Natural Computing
https://doi.org/10.1007/s11047-025-10044-7

Model checking for distributed reaction systems with temporal-epistemic properties

Artur Meski[1,2] · Maciej Koutny[3] · Łukasz Mikulski[4] · Ion Petre[5] · Wojciech Penczek[2] · Marcin Piatkowski[4,6]

Accepted: 22 July 2025

Abstract
Reaction systems are a model of computation inspired by the biochemistry exhibited by living cells. This paper introduces the notion of agency as an extension to the reaction systems formalism, leading to *distributed* reaction systems. Adding agents in the reaction systems setting, allows for the natural modelling and representation of multi-agent and distributed systems. To support the specification of temporal-epistemic properties of distributed reaction systems, we introduce the logic rsCTLK and present experimental results of its associated model checking procedure run on a biological benchmark of within-cell signal transduction networks. The experimental results are encouraging despite the complexity of the rsCTLK model checking problem that is shown to be PSPACE-complete.

Keywords Reaction systems · Model checking · Temporal-epistemic logics

1 Introduction and related work

Natural computing is a fast developing research field concerned with investigating models and computational techniques inspired by nature, as well as investigating, in terms of information processing, complex phenomena taking place in nature (Kari and Rozenberg 2008) and Rozenberg et al. (2012). Within the sub-field of bio-molecular computation, a particular strand dealing with the latter kind of investigation aims at establishing how bio-computations drive natural processes. A prominent formal model introduced to support such an endeavour is the model of *reaction systems* (Ehrenfeucht and Rozenberg 2007b; Ehrenfeucht et al. 2013; Dennunzio et al. 2014). They were proposed as a formal model of the functioning of the living cell, where this functioning is viewed in terms of formal processes resulting from interactions between biochemical reactions. Crucially, these interactions are driven by the mechanisms of facilitation and inhibition: the (products of the) reactions may facilitate or inhibit each other. The basic model of reaction systems is qualitative rather than quantitative. However, it takes into account the basic flow of energy of the living cell (Lehninger 1965), and the fact that the behaviour of the living cell is influenced by its environment. Having originally been inspired by the behaviour of the living cell, recent research on reaction systems has been motivated by both biological and medical considerations (Ehrenfeucht and Rozenberg 2007a, 2009;

Maciej Koutny, Łukasz Mikulski, Ion Petre, Wojciech Penczek and Marcin Piatkowski have contributed equally to this work.

✉ Łukasz Mikulski
lukasz.mikulski@mat.umk.pl

Artur Meski
meski@ipipan.waw.pl

Maciej Koutny
maciej.koutny@ncl.ac.uk

Ion Petre
ion.petre@utu.fi

Wojciech Penczek
penczek@ipipan.waw.pl

Marcin Piatkowski
marcin.piatkowski@mat.umk.pl

1 Vector GB Ltd., London, UK

2 Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

3 School of Computing, Newcastle University, Newcastle upon Tyne, UK

4 Faculty of Mathematics and Computer Science, Nicolaus Copernicus University in Toruń, Torun, Poland

5 Department of Mathematics and Statistics, University of Turku, Turku, Finland

6 Interdisciplinary Centre for Mathematical and Computational Modeling, University of Warsaw, Warsaw, Poland

Published online: 05 November 2025

Brijder et al. 2011; Corolli et al. 2012; Azimi et al. 2015b; Ehrenfeucht et al. 2017; Bowles et al. 2024) as well as the general considerations concerned with the understanding of computations taking place in reaction systems (Ehrenfeucht et al. 2010, 2011; Kleijn et al. 2011; Salomaa 2013; Dennunzio et al. 2014; Genova et al. 2017). Reaction systems are perceived as a novel general model of interactive computation.

Verification is a key area of reaction systems research, e.g. Azimi et al. (2015a, 2016); Męski et al. (2015); Dennunzio et al. (2019); Brodo et al. (2023). There are results showing model checking methods for branching time temporal properties specified in CTL for reaction systems (rsCTL) (Męski et al. 2015) as well as for linear time temporal properties (Męski et al. 2016; Męski et al. 2017). An encoding of a parametric version of reaction systems into SMT was defined and a synthesis based on bounded model checking for properties specified in linear time temporal logic was proposed (Męski et al. 2018), while languages of distributed reaction systems (based on the definition of the preliminary technical report (Męski et al. 2019) of this paper) are studied in Ciencialová et al. (2022, 2023).

Temporal-epistemic properties of multi-agent systems can be expressed using CTL*K (van der Meyden and Wong 2003; Jones et al. 2012)—a logic combining epistemic and temporal operators of CTL* (Emerson and Halpern 1986; Ferrando and Malvone 2021). A symbolic model checking method (Clarke et al. 2001) for interpreted systems (Lomuscio and Ryan 1997) and CTLK specifications was proposed in Raimondi and Lomuscio (2004, 2005) and a bounded model checking approach for multi-agent systems (Wooldridge 2002; Maubert et al. 2023) and temporal-epistemic properties was put forward in Penczek and Lomuscio (2003) and considered also in Męski et al. (2014).

Below, we discuss the main contributions of this paper. First of all, we introduce the notion of agency as an extension to the reaction systems, leading to *distributed* reaction systems (DRS). Adding agents in the reaction systems setting allows for the natural modelling and verification of multi-agent and distributed systems as well as different synchronisation schemes. Crucially, it is possible to model systems employing both synchronous and asynchronous execution semantics, and an extended notion of context automata allowing conditional generation of contexts. We demonstrate how the formalism of DRS can be applied to modelling of multi-agent systems.

To enable specifying temporal-epistemic properties of DRS we define a new logic for reaction systems, rsCTLK, which extends rsCTL (Męski et al. 2017) with epistemic operators. Then, we introduce a model checking method for rsCTLK. The method proposed is implemented using binary decision diagrams for symbolic model checking and the method is evaluated experimentally. Finally, we prove that model checking for rsCTLK is PSPACE-complete.

We present the running example inspired by the mechanistic model of within-cell signal transduction networks of Zañudo et al. (2017). The modelled signal transduction pathway consist of signalling proteins, enzymes, receptor proteins, and signaling molecules that can take two states. We define signalling pathway starting with growth factor GF which, through receptor tyrosine kinase RTK, activates signalling protein RAS and subsequently recruits other signaling proteins like kinase RAF and mitogen-activated protein kinase cascade MEK. This, together with phosphorylated kinase AKT, activates transcription factors TF. We check if an external agent always knows whether or nor TF was observed in each pathway.

The paper is organised as follows. In Sect. 2, we recall the basic notions and definitions used by reaction systems. Section 3 introduces distributed reaction systems and Sect. 4 defines rsCTLK. In Sect. 5, a model checking method for DRS and rsCTLK is introduced. In Sect. 7, we evaluate experimentally an implementation of the proposed verification method. In the last section of this paper we draw some conclusions.

2 Preliminaries

A *reaction system* is a pair $\mathcal{R} = (S, A)$, where S is a finite *background* set and A is a set of *reactions* over S. Each reaction in A is a triple $b = (R, I, P)$ such that R, I, P are nonempty subsets of S with $R \cap I = \emptyset$. 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. By $racS$ we denote all reactions over S.

A reaction $b \in A$ is *enabled* by $T \subseteq S$, denoted $en_b(T)$, if $R_b \subseteq T$ and $I_b \cap T = \emptyset$. The *result* of b on T is given by $res_b(T) = P_b$ if $en_b(T)$, and by $res_b(T) = \emptyset$ 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.

Example 1 The running example is inspired by the mechanistic model of within-cell signal transduction networks of Zañudo et al. (2017). We start with a simple reaction system that mimics a simplified discrete dynamic model of the signalling network.

The modelled signal transduction pathway consist of signalling proteins, enzymes, receptor proteins, and signalling molecules that can take two states – ON and OFF. We define signalling pathway starting with growth factor GF which,

through receptor tyrosine kinase RTK, activates signalling protein RAS and subsequently recruits other signalling proteins like kinase RAF and mitogen-activated protein kinase cascade MEK (MEK/ERK). This, together with phosphorylated kinase AKT, activates transcription factors TF. Note that in the initial model presented in this example, kinase AKT is not produced by the considered pathway and needs to be provided by the environment. We will model the second pathway providing kinase AKT in subsequent examples.

Moreover, we simulate drug inhibition by adding to initial network RTK inhibitor RTKi and MEK inhibitor MEKi, and introduce dummy inhibitor $\star$ (necessary to comply with the definition, see Męski et al. (2015)). More details are in Example 3, which demonstrates that the experiment of Zañudo et al. (2017) can be recreated with distributed reaction systems.

Let $(S, A_1) = (\{\text{GF}, \text{RTK}, \text{RTK}i, \text{RAS}, \text{RAF}, \text{MEK}, \text{MEK}i, \text{AKT}, \text{TF}, \star\}, \{a_1, \ldots, a_6\})$ be a reaction system, with reactions:

$a_1 : (\{\text{GF}\}, \{\star\}, \{\text{GF}\})$
$a_2 : (\{\text{GF}\}, \{\star, \text{RTK}i\}, \{\text{RTK}\})$
$a_3 : (\{\text{RTK}\}, \{\star\}, \{\text{RAS}\})$
$a_4 : (\{\text{RAS}\}, \{\star\}, \{\text{RAF}\})$
$a_5 : (\{\text{RAF}\}, \{\star, \text{MEK}i\}, \{\text{MEK}\})$
$a_6 : (\{\text{MEK}, \text{AKT}\}, \{\star\}, \{\text{TF}\})$

Note that reaction a_2 specifies that the presence (ON) of growth factor GF and the absence (OFF) of both dummy inhibitor $\star$ and inhibitor RTKi, implies the activation (changing state to ON) or maintaining the activity (staying in ON) of the receptor tyrosine kinase RTK. No other reaction activates the receptor tyrosine kinase RTK. Hence, if one of the conditions is not met (presence of GF or absence of both $\star$ and RTKi), the receptor tyrosine kinase RTK is deactivated (changing state to OFF).

Note that in the state $T = \{\text{GF}, \text{RTK}i, \text{RTK}, \text{RAF}, \text{AKT}\}$ the reactions a_1, a_3 and a_5 are enabled, while a_2, a_4 and a_6 are not. Hence $res_{A_1}(T) = res_{a_1}(T) \cup res_{a_3}(T) \cup res_{a_5}(T) = \{\text{GF}, \text{RAS}, \text{MEK}\}$.

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 had to be produced (sustained) by a reaction enabled by T (and so $x \in res_A(T)$). Also, there are no conflicts between reactions.

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 decide 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. The definition of context automaton we provide allows for non-determinism in two dimensions. Not only can one define different outgoing edges that connect two different states, but also multiple edges providing different contexts between the same pair of states.

A *context automaton* (CA) over Σ, is a triple $\mathfrak{A} = (\mathcal{Q}, \mathfrak{q}^{init}, R)$, where: Σ is a finite set of labels, $\mathcal{Q}$ is a finite set of *locations*, $\mathfrak{q}^{init} \in \mathcal{Q}$ is the *initial location*, and $R \subseteq \mathcal{Q} \times \Sigma \times \mathcal{Q}$ is a *transition relation* labelled with elements of Σ. We assume that R is serial, i.e., for each $\mathfrak{q} \in \mathcal{Q}$ there is $c \in \Sigma$ and $\mathfrak{q}' \in \mathcal{Q}$ such that $(\mathfrak{q}, c, \mathfrak{q}') \in R$.

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

- $\zeta = (z_0, z_1, \ldots, z_n)$, $\gamma = (C_0, C_1, \ldots, C_n)$, $\delta = (D_0, D_1, \ldots, D_n)$,
- $z_0, z_1, \ldots, z_n \in Q$ with $z_0 = \mathfrak{q}^{init}$,
- $C_0, C_1, \ldots, C_n, D_0, D_1, \ldots, D_n \subseteq S$ 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 $i \in \{1, \ldots, n\}$.

The *state sequence* of π is $\tau = (C_0 \cup D_0, \ldots, C_n \cup D_n)$.

Intuitively, the state sequence of π captures the behaviour of CR-$\mathcal{R}$ by recording the states obtained by the evolution of the reaction system $\mathcal{R}$ in the environment modelled by the context automaton $\mathfrak{A}$.

3 Distributed reaction systems

A distributed reaction system models a system that consists of many *agents*, where each agent has its own set of reactions defined over a common background set. Let $\mathcal{A}$ be an nonempty finite set, the elements of which are called *agents*. A *distributed reaction system* (DRS) is a tuple $\mathcal{D} = (S, \{A_i\}_{i \in \mathcal{A}})$, where S is a finite nonempty set, and $A_i \subseteq rac(S)$ for each $i \in \mathcal{A}$.

Given a DRS $\mathcal{D}$, we refer to S as $\mathcal{D}$'s *background set*. Throughout this paper, $\mathcal{A} = \{1, \ldots, m\}$ is a fixed set of agents, unless it is specified otherwise. Each agent maintains its own local state, and the tuples of $St_{\mathcal{D}} = \prod^m 2^S$ are called the *states* of $\mathcal{D}$.

At each transition from a global state to its successor, the environment provides the DRS with a *context*. Each context contains a set of entities for each agent, and specifies the activated agents. The contexts are defined as pairs $Ct_{\mathcal{D}} = St_{\mathcal{D}} \times 2^{\mathcal{A}}$, and we use $\mathbf{C}^c$ and $\mathbf{C}^a$ to respectively denote the first and the second component of a context $\mathbf{C} \in Ct_{\mathcal{D}}$. Thus $\mathbf{C}^c$ represent a tuple of sets of entities provided by the environment, and $\mathbf{C}^a$ denotes the *activated* agents.

