

Dear Author,

Here are the proofs of your article.

- You can submit your corrections **online**, via **e-mail** or by **fax**.
- For **online** submission please insert your corrections in the online correction form. Always indicate the line number to which the correction refers.
- You can also insert your corrections in the proof PDF and **email** the annotated PDF.
- For fax submission, please ensure that your corrections are clearly legible. Use a fine black pen and write the correction in the margin, not too close to the edge of the page.
- Remember to note the **journal title, article number**, and **your name** when sending your response via e-mail or fax.
- **Check** the metadata sheet to make sure that the header information, especially author names and the corresponding affiliations are correctly shown.
- **Check** the questions that may have arisen during copy editing and insert your answers/corrections.
- **Check** that the text is complete and that all figures, tables and their legends are included. Also check the accuracy of special characters, equations, and electronic supplementary material if applicable. If necessary refer to the *Edited manuscript*.
- The publication of inaccurate data such as dosages and units can have serious consequences. Please take particular care that all such details are correct.
- Please **do not** make changes that involve only matters of style. We have generally introduced forms that follow the journal's style.
Substantial changes in content, e.g., new results, corrected values, title and authorship are not allowed without the approval of the responsible editor. In such a case, please contact the Editorial Office and return his/her consent together with the proof.
- If we do not receive your corrections **within 48 hours**, we will send you a reminder.
- Your article will be published **Online First** approximately one week after receipt of your corrected proofs. This is the **official first publication** citable with the DOI. **Further changes are, therefore, not possible.**
- The **printed version** will follow in a forthcoming issue.

Please note

After online publication, subscribers (personal/institutional) to this journal will have access to the complete article via the DOI using the URL: [http://dx.doi.org/\[DOI\]](http://dx.doi.org/[DOI]).

If you would like to know when your article has been published online, take advantage of our free alert service. For registration and further information go to: <http://www.link.springer.com>.

Due to the electronic nature of the procedure, the manuscript and the original figures will only be returned to you on special request. When you return your corrections, please inform us if you would like to have these documents returned.

Metadata of the article that will be visualized in OnlineFirst

ArticleTitle	Experimenting with reaction systems using graph transformation and GROOVE	
Article Sub-Title		
Article CopyRight	The Author(s), under exclusive licence to Springer Nature B.V. (This will be the copyright line in the final PDF)	
Journal Name		
Corresponding Author	Family Name Particle Given Name Suffix Division Organization Address Phone Fax Email URL ORCID	Rensink Arend CS Department University of Twente Hallenweg 19, 7522, Enschede, Netherlands arend.rensink@utwente.nl
Author	Family Name Particle Given Name Suffix Division Organization Address Phone Fax Email URL ORCID	Bruni Roberto CS Department University of Pisa Largo B. Pontecorvo, 3, 56127, Pisa, Italy roberto.bruni@unipi.it
Schedule	Received Revised Accepted	27 July 2025
Abstract	We explore the capabilities of GROOVE, a state-of-the-art toolset based on graph transformation systems, to perform different kinds of analyses of Reaction Systems, ranging from reachability and causal analysis to model checking. Our results are encouraging, as in the presence of large state spaces GROOVE improves the time required for both reachability and causal analyses by an order of magnitude, compared to other available tools. From the point of view of GROOVE, the implementation of Reaction Systems provided some interesting insights on the most convenient way to model certain computational requirements through negative and nested application conditions.	
Keywords (separated by '-')	Reaction systems - Graph transformation - GROOVE - Causal analysis - Formal methods	
Footnote Information		



Experimenting with reaction systems using graph transformation and GROOVE

Roberto Bruni¹ · Arend Rensink²

Accepted: 27 July 2025
© The Author(s), under exclusive licence to Springer Nature B.V. 2025

1 Abstract

2 We explore the capabilities of GROOVE, a state-of-the-art toolset based on graph transformation systems, to perform different
3 kinds of analyses of Reaction Systems, ranging from reachability and causal analysis to model checking. Our results are
4 encouraging, as in the presence of large state spaces GROOVE improves the time required for both reachability and causal
5 analyses by an order of magnitude, compared to other available tools. From the point of view of GROOVE, the implementation of
6 Reaction Systems provided some interesting insights on the most convenient way to model certain computational requirements
7 through negative and nested application conditions.

8 **Keywords** Reaction systems · Graph transformation · GROOVE · Causal analysis · Formal methods

9 1 Introduction

10 We explore the capabilities of a state-of-the-art toolset based
11 on graph transformation to perform reachability, causal anal-
12 ysis and verification of complex systems modelled using
13 Reaction Systems.

14 Reaction systems (RSs) (Ehrenfeucht and Rozenberg
15 2007) are a computational model inspired by the function-
16 ing of biochemical reactions within living cells. RSs focus
17 on the interaction of entities through a set of reactions. Each
18 reaction relies on some reactants, inhibitors, and products to
19 mimic two fundamental mechanisms found in nature: facilita-
20 tion and inhibition. At each time instant, the next state of the
21 system is determined by the products of all enabled reactions
22 plus some additional entities that are possibly provided by
23 the environment. Unlike traditional models of concurrency,
24 like Petri nets, the theory of RSs is based on three principles:
25 *no permanency*: any entity vanishes unless it is sustained by
26 a reaction; *no competition*: an entity is either available for

27 all reactions, or it is not available at all; and *no counting*:
28 the exact concentration level of available entities is ignored.
29 Moreover, due to the use of inhibitors, RS can exhibit non-
30 monotonic behaviour, in the sense that what can be done
31 with fewer resources is not necessarily replicable with more
32 resources.

33 While *closed* RSs evolve in isolation, *interactive pro-
34 cesses* are dealt with by providing suitable environments that
35 provide a sequence of stimuli at each step: these are sets of
36 entities that can be used to trigger or inhibit some reactions.
37 A common example involves using contexts to analyse how
38 drug administration affects organisms that are modelled as
39 sets of reactions.

40 Since their introduction, RSs have been successfully
41 applied to the analysis of complex systems in many dif-
42 ferent fields (Azimi et al. 2014; Corolli et al. 2012; Azimi
43 2017; Okubo and Yokomori 2016; Ehrenfeucht et al. 2010,
44 2011). Recent applications concerned, e.g., with the effi-
45 cacy of medical treatments for comorbidities and with the
46 selection of the best environment to achieve some desired
47 phenomena (Bowles et al. 2024; Brodo et al. 2025a), led to
48 experimenting with environments that exhibit nondeterministic
49 and recursive behaviour. As a consequence, performing
50 reachability and causal analysis requires the exploration of
51 large state spaces, for which the prototype tool BioResolve
52 (Brodo et al. 2021) struggled in terms of memory consump-
53 tion and response time.

✉ Arend Rensink
arend.rensink@utwente.nl

Roberto Bruni
roberto.bruni@unipi.it

¹ CS Department, University of Pisa, Largo B. Pontecorvo, 3,
56127 Pisa, Italy

² CS Department, University of Twente, Hallenweg 19, 7522
Enschede, Netherlands

Graph Transformation (GT) (Ehrig et al. 2006; Heckel and Taentzer 2020) is a modelling technique that is widely applicable in problem domains where the objects of study have an inherent graphical structure, and the task at hand is to study their properties and evolution. Besides the graphs themselves, the core concept is that of a (transformation) *rule* capturing a particular change to such a graph. Rules can be used, for instance, to describe the change of a system over time, but can also be instrumental in composing and decomposing graphs and so exposing structural properties. Since RSs can be derived from Boolean network models and visualised themselves as suitable networks of reactions, it is quite natural trying to embed them within the GT framework to take advantage of well established analysis techniques.¹

Importantly, from a practical point of view, there are a number of (academic) tools supporting the use of GT. The research described here crucially relies on Ghamarian et al. (2012), one of the most prominent tools in this area, which was designed precisely to enable GT-based system analysis of the kind described above. The features of GROOVE that are essential for the purpose of this research are:

1. Nested (i.e., quantified) rules, which capture simultaneous changes in all neighbourhoods that satisfy certain application conditions, rather than only locally in one such neighbourhood at a time;
2. Complete exploration of the set of reachable states (under the given rules) using various strategies;
3. Model checking functionalities that can be used to validate previous findings as well to explore and support the study of new behavioural and structural properties.

The main research question that motivated our study is: *how can GT help in addressing the analysis of Reaction Systems?* To this aim, we encode a given RS as a single graph, upon which a small number of (fixed) rules can simulate the correct semantics.

