre, I., Porreca, A.E.: Complexity of model checking for reaction systems. Theor. Comput. Sci. **623**, 103–113 (2016)
3. Azimi, S., Gratie, C., Ivanov, S., Petre, I.: Dependency graphs and mass conservation in reaction systems. Theor. Comput. Sci. **598**, 23–39 (2015)
4. Azimi, S., Iancu, B., Petre, I.: Reaction system models for the heat shock response. Fundamenta Informaticae **131**(3–4), 299–312 (2014)
5. Brijder, R., Ehrenfeucht, A., Rozenberg, G.: Reaction systems with duration. In: Kelemen, J., Kelemenová, A. (eds.) Computation, Cooperation, and Life. LNCS, vol. 6610, pp. 191–202. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20000-7_16
6. Corolli, L., Maj, C., Marini, F., Besozzi, D., Mauri, G.: An excursion in reaction systems: from computer science to biology. Theor. Comput. Sci. **454**, 95–108 (2012)
7. Dennunzio, A., Formenti, E., Manzoni, L.: Reaction systems and extremal combinatorics properties. Theor. Comput. Sci. **598**, 138–149 (2015)
8. Dennunzio, A., Formenti, E., Manzoni, L., Porreca, A.E.: Ancestors, descendants, and gardens of eden in reaction systems. Theor. Comput. Sci. **608**, 16–26 (2015)

9. Ehrenfeucht, A., Kleijn, J., Koutny, M., Rozenberg, G.: Reaction systems: a natural computing approach to the functioning of living cells. In: A Computable Universe, Understanding and Exploring Nature as Computation, pp. 189–208 (2012)
10. Ehrenfeucht, A., Kleijn, J., Koutny, M., Rozenberg, G.: Evolving reaction systems. Theor. Comput. Sci. **682**, 79–99 (2017)
11. Ehrenfeucht, A., Rozenberg, G.: Reaction systems. Fundamenta Informaticae **75**(1–4), 263–280 (2007)
12. Ehrenfeucht, A., Rozenberg, G.: Introducing time in reaction systems. Theor. Comput. Sci. **410**(4–5), 310–322 (2009)
13. Formenti, E., Manzoni, L., Porreca, A.E.: Cycles and global attractors of reaction systems. In: Jürgensen, H., Karhumäki, J., Okhotin, A. (eds.) DCFS 2014. LNCS, vol. 8614, pp. 114–125. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09704-6_11
14. Formenti, E., Manzoni, L., Porreca, A.E.: Fixed points and attractors of reaction systems. In: Beckmann, A., Csuhaj-Varjú, E., Meer, K. (eds.) CiE 2014. LNCS, vol. 8493, pp. 194–203. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08019-2_20
15. Formenti, E., Manzoni, L., Porreca, A.E.: On the complexity of occurrence and convergence problems in reaction systems. Nat. Comput. **14**, 1–7 (2014)
16. Hirvensalo, M.: On probabilistic and quantum reaction systems. Theor. Comput. Sci. **429**, 134–143 (2012)
17. Horn, F., Jackson, R.: General mass action kinetics. Arch. Ration. Mech. Anal. **47**(2), 81–116 (1972)
18. Kroening, D., Strichman, O.: Decision Procedures - An Algorithmic Point of View. Texts in Theoretical Computer Science. An EATCS Series, 2nd edn. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-50497-0
19. Męski, A., Koutny, M., Penczek, W.: Towards quantitative verification of reaction systems. In: Amos, M., Condon, A. (eds.) UCNC 2016. LNCS, vol. 9726, pp. 142–154. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41312-9_12
20. Męski, A., Penczek, W., Rozenberg, G.: Model checking temporal properties of reaction systems. Inf. Sci. **313**, 22–42 (2015)
21. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
22. Męski, A., Koutny, M., Penczek, W.: Verification of linear-time temporal properties for reaction systems with discrete concentrations. Fundam. Inform. **154**(1–4), 289–306 (2017)
23. Salomaa, A.: Functions and sequences generated by reaction systems. Theor. Comput. Sci. **466**, 87–96 (2012)
24. Salomaa, A.: On state sequences defined by reaction systems. In: Constable, R.L., Silva, A. (eds.) Logic and Program Semantics. LNCS, vol. 7230, pp. 271–282. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29485-3_17
25. Salomaa, A.: Functional constructions between reaction systems and propositional logic. Int. J. Found. Comput. Sci. **24**(1), 147–160 (2013)
26. Salomaa, A.: Minimal and almost minimal reaction systems. Nat. Comput. **12**(3), 369–376 (2013)