The evolution of a DRS in an unconstrained environment is captured by a suitably redefined notion of an interactive

process, where the activated agents combine (share) their local states to derive the next local states. For dealing with tuples we use the following notation: if $x = (x_1, x_2, \ldots, x_n)$ is a tuple, then $x[i] = x_i$, for every $i \leq n$. With $\overline{\varnothing}^m$ we denote the tuple consisting of m empty sets.

We now introduce the notion of an interactive process for DRS. Let $\mathcal{D} = (S, \{A_i\}_{i \in \mathcal{A}})$ be a DRS. An n-step *interactive process in* $\mathcal{D}$ is $\pi = (\gamma, \delta)$, where:

- $\gamma = (\mathbf{C}_0, \mathbf{C}_1, \ldots, \mathbf{C}_n)$ and $\delta = (\mathbf{D}_0, \mathbf{D}_1, \ldots, \mathbf{D}_n)$,
- $\mathbf{C}_0, \mathbf{C}_1, \ldots, \mathbf{C}_n \in Ct_{\mathcal{D}}$,
- $\mathbf{D}_0, \mathbf{D}_1, \ldots, \mathbf{D}_n \in St_{\mathcal{D}}$ with $\mathbf{D}_0 = \overline{\varnothing}^m$,
- for each $k \in \{0, \ldots, n-1\}$ and each $i \in \mathcal{A}$:

$$\mathbf{D}_{k+1}[i] = \begin{cases} res_{A_i}\left(\mathbf{C}_k^c[i] \cup \bigcup_{j \in \mathbf{C}_k^a} \mathbf{D}_k[j]\right) & \text{if } i \in \mathbf{C}_k^a \\ \mathbf{D}_k[i] & \text{if } i \notin \mathbf{C}_k^a \end{cases}.$$

Now we give an intuition for the above definition. For an activated agent $i \in \mathcal{A}$, i.e., $i \in \mathbf{C}_k^a$, to calculate its local successor state $\mathbf{D}_{k+1}[i]$, we take its context set $\mathbf{C}_k^c[i]$ and the union of all local states of the activated agents (state sharing). If $i \in \mathcal{A}$ is not activated ($i \notin \mathbf{C}_k^a$), then $\mathbf{D}_k[i]$ remains unchanged, i.e., $\mathbf{D}_{k+1}[i] = \mathbf{D}_k[i]$.

Example 2 We modify Example 1 by adding the second signalling pathway, which using signalling protein PI3K and phosphorylating the phospholipid PIP2 into PIP3, leads to the phosphorylation of the kinase AKT. So, we get a simple example where two agents synchronise to activate transcription factors TF by combining their local states.

Let $\mathcal{D}_1 = (S', \{A_1, A_2\})$ be a DRS, where $S' = S \cup \{\text{PI3K}, \text{PIP3}\}$ is the background set and $A_2 = \{a_7, a_8, \ldots, a_{13}\}$ is the set of reactions defined as follows:

$a_7 : (\{\text{GF}\}, \{\star\}, \{\text{GF}\})$,
$a_8 : (\{\text{GF}\}, \{\star, \text{RTK}i\}, \{\text{RTK}\})$,
$a_9 : (\{\text{RTK}\}, \{\star\}, \{\text{PI3K}\})$,
$a_{10} : (\{\text{RAS}\}, \{\star\}, \{\text{PI3K}\})$,
$a_{11} : (\{\text{PI3K}\}, \{\star\}, \{\text{PIP3}\})$,
$a_{12} : (\{\text{PIP3}\}, \{\star\}, \{\text{AKT}\})$,
$a_{13} : (\{\text{MEK}, \text{AKT}\}, \{\star\}, \{\text{TF}\})$.

Recall that the entity $\star$ is used as a dummy inhibitor. Consider the process $\pi = (\gamma, \delta)$ in $\mathcal{D}_1$ s.t. $\gamma = (\mathbf{C}_0, \mathbf{C}_1, \ldots)$ and $\delta = (\mathbf{D}_0, \mathbf{D}_1, \ldots)$ where $\mathbf{C}_i$ and $\mathbf{D}_i$ for $i \in \{0, \ldots, 9\}$ are defined as in Fig. 1.

We begin with the context $\mathbf{C}_0 = ((\{\text{GF}\}, \{\text{GF}\}), \{1, 2\})$ and the state $\mathbf{D}_0 = \overline{\varnothing}^2$, where both agents 1 and 2 are active and *receive* GF. By combining the local states of agents 1 and 2 we get $\varnothing$. We apply the reactions of A_1 to $\{\text{GF}\} \cup \varnothing$ and the reactions of A_2 to $\{\text{GF}\} \cup \varnothing$. As the result, we get the state $\mathbf{D}_1 = (\{\text{GF}, \text{RTK}\}, \{\text{GF}, \text{RTK}\})$, where $res_{A_1}(\{\text{GF}\} \cup \varnothing) = \{\text{GF}, \text{RTK}\}$ is the local state of agent 1 and $res_{A_2}(\{\text{GF}\} \cup \varnothing) = \{\text{GF}, \text{RTK}\}$ is the local state of agent 2.

i	$\mathbf{C}_i$	$\mathbf{D}_i$
0	$((\{\text{GF}\}, \{\text{GF}\}), \{1, 2\})$	$\overline{\varnothing}^2$
1	$(\overline{\varnothing}^2, \{1\})$	$(\{\text{GF}, \text{RTK}\}, \{\text{GF}, \text{RTK}\})$
2	$(\overline{\varnothing}^2, \{1\})$	$(\{\text{GF}, \text{RTK}, \text{RAS}\}, \{\text{GF}, \text{RTK}\})$
3	$(\overline{\varnothing}^2, \{1\})$	$(\{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}\}, \{\text{GF}, \text{RTK}\})$
4	$(\overline{\varnothing}^2, \{1, 2\})$	$(\{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}, \text{MEK}\}, \{\text{GF}, \text{RTK}\})$
5	$(\overline{\varnothing}^2, \{2\})$	$(\{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}, \text{MEK}\}, \{\text{GF}, \text{RTK}, \text{PI3K}\})$
6	$(\overline{\varnothing}^2, \{2\})$	$(\{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}, \text{MEK}\}, \{\text{GF}, \text{RTK}, \text{PI3K}, \text{PIP3}\})$
7	$(\overline{\varnothing}^2, \varnothing)$	$(\{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}, \text{MEK}\}, \{\text{GF}, \text{RTK}, \text{PI3K}, \text{PIP3}, \text{AKT}\})$
8	$(\overline{\varnothing}^2, \{1, 2\})$	$(\{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}, \text{MEK}\}, \{\text{GF}, \text{RTK}, \text{PI3K}, \text{PIP3}, \text{AKT}\})$
9	$(\overline{\varnothing}^2, \varnothing)$	$(\{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}, \text{MEK}, \text{TF}\}, \{\text{GF}, \text{RTK}, \text{PI3K}, \text{PIP3}, \text{AKT}, \text{TF}\})$

Fig. 1 Sample interactive process

Then $\mathbf{D}_9 = (\{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}, \text{MEK}, \text{TF}\}, \{\text{GF}, \text{RTK}, \text{PI3K}, \text{PIP3}, \text{AKT}, \text{TF}\})$ is a DRS obtained after eight steps presented in Fig. 1.

Notice that AKT is required for TF to be produced by agent 1 and the entity is provided by agent 2 by sharing the local states of the active agents. Similarly MEK is required for TF to be produced in 2. The context $\mathbf{C}_9$ is defined only for completeness and it could be used only to calculate the successor of $\mathbf{D}_9$. In this way, we have modelled two asynchronous signalling pathways. They are independent in the middle of the process, while the synchronisation takes place at the initial stage (appearance of RTK, common activation signal), as well as at the end (activation of the transcription factor requires signals from both pathways). Note that the interaction between both pathways is hardcoded in the utilised formalism and can be further specified and fitted using context automata. Note that in the provided example we have an apparent synchronisation in step 4, where both agents are activated. However, it does not affect the entire process, as for agent 1 it is an idle step (context is empty, second agent does not provide anything interesting, and the set of performed reactions is equal to the previous step). In Example 3, we provide an alternative context sequence with the same result, but without this apparent synchronisation. The simultaneous activation of two agents is necessary to synchronise them; however, it might be not sufficient and result in parallel execution of independent activities.

Interactive processes for DRS capture all possible evolutions of a DRS. However, the behaviour of an environment is constrained on the current state of the DRS. Note that this solution is biologically motivated and allows feedback loops between an organism and the environment (Meacock

and Mitri 2025). In general, this gives a possibility to treat the environment as a (collective) agent that may respond to the actions of other agents. We capture such an environment through the notion of an extended context automaton, where the transitions between the states are guarded by formulae restricting the allowed transitions for local states of the agents.

The *state constraints* $\mathcal{SC}_\mathcal{D}$ are Boolean formulae with propositions in the form of $i.ent$, where $i \in \mathcal{A}$ and $ent \in S$. Their grammar is:

$\mathfrak{sc} ::= true \mid i.ent \mid \neg\mathfrak{sc} \mid \mathfrak{sc} \vee \mathfrak{sc}$.

By $\mathbf{W} \models_{sc} \mathfrak{sc}$ we denote that $\mathfrak{sc} \in \mathcal{SC}_\mathcal{D}$ holds in $\mathbf{W} \in St_\mathcal{D}$. The satisfaction relation $\models_{sc}$ is defined recursively as follows:

- $\mathbf{W} \models_{sc} true$ for any $\mathbf{W} \in St_\mathcal{D}$,
- $\mathbf{W} \models_{sc} i.e$ iff $e \in \mathbf{W}[i]$,
- $\mathbf{W} \models_{sc} \neg\mathfrak{sc}$ iff $\mathbf{W} \not\models_{sc} \mathfrak{sc}$,
- $\mathbf{W} \models_{sc} \mathfrak{sc}_1 \vee \mathfrak{sc}_2$ iff $\mathbf{W} \models_{sc} \mathfrak{sc}_1$ or $\mathbf{W} \models_{sc} \mathfrak{sc}_2$.

An *extended context automaton* (ECA) over $\mathcal{D}$, is a triple $\mathfrak{E} = (\mathcal{Q}, \mathfrak{q}^{init}, R)$ where: $\mathcal{Q}$ is a finite set of *locations*, $\mathfrak{q}^{init} \in \mathcal{Q}$ is the *initial location*, and $R \subseteq \mathcal{Q} \times \mathcal{SC}_\mathcal{D} \times Ct_\mathcal{D} \times \mathcal{Q}$ is a *transition relation*.

For $(\mathfrak{q}, \mathfrak{sc}, \mathbf{C}, \mathfrak{q}') \in R$ we also write $\mathfrak{q} \xrightarrow{\mathfrak{sc},\mathbf{C}} \mathfrak{q}'$. An extended context automaton over $\mathcal{D}$ is simply called *extended context automaton* if $\mathcal{D}$ is clear from the context.

$\mathfrak{E}$ is *progressive* if for all $\mathfrak{q} \in \mathcal{Q}$ and $\mathbf{W} \in St_\mathcal{D}$ there exist $\mathfrak{sc} \in \mathcal{SC}_\mathcal{D}$, $\mathbf{C} \in Ct_\mathcal{D}$, and $\mathfrak{q}' \in \mathcal{Q}$ such that $(\mathfrak{q}, \mathfrak{sc}, \mathbf{C}, \mathfrak{q}') \in R$ and $\mathbf{W} \models_{sc} \mathfrak{sc}$.

We formalise the notion of a DRS evolving in an environment provided by a progressive ECA: a *context-restricted distributed reaction system* (CRDRS) is a pair CR-$\mathcal{D}$ $= (\mathcal{D}, \mathfrak{E})$ such that $\mathcal{D}$ is a DRS and $\mathfrak{E}$ is a progressive ECA. The set of states of CR-$\mathcal{D}$ is defined as $St_{\text{CR-}\mathcal{D}} = St_\mathcal{D} \times \mathcal{Q}$. Let $\mathcal{S} \in St_{\text{CR-}\mathcal{D}}$ and $\mathcal{S} = (\mathbf{W}, \mathfrak{q})$, then we write $\mathcal{D}(\mathcal{S})$ for $\mathbf{W}$ and $\mathfrak{E}(\mathcal{S})$ for $\mathfrak{q}$.

In CA we assume that the transition relation is serial. This allows us to avoid the situation where 'the environment dies', i.e., where the context automaton stops providing contexts to the reaction system. Similarly, given a CRDRS, we assume that the ECA is progressive. In ECAs, for a transition to be enabled, the state constraint associated with the transition must be satisfied. Checking progressiveness of an ECA amounts to checking, if for each location and each $\mathbf{W} \in St_\mathcal{D}$ there exists a transition for which its state constraint is satisfied. This means that checking if an ECA is progressive could be involved. We take a different approach and instead of checking if a given ECA is progressive, we construct a corresponding progressive ECA.