The core rule describes the simultaneous firing of all **Reactions** of which no inhibitors and all reactants are present; the firing results in the presence of all products. Simultaneously, all currently present **Entities** are removed. In GROOVE syntax, the core rule is drawn as in Fig. 1. To parse this, note that (in the GROOVE notation) the red structure must be absent for the rule to apply; moreover, green labels are added and blue ones deleted upon rule application. The *present* flag signals whether an **Entity** is considered to be currently present;

¹ It should be noted that there is another, quite unrelated, way in which the concepts of Graph Transformation and Reaction systems may be combined, namely by extending the latter from pure entities (which are analogous to graph nodes) to entities with relations between them (which can be encoded through graph edges). In Kreowski and Rozenberg (2019) this has inspired a new methodology for Graph Transformation, there called *graph surfing*.

hence, creating or deleting that flag comes down to creating or deleting the **Entity**. The \forall -nodes impose the desired quantification, causing a single application of this rule to model the firing of all enabled **Reactions**, even if there are thousands of them. Similarly for the simultaneous deletion of all **Entitys**.

Our model also supports environments that inject **Entitys** in a controlled manner. This is achieved by encoding the context specification in the initial graph and exploiting a second predefined rule, not shown here (see Fig. 7). A configuration is reachable if it can be constructed by the alternating application of both rules: first the context produces a set of stimuli (possibly upon inspection of the current state, but also nondeterministically or a combination of both), and then all enabled reactions are executed.

After a brief account of RSs (Sect. 2.1) and GROOVE (Sect. 2.2), we present the overall rationale, blueprint and key features of our approach in Sect. 4 through a simple vending machine example (Sect. 3). The subsequent experimentation in Sect. 5 is concerned with revisiting existing RS case studies to assess how GROOVE can enhance their analysis. In particular, we tackle the comorbidity treatment scenario from Bowles et al. (2024) in Sect. 5.1, the protein signalling networks analysis from Ballis et al. (2024) in Sect. 5.2, and the T cell differentiation study from Brodo et al. (2025a) in Sect. 5.3. Our results are encouraging: GROOVE is capable to explore large RSs by using roughly one tenth of the time compared to BioResolve for both reachability and causal analyses. More precisely, we are able to find a trace towards unwanted patterns (if they exist) among hundreds of thousands of reachable configurations using different heuristics; then, we can also prune the trace to extract a graphical representation of the causal history of that entities. Using the built-in model checker it is also possible to confirm the results of previous studies, as well as proving new facts. A comparison with related tools is drawn in Sect. 6. Concluding remarks and future directions are discussed in Sect. 7.

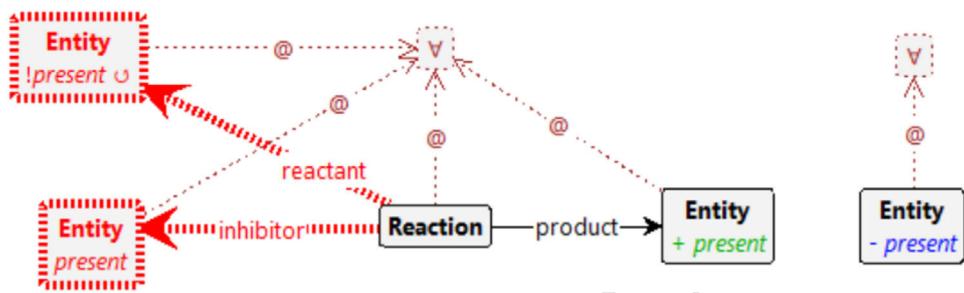
2 Background

2.1 RSs with guarded contexts

First, we briefly account for the classical set theoretic definition of Reaction Systems (RSs) (Ehrenfeucht and Rozenberg 2007). Then, we focus on their process algebraic version (Brodo et al. 2021) and its further extension with guarded contexts in Bowles et al. (2024) that are supported by the RS analysis framework BioResolve.²

RS basics. A Reaction System is a pair $A = (S, A)$, where S is the finite set of *entities*, and A is a finite set of *reactions* of

² <https://www.di.unipi.it/~bruni/LTSRS/>.

Fig. 1 Rule for reactions firing

the form $a = (R, I, P)$, with $R, I, P \subseteq S$ and $R \cap I = \emptyset$.
 The sets R, I, P are the sets of *reactants*, *inhibitors*, and
products, respectively. Without loss of generality, we admit
 the use of empty sets as reactants, inhibitors, and products.
 Given the current state $W \subseteq S$, a reaction $a = (R, I, P)$ is
 enabled in W if all its reactants are present (i.e., $R \subseteq W$)
 and all its inhibitors absent (i.e., $W \cap I = \emptyset$). The *result* of
 the reaction a on the current state W is P if a is enabled,
 and \emptyset otherwise. The result of all reactions A on the current
 state W , is the union of the results of all reactions. The no-
 permanency principle of RSs dictates that entities disappear
 if not sustained by some reaction. Thus, the current state
 $W = D \cup C$ is determined by the result D of all reactions
 on the previous state, together with some additional entities
 C that can be provided by the *context* at each step.

Process algebraic RSs and guarded contexts. Inspired by Plotkin's Structural Operational Semantics approach (Plotkin 2004) and process algebras such as CCS (Milner 1980), the key features of the process algebraic version of RSs are compositionality and the ability to account for a quite general notion of context (guarded, nondeterministic, recursive) using a friendly syntax. This way, we derive a Labelled Transition System (LTS) semantics for RSs by means of inductive inference rules, where LTS states are terms of an algebra, each transition defines a computation step of the RS and its label records the entities involved in that step.

RS processes are defined by the grammar below:

```

P ::= [M]
M ::= (R, I, P) | D | K | M|M
K ::= 0 | (R, I, C).K | K + K | X
  
```

where R, I, P, C , and D are sets of entities (with $R \cap I = \emptyset$) and X is a context identifier drawn from a family of (possibly recursive) definitions $\Delta \triangleq \{X_j = K_j\}_{j \in J}$, called the *environment*.

Roughly, a RS process P embeds a *mixture* process M obtained as the parallel composition of some reactions (R, I, P) , some available entities D (if any), and some *context* process K . A context process K is either: the nil context 0 that stops the computation; the guarded context $(R, I, C).K$ that makes the entities C available to the reactions if the reac-

tants R are present and the inhibitors I are absent, and then will behave as K at the next step; the non-deterministic choice $K_1 + K_2$ that can behave as either K_1 or K_2 ; the context identifier X that behaves as K for $X = K \in \Delta$. We write $C.K$ as a shorthand for the trivially guarded process $(\emptyset, \emptyset, C).K$ and we assume the recursive context $\text{Emp} = \emptyset.\text{Emp}$ is always defined.

We say that P and P' are structurally equivalent, written $P \equiv P'$, when they denote the same term up to the laws of Abelian monoids (unit, associativity and commutativity) for parallel composition $\cdot|.$, with \emptyset as the unit, and the laws of idempotent Abelian monoids for choice $\cdot + \cdot$, with 0 as the unit. We also assume $D_1|D_2 \equiv D_1 \cup D_2$ for any $D_1, D_2 \subseteq S$. Indexed sums and parallel compositions are denoted, respectively, by $\sum_{i \in I} K_i$ and $\prod_{i \in I} M_i$.

The SOS semantics of RS processes is defined by the SOS rules in Fig. 2. A transition label ℓ , written $\langle\langle D \triangleright R', I', C \rangle\rangle \triangleright R, I, P$, records: the available entities D ; the entities C provided by the guarded contexts, assuming all entities in R' are present and those in I' are absent; the set R of entities whose presence enables or disables some reactions; the set I of entities whose absence enables or disables some reactions; and the set P of reaction products. The rules guarantee that, whenever $P \xrightarrow{\langle\langle D \triangleright R', I', C \rangle\rangle \triangleright R, I, P} P'$, it holds that (R', I', C) is enabled in D and that (R, I, P) is enabled in $W \triangleq (D \cup C)$.

The rule (*Ent*) records the set of current entities D . By rule (*Cxt*), a guarded context process $(R, I, C).K$ makes available the entities in C if the reactants R are present in the current state and the inhibitors I are absent, and then reduces to K . Rules (*Suml*) and (*Sumr*) select a move of either the left or the right context, resp., discarding the other process. By rule (*Rec*), a context identifier X behaves according to its defining process K .