Progressiveness can be easily imposed on any extended context automaton $\mathfrak{E} = (\mathcal{Q}, \mathfrak{q}^{init}, R)$. Let $\perp \notin \mathcal{Q}$ be a location, $\mathcal{Q}' = \mathcal{Q} \cup \{\perp\}$, and let $prg(\mathfrak{E}) = (\mathcal{Q}', \mathfrak{q}^{init}, R')$ be an ECA such that $R' = R \cup R_\perp \cup \{(\perp, true, (\overline{\varnothing}^m, \varnothing), \perp)\}$, where
$R_\perp = \{(q, \neg\mathfrak{sc}_q, (\overline{\varnothing}^m, \varnothing), \perp) \mid q \in Q\}$ and $\mathfrak{sc}_q = \bigvee\{\mathfrak{sc} \mid (\mathfrak{q}, \mathfrak{sc}, \mathbf{C}, \mathfrak{q}') \in R\}$,
for every $\mathfrak{q} \in \mathcal{Q}$.

In the above construction, we add a special location $\perp$ with a self-loop, and for each transition we add an additional transition with the state constraint negated. The added transitions lead to $\perp$, provide empty contexts, and do not activate any agents. The following result follows immediately from the above construction.

Lemma 1 *$prg(\mathfrak{E})$ is a progressive ECA, for every $\mathfrak{E}$.*

An n-step *interactive process in* CR-$\mathcal{D}$ is a triple (ζ, γ, δ), where:

- $\zeta = (\mathfrak{q}_0, \ldots, \mathfrak{q}_n)$, $\gamma = (\mathbf{C}_0, \ldots, \mathbf{C}_n)$, and $\delta = (\mathbf{D}_0, \ldots, \mathbf{D}_n)$,
- $\mathfrak{q}_0, \ldots, \mathfrak{q}_n \in \mathcal{Q}$ with $\mathfrak{q}_0 = \mathfrak{q}^{init}$, and $\mathbf{C}_0, \ldots, \mathbf{C}_n \in Ct_\mathcal{D}$,
- $\mathbf{D}_0, \ldots, \mathbf{D}_n \in St_\mathcal{D}$ with $\mathbf{D}_0 = \overline{\varnothing}^m$,
- for each $k \in \{0, \ldots, n-1\}$ there exists $\mathfrak{sc} \in \mathcal{SC}_\mathcal{D}$ such that $\mathbf{D}_k \models_{sc} \mathfrak{sc}$ as well as $(\mathfrak{q}_k, \mathfrak{sc}, \mathbf{C}_k, \mathfrak{q}_{k+1}) \in R$ and, for each $i \in \mathcal{A}$:

$$\mathbf{D}_{k+1}[i] = \begin{cases} res_{A_i}(\mathbf{C}_k^c[i] \cup \bigcup_{j \in \mathbf{C}_k^a} \mathbf{D}_k[j]) & \text{if } i \in \mathbf{C}_k^a, \\ \mathbf{D}_k[i] & \text{if } i \notin \mathbf{C}_k^a. \end{cases}$$

Example 3 We continue with the discrete dynamic network model of Zañudo et al. (2017), which was exploited to give Examples 1 and 2. Its dynamic part is defined through a series of regulatory functions (denoted with the symbol f) assigned to positive values of state variables (denoted with the symbol σ) with the use of logical operators AND, OR, and NOT. This way $f_{\text{PI3K}} = \sigma_{\text{RTK}}$ OR σ_{RAS} means that a regulatory function f_{PI3K} activates or maintains active the signalling protein PI3K (resulting in setting the state of the variable σ_{PI3K} to ON) assuming that the state of at least one of the variables σ_{RTK} or σ_{RAS} is equal to ON (meaning that either receptor tyrosine kinase RTK or signalling protein RAS or both of them are active). A simplified network of PI3K and MAPK signalling initialized by receptor tyrosine kinases (RTK) can be represented by the following logical rules (see Fig. 2):

- $f_{\text{PI3K}} = \sigma_{\text{RTK}}$ OR σ_{RAS}
- $f_{\text{RAF}} = \sigma_{\text{RAS}}$
- $f_{GrowthFactor} = \sigma_{GrowthFactor}$
- $f_{\text{RAS}} = \sigma_{\text{RTK}}$
- $f_{\text{MEK}} = \sigma_{\text{RAF}}$ AND NOT $\sigma_{\text{MEK}i}$
- $f_{\text{AKT}} = \sigma_{\text{PIP3}}$
- $f_{\text{RTK}} = \sigma_{GrowthFactor}$ AND NOT $\sigma_{\text{RTK}i}$
- $f_{\text{PIP3}} = \sigma_{\text{PI3K}}$
- $f_{TranscriptionFactors} = \sigma_{\text{AKT}}$ AND σ_{MEK}

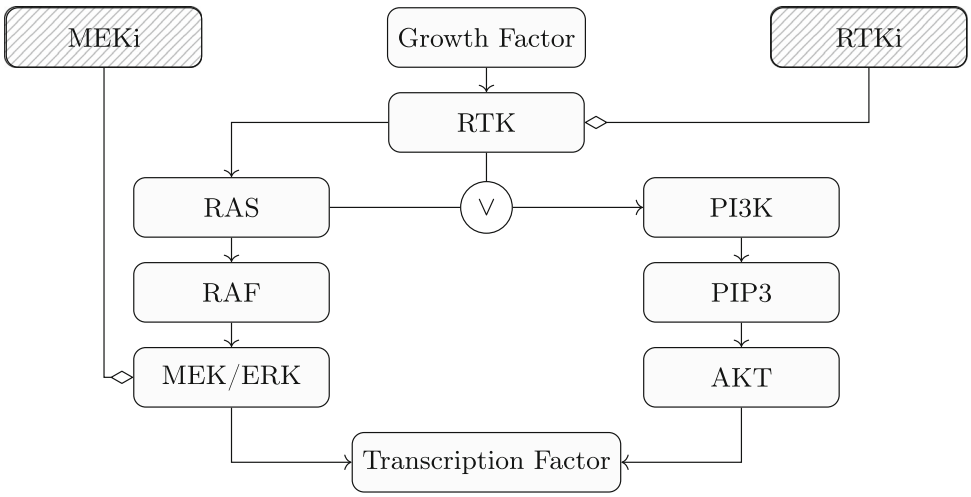


Fig. 2 The network model inspired by Zañudo et al. (2017). In the network model two or more inputs can influence one node being combined via logical AND or logical OR operator. We treat AND operator as default, putting OR operator implicitly

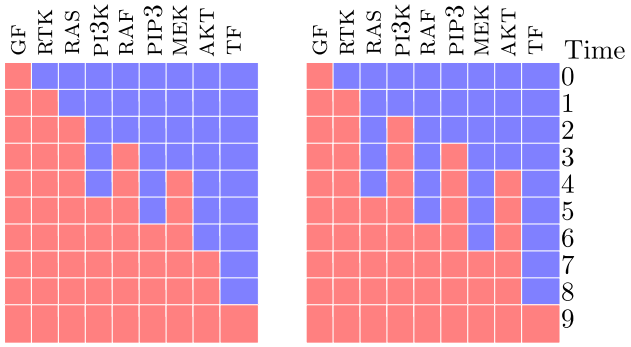


Fig. 3 Two possible trajectories taken from Zañudo et al. (2017). Rows represent quanta of time, while columns represent states of the considered signals, where a red box depicts the ON state and a blue box depicts the OFF state

It is quite natural to treat the activated elements as products, the positive state variables as reactants, and the negative state variables as inhibitors. This way one can use logical formulas in disjunctive normal form to identify each logical rule with a set of reactions. Note that all presented logical rules were mapped to reactions from either A_1 or A_2 defined in Examples 1 and 2. Moreover, the presented evolution of the system from Example 2 can be identified with the one presented in the leftmost diagram of Fig. 1b of Zañudo et al. (2017) (reprinted as Fig. 3(left)) as one of possible effects of the application of the above set of logical rules.

Recall that $\mathbf{C}_0^c = (\{\text{GF}\}, \{\text{GF}\})$ is the initial set of entities provided by the environment and, for any $i > 0$, we have $\mathbf{C}_i^c = \overline{\varnothing}^2$. Moreover, the sequence of active agents is

$(\{1, 2\}, \{1\}, \{1\}, \{1\}, \{1, 2\}, \{2\}, \{2\}, \varnothing, \{1, 2\}, \varnothing)$.

We obtain the same result taking $\mathbf{C}_0^c = (\{\text{GF}\}, \varnothing)$ and the sequence of active agents

$(\{1, 2\}, \{1\}, \{1\}, \{1\}, \{2\}, \{2\}, \{2\}, \varnothing, \{1, 2\}, \varnothing)$.

However, if we leave $\mathbf{C}_0^c = (\{\text{GF}\}, \{\text{GF}\})$ and change the sequence of active agents to

$(\{1, 2\}, \{2\}, \{2\}, \{2\}, \{1\}, \{1\}, \{1\}, \varnothing, \{1, 2\}, \varnothing)$

then the obtained sequence of resulting states could be indentified with the middle diagram of Fig. 1b of Zañudo et al. (2017) (reprinted as Fig. 3(right)).

To complete the example we can obtain the sequence of resulting states identified with the rightmost diagram of Fig. 1b of Zañudo et al. (2017) by taking the sequence of active agents

$(\{1\}, \{1\}, \{2\}, \{2\}, \{1\}, \{1\}, \{2\}, \varnothing, \{1, 2\}, \varnothing)$.

Example 4 We now present a parameterised variant of the distributed reaction system $\mathcal{D}[x, y, z] = (S_{x,y}, \{A_1, \ldots, A_y\})$ for $x, y, z \geq 2$ and $y \geq z$, inspired by Examples 1, 2 and 3, as follows:

- $S_{x,y} = \{\text{GF}, \text{RTK}, \text{RTK}i, \text{TF}, \star, \text{EN}_1^1, \ldots, \text{EN}_y^x, \text{EN}i_1^1, \ldots, \text{EN}i_y^x\}$;
- $A_j = \{(\{\text{GF}\}, \{\star\}, \{\text{GF}\}), (\{\text{GF}\}, \{\star, \text{RTK}i\}, \{\text{RTK}\}), (\{\text{RTK}\}, \{\text{EN}i_j^1\}, \{\text{EN}_j^1\}), (\{\text{EN}_j^1\}, \{\text{EN}i_j^2\}, \{\text{EN}_j^2\}), \ldots, (\{\text{EN}_j^{x-1}\}, \{\text{EN}i_j^x\}, \{\text{EN}_j^x\})\}$

  $\cup \bigcup_{\{a_1,\ldots,a_z\}\subseteq\{1,\ldots,y\}} \{(\{\text{EN}_{a_1}^x, \ldots, \text{EN}_{a_z}^x\}, \{\star\}, \{\text{TF}\})\}$.

This way we introduce a system consisting of y pathways of length $x+3$ each, where any z signalling pathways need to simultaneously activate the last signal EN_i^x in order to jointly activate transcription factor. Note that the i-th pathway has x unique reactants EN_i^j and three common reactants GF, RTK and TF together with inhibitor $\text{RTK}i$ as well as dummy inhibitor $\star$. Moreover, we have a specified inhibitor $\text{EN}i_i^j$ that blocks the propagation of the signal EN_i^j. Each pathway ends with a set of reactions which activate Transcription factor TF.

We can specify a system almost equivalent to our running example (without creation of PI3K on a base of RAS and with many additional inhibitors) as $\mathcal{D}[3, 2, 2]$.

For further examples, we also introduce an additional agent with the set of reactions $A_0^{[x,y,z,B]}$, where B is a set of reactants of the shape EN_i^j, that is responsible for the

treatment application: $A_0^{[x,y,z,B]} = \{(\{\text{RTK}\}, \{\star\}, \{\text{RTK}i\}), (\{\text{EN}_i^j\}, \{\star\}, \{\text{EN}i_i^x\})\}$ for all $\text{EN}_i^j \in B$. Intuitively, agent 0 reacts to the activation of receptor tyrosine kinase and several other signals related to independent pathways introducing the appropriate inhibitors into the model.

4 Logic for temporal-epistemic properties

Recall that defining a context automaton we have already introduced the satisfaction relation $\models_{sc}$.

The language of *computation tree logic of knowledge for reaction systems*, rsCTLK for short, is defined by the following grammar: $\phi := i.ent \mid \neg\phi \mid \phi \vee \phi \mid \mathsf{E}_{\mathfrak{sc}}\mathsf{X}\phi \mid \mathsf{E}_{\mathfrak{sc}}\mathsf{G}\phi \mid \mathsf{E}_{\mathfrak{sc}}[\phi\mathsf{U}\phi] \mid \overline{\mathrm{K}}_i\phi \mid \overline{\mathrm{C}}_\Gamma\phi$, where $i \in \mathcal{A}$, $ent \in S$, $\mathfrak{sc} \in \mathcal{SC}_\mathcal{D}$ and $\Gamma \subseteq \mathcal{A}$.

The aim of the logic is to resemble CTLK (Penczek and Lomuscio 2003), while retaining the expressive power of rsCTL. The grammar uses the propositional and temporal operators of rsCTL (Męski et al. 2015). In place of propositional variables we use $i.ent$, which allows for specifying entities in local states of the individual agents. Additionally, the logic uses epistemic operators for specifying knowledge properties: the operators $\overline{\mathrm{K}}_i\phi$ and $\overline{\mathrm{C}}_\Gamma\phi$ are the existential counterparts of the universal $\mathrm{K}_i\phi$ and $\mathrm{C}_\Gamma\phi$, respectively, which we derive later. Here we provide the intuitive meaning of the universal operators: $\mathrm{K}_i\phi$ means the agent $i \in \mathcal{A}$ *knows* ϕ and $\mathrm{C}_\Gamma\phi$ means ϕ is *common knowledge* amongst the agents of Γ.