The rule (*Pro*) assumes the reaction (R, I, P) is enabled: it records its reactants, inhibitors, and products in the label, and leaves the reaction available at the next step, together with its products P . The rule (*Inh*) records in the label the reasons why the reaction (R, I, P) should not be executed: possibly some inhibiting entities ($J \subseteq I$) are present or some reactants ($Q \subseteq R$) are missing, with $J \cup Q \neq \emptyset$, as at least one cause is needed.

$$\begin{array}{c}
\frac{}{D \xrightarrow{\langle\langle D \triangleright \emptyset, \emptyset, \emptyset \rangle \triangleright \emptyset, \emptyset, \emptyset \rangle} \emptyset} (Ent) \\
\frac{\mathsf{K}_1 \xrightarrow{\ell} \mathsf{K}'_1}{\mathsf{K}_1 + \mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}'_1} (Suml) \quad \frac{\mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}'_2}{\mathsf{K}_1 + \mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}'_2} (Sumr) \quad \frac{X = \mathsf{K} \in \Delta \quad \mathsf{K} \xrightarrow{\ell} \mathsf{K}'}{X \xrightarrow{\ell} \mathsf{K}'} (Rec) \\
\\
\frac{(R, I, P) \xrightarrow{\langle\langle \emptyset \triangleright \emptyset, \emptyset, \emptyset \rangle \triangleright R, I, P \rangle} (R, I, P)|P}{(R, I, P)} (Pro) \quad \frac{J \subseteq I \quad Q \subseteq R \quad J \cup Q \neq \emptyset}{(R, I, P) \xrightarrow{\langle\langle \emptyset \triangleright \emptyset, \emptyset, \emptyset \rangle \triangleright J, Q, \emptyset \rangle} (R, I, P)} (Inh) \\
\\
\frac{\mathsf{M}_1 \xrightarrow{\ell_1} \mathsf{M}'_1 \quad \mathsf{M}_2 \xrightarrow{\ell_2} \mathsf{M}'_2 \quad \ell_1 \frown \ell_2}{\mathsf{M}_1 \mid \mathsf{M}_2 \xrightarrow{\ell_1 \cup \ell_2} \mathsf{M}'_1 \mid \mathsf{M}'_2} (Par) \quad \frac{\mathsf{M} \xrightarrow{\langle\langle D \triangleright R', I', C \rangle \triangleright R, I, P \rangle} \mathsf{M}' \quad R' \subseteq D \quad R \subseteq D \cup C}{[\mathsf{M}] \xrightarrow{\langle\langle D \triangleright R', I', C \rangle \triangleright R, I, P \rangle} [\mathsf{M}']} (Sys)
\end{array}$$

where $\ell_1 \frown \ell_2$ and $\ell_1 \cup \ell_2$ are defined as follows:

$$\begin{aligned}
\langle\langle D_1 \triangleright R'_1, I'_1, C_1 \rangle \triangleright R_1, I_1, P_1 \rangle \frown \langle\langle D_2 \triangleright R'_2, I'_2, C_2 \rangle \triangleright R_2, I_2, P_2 \rangle &\triangleq (\bigcup_{i=1,2} D_i \cup R'_i) \cap (I'_1 \cup I'_2) = \emptyset \wedge (\bigcup_{i=1,2} D_i \cup C_i \cup R_i) \cap (I_1 \cup I_2) = \emptyset \\
\langle\langle D_1 \triangleright R'_1, I'_1, C_1 \rangle \triangleright R_1, I_1, P_1 \rangle \cup \langle\langle D_2 \triangleright R'_2, I'_2, C_2 \rangle \triangleright R_2, I_2, P_2 \rangle &\triangleq \langle\langle D_1 \cup D_2 \triangleright R'_1 \cup R'_2, I'_1 \cup I'_2, C_1 \cup C_2 \rangle \triangleright R_1 \cup R_2, I_1 \cup I_2, P_1 \cup P_2 \rangle
\end{aligned}$$

Fig. 2 SOS semantics of the RS processes

The rule (*Par*) puts two processes in parallel by pooling their labels and joining all labels components. We write $\ell_1 \cup \ell_2$ for the component-wise union of labels, while the sanity check $\ell_1 \frown \ell_2$ is required to guarantee that labels of reactants and inhibitors are consistent (see definitions in Fig. 2).

Finally, the rule (*Sys*) checks that all the needed reactants are available in the system. Checking the absence of inhibitors is not necessary, thanks to the sanity check in rule (*Par*). Note that, while the enabling of $(R, I, C).K$ requires the presence of reactants R and the absence of inhibitors I w.r.t. the set of current entities D , in the case of reactions (R, I, P) , the check is performed w.r.t. the current state $W = D \cup C$. More importantly, the products C are made available immediately from the context, not at the next step. It is worthy to mention that a conditional prefixed process that is not enabled behaves as the $\mathbf{0}$ process.

Remark 1 It is worth noting that any RS process can be written in the form $[Rs \mid Ks \mid D]$ where $Rs = \prod_i (R_i, I_i, P_i)$ is the parallel composition of all reactions in the system, $Ks = \prod_j K_j$ is the parallel composition of all contexts and D is the set of currently available entities. Moreover, it can be proved that a generic transition has the shape: $[Rs \mid Ks \mid D] \xrightarrow{\langle\langle D \triangleright R', I', C \rangle \triangleright R, I, P} [Rs \mid Ks' \mid P]$, i.e., reactions are always preserved by transitions, the first component D of the transition label is just the set of available entities in the source state and the new result set in the target state is just the product set P observed in the label of the transition. The choices in Ks are taken by considering the set of available entities D (in the case of guarded prefixes) and will determine the context C appearing in the label as well as the

continuation Ks' appearing in the target state. Given D and C , the product P is then uniquely determined by Rs . For brevity, we will sometimes draw the LTS, by recording only the strict amount of information in nodes and labels. Thus the above sample transition will be abbreviated as $[Ks \mid D] \xrightarrow{C} [Ks' \mid P]$, assuming reactions Rs are known a priori.

A first concrete example of RS that exposes most features is presented in Sect. 3.

2.2 GT and GROOVE

Graph Transformation (GT, sometimes called Graph Rewriting) is a well-established rule-based formalism, the core of which is to specify precisely how graphs may evolve. Each rule embodies a particular change, which can be applied to a given graph (in the simplest form consisting of nodes and binary edges) by establishing where in that graph the pre-conditions of the rule are met, and then adding and deleting nodes and edges as prescribed.

In this paper, we use the so-called *algebraic approach* to graph transformation (see Ehrig et al. 2006 for a formal exposition and Heckel and Taentzer (2020) for applications in the context of software engineering); moreover, we rely on the particular flavour implemented in the tool GROOVE (Ghamarian et al. 2012; Rensink 2024). Some of the relevant features of the approach and the tool are highlighted below.

Graphs are *simple* and *typed*, meaning that there is at most one edge of a given type between any two nodes and that all nodes and edges are labelled through a morphism to a given (fixed) *type graph*. Edges are *directed* (going from their

source to their *target*). Besides binary edges, nodes can also have *flags* (which are actually self-loops that act as additional, optional labels on nodes) and *attributes* (which are actually binary edges whose target node is a data value, e.g., an integer or a string).

Rules, in their simplest form, consist of a left hand side (LHS) and right hand side (RHS). Rule applicability is established by *matching* the LHS to the graph in question, and where a match exists, removing nodes and edges that are in the LHS but not in the RHS, and vice versa, adding nodes and edges that are in the RHS but not in the LHS. In addition, however, GROOVE supports *quantified* rules, which can simultaneously be applied to multiple places in the same graph. An example is shown in Fig. 7 below.

Evolution of a graph is defined on the basis of a graph transformation system, which is a set of rules applied to a graph at hand, giving rise to a modified graph to which every rule can be applied again, and so forth. On top of this, GROOVE allows for *control programs* that can specify in what order rules may be applied. By exploring the potential evolution of a graph in all ways allowed by the control program, GROOVE constructs the *state space* of the graph transformation system, in the form of a *labelled transition system* consisting of all reachable graphs and the rule applications between them.

Analysis consists of the exploration of the state space for a given initial graph, rule system and (optional) control program. The exploration can be tuned by built-in strategies for searching and model checking.

The power of graph transformation lies in its generality: many systems naturally lend themselves to be modelled as graphs, and algebraic rules—especially quantified ones—provide a rich framework to specify their evolution. This is in fact our motivation for using it the current paper: reaction systems can straightforwardly be interpreted as graphs. Figure 3 shows the core types for the relevant concepts of that interpretation. (The colours just support the visualisation and have no semantics of their own.)