In contrast to rsCTL (Męski et al. 2015), to restrict the scope of the path quantifiers in this logic we use state constraints. One advantage of such an approach is the ability to obtain compact representations for families of sets of entities (sets of the allowed actions) under path quantifiers. Let ϕ be an rsCTLK formula. Then, $\mathsf{d}(\phi)$ is the *depth of* ϕ defined recursively as follows:

- if $\phi = i.ent$, where $i \in \mathcal{A}$ and $ent \in S$, then $\mathsf{d}(\phi) = 1$,
- if $\phi \in \{\neg\phi', \mathsf{E}_{\mathfrak{sc}}\mathsf{X}\phi', \mathsf{E}_{\mathfrak{sc}}\mathsf{G}\phi', \overline{\mathrm{K}}_i\phi', \overline{\mathrm{C}}_\Gamma\phi'\}$, then $\mathsf{d}(\phi) = \mathsf{d}(\phi') + 1$,
- if $\phi \in \{\phi' \vee \phi'', \mathsf{E}_{\mathfrak{sc}}[\phi'\mathsf{U}\phi'']\}$, then $\mathsf{d}(\phi) = max(\{\mathsf{d}(\phi'), \mathsf{d}(\phi'')\}) + 1$.

Let CR-$\mathcal{D} = (\mathcal{D}, \mathfrak{E})$ where $\mathcal{D} = (S, \{A_i\}_{i\in\mathcal{A}})$ is a DRS and $\mathfrak{E} = (\mathcal{Q}, \mathfrak{q}^{init}, R)$ is an extended context automaton over $\mathcal{D}$. The *model for* CR-$\mathcal{D}$ is a tuple $\mathcal{M}_{\text{CR-}\mathcal{D}} = (St_{\text{CR-}\mathcal{D}}, \mathcal{S}_{init}, \longrightarrow)$, where:

1. $St_{\text{CR-}\mathcal{D}}$ is the set of states of $\mathcal{M}_{\text{CR-}\mathcal{D}}$,
2. $\mathcal{S}_{init} = \big((\varnothing, \ldots, \varnothing), \mathfrak{q}^{init}\big) \in St_{\text{CR-}\mathcal{D}}$ is the initial state,
3. $\longrightarrow \subseteq St_{\text{CR-}\mathcal{D}} \times Ct_\mathcal{D} \times St_{\text{CR-}\mathcal{D}}$ is the transition relation such that for all $\mathbf{S}, \mathbf{S}' \in St_\mathcal{D}$, $\mathfrak{q}, \mathfrak{q}' \in \mathcal{Q}$, $\mathbf{C} \in Ct_\mathcal{D}$: $\big((\mathbf{S}, \mathfrak{q}), \mathbf{C}, (\mathbf{S}', \mathfrak{q}')\big) \in \longrightarrow$ iff
   (a) $(\mathfrak{q}, \mathfrak{sc}, \mathbf{C}, \mathfrak{q}') \in R$ for some $\mathfrak{sc} \in \mathcal{SC}_\mathcal{D}$ and $\mathbf{S} \models_{sc} \mathfrak{sc}$,
   (b) for each $k \in \mathcal{A}$:
   - $\mathbf{S}'[k] = res_{A_k}(\mathbf{C}^c[k] \cup \bigcup_{j\in\mathbf{C}^a} \mathbf{S}[j])$ if $k \in \mathbf{C}^a$ and
   - $\mathbf{S}'[k] = \mathbf{S}[k]$ if $k \notin \mathbf{C}^a$.

Each element $(\mathcal{S}, \mathbf{C}, \mathcal{S}') \in \longrightarrow$ is denoted by $\mathcal{S} \xrightarrow{\mathbf{C}} \mathcal{S}'$.

The following lemma follows from the above definition and seriality of the transition relation of $\mathfrak{E}$.

Lemma 2 *For each $\mathcal{S} \in St_{\text{CR-}\mathcal{D}}$ there exist $\mathbf{C} \in Ct_\mathcal{D}$ and $\mathcal{S}' \in St_{\text{CR-}\mathcal{D}}$ such that $\mathcal{S} \xrightarrow{C} \mathcal{S}'$.*

A *path* over $\mathfrak{sc} \in \mathcal{SC}_\mathcal{D}$ in $\mathcal{M}_{\text{CR-}\mathcal{D}}$ is an infinite sequence $\sigma = (\mathcal{S}_0, \mathbf{C}_0, \mathcal{S}_1, \mathbf{C}_1, \ldots)$ s.t. $\mathcal{S}_i \xrightarrow{\mathbf{C}_i} \mathcal{S}_{i+1}$ and $\mathbf{C}_i^c \models_{sc} \mathfrak{sc}$ for each $i \geq 0$. The set of all paths over $\mathfrak{sc}$ is denoted by $\Pi_{\mathfrak{sc}}^{\mathbf{inf}}$. For each $i \geq 0$, the i^{th} state of the path σ is denoted by $\sigma_s(i)$ and the i^{th} action (i.e. a step of reaction sytem) of the path σ is denoted by $\sigma_a(i)$. By $\Pi_{\mathfrak{sc}}^{\mathbf{inf}}(\mathcal{S})$ we denote the set of all paths over $\mathfrak{sc}$ that start in $\mathcal{S} \in St_{\text{CR-}\mathcal{D}}$, i.e., $\Pi_{\mathfrak{sc}}^{\mathbf{inf}}(\mathcal{S}) = \{\sigma \in \Pi_{\mathfrak{sc}}^{\mathbf{inf}} \mid \sigma_s(0) = \mathcal{S}\}$.

Let $\mathcal{S}, \mathcal{S}' \in St_{\text{CR-}\mathcal{D}}$ and $\mathfrak{sc} \in \mathcal{SC}_\mathcal{D}$. We say that $\mathcal{S}'$ is an $\mathfrak{sc}$-successor of $\mathcal{S}$ (denoted by $\mathcal{S} \longrightarrow_{\mathfrak{sc}} \mathcal{S}'$) iff there exists $\mathbf{C} \in Ct_\mathcal{D}$ such that $\mathbf{C}^c \models_{sc} \mathfrak{sc}$ and $\mathcal{S} \xrightarrow{\mathbf{C}} \mathcal{S}'$. The relation $\twoheadrightarrow \subseteq St_{\text{CR-}\mathcal{D}} \times St_{\text{CR-}\mathcal{D}}$ is defined as follows: $(\mathcal{S}, \mathcal{S}') \in \twoheadrightarrow$ iff there exists $\mathbf{C} \in Ct_\mathcal{D}$ such that $\mathcal{S} \xrightarrow{\mathbf{C}} \mathcal{S}'$. The relation $\twoheadrightarrow_r \subseteq St_{\text{CR-}\mathcal{D}} \times St_{\text{CR-}\mathcal{D}}$ is the transitive closure of $\twoheadrightarrow$. A state $\mathcal{S} \in St_{\text{CR-}\mathcal{D}}$ is *reachable* iff $\mathcal{S}_{init} \twoheadrightarrow_r \mathcal{S}$. Then, $Reach(\text{CR-}\mathcal{D}) \subseteq St_{\text{CR-}\mathcal{D}}$ is the set of the reachable states of $\mathcal{M}_{\text{CR-}\mathcal{D}}$, i.e., $Reach(\text{CR-}\mathcal{D}) = \{\mathcal{S} \in St_{\text{CR-}\mathcal{D}} \mid \mathcal{S}_{init} \twoheadrightarrow_r \mathcal{S}\}$. To denote $Reach(\text{CR-}\mathcal{D})$ we also write $Reach(\mathcal{M}_{\text{CR-}\mathcal{D}})$.

For each agent $i \in \mathcal{A}$, the *epistemic indistinguishability* relation $\sim_i \subseteq St_{\text{CR-}\mathcal{D}} \times St_{\text{CR-}\mathcal{D}}$ is defined by: $\mathcal{S} \sim_i \mathcal{S}'$ iff $\mathcal{D}(\mathcal{S})[i] = \mathcal{D}(\mathcal{S}')[i]$ and $\mathcal{S}, \mathcal{S}' \in Reach(\text{CR-}\mathcal{D})$. For a group of agents $\Gamma \subset \mathcal{A}$, the union of the indistinguishability relations of Γ is defined as $\sim_\Gamma^{\mathrm{E}} = \bigcup_{i\in\Gamma} \sim_i$. By $\sim_\Gamma^{\mathrm{C}}$ we denote the transitive closure of $\sim_\Gamma^{\mathrm{E}}$. If $\mathcal{S} \sim_i \mathcal{S}'$ for some $i \in \Gamma$, we say $\mathcal{S}$ is a Γ-neighbour, or an i-neighbour, of $\mathcal{S}'$ (denoting them also as nb_Γ and $\mathsf{nb}_{\{i\}}$).

Let $\mathcal{M}_{\text{CR-}\mathcal{D}} = (St_{\text{CR-}\mathcal{D}}, \mathcal{S}_{init}, \longrightarrow)$ be a model for CR-$\mathcal{D}$, $\mathcal{S} \in St_{\text{CR-}\mathcal{D}}$ be a state of $\mathcal{M}_{\text{CR-}\mathcal{D}}$, and ϕ be an rsCTLK formula.

The fact that ϕ holds in $\mathcal{S}$ is denoted by $\mathcal{M}_{\text{CR-}\mathcal{D}}, \mathcal{S} \models \phi$, or simply $\mathcal{S} \models \phi$ when $\mathcal{M}_{\text{CR-}\mathcal{D}}$ is clear from the context.

The relation $\models$ is defined recursively as follows:

- $\mathcal{S} \models i.ent$ iff $ent \in \mathcal{D}(\mathcal{S})[i]$,
- $\mathcal{S} \models \neg\phi$ iff $\mathcal{S} \not\models \phi$,
- $\mathcal{S} \models \phi \vee \psi$ iff $\mathcal{S} \models \phi$ or $\mathcal{S} \models \psi$,
- $\mathcal{S} \models \mathsf{E}_{\mathfrak{sc}}\mathsf{X}\phi$ iff $(\exists\sigma \in \Pi_{\mathfrak{sc}}^{\mathbf{inf}}(\mathcal{S}))\ \sigma_s(1) \models \phi$,
- $\mathcal{S} \models \mathsf{E}_{\mathfrak{sc}}\mathsf{G}\phi$ iff $(\exists\sigma \in \Pi_{\mathfrak{sc}}^{\mathbf{inf}}(\mathcal{S}))(\forall i \geq 0)(\sigma_s(i) \models \phi)$,
- $\mathcal{S} \models \mathsf{E}_{\mathfrak{sc}}[\phi\mathsf{U}\psi]$ iff $(\exists\sigma \in \Pi_{\mathfrak{sc}}^{\mathbf{inf}}(\mathcal{S}))(\exists i \geq 0)\big(\sigma_s(i) \models \psi$ and $(\forall 0 \leq j < i)\ \sigma_s(j) \models \phi\big)$,
- $\mathcal{S} \models \overline{\mathrm{K}}_i\phi$ iff $(\exists\mathcal{S}' \in St_{\text{CR-}\mathcal{D}})(\mathcal{S} \sim_i \mathcal{S}'$ and $\mathcal{S}' \models \phi)$

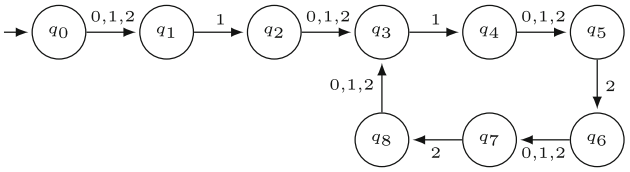


Fig. 4 Automaton $\mathfrak{E}_1$. Since we do not specify any sophisticated state constraints and in all arcs except for the first one (between q_0 and q_1) we provide empty contexts, on the labels related to the transitions we give only the sets of activated agents

- $\mathcal{S} \models \overline{\mathrm{C}}_\Gamma \phi$ iff $(\exists \mathcal{S}' \in St_{\text{CR-}\mathcal{D}})(\mathcal{S} \sim_\Gamma^{\mathrm{C}} \mathcal{S}'$ and $\mathcal{S}' \models \phi)$.

Next, we introduce some useful derived operators. First, $true \stackrel{def}{=} i.ent \vee \neg i.ent$, for any $i \in \mathcal{A}$ and $ent \in S$. Then we define:

$\phi \Rightarrow \psi \stackrel{def}{=} \neg\phi \vee \psi, \quad \mathsf{E}_{\mathfrak{sc}}\mathsf{F}\phi \stackrel{def}{=} \mathsf{E}_{\mathfrak{sc}}[\,true\mathsf{U}\phi]$.

We also introduce the universal path quantifier $\mathsf{A}_{\mathfrak{sc}}$ meaning 'for all the paths over $\mathfrak{sc}$':

$\mathsf{A}_{\mathfrak{sc}}\mathsf{F}\phi \stackrel{def}{=} \neg\mathsf{E}_{\mathfrak{sc}}\mathsf{G}\neg\phi, \quad \mathsf{A}_{\mathfrak{sc}}\mathsf{X}\phi \stackrel{def}{=} \neg\mathsf{E}_{\mathfrak{sc}}\mathsf{X}\neg\phi, \quad \mathsf{A}_{\mathfrak{sc}}\mathsf{G}\phi \stackrel{def}{=} \neg\mathsf{E}_{\mathfrak{sc}}[\,true\mathsf{U}\neg\phi]$.