Note especially the (abstract) supertype **Rule** with subtypes **Reaction** and **Step**: the former is the type for the elements of A in a Reaction System $A = (S, A)$, whereas the latter is used to represent triples (R, I, P) in a context process K . The flag *fired* is used to mark **Rules** that have triggered in the most recent step. The set S is represented by nodes of type **Entity**; for a given **Rule**, the subsets R , I , and P of S are those **Entity**s to which there is an outgoing edge labelled reactant, inhibitor or product. The subtype **Forbidden** anticipates the principle, demonstrated later in this paper, of identifying undesirable entities and specifically searching for scenarios in which those are produced. The flag *present* is used to label the entities occurring in a state W . Finally, the structure of (guarded) context processes is captured by **State** entities, with next-edges to the **Steps** that

can be non-deterministically chosen; the subsequent process after such a **Step** is determined by its outgoing move-edge. **Token** nodes are used to model which **States** are currently active.

3 Running example: a toy vending machine

To illustrate some basic concepts of RSs and, in the next section, of the proposed GROOVE encoding, we model a system composed of a student and a vending machine as a toy example. The vending machine accepts two different kinds of coins and can dispense either a cappuccino or an espresso when a coffee coin is inserted or a tea if a tea coin is inserted. A cappuccino is dispensed if some milk is available, otherwise espresso is produced. Assuming the powder for preparing coffee and tea are always present, the corresponding process can be written as follows:

$$\begin{aligned} \text{VM} \triangleq & \{\{\text{ccoin}, \text{cpowder}\}, \{\text{nomilk}\}, \{\text{cappuccino}\}\} \\ | & \{\{\text{ccoin}, \text{cpowder}, \text{nomilk}\}, \emptyset, \{\text{espresso}\}\} \\ | & \{\{\text{tcoin}, \text{tpowder}\}, \emptyset, \{\text{tea}\}\} \\ | & \{\{\text{cpowder}\}, \emptyset, \{\text{cpowder}\}\} \\ | & \{\{\text{tpowder}\}, \emptyset, \{\text{tpowder}\}\} \end{aligned}$$

A refill context process can, nondeterministically, refill the machine with milk.

$$\text{Refill} \triangleq \{\text{nomilk}\}. \text{Refill} + \emptyset. \text{Refill}$$

The student process is very simple: she takes cappuccino in the morning and tea in the afternoon, otherwise she gets angry.

$$\begin{aligned} \text{Student} \triangleq & (\{\text{am}\}, \emptyset, \{\text{ccoin}\}). \text{GetCappuccino} \\ & + (\emptyset, \{\text{am}\}, \{\text{tcoin}\}). \text{GetTea} \\ & + \{\text{idle}\}. \text{Student} \\ \text{GetCappuccino} \triangleq & (\{\text{cappuccino}\}, \emptyset, \emptyset). \text{Student} \\ & + (\{\text{espresso}\}, \emptyset, \{\text{anger}\}). \text{Student} \\ \text{GetTea} \triangleq & (\{\text{tea}\}, \emptyset, \emptyset). \text{Student} \\ & + (\emptyset, \{\text{tea}\}, \{\text{anger}\}). \text{Student} \end{aligned}$$

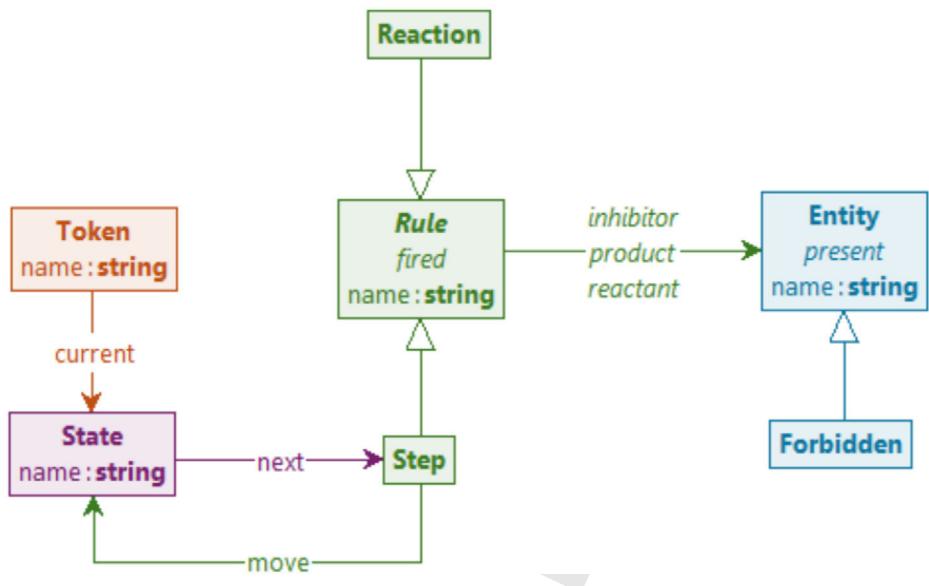
If the student is angry, she will bang the machine:

$$\text{Anger} \triangleq (\{\text{anger}\}, \emptyset, \{\text{bang}\})$$

Finally, two more reactions model the passage of time (morning vs afternoon) while the student is idle (i.e., not in the process of getting beverages).

$$\begin{aligned} \text{Day} \triangleq & (\{\text{idle}\}, \{\text{am}\}, \{\text{am}\}) \\ | & (\{\text{am}\}, \{\text{idle}\}, \{\text{am}\}) \end{aligned}$$

Fig. 3 Core type graph for reaction systems



367 We assume that, initially, both entities `cpowder` and
 368 `tpowder` are present. So the system can be coded as the
 369 guarded RS process

370 [Refill|Student|{cpowder, tpowder}|VM|Anger|Day].

371 The complete encoding of the above RS in BioResolve
 372 syntax is reported in Fig. 19 in the Appendix. Using the
 373 BioResolve directive `main_do(digraph)`, we can auto-
 374 matically generate the underlying LTS as in Fig. 4: the initial
 375 state is in light blue, while there is also a “bad” state in which
 376 the student is banging the machine, shown in light coral. Note
 377 that, as the entity `am` is initially not present, it means that the
 378 initial time is in the afternoon. In drawing the transition sys-
 379 tem, we have used the representational convention explained
 380 in Remark 1).

381 In this particular example, we wish to analyse why `bang` is
 382 produced, i.e., what are its causes. By manual inspection we
 383 can recover a trace starting from the initial state and leading
 384 to the “bad” state, e.g., $\xrightarrow{\text{idle}} \xrightarrow{\text{ccoin;nomilk}} \xrightarrow{\text{anger}}$ (highlighted
 385 in red in Fig. 4). From this trace, using the dynamic slicing
 386 process described in Brodo et al. (2024a), we can reconstruct
 387 that the production of `bang` was due to the prior production
 388 of `anger`, which in turn was caused by the student getting
 389 espresso instead of cappuccino, which is because there was
 390 no `milk` when a `ccoin` was inserted at `am`.

4 Encoding of RS in GT

392 As an alternative to BioResolve, we investigate the use of
 393 graph transformation and GROOVE to generate the under-
 394 lying LTS of a given Reaction System, on the basis of a

395 start graph obtained by transformation from the RS specifi-
 396 cation. Because the start graph will typically include special
 397 **Entity** subtype, it comes together with an additional type
 398 graph where those are specified. Depending on what one
 399 wants to analyse, the various strengths and capabilities of
 400 GROOVE then come into play.

401 For instance, one possibility is to use GROOVE’s model
 402 checking capabilities to check for temporal patterns of entity
 403 generation in the transition system. Another way to proceed
 404 is to focus on a given trace and build its *occurrence graph*,
 405 which contains all the rule occurrences and entity instances
 406 present in that trace—analogue, in fact, to the way a Petri net
 407 process captures a particular behaviour. If the trace in ques-
 408 tion leads to a state in which a **Forbidden** entity is present
 409 (such as the `bang` entity in our toy example), we can also
 410 *prune* the occurrence graph, again using graph transfor-
 411 mation, to keep only those rule occurrences and entity instances
 412 that directly contributed to the existence of the forbidden
 413 entity.

414 This gives rise to the tool chain depicted in Fig. 5, the
 415 phases of which will be explained in some more detail in the
 416 remainder of this section.

417 **Transform.** The first step is a text-to-model transforma-
 418 tion from a problem specification in BioResolve syntax
 419 into GROOVE syntax. This is achieved by running the
 420 `main_do(rs2gts)` directive of BioResolve, which pro-
 421 duces two artefacts: firstly, an additional type graph, com-
 422 plementary to the one shown in Fig. 3, which specifies one
 423 subtype of **Entity** for each of the entities in the problem at
 424 hand (essentially for performance reasons: relying on dedi-
 425 cated types speeds up the matching step of GROOVE); and
 426 secondly (more importantly) a start graph in which the entire
 427 BioResolve system is encoded as suggested by Fig. 3. For

Fig. 4 LTS of the toy example. For brevity we use the notation introduced in Remark 1, where node labels only account for the list of (semicolon separated) current contexts and entities and transition labels carry the entities provided by the context

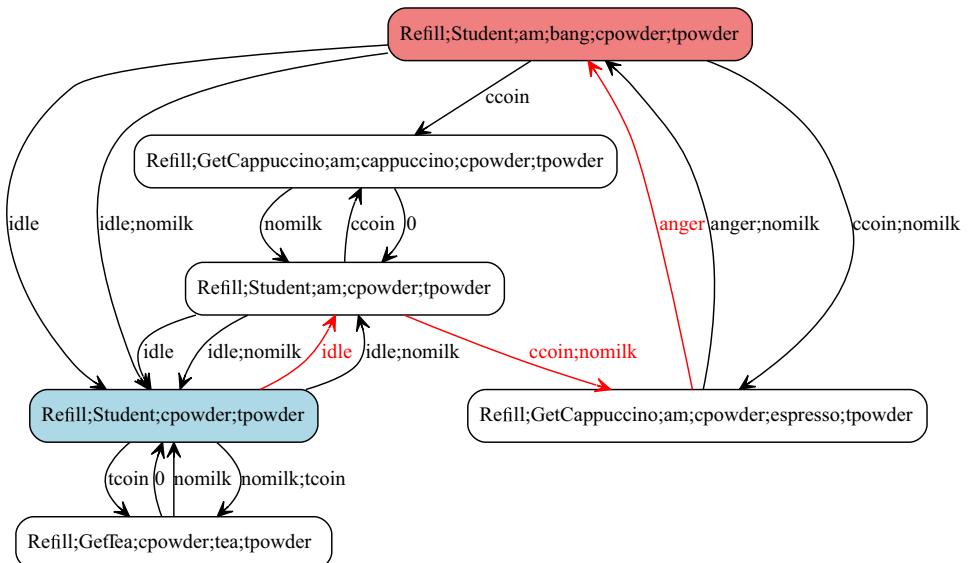
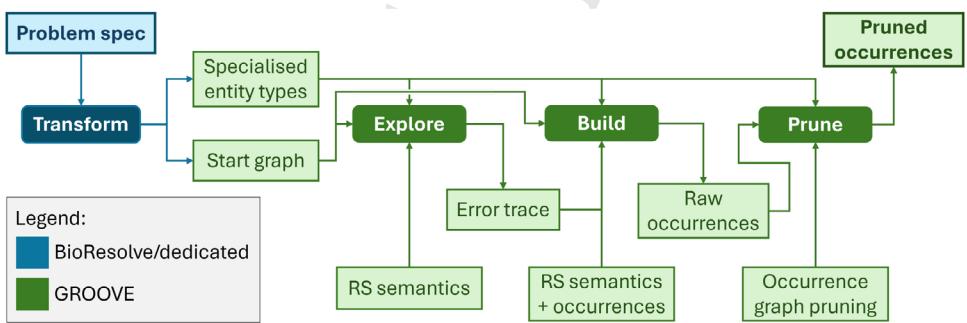


Fig. 5 Reaction system exploration and analysis using GROOVE



the example system, the additional types as well as two self-explanatory fragments of the start graph are shown in Fig. 6.

We claim that this transformation is semantics-preserving; Appendix A gives a sketch of the argument. A fully formal statement and proof of semantic correspondence, however, is outside of the scope of this paper.

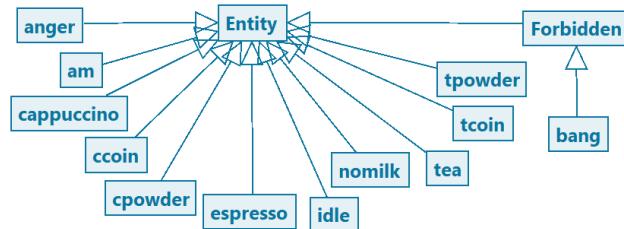
Explore. The dynamics of Reaction Systems is encoded as a combination of two rules, **context** and **react**, which are scheduled to fire in alternation. The rule **context** encodes the simultaneous firing of all context processes (nondeterministically selecting an enabled **Step** from every **State** with a **Token**), whereas **react** encodes the (deterministic) simultaneous firing of all enabled **Reactions**, while simultaneously erasing all **Entity**s that were not just produced. The production or erasure of an **Entity** is encoded through the creation or deletion of a *present* flag on a (persistent) **Entity** node, *not* by the creation or deletion of the node itself. In addition, to keep track of which nondeterministic choices were actually taken, the **context** rule marks the **Steps** that were selected with a *-fired* flag, which is subsequently erased by the **react** rule.

Figure 7 shows the first (and most intricate) of these rules, viz. the one for the context firing. This is a quantified rule, which can be read as follows: For all **States** with a **Token**, there is a next **Step** such that for all inhibitors there is no *present* flag whereas for all reactants there is a *present* flag; moreover, when the rule is applied, all products of the selected **Steps** receive a *present* flag, the **Steps** themselves receive a *fired* flag, and all **Tokens** move to the successor **States**. Colour coding is used in the visual representation to distinguish the quantifier nodes \forall and \exists (both in purple), as well as the mandatory absence (red), deletion (blue) and creation (green) of edges and flags.³

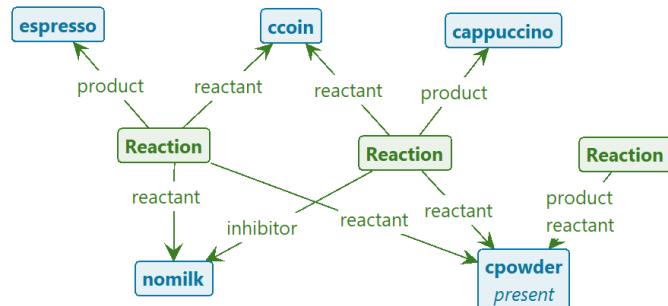
To mimic the BioResolve semantics as closely as possible, we can instruct GROOVE to regard every pair of **context**- and **react**-transitions as an atomic transaction, corresponding precisely to a transition in BioResolve (though not with the same label), and then generate the entire state space. This is achieved through a control program of the form

³ This colour coding is GROOVE-specific and entirely separate from the problem-specific colouring of the graph nodes in Figs. 3 and 6; in fact, to avoid confusion, the problem-specific colouring is *not* used in the rule view.

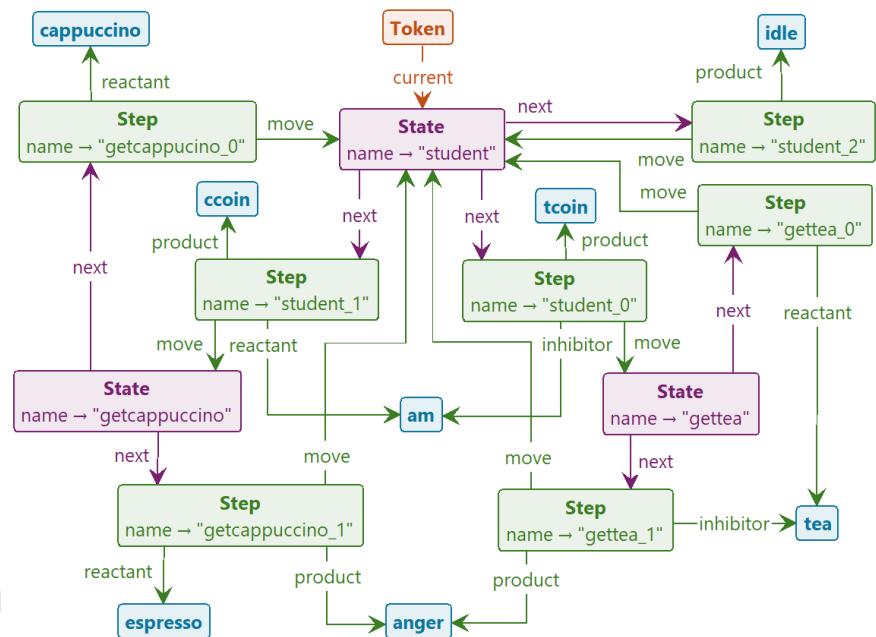
Fig. 6 Graph representation of running example