Moreover, we assume $\mathfrak{sc} = true$ when $\mathfrak{sc}$ is not explicitly specified for any of the rsCTLK operators, e.g., $\mathsf{EF}\phi \stackrel{def}{=} \mathsf{E}_{true}\mathsf{F}\phi$. We also introduce derived epistemic operators:

$\mathrm{K}_i\phi \stackrel{def}{=} \neg\overline{\mathrm{K}}_i\neg\phi, \quad \mathrm{C}_\Gamma\phi \stackrel{def}{=} \neg\overline{\mathrm{C}}_\Gamma\neg\phi, \quad \overline{\mathrm{E}}_\Gamma\phi \stackrel{def}{=} \bigvee_{i\in\Gamma}\overline{\mathrm{K}}_i\phi, \quad \mathrm{E}_\Gamma\phi \stackrel{def}{=} \neg\overline{\mathrm{E}}_\Gamma\neg\phi$.

We say that an rsCTLK formula ϕ *holds in the model* $\mathcal{M}_{\text{CR-}\mathcal{D}}$ if and only if ϕ holds in the initial state of $\mathcal{M}_{\text{CR-}\mathcal{D}}$: $\mathcal{M}_{\text{CR-}\mathcal{D}} \models \phi$ iff $\mathcal{S}_{init} \models \phi$.

Example 5 Here we specify some rsCTLK properties of the biological running example presented in Examples 1–3.

Let us recall two agents 1 and 2 with sets of reactions A_1 and A_2 from Example 2, and add another agent 0 (similar to the treatment agent in Example 4) with the reaction set $A_0 = \{(\{\text{RTK}\}, \{\star\}, \{\text{RTK}i\}), (\{\text{RAF}\}, \{\star\}, \{\text{MEK}i\}), (\{\text{RAS}\}, \{\star\}, \{\text{MEK}i\})\}$. Together they form a DRS $\mathcal{D}_2 = (S', \{A_0, A_1, A_2\})$.

Let us check whether agent 0 always knows whether Transcription Factor was observed in any pathway or in neither of them. This property is specified as the rsCTLK formula $\phi = \mathsf{AG}(\mathrm{K}_0(1.\text{TF} \vee 2.\text{TF}) \vee \mathrm{K}_0(\neg 1.\text{TF} \wedge \neg 2.\text{TF}))$. Moreover, let us define three extended context automata over $\mathcal{D}$ as presented in Figs. 4, 5, 6.

Consider two processes $\pi_j = (\gamma_j, \delta_j)$, where $j \in \{1, 2\}$, in $\mathcal{D}_2$ such that $\gamma_j = (\mathbf{C}_0^j, \mathbf{C}_1^j, \mathbf{C}_2^j, \ldots)$ and $\delta_j = (\mathbf{D}_0^j, \mathbf{D}_1^j, \mathbf{D}_2^j, \ldots)$ where $\mathbf{C}_i^j$ and $\mathbf{D}_i^j$ for $i \in \{0, \ldots, 9\}$ are defined by tables presented in Fig. 7.

Notice that both processes are (at least up to this point) indistinguishable for agent 0. Moreover, π_1 and π_2 are both finite processes of $\mathcal{D}_2$ with automaton $\mathfrak{E}_3$. Hence, ϕ is not satisfied for this CRDRS. On the other hand, π_1 is a finite

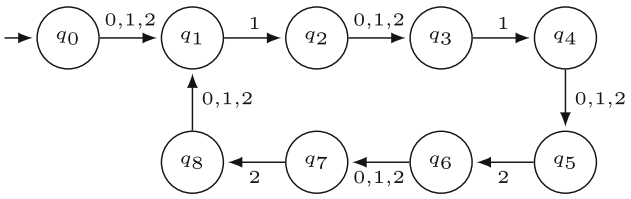


Fig. 5 Automaton $\mathfrak{E}_2$

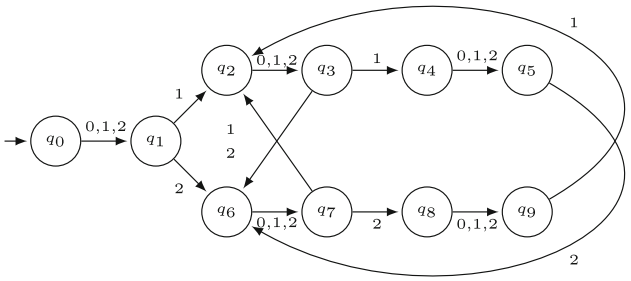


Fig. 6 Automaton $\mathfrak{E}_3$

prefix of the only infinite process of $\mathcal{D}_2$ with automaton $\mathfrak{E}_1$. After that the last six states repeat periodically. Hence, ϕ is satisfied for this CRDRS (with the first component of the internal formula true). Finally, π_2 is a finite prefix of the only infinite process of $\mathcal{D}_2$ with automaton $\mathfrak{E}_2$. For similar reasons as before, ϕ is not satisfied for this CRDRS.

Example 6 Let us recall the parametric variant of the DRS $\mathcal{D}[x, y, z]$ of Example 4 together with the medical treatment agent 0 with the set of reactions $A_0^{x,y,z,B}$. This way we can model the application of drugs (inhibitors specified by parameter B) in the situation where more then two independent signalling pathways (of equal lengths) are considered.

Basing on finite automata $\mathfrak{E}_3$ of Example 5, we consider parameterized context automata $\mathfrak{E}_4$ that start with the transition engaging all agents with context of the form $(\{\text{GF}\}\ldots\{\text{GF}\})$ and then engage alternately one non-zero agent and all agents (always with empty contexts and with a special non-zero agent chosen no more than twice in a row) (Figs. 8, 9).

$\mathfrak{E}_4 = (\{q_I, q_S\} \cup \bigcup_{i=1,\ldots,y}\{q_{iA}, q_{iB}, q_{iC}, q_{iD}\}, q_I,$
$\{(q_I, true, (\{\text{GF}\}, \{\text{GF}\}, \ldots, \{\text{GF}\}), \{0, 1, \ldots, y\}), q_S)\}$
$\cup \bigcup_{i=1,\ldots,y}\{(q_S, true, (\overline{\varnothing}^{y+1}, \{i\}), q_{iA}),$
$(q_{iA}, true, (\overline{\varnothing}^{y+1}, \{0, 1, \ldots, y\}), q_{iB}),$
$(q_{iB}, true, (\overline{\varnothing}^{y+1}, \{i\}), q_{iC}),$
$(q_{iC}, true, (\overline{\varnothing}^{y+1}, \{0, 1, \ldots, y\}), q_{iD})\}$
$\cup \bigcup_{i=1,\ldots,y}\bigcup_{j=1,\ldots,y\wedge j\neq i}\{(q_{iB}, true, (\overline{\varnothing}^{y+1}, \{j\}), q_{jA}),$
$(q_{iD}, true, (\overline{\varnothing}^{y+1}, \{j\}), q_{jA})\}$).

Moreover, let us consider some specific automata that generate single infinite paths accepted by $\mathfrak{E}_4$. As an example one can define the automaton inspired by $\mathfrak{E}_2$.

$\mathfrak{E}_5 = (\{q_0, q_1, \ldots, q_{4y}\}, q_0,$

Fig. 7 We present prefixes of two interactive processes π_1 and π_2. Since γ_1 is almost equal to γ_2, we present them in the common, second column, proving two corresponding values in the rows 11 and 15 that distinguish them

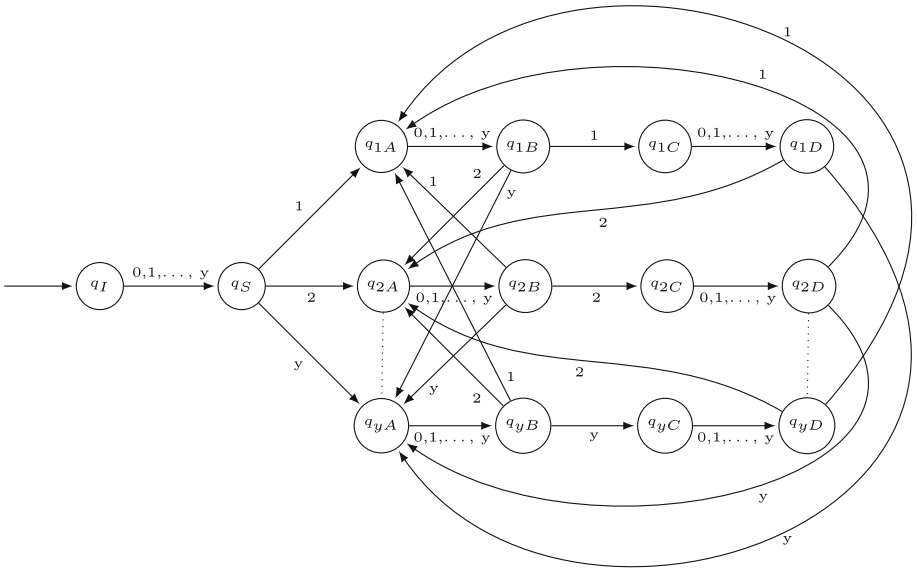
i	$\mathbf{C}_i^1$ / $\mathbf{C}_i^2$	$\mathbf{D}_i^1$	$\mathbf{D}_i^2$
0	$(\{\text{GF}\}^3, \{0,1,2\})$	$\overline{\varnothing}^3$	$\overline{\varnothing}^3$
1	$(\overline{\varnothing}^3, \{1\})$	$(\varnothing, \{\text{GF}, \text{RTK}\}, \{\text{GF}, \text{RTK}\})$	$(\varnothing, \{\text{GF}, \text{RTK}\}, \{\text{GF}, \text{RTK}\})$
2	$(\overline{\varnothing}^3, \{0,1,2\})$	$(\varnothing, \{\text{GF}, \text{RTK}, \text{RAS}\}, \{\text{GF}, \text{RTK}\})$	$(\varnothing, \{\text{GF}, \text{RTK}, \text{RAS}\}, \{\text{GF}, \text{RTK}\})$
3	$(\overline{\varnothing}^3, \{1\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}\}, \{\text{GF}, \text{RTK}, \text{PI3K}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}\}, \{\text{GF}, \text{RTK}, \text{PI3K}\})$
4	$(\overline{\varnothing}^3, \{0,1,2\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}, \text{MEK}\}, \{\text{GF}, \text{RTK}, \text{PI3K}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RTK}, \text{RAS}, \text{RAF}, \text{MEK}\}, \{\text{GF}, \text{RTK}, \text{PI3K}\})$
5	$(\overline{\varnothing}^3, \{2\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAS}, \text{RAF}\}, \{\text{GF}, \text{PI3K}, \text{PIP3}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAS}, \text{RAF}\}, \{\text{GF}, \text{PI3K}, \text{PIP3}\})$
6	$(\overline{\varnothing}^3, \{0,1,2\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAS}, \text{RAF}\}, \{\text{GF}, \text{RTK}, \text{PIP3}, \text{AKT}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAS}, \text{RAF}\}, \{\text{GF}, \text{RTK}, \text{PIP3}, \text{AKT}\})$
7	$(\overline{\varnothing}^3, \{2\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAF}\}, \{\text{GF}, \text{RTK}, \text{PI3K}, \text{AKT}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAF}\}, \{\text{GF}, \text{RTK}, \text{PI3K}, \text{AKT}\})$
8	$(\overline{\varnothing}^3, \{0,1,2\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAF}\}, \{\text{GF}, \text{RTK}, \text{PIP3}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAF}\}, \{\text{GF}, \text{RTK}, \text{PIP3}\})$
9	$(\overline{\varnothing}^3, \{1\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}\}, \{\text{GF}, \text{PI3K}, \text{AKT}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}\}, \{\text{GF}, \text{PI3K}, \text{AKT}\})$
10	$(\overline{\varnothing}^3, \{0,1,2\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RTK}\}, \{\text{GF}, \text{PI3K}, \text{AKT}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RTK}\}, \{\text{GF}, \text{PI3K}, \text{AKT}\})$
11	$(\overline{\varnothing}^3, \{2\})$ $(\overline{\varnothing}^3, \{1\})$	$(\{\text{RTK}i\}, \{\text{GF}, \text{RAS}\}, \{\text{GF}, \text{PIP3}\})$	$(\{\text{RTK}i\}, \{\text{GF}, \text{RAS}\}, \{\text{GF}, \text{PIP3}\})$
12	$(\overline{\varnothing}^3, \{0,1,2\})$	$(\{\text{RTK}i\}, \{\text{GF}, \text{RAS}\}, \{\text{GF}, \text{RTK}, \text{AKT}\})$	$(\{\text{RTK}i\}, \{\text{GF}, \text{RTK}, \text{RAF}\}, \{\text{GF}, \text{PIP3}\})$
13	$(\overline{\varnothing}^3, \{2\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAF}\}, \{\text{GF}, \text{PI3K}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAS}, \text{MEK}\}, \{\text{GF}, \text{PIP3}\})$
14	$(\overline{\varnothing}^3, \{0,1,2\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAF}\}, \{\text{GF}, \text{RTK}, \text{PIP3}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAS}, \text{MEK}\}, \{\text{GF}, \text{RTK}, \text{AKT}\})$
15	$(\overline{\varnothing}^3, \{1\})$ $(\overline{\varnothing}^3, \{2\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}\}, \{\text{GF}, \text{PI3K}, \text{AKT}\})$	$(\{\text{RTK}i, \text{MEK}i\}, \{\text{GF}, \text{RAF}, \text{TF}\}, \{\text{GF}, \text{PI3K}, \text{TF}\})$

$\{(q_0, true, (\{\text{GF}\}, \{\text{GF}\}, \ldots, \{\text{GF}\}), \{0,1,\ldots,y\}), q_1)\}$
$\cup \bigcup_{i=0,\ldots,y-1} \{(q_{4i+1}, true, (\overline{\varnothing}^{y+1}, \{i+1\}), q_{4i+2}),$
$(q_{4i+2}, true, (\overline{\varnothing}^{y+1}, \{0,1,\ldots,y\}), q_{4i+3}),$
$(q_{4i+3}, true, (\overline{\varnothing}^{y+1}, \{i+1\}), q_{4i+4}),$
$\cup \bigcup_{i=0,\ldots,y-2} \{(q_{4i+4}, true, (\overline{\varnothing}^{y+1}, \{0,1,\ldots,y\}), q_{4i+5})\}$
$\cup \{(q_{4y}, true, (\overline{\varnothing}^{y+1}, \{0,1,\ldots,y\}), q_1)\}$).