(a) Specialised entity types

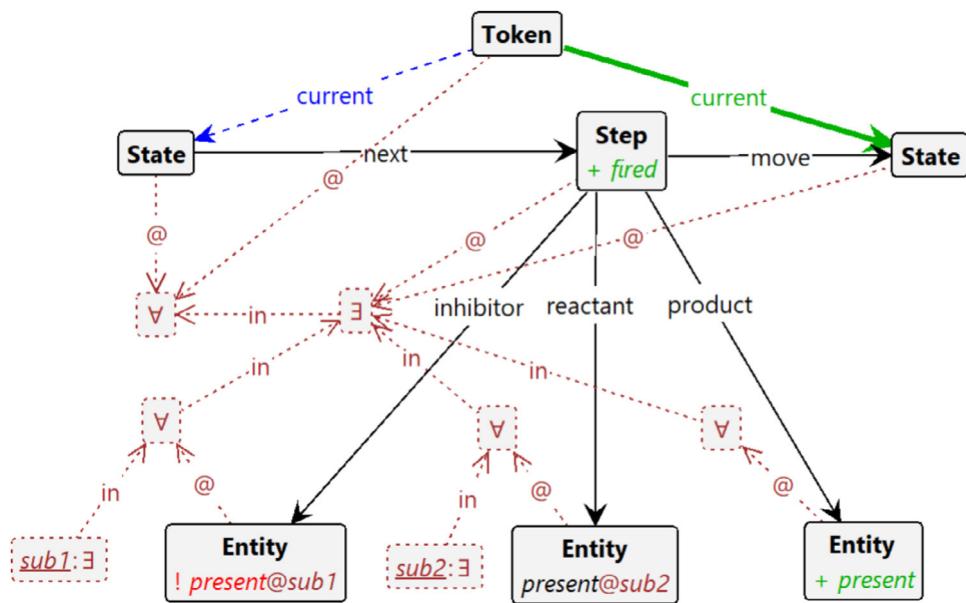


(b) Start graph fragment: Three reactions from VM



(c) Start graph fragment: The Student context process

Fig. 7 Rule for context firing



```
468     recipe fire() {  
469         context; react;  
470     }
```

```
fired("getcappuccino_1");
react;
```

471 where a “recipe” is the keyword for a transaction wrap-
 472 ping the body. With this in place, the GUI-based version
 473 of GROOVE produces the transition system displayed in Fig.
 474 8 (which can also be exported to a range of standard for-
 475 mats), which is easily (visually) checked to be essentially
 476 isomorphic to the one in Fig. 4.

Alternatively, we can (for instance) ask GROOVE to search the for the first reachable state in which a **Forbidden** entity appears, using breadth-first search. (In fact, the state property for which GROOVE searches is itself determined by a *rule*, which in this case merely tests for the presence of a **Forbidden** entity). When found, the trace to the forbidden state can be saved as a control program that drives the next stage of GROOVE exploration. In particular, using the alternating application of the context and react rules (rather than the transactional variant used for Fig. 8) this control program also records the fired-applications that tell which **Steps** have fired: this completely determines how the non-determinism in the context process has been resolved in order to arrive at the forbidden state. Here is the control program for the shortest trace to state *s17* in our running example:

```
492     context;  
493     fired("student_2");  
494     fired("refill_1");  
495     react;  
496     context;  
497     fired("student_1");  
498     fired("refill_0");  
499     react;  
500     context;  
501     fired("refill_1");
```

Here `student_2`, `student_1` and `getcappuccino_1` are the **Steps** of the **Student** process visualised in Fig. 6; `refill_1` and `refill_0` are the steps of the **Refill** process given in Sect. 3.

Build. The purpose of this phase is to build an occurrence graph that explains how **Forbidden** was produced, by collecting its (transitive) dependencies (Brodo et al. 2024b). Concretely, we record the following dependencies:

- From each non-initial **Entity** instance to the **Rule** occurrence of which it is the product;
 - From each **Rule** occurrence to all its reactant **Entity** instances;
 - From each **Step** occurrence to all directly preceding **Step** occurrences.

Figure 9 shows the occurrence type graph.

With respect to the slicing algorithm used in, e.g., Brodo et al. (2025a), Brodo et al. (2024a), the difference is that our dependencies are based on instances, and that we explicitly include the rule occurrences and their predecessors. Each of these different kinds of dependencies is visualised both through a specific edge (product, reactant or predecessor) between the relevant instance- and occurrence-nodes, and (for the sake of a more uniform treatment during the next step of *pruning*) through an auxiliary `depends`-edge on the level of **TreeNode**, of which all others are subtypes.

Note that all this is restricted to *positive* dependencies. In fact, we have constructed our example so that there are no inhibitors in the **Rules** that fire in the trace above; if we

Fig. 8 GROOVE LTS of the toy example

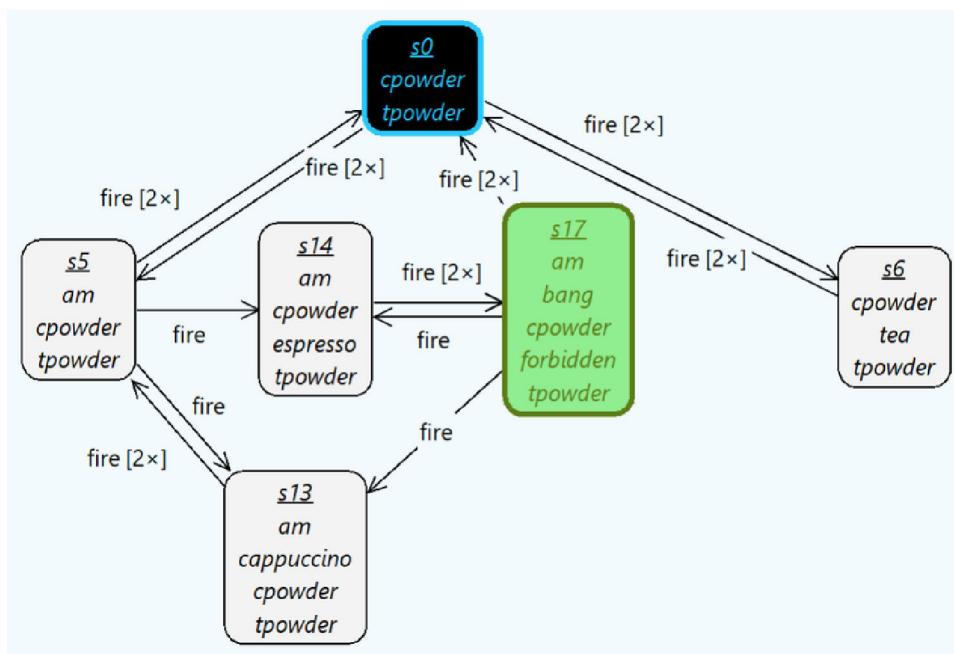
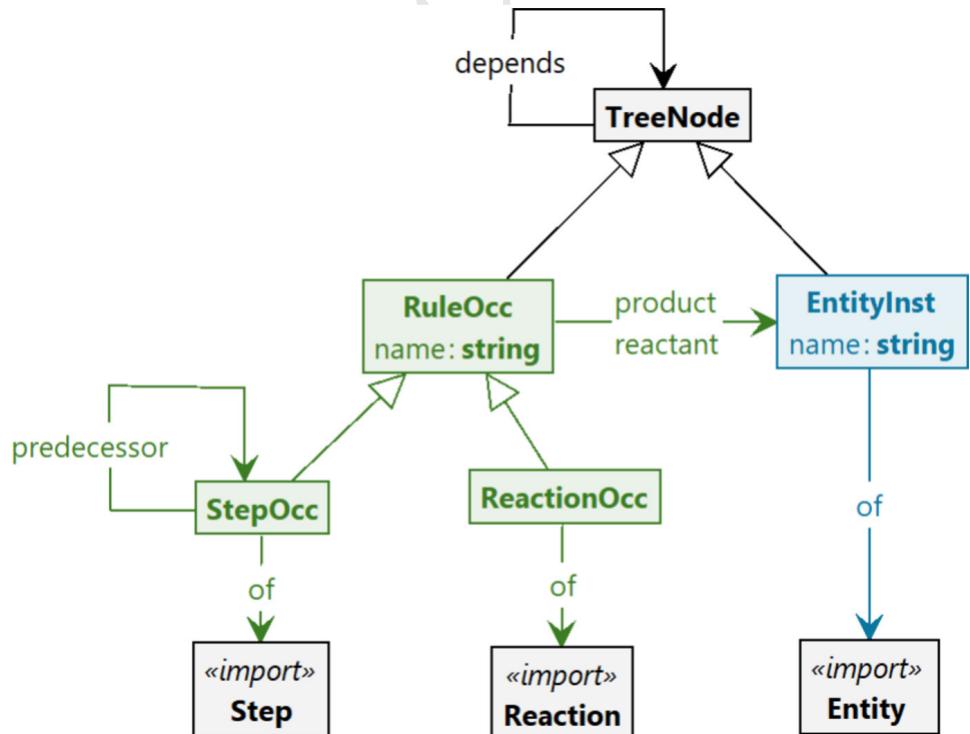