Finally, for the discussed parametric distributed reaction system with both variants of context automaton we can verify the following sample logical formula:

$$\phi = \mathsf{AG}(\mathrm{K}_0(\bigvee_{i=1}^{y} i.\text{TF}) \vee \mathrm{K}_0(\bigwedge_{i=1}^{y} \neg i.\text{TF})). \tag{1}$$

Fig. 8 Automaton $\mathfrak{E}_4$



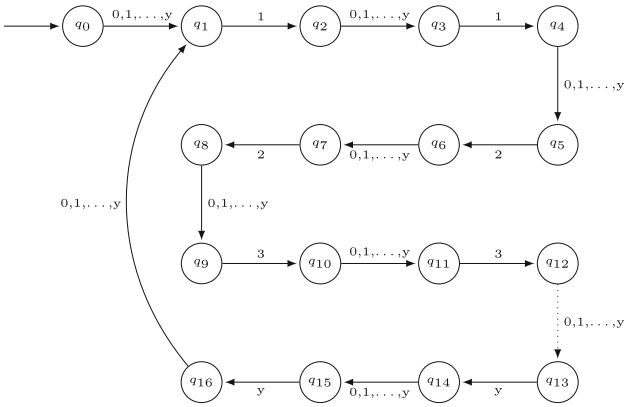


Fig. 9 Automaton $\mathfrak{E}_5$

Similarily to the formula from Example 5, we are asking if the treatment agent 0 always knows whether the proposed therapy is effective or not.

5 Model checking for rsCTLK

In this section we describe a model checking method for rsCTLK, which is based on computing fixed points. To calculate the set of the reachable states and the results for temporal operators we use algorithms similar to the ones presented in Męski et al. (2015). The difference is in how the set of successor and predecessor states are defined. In this section we restrict the sets of states to only those which are obtained via transitions restricted with state constraints, whereas in Męski et al. (2015) sets of actions were used. Firstly, we give an algorithm for computing all reachable states of $\mathcal{M}_{\text{CR-}\mathcal{D}}$, and then we provide a method for computing the set of states, where an rsCTLK formula holds.

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

Let $\mathcal{M}_{\text{CR-}\mathcal{D}} = (St_{\text{CR-}\mathcal{D}}, \mathcal{S}_{init}, \longrightarrow)$ be a model. We define the function that assigns the set of the $\mathfrak{sc}$-successors to the states in $W \subseteq St_{\text{CR-}\mathcal{D}}$: $\mathsf{post}_{\mathfrak{sc}}(W) = \{\mathcal{S}' \in St_{\text{CR-}\mathcal{D}} \mid (\exists \mathcal{S} \in W)\ \mathcal{S} \longrightarrow_{\mathfrak{sc}} \mathcal{S}')\}$, where $\mathfrak{sc} \in \mathcal{SC}_{\mathcal{D}}$. The set $Reach(\text{CR-}\mathcal{D}) \subseteq \mathcal{S}$ can be characterised by the fixed point equation: $Reach(\text{CR-}\mathcal{D}) = \mu X.\big(\mathcal{S}_{init} \cup X \cup \mathsf{post}_{true}(X)\big)$.

Algorithm 1: checkNC(Γ, ϕ_1)

```
X := ∅;
X_p := Reach(CR-D);
Y_φ1 = check_rsCTLK(φ1);
while X ≠ X_p do
    X_p := X, X := nb_Γ(Y_φ1 ∪ X);
end
return X;
```

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

$\mathsf{pre}^{\exists}_{\mathfrak{sc}}(W) \stackrel{def}{=} \{\mathcal{S} \in Reach(\text{CR-}\mathcal{D}) \mid (\exists \mathcal{S}' \in W)\ \mathcal{S} \longrightarrow_{\mathfrak{sc}} \mathcal{S}')\}$.

Algorithm 2: $\mathsf{check}_{rs\text{CTLK}}(\phi)$

```
if φ = i.ent then return
  {S ∈ St_CR-D | ent ∈ D(S)[i]} ∩ Reach(CR-D) else if φ = ¬φ1
  then return Reach(CR-D) \ check_rsCTLK(φ1) else if
  φ = φ1 ∨ φ2 then return check_rsCTLK(φ1) ∪ check_rsCTLK(φ2)
  else if φ = E_sc X φ1 then return pre^∃_sc(check_rsCTLK(φ1)) else if
  φ = E_sc G φ1 then return checkEG(sc, φ1) else if
  φ = E_sc[φ1 U φ2] then return checkEU(sc, φ1, φ2) else if
  φ = K̄_i φ1 then
    X := check_rsCTLK(φ1);
    return nb_{i}(X);
end
else if φ = C̄_Γ φ1 then return checkNC(Γ, φ1)
```

Let ϕ_1, ϕ_2 be rsCTLK formulae. For the non-epistemic formulae of rsCTLK the sets of states in which they hold are defined as follows:

$[\![\neg\phi_1]\!] \stackrel{def}{=} Reach(\text{CR-}\mathcal{D}) \setminus [\![\phi_1]\!]$, $[\![\phi_1 \vee \phi_2]\!] \stackrel{def}{=} [\![\phi_1]\!] \cup [\![\phi_2]\!]$, $[\![\phi_1 \wedge \phi_2]\!] \stackrel{def}{=} [\![\phi_1]\!] \cap [\![\phi_2]\!]$,

$[\![\mathsf{E}_{\mathfrak{sc}}\mathsf{X}\phi_1]\!] \stackrel{def}{=} \mathsf{pre}^{\exists}_{\mathfrak{sc}}([\![\phi_1]\!])$, $[\![\mathsf{E}_{\mathfrak{sc}}\mathsf{G}\phi_1]\!] \stackrel{def}{=} \nu X.\big([\![\phi_1]\!] \cap \mathsf{pre}^{\exists}_{\mathfrak{sc}}(X)\big)$,

$[\![\mathsf{E}_{\mathfrak{sc}}[\phi_1\mathsf{U}\phi_2]]\!] \stackrel{def}{=} \mu X.\big([\![\phi_2]\!] \cup ([\![\phi_1]\!] \cap \mathsf{pre}^{\exists}_{\mathfrak{sc}}(X))\big)$.

The above definitions differ from the ones presented in Męski et al. (2015) in how the predecessors are defined – here we restrict the set of predecessors with state constraints.

Recall that we have already defined *neighbours* and *epistemic indistinguishability* relation.

The sets for the epistemic operators (Fagin et al. 2003) are defined as follows:

$[\![\overline{\mathsf{K}}_i\phi_1]\!] \stackrel{def}{=} \mathsf{nb}_{\{i\}}([\![\phi_1]\!])$, $[\![\overline{\mathsf{C}}_\Gamma\phi_1]\!] \stackrel{def}{=} \mu X.\big([\![\phi_1]\!] \cup \mathsf{nb}_\Gamma(X)\big)$.

To compute the set of states for $\overline{\mathsf{K}}_i\phi_1$ we find all the i-neighbours of the states in which ϕ_1 holds. In the case of $\overline{\mathsf{C}}_\Gamma\phi_1$, to obtain the set of states in which the formula holds, we calculate the least fixed point. The procedure $\mathsf{check}_{rs\text{CTLK}(\phi)}$ for computing the set of states in which an rsCTLK formula ϕ holds is outlined in Algorithm 2. It uses the procedure for calculating $[\![\overline{\mathsf{C}}_\Gamma\phi_1]\!]$ presented in Algorithm 1.

Given CR-$\mathcal{D}$ and an rsCTLK formula ϕ, the *rs*CTLK *model checking* problem is the problem of deciding whether $\mathcal{M}_{\text{CR-}\mathcal{D}} \models \phi$. See Męski et al. (2015) for the pseudo-code for calculating $[\![\mathsf{E}_{\mathfrak{sc}}\mathsf{G}\phi_1]\!]$ and $[\![\mathsf{E}_{\mathfrak{sc}}[\phi_1\mathsf{U}\phi_2]]\!]$ ($\mathsf{checkEG}(\mathfrak{sc}, \phi_1)$ and $\mathsf{checkEU}(\mathfrak{sc}, \phi_1, \phi_2)$, respectively).

Lemma 3 *The rs*CTLK *model checking problem is* PSPACE*-hard.*

Proof Follows from the fact that QSAT can be reduced to the rsCTLK model checking problem. It is easy to see that every initialised context restricted reaction system ICRRS can be translated into CRDRS with a single agent and rsCTL is a subset of rsCTLK. Therefore, the reduction is similar to the one for ICRRS and rsCTL (Męski et al. 2015). □

Lemma 4 *The rs*CTLK *model checking problem is in* PSPACE.

Proof We show a nondeterministic algorithm for deciding whether $\mathcal{M}_{\text{CR-}\mathcal{D}} \models \phi$, which requires at most polynomial space in the size of the input, i.e., the formula ϕ and CR-$\mathcal{D}$.

The proof for the operators common with rsCTL is similar to the one of Męski et al. (2015) for rsCTL and initialised context-restricted reaction systems.

The algorithm uses the recursive procedure $label(\mathcal{S}, \phi)$, which returns *true* iff $\mathcal{M}_{\text{CR-}\mathcal{D}}, \mathcal{S} \models \phi$, where $\mathcal{S} \subseteq St_{\text{CR-}\mathcal{D}}$; otherwise, it returns *false*.

The encoding of each state requires space $\mathcal{O}(|S| \cdot |\mathcal{A}|)$ and each successor can be generated in space $\mathcal{O}(|S| \cdot |\mathcal{A}|)$, whereas the overall algorithm requires space $\mathcal{O}\big(|S| \cdot |\mathcal{A}| \cdot \mathsf{d}(\phi)\big)$.

The proof follows by the induction on the length of the formula ϕ. The cases in which ϕ does not contain any temporal and epistemic operators, or $\phi = \mathsf{E}_{\mathfrak{sc}}\mathsf{X}\phi_1$, are straightforward.

We omit the part of the proof for the temporal operators since it is similar to the one for rsCTL (Męski et al. 2015).

The intuition for the procedure for checking $\phi = \overline{\mathsf{C}}_\Gamma\phi_1$ in $\mathcal{S} \in St_{\text{CR-}\mathcal{D}}$ is outlined in Algorithm 3.

The algorithm searches for a path via the relation $\sim^{\mathrm{C}}_{\Gamma}$ from $\mathcal{S}$ to a state in which ϕ_1 holds and it is a reachable state of the model. Initially, $\widehat{\mathcal{S}}$ is set to $\mathcal{S}$. If ϕ_1 holds in $\widehat{\mathcal{S}}$ and the state is reachable, then the algorithm returns *true*; otherwise it nondeterministically selects $\widehat{\mathcal{S}}' \in St_{\text{CR-}\mathcal{D}}$ accessible via $\sim_i$ for $i \in \Gamma$ and then it jumps to the beginning of the procedure and checks if ϕ_1 holds in that state.

Algorithm 3: Nondeterministic procedure for checking $\overline{\mathsf{C}}_\Gamma\phi_1$

```
Ŝ := S;
checking:;
if label(Ŝ, φ1) and label(S_init, EF φ_Ŝ) then return true ;
guessing:;
guess Ŝ' ∈ St_CR-D and i ∈ Γ;
if ¬(Ŝ ∼_i Ŝ') then goto guessing ;
Ŝ := Ŝ';
goto checking;
```

We do not explicitly refer to the context automaton of CR-$\mathcal{D}$ in the algorithms as it does not affect the complexity considerations. However, when selecting a CRDRS state $\widehat{\mathcal{S}}$, in fact, a state of $\mathcal{D}$ and a location of $\mathfrak{E}$ are selected.

To verify if the rsCTLK formula ϕ holds in the model $\mathcal{M}_{\text{CR-}\mathcal{D}}$, the *label* procedure is called for the initial state of the model, i.e., $\mathcal{M}_{\text{CR-}\mathcal{D}} \models \phi$ iff $label(\mathcal{S}_{init}, \phi)$.