Fig. 9 Occurrence type graph



532 would rely on an entity milk that inhibits the production of
 533 espresso, rather than on nomilk as a reactant, the occurrence
 534 graph for bang would not include milk; and likewise if we
 535 would use cappuccino as an inhibitor for anger rather than
 536 espresso as a reactant for it. The representation of negative
 537 dependencies is a research question in its own, and is outside
 538 the scope of this paper (although a possibility to capture neg-
 539 ative dependencies is to focus the analysis on the Positive RS

540 that results from the transformation defined in Brodo et al.
 541 (2024b)).

542 As indicated in Fig. 5, the occurrence graph is produced
 543 by another GROOVE rule system, using the same start graph
 544 but driven by the control program previously created in the
 545 explore phase, which encodes (as we have seen) the trace to
 546 the undesirable state. The occurrence graph semantics con-
 547 sists of rules with the same names (react, context and fired),

but different functionality: in particular, rather than manipulating *present* flags, the react rule now creates **RuleOcc**- and **EntityInst**-nodes together with their dependencies. This is a non-trivial procedure that in fact itself requires several successive stages. Though the details of these stages are not of sufficient interest to include in this paper, we want to point out that breaking down a single rule (react, in this case) into multiple stages would seem to contradict the tenet of algebraic graph transformation that a rule embodies a single, atomic change to a graph. This contradiction is solved by relying once more on GROOVE recipes: in the occurrence graph semantics, react is actually not a rule but a recipe, defined as

```
561   recipe react() {
562     entities-age;
563     react-produce;
564     merge;
565   }
```

of which the three atomic steps perform the necessary book-keeping to correctly produce the occurrence graph.

Prune. The occurrence graph built by the rule system described above is too large to be useful: it contains *all* entity instances and rule occurrences produced by the trace, not just the dependencies of the undesired **Forbidden** entity. Moreover, the entire start graph is also (still) present. Therefore, in a third phase, all redundant information is pruned. This is achieved by first marking all transitive backward dependencies, and then removing all unmarked nodes. Since this is straightforward, and of no particular interest in the context of this paper, we omit the details of the GROOVE rule system for this phase. Its outcome for our running example is shown in Fig. 10 (where we have elided the auxiliary depends-edges).

The pruned occurrence graph visualises the causal effect chain already explained informally at the end of Sect. 3: the presence of a ccoin, which itself depends on am, combined with cpowder and nomilk causes the production of espresso, after which the student produces anger and then bang, which is **Forbidden**.

5 Experimentation

Here we consider three larger case studies whose RS specifications have already appeared in the recent literature. For each case study, we briefly describe its main features and then show how the methodology outlined in the previous sections can be applied for carrying out some fruitful experimentation with GROOVE.

All GROOVE experiments were carried out using GROOVE version 7.4.3 on a Dell Precision 3551 laptop with an Intel i7 CPU running at 2.6 GHz; GROOVE was run in a Java 24

JVM with 12GB of memory. No attempt was made to measure running time with precision, and repeated experiments have shown that the reported durations can deviate up to 25%. In order to facilitate replication of the experiments, we have included supplementary materials with this paper, including the required rule systems and start graphs and instructions for invoking GROOVE; see Sect. C.5.

5.1 Comorbidity treatment analysis

This case was studied in Bowles et al. (2024), where guarded contexts were introduced to handle key features of medical treatments. It concerns the risk mitigation of medication harm in the treatment of patients with comorbidities; i.e., patients with two or more long-term chronic conditions (such as diabetes, hypertension, cardiovascular diseases, chronic kidney disease, cancer, chronic obstructive pulmonary disease, among many others), who are therefore subject to follow several treatment plans simultaneously, called *clinical guidelines* (Feder et al. 1999; Woolf et al. 1999). Since clinical guidelines address a single disease, comorbidities easily lead to *polypharmacy*, where five or more medications must be administered, increasing the risk of adverse drug reactions, or of making certain drugs less effective when combined (Hughes et al. 2013).

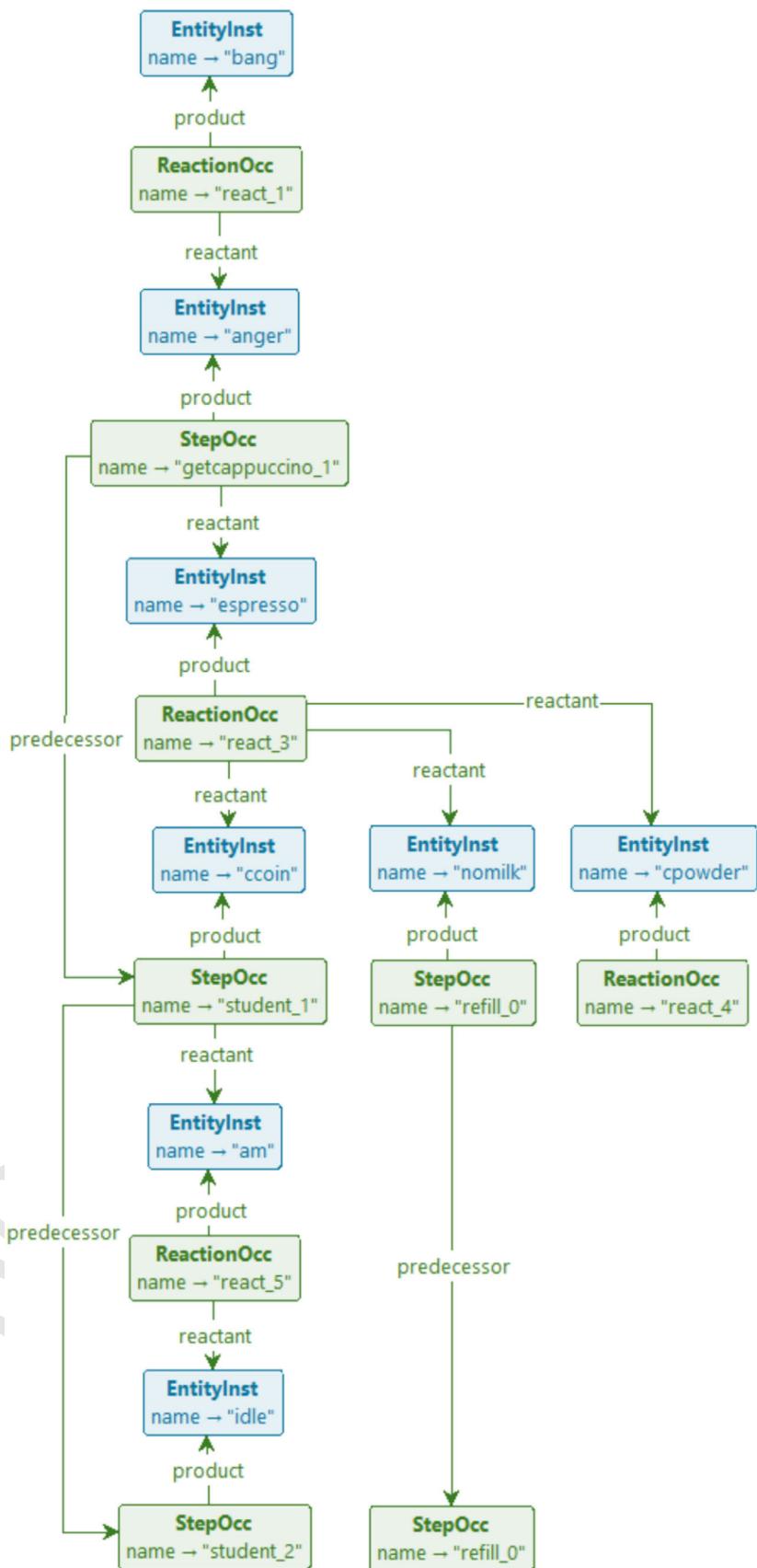
Analysis goals. The goal of the analysis is to explore the combination of clinical guidelines in the presence of comorbidities and for different patient profiles to detect if major risks can arise from the treatments and which profiles are exposed at severe risks. Using formal methods for risk mitigation intends to help doctors choose between alternative treatment options as well as to point out missing conditions that could be helpful to revise and update clinical guidelines.