The procedure is called recursively for each subformula of ϕ. At a given recursion level the procedure requires only a constant number of variables to be stored. The total space

requirement depends on $\mathcal{O}(\mathsf{d}(\phi))$ calls of *label*, where a single call needs space $\mathcal{O}(|S| \cdot |\mathcal{A}|)$. The space requirement for *label* is not affected by the size of $\mathfrak{sc}$ as it is only used in nondeterministic choices. For each call of *label*, i.e., for each nesting level of ϕ, *label* is called recursively at most twice, as each operator of rsCTLK has at most two arguments.

Thus, the overall space requirement of the procedure is $\mathcal{O}\big(|S| \cdot |\mathcal{A}| \cdot \mathsf{d}(\phi)\big)$.

Therefore, by Savitch's theorem, the deterministic algorithm can be implemented in polynomial space. □

The following theorem follows directly from Lemmas 3 and 4.

Theorem 1 *he rs*CTLK *model checking problem is* PSPACE*-complete.*

6 Model checking toolkit

ReactICS is a comprehensive toolkit for the modelling and verification of reaction systems. It consists of two modules: one based on binary decision diagrams (BDDs) for compact storage and efficient state-space manipulation, and another that reduces verification problems to *satisfiability modulo theories* (SMT). Moreover, ReactICS supports symbolic model checking for rsCTLK, enabling the formal verification of systems in which both temporal and epistemic properties are essential aspects of the modelled behaviour.

6.1 Translation to ISPL

We validated the effectiveness of ReactICS by comparing it with another model checking tool. To this end, we simulated a distributed reaction system with a context automaton and verified rsCTL formulae using MCMAS.

MCMAS (Lomuscio et al. 2017) is an open-source model checker for the verification of multi-agent systems. It supports symbolic techniques for verifying such systems against specifications expressing temporal, epistemic, and strategic properties. Due to differing operational paradigms, simulating the behaviour of distributed reaction systems required translating the DRS descriptions into the *Interpreted Systems Programming Language* (ISPL) used by MCMAS.

Due to technical restrictions on accessing the values of local variables, the translation does not fully capture the described framework. We simulated distributed reaction systems and validated rsCTL formulae. The only difference lies in the operation of the context automaton, which was left unconstrained; specifically, we universally applied trivial state constraints, always equal to *true*. Nevertheless, we succeeded in preserving key properties of reaction systems – most notably, the qualitative nature of non-permanent entities.

A multi-agent system described by an ISPL program consists of several independent agents and a single moderating agent called the *Environment*. The local state of each agent is represented by internal private variables and cannot be observed by other agents, including the *Environment*. Communication between agents is realised through publicly observable local actions. Moreover, the *Environment* agent may provide a set of so-called *observable variables*, which can be read by other agents.

The environment state is represented by the local private variables of the agent *Environment*. Moreover, the *Environment* is responsible for handling the context automaton, storing its current state q and the transition t used to reach q. Information about reactants and inhibitors available to agents is shared via observable variables of the agent *Environment*. Note that separate sets of observable variables are required for each agent, as some agents may receive additional entities from the context. The simulation proceeds step by step, guided by the actions of the *Environment* (see below for details).

Each agent updates its local state by copying the relevant values (activity status, available entities, etc.) from the observable variables of the *Environment*. After performing all reactions, each agent informs the *Environment* of the products it possesses (or their absence) by triggering appropriate actions in response to the *Environment*'s queries.

A single computation step of a distributed reaction system is simulated by executing the following sequence of computational steps in the ISPL program:

1. The agent *Environment* resets the system state by clearing all variables associated with entity presence and agent activity, ensuring the qualitative (non-permanent) semantics of entities.
2. Active agents are determined based on the current transition of the context automaton. Activation status is stored in observable variables, which agents use to update their local state.
3. The *Environment* issues queries to active agents regarding products produced in the previous step. Each agent responds by invoking the corresponding action, indicating whether it possesses the queried product.
4. Observable variables representing product possession are synchronised with the environment state.
5. Additional entities, if specified by the context automaton, are assigned to each active agent.
6. Each active agent updates its local possession state using the values of the corresponding observable variables.
7. Each active agent executes all enabled reactions.
8. The *Environment* selects the next transition of the context automaton.

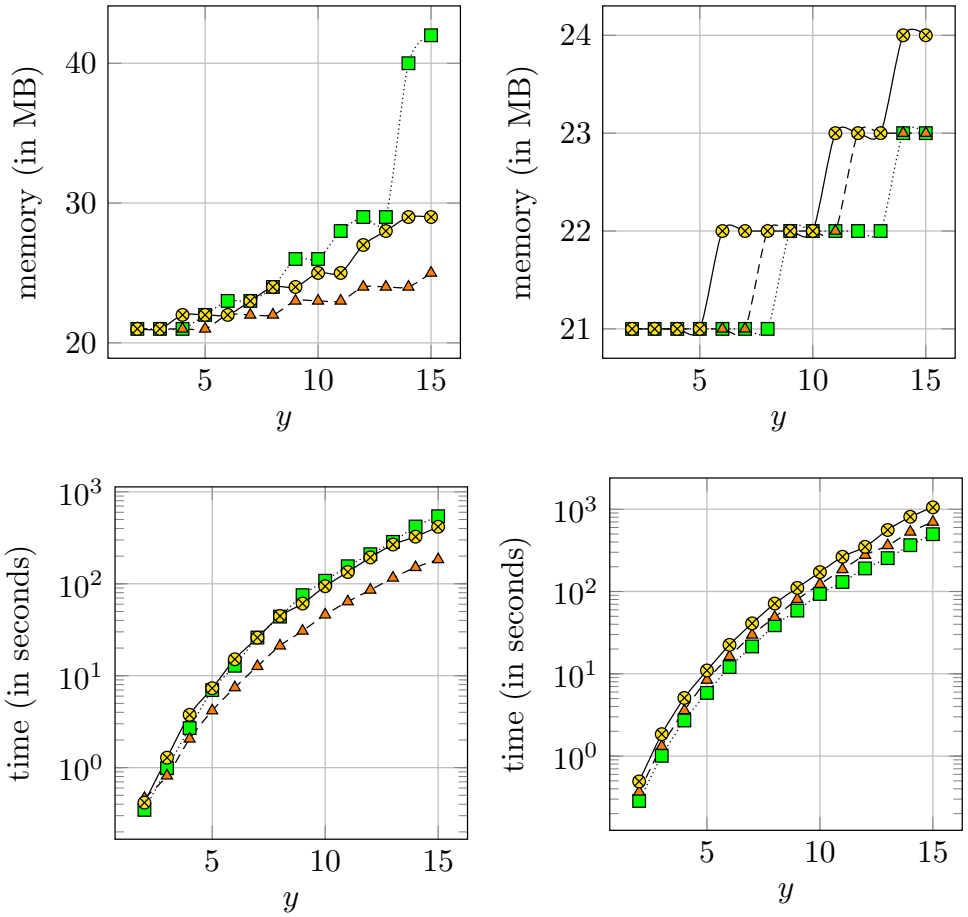


Fig. 10 The graphs of the operation time and memory used versus time. The presented data are related to values of x equal to 2 (□), 3 (△) and 4 (⊗). The graphs on the left-hand side depict systems with context automaton $\mathfrak{E}_4$, while the ones on the right-hand side with $\mathfrak{E}_5$

All active agents execute their reactions synchronously within a single ISPL computation step.

The translation is defined for formulae interpreted under CTLK semantics. Its correctness is ensured by the stuttering nature of global state sequences. Recall that a global state of a distributed reaction system is a composition of the local states of all agents. The translation modifies all variables corresponding to local states referenced in CTLK formulae simultaneously, during step 7. Between such updates, the global state remains constant. This is sufficient for the correct interpretation of G and U operators, but introduces difficulties in handling the X operator. Each occurrence of X in rsCTL must be replaced with a sequence (of appropriate length) of X operators in CTLK.

Furthermore, to ensure the correct evaluation of knowledge operators, only those entities that will be used in step 7 are distributed in step 6. As a result, agents cannot distinguish between global states in which particular reactions are disabled due to different causes – whether due to the absence of required products or the presence of inhibitors.

7 Experimental results

The proposed model checking method has been implemented in our toolkit—ReactICS (the results can be reproduced using https://github.com/arturmeski/reactics). The implementation uses C++ as the implementation language and the CUDD library (Somenzi 2009) for creating binary decision diagrams (BDD) and handling the operations on them. The tool uses the algorithms from Sect. 5.

The experiments were conducted on a machine equipped with an Intel® Xeon® Platinum 8260 processor and 1TB of RAM, running the Debian Linux operating system.

In the experimental evaluation we used our implementation with the models presented in Examples 4 and 6, employing both provided context automata. We configured the CUDD library to attempt to optimise the BDD variables by using its implementation of the group sifting algorithm from Panda and Somenzi (1995). However, an analysis of BDD variable orders and their optimal reordering strategies is beyond the scope of this paper. The experimental results are presented in Fig. 10. We considered values of the scaling parameter $y = z$ between 2 and 15. For each value of the scaling parameter we evaluated the rsCTLK formula (1),

Table 1 The comparison of model checking time (MC, sec.), total running time (TT, sec.), and total memory used (M, MB) by ReactICS and MCMAS for the formula (1) and systems with context automata $\mathfrak{E}_4$

x	y, z	Context automaton $\mathfrak{E}_4$						Value
		ReactICS			MCMAS			
		MC	TT	M	MC	TT	M	
2	2	0.30	0.36	21.00	0.23	1.09	17	False
2	3	0.87	0.99	21.00	1.05	3.64	37	False
2	4	2.45	2.74	21.00	3.56	15.19	54	False
2	5	6.42	7.00	22.00	14.03	29.25	58	False
2	6	12.22	13.06	23.00	35.55	62.63	65	False
2	7	25.13	26.53	23.00	61.03	115.01	83	False
2	8	42.64	44.95	24.00	176.35	315.77	147	False
2	9	73.42	77.09	26.00	366.02	579.12	157	False
2	10	105.70	111.50	26.00	328.02	667.80	187	False
2	11	153.40	160.30	28.00	1006.19	1924.15	314	False
2	12	204.60	214.10	29.00	2819.00	3695.35	665	False
2	13	278.30	289.70	29.00	1643.08	3323.01	634	False
2	14	410.00	424.70	40.00	4665.78	6213.95	945	False
2	15	521.30	539.50	42.00	7699.47	11507.60	1910	False
3	2	0.41	0.47	21.00	0.40	1.89	19	False
3	3	0.67	0.82	21.00	0.13	5.45	38	True
3	4	1.72	2.07	21.00	0.34	14.10	55	True
3	5	3.52	4.28	21.00	0.81	33.49	58	True
3	6	6.37	7.61	22.00	1.83	60.86	74	True
3	7	11.08	13.37	22.00	3.77	162.08	124	True
3	8	17.99	21.00	22.00	4.86	188.89	135	True
3	9	27.10	31.32	23.00	11.82	508.04	267	True
3	10	38.71	45.65	23.00	14.34	527.34	270	True
3	11	53.95	64.31	23.00	30.91	1026.20	470	True
3	12	72.77	85.35	24.00	63.16	2422.59	973	True
3	13	102.60	118.80	24.00	98.60	1840.90	1178	True
3	14	124.70	150.40	24.00	170.55	3579.14	1562	True
3	15	157.60	185.20	25.00	290.35	18414.90	2496	True
4	2	0.34	0.41	21.00	0.03	2.40	21	True
4	3	1.13	1.31	21.00	0.13	9.26	38	True
4	4	3.28	3.74	22.00	0.67	27.00	58	True
4	5	6.47	7.36	22.00	1.33	52.68	69	True
4	6	13.85	15.46	22.00	3.04	155.44	114	True
4	7	23.57	26.01	23.00	6.50	250.04	180	True
4	8	41.23	45.09	24.00	10.65	691.44	250	True
4	9	56.48	62.59	24.00	30.24	1449.25	502	True
4	10	85.82	93.51	25.00	37.61	1601.89	710	True
4	11	123.50	135.40	25.00	99.06	3302.76	1402	True
4	12	177.50	196.70	27.00	282.10	25488.80	4509	True
4	13	242.30	265.30	28.00	416.01	16638.00	4239	True

which holds in most of the tested models. It does not hold for $x = 2$ with $\mathfrak{E}_4$ and $x = 3$, $y = z = 2$ with $\mathfrak{E}_4$, as expected. For the remaining cases, we have positively verified the formula. This means that the outcome of the treatment could be determined (the activation of the transcription factor can be deduced from the local states of the 0^{th} agent).

The verification time scales exponentially. The only non-obvious case concerns the execution times for $\mathfrak{E}_4$. The computation times for two agents are shorter than the com-

Table 2 The comparison of model checking time (MC, sec.), total running time (TT, sec.), and total memory used (M, MB) by ReactICS and MCMAS for the formula (1) and systems with context automata $\mathfrak{E}_5$

x	y, z	Context automaton $\mathfrak{E}_5$						Value
		ReactICS			MCMAS			
		MC	TT	M	MC	TT	M	
2	2	0.24	0.28	21.00	0.01	0.74	13	True
2	3	0.91	1.01	21.00	0.03	2.00	18	True
2	4	2.51	2.72	21.00	0.06	5.18	36	True
2	5	5.66	6.06	21.00	0.12	10.49	40	True
2	6	11.71	12.35	21.00	0.27	16.34	51	True
2	7	21.28	22.33	21.00	0.46	29.73	49	True
2	8	38.80	40.22	21.00	0.60	58.69	52	True
2	9	57.55	59.61	22.00	1.17	79.54	58	True
2	10	92.56	95.85	22.00	2.03	150.54	66	True
2	11	129.40	133.40	22.00	2.14	206.50	70	True
2	12	187.10	192.60	22.00	3.63	281.41	94	True
2	13	252.60	259.40	22.00	4.46	353.73	107	True
2	14	356.70	365.10	23.00	8.22	476.67	159	True
2	15	486.90	498.00	23.00	8.61	959.82	160	True
3	2	0.32	0.39	21.00	0.01	1.21	16	True
3	3	1.29	1.48	21.00	0.05	3.83	28	True
3	4	3.41	3.65	21.00	0.11	9.18	34	True
3	5	8.17	8.69	21.00	0.30	16.28	54	True
3	6	15.71	16.49	21.00	0.51	27.13	41	True
3	7	29.25	30.63	21.00	0.62	51.31	57	True
3	8	47.92	50.03	22.00	1.53	115.99	64	True
3	9	79.18	82.12	22.00	1.82	166.46	68	True
3	10	119.00	123.10	22.00	2.48	289.47	85	True
3	11	180.10	185.80	22.00	4.21	364.15	110	True
3	12	270.20	277.20	23.00	7.77	467.15	168	True
3	13	355.80	364.80	23.00	34.38	1041.26	212	True
3	14	530.70	542.70	23.00	16.86	1828.61	286	True
3	15	701.50	716.70	23.00	22.63	3237.66	273	True
4	2	0.43	0.51	21.00	0.02	2.08	17	True
4	3	1.70	1.85	21.00	0.07	6.14	29	True
4	4	4.81	5.18	21.00	0.19	16.74	45	True
4	5	10.47	11.14	21.00	0.36	33.37	43	True
4	6	21.68	22.71	22.00	0.88	54.07	59	True
4	7	39.60	41.42	22.00	1.60	98.00	67	True
4	8	70.11	72.72	22.00	2.81	219.00	91	True
4	9	109.90	114.00	22.00	5.64	261.68	114	True
4	10	167.20	172.30	22.00	5.89	440.83	147	True
4	11	258.10	265.80	23.00	9.57	566.65	213	True
4	12	341.50	350.40	23.00	165.30	2799.84	383	True

putation times for three and four agents. Note that this corresponds to the result of the verification, where the verified formula did not hold for $x = 2$. This can be explained by the following observation: for $x > 2$, where the verified formula holds, it was sufficient to explore only a portion of the state space; in contrast, when the formula did not hold, the entire state space had to be explored.

As an additional step in evaluating our tool, we conducted comparative experiments with MCMAS (Lomuscio et al. 2017). For each considered scaling parameter value ($2 \leq x \leq 4$, $2 \leq y = z \leq 15$) and formula (1), we per-

formed model checking using ReactICS. Then, the input file was translated to ISPL using the method described in Sect. 6.1 and the resulting output file was processed by MCMAS. In both cases, the model checking time, total computation time, and memory used were measured. The results are presented in Tables 1 and 2. In most cases—particularly for larger values of the scaling parameters—both the model checking time and total running time are better for ReactICS. The maximum total running time observed for ReactICS was 539.5 s, compared to 25488.8 s for MCMAS, making MCMAS nearly two orders of magnitude slower in the worst case among the tested parameters. The memory consumption gap widens in favour of ReactICS, with MCMAS demonstrating significantly higher usage.

Our experimental results demonstrated that a model checker specialised for DRS outperforms a comparable tool designed for general multi-agent systems when verifying DRS models. One reason for this is that DRS are a qualitative formalism that also assumes non-permanency. As demonstrated by our translation in Sect. 6.1, emulating this behaviour is non-trivial and, as the results indicate, incurs a performance penalty.

8 Conclusions

We introduced a generalisation of reaction systems that allows for modelling distributed systems. We also extended the notion of context automata by allowing the behaviour of the environment also depend on the state of the reaction system. To allow for expressing temporal and epistemic properties of multi-agent systems, we defined rsCTLK, which is a logic combining rsCTL with CTLK. For the introduced formalisms we described a symbolic model checking method based on binary decision diagrams. The approach was implemented in ReactICS and evaluated experimentally on a biological benchmark of within-cell signal transduction networks. We also provided a comparison with another model checking tool that is, to the best of our knowledge, functionally most similar to ReactICS.

Distributed reaction systems offer a range of new opportunities for modelling complex biological processes. Our framework enables the modelling of competitive and collaborative systems where multiple agents (each with their own dynamics) interact, compete, or cooperate. Such systems are common in biology, including tumour-immune system interactions (with drug interventions), bacterial competition within microbiomes (where different strains vie for space and nutrients), and immune system coordination (where different cell types work together to identify and eliminate pathogens).

An important advantage of distributed reaction systems is their support for modular modelling. As shown in our examples, components can be developed independently and later integrated into a more comprehensive model, with their interactions governed by the distributed environment. This environment can be used to indicate which agents are active at different stages of the dynamic process. Furthermore, the context automaton adds an additional layer of flexibility, making it possible to explore alternative model scenarios, such as the effects of external interventions like drug treatments in disease models.

The ability to model multi-agent interactions with reaction systems can be further extended by incorporating quantitative descriptions of the models, for instance, through various kinetic laws. These can be implemented in either continuous or discrete frameworks and may involve deterministic, non-deterministic, or stochastic dynamics. Introducing such quantitative elements would bring reaction systems closer to traditional mathematical modelling approaches and make them more compatible with existing computational tools for model analysis.

Author contributions All authors contributed equally to this work.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

Azimi S, Gratie C, Ivanov S, Petre I (2015a) Dependency graphs and mass conservation in reaction systems. Theoret Comput Sci 598:23–39

Azimi S, Panchal C, Czeizler E, Petre I (2015b) Reaction systems models for the self-assembly of intermediate filaments. Ann Univ Bucharest LXI I(2):9–24

Azimi S, Gratie C, Ivanov S, Manzoni L, Petre I, Porreca AE (2016) Complexity of model checking for reaction systems. Theoret Comput Sci 623:103–113

Bowles J, Brodo L, Bruni R, Falaschi M, Gori R, Milazzo P (2024) Enhancing reaction systems with guards for analysing comorbidity treatment strategies. In: International conference on computational methods in systems biology. Springer, pp 27–44

Brijder R, Ehrenfeucht A, Rozenberg G (2011) Reaction systems with duration. In: Computation, cooperation, and life - essays dedicated to gheorghe păun on the occasion of his 60th birthday. LNCS, 6610, 191–202. Springer

Brodo L, Bruni R, Falaschi M (2023) Verification of reaction systems processes. In: Challenges of software verification. Springer, pp 243–264

Ciencialová L, Cienciala L, Csuhaj-Varjú E (2022) Languages of distributed reaction systems. In: International conference on machines, computations, and universality. Springer, pp 75–90

Ciencialová L, Cienciala L, Csuhaj-Varjú E (2023) Language classes of extended distributed reaction systems. Int J Found Comput Sci 2023:1–24

Clarke EM, Grumberg O, Peled DA (2021) Model checking. MIT Press

Corolli L, Maj C, Marini F, Besozzi D, Mauri G (2012) An excursion in reaction systems: From computer science to biology. Theoret Comput Sci 454:95–108

Dennunzio A, Formenti E, Manzoni L (2014) Extremal combinatorics of reaction systems. In: International conference on language and automata theory and applications. Springer, pp 297–307

Dennunzio A, Formenti E, Manzoni L, Porreca AE (2019) Complexity of the dynamics of reaction systems. Inf Comput 267:96–109

Ehrenfeucht A, Rozenberg G (2007a) Events and modules in reaction systems. Theoret Comput Sci 376(1–2):3–16

Ehrenfeucht A, Rozenberg G (2007b) Reaction systems. Fund Inform 75(1–4):263–280

Ehrenfeucht A, Rozenberg G (2009) Introducing time in reaction systems. Theoret Comput Sci 410(4–5):310–322

Ehrenfeucht A, Main M, Rozenberg G (2010) Combinatorics of life and death for reaction systems. Int J Found Comput Sci 21(03):345–356

Ehrenfeucht A, Main MG, Rozenberg G (2011) Functions defined by reaction systems. Int J Found Comput Sci 22(1):167–178

Ehrenfeucht A, Kleijn J, Koutny M, Rozenberg G (2013) Reaction systems: a natural computing approach to the functioning of living cells. In: A computable universe: understanding and exploring nature as computation. World Scientific, pp 189–208

Ehrenfeucht A, Kleijn J, Koutny M, Rozenberg G (2017) Evolving reaction systems. Theoret Comput Sci 682:79–99

Emerson EA, Halpern JY (1986) "Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic. J ACM 33(1):151–178

Fagin R, Halpern JY, Moses Y, Vardi MY (2003) Reasoning about knowledge. MIT Press, Cambridge

Ferrando A, Malvone V (2021) Towards the verification of strategic properties in multi-agent systems with imperfect information. Preprint at http://arxiv.org/abs/2112.13621

Genova D, Hoogeboom HJ, Jonoska N (2017) A graph isomorphism condition and equivalence of reaction systems. Theoret Comput Sci 701:109–119

Jones AV, Knapik M, Penczek W, Lomuscio A (2012) Group synthesis for parametric temporal-epistemic logic. In: International conference on autonomous agents and multiagent systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes), pp 1107–1114

Kari L, Rozenberg G (2008) The many facets of natural computing. Commun ACM 51(10):72–83

Kleijn J, Koutny M, Rozenberg G (2011) Modelling reaction systems with Petri nets. In: International workshop on biological processes & petri nets (BioPPN-2011)

Lehninger AL (1965) Bioenergetics: the molecular basis of biological energy transformations. Biology teaching monograph series. W. A. Benjamin

Lomuscio A, Ryan M (1997) On the relation between interpreted systems and Kripke models. In: Agents and multi-agent systems formalisms, methodologies, and applications. Lecture notes in artificial intelligence, vol 1441, Springer, pp 46–59

Lomuscio A, Qu H, Raimondi F (2017) MCMAS: an open-source model checker for the verification of multi-agent systems. Int J Softw Tools Technol Transfer 19(1):9–30

Maubert B, Murano A, Rubin S (2023) Logical aspects of multi-agent systems. Ann Math Artif Intell 91(4):373–374

Meacock OJ, Mitri S (2025) Environment-organism feedbacks drive changes in ecological interactions. Ecol Lett 28(1):70027

Męski A, Penczek W, Szreter M, Woźna-Szcześniak B, Zbrzezny A (2014) BDD- versus SAT-based bounded model checking for the existential fragment of linear temporal logic with knowledge: algorithms and their performance. Auton Agent Multi-Agent Syst 28(4):558–604

Męski A, Penczek W, Rozenberg G (2015) Model checking temporal properties of reaction systems. Inf Sci 313:22–42

Męski A, Koutny M, Penczek W (2016) Towards quantitative verification of reaction systems. In: Unconventional Computation and Natural Computation: 15th International Conference, UCNC 2016, Manchester, UK, July 11-15, 2016, Proceedings, 142–154

Męski A, Koutny M, Penczek W (2017) Verification of linear-time temporal properties for reaction systems with discrete concentrations. Fundam Inform 154(1–4):289–306

Męski A, Koutny M, Penczek W (2018) Reaction mining for reaction systems. In: Unconventional computation and natural computation - 17th international conference, UCNC 2018, Fontainebleau, France, June 25–29, 2018, Proceedings, pp 131–144

Męski A. Koutny M, Penczek W (2019) Model checking for temporal-epistemic properties of distributed reaction systems. Technical Report CS-TR-1526, School of Computing, Newcastle University, Newcastle upon Tyne, UK

Meyden R, Wong K-S (2003) Complete axiomatizations for reasoning about knowledge and branching time. Stud Logica 75(1):93–123

Panda S, Somenzi F (1995) Who are the variables in your neighborhood. In: Proceedings of the 1995 IEEE/ACM international conference on computer-aided design, ICCAD 1995, San Jose, California, USA, November 5-9, 1995, 74–77

Penczek W, Lomuscio A (2003) Verifying epistemic properties of multi-agent systems via bounded model checking. In: Proceedings of the second international joint conference on autonomous agents and multiagent systems. AAMAS '03. ACM, New York, pp 209–216

Raimondi F, Lomuscio A (2004) Automatic verification of deontic interpreted systems by model checking via OBDD's. In: Mántaras RL, Saitta L (eds.) Proceedings of ECAI, pp 53–57

Raimondi F, Lomuscio A (2005) Symbolic model checking of multi-agent systems using OBDDs. In: Proc. of the 3rd NASA workshop on formal approaches to agent-based systems (FAABS III), Volume 3228 of LNCS. Springer, Cham, pp 213–221

Rozenberg G, Bäck T, Kok JN (eds) (2012) Handbook of natural computing. Springer, Cham

Salomaa A (2013) Functional constructions between reaction systems and propositional logic. Int J Found Comput Sci 24(01):147–159

Somenzi F (2009) CUDD: CU decision diagram package-release 2.4.0, vol 21, University of Colorado at Boulder

Wooldridge M (2002) An introduction to multi-agent systems. Wiley, England

Zañudo JGT, Scaltriti M, Albert R (2017) A network modeling approach to elucidate drug resistance mechanisms and predict combinatorial drug treatments in breast cancer. Cancer Converg 1(1):1–25