Features of interest. In this case study, reachability and causal analysis are key issues. Specifically, reachability is used to address questions such as *can the combination of clinical guidelines expose the patient at serious risks because of drugs interference?* Then, in the affirmative case, causal analysis can help to detect which medical decisions would be directly responsible for causing serious harm as well as to point out which alternative treatment would be available, if any. We selected this case study because the use of guarded contexts introduces new challenges for the causal analysis of RSs: while existing approaches typically focus on identifying combinations of drugs administered within the context that may have caused harm, they often fail to highlight the medical decisions that led to their administration.

Experimental set up. The RS encoding proposed in Bowles et al. (2024) relies on a formal representation of patient profiles, medical guidelines and adverse drug reactions.

For each drug d that appears in the therapies, we consider three corresponding entities get_d , stop_d and d : the first represents the prescription of d by the doctor, the second the

Fig. 10 Pruned occurrence graph of the running example



removal of d from the current treatment and the third the intake of the drug by the patient. For handling multiple drugs of the same class c , we exploit analogous entities `stop_c` and c .

For medical guidelines, it takes in input the event structure modelling of therapies introduced in Bowles and Caminati (2017). Roughly, to each event e there is an identifier E_e defined as a sum of processes, one for each outgoing arc of e . If some guard is attached to the arc, then the corresponding alternative is also guarded. The prescription of a drug d is modelled by the provision of the entity `get_d`. Similarly, if the therapy requires stopping the drug d , the entity `stop_d` is produced.

The patient profile is determined by the conditions that trigger the treatment (e.g., headache, hypertension) and by the conditions that appear in the arc labels of the event structure (e.g., pregnant, asthma). We call them *features*. Correspondingly, there is one context $K_f = \{f\}.\text{Emp}$ for each feature f , and a patient profile is just a combination of some features $K_{f_1} \mid \dots \mid K_{f_n}$. Once the profile is determined by the context, it is preserved during the rest of the computation by reactions of the form $(\{f\}, \emptyset, \{f\})$, one for each feature. Accounting for all possible patient profiles within a single model can be done by considering the context $\prod_f (K_f + \text{Emp})$.

For each drug d of class c , there will be the following reactions: $(\{\text{get}_d\}, \{\text{stop}_d, \text{stop}_c\}, \{d, c\})$ modelling the intake of the drug d as for doctor prescription, and $(\{d\}, \{\text{stop}_d, \text{stop}_c\}, \{d, c\})$ modelling the prosecution of the therapy. Adverse drug reactions are provided in the form of so-called ADR tables. Each row corresponds to a set of medications M , a textual description of their side effects and risks when used in combination, and a severity level m (e.g., minor, moderate, major). Each row translates to a reaction $(M, \emptyset, \{m\})$.

The whole RS specification can be found in the Appendix: the list of reactions is in Fig. 20 and context processes are defined in Fig. 21.

Previous approach.

The approach outlined in Bowles et al. (2024) has been used to synthesise the patient profiles that are more at risk, as a support for dynamic guideline revision: by refining guarded contexts to prevent severe effects for specific patients, we can readily check the efficacy of the changes.

GROOVE experimentation.

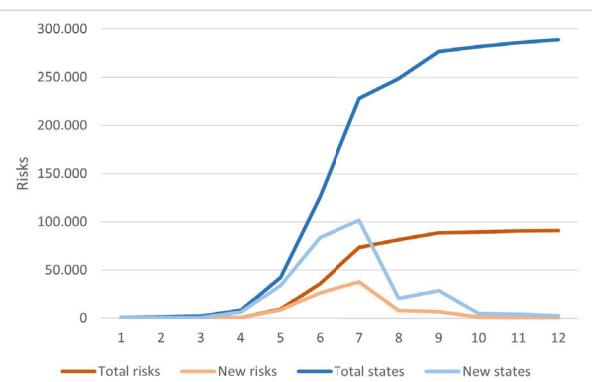
The benefit of using GROOVE in this case study is that, besides identifying situations where a risk is found, for any risk so identified the corresponding occurrence graph of a risk can be generated, using the process outlined in Sect. 4. This provides a means for medical experts to more easily analyse root causes: for any risk that has been identified, what is the causal structure of the steps and entities leading up to it?

Measurement	Search method	Count	Time (s)
Total states	BFS	309 798	3 368
Total states	DFS	309 798	2 464
Major risks	DFS	91 113	2 447
Moderate risks	DFS	97 805	1 534
Maj&Mod risks	DFS	61 976	2 727
Minor risks	DFS	24	2 218

(a) Full exploration

Explore depth	Total risks	New risks	Total states	New states
1	0	0	769	769
2	0	0	1 345	576
3	0	0	1 873	528
4	716	716	8 133	6 260
5	9 317	8 601	42 293	34 160
6	35 899	26 582	126 012	83 719
7	73 591	37 693	227 866	101 854
8	81 691	8 100	248 420	20 554
9	88 631	6 940	276 924	28 504
10	89 733	1 102	281 854	4 930
11	90 573	840	286 054	4 200
12	91 133	560	288 854	2 800

(b) Bounded BFS exploration of major risks (tabular)



(c) Bounded BFS exploration of major risks (chart)

Fig. 11 States, risks and execution time

GROOVE can be used for full state space generation, for instance to count the number of ways a minor, moderate or major risk can arise. Some statistics can be found in Fig. 11.

The first two lines show that depth-first exploration strategy (DFS) is generally more convenient than breadth-first (BFS), so we rely on the former for the subsequent entries. At first sight, the number of major (and moderate) risks reported in Fig. 11 seems impossibly large, and evidently implies that there are patient profiles that give rise to many risks. However, this should be interpreted with care: the count refers to

709 the total number of configurations containing a **Forbidden**
 710 (i.e., major or minor) entity, and there may very well be entities
 711 whose presence or absence does not causally contribute
 712 to that **Forbidden** entity—in other words, which would not
 713 appear in its occurrence graph. Configurations counted as
 714 separate risks may well reduce to the same causation. To
 715 analyse this further, one would have to construct (and prune)
 716 the occurrence graphs for all risk configurations, and com-
 717 pare them on that basis. Though this is beyond the scope of
 718 this paper, such an analysis is in principle straightforward to
 719 carry out in GROOVE—it is a matter of combining the three
 720 steps in Fig. 5 into a single rule system.

721 The line of Fig. 11a headed “Maj&Mod risks” reports the
 722 number of configurations at which *both* a major and a mod-
 723 erate entity appear during the same step; hence, these are
 724 counted as both major and moderate risks (partially explain-
 725 ing their high numbers).

726 By exploring only up to a certain depth, we can get some idea
 727 of the number of steps after which a risk typically appears,
 728 which in turn indicates the complexity of the context in which
 729 it appears. Figure 11b shows how many major risks occur
 730 after a fixed number of reaction steps, and also how much of
 731 the total state space is involved. Note that, in this case, the
 732 total state count (reached after 12 steps) stays below that of
 733 Fig. 11b; this is because in the experiments of Fig. 11b we
 734 stop exploration at states where a risk has been found. Figure
 735 11c shows the same data in a graphical form.

736 Alternatively, we can stop exploring after having found a
 737 predetermined number of risks. By setting the exploration
 738 strategy to breadth-first search, it is guaranteed that the risks
 739 found are those reached after the shortest number of steps,
 740 meaning they are the easiest to analyse visually. For instance,
 741 Fig. 12 shows the occurrence graph of the first major risk
 742 found in this way.

743 The patient configuration in question is a combination of
 744 `has_fib`, `afib` and `heart_rate`; the combination of
 745 the first two leads to the prescription of `fleacinide` and
 746 the second to the prescription of `diltiazem`, the combi-
 747 nation of which should, however, be avoided. By counting
 748 the longest chain of **StepOcc**-nodes, it is confirmed that it
 749 indeed takes 4 steps to establish this risk.

750 We can also use GROOVE to replicate the findings of Bowles
 751 et al., (2024, Fig. 6) in terms of the relation between patient
 752 profiles and risks, using model checking. Recall that the
 753 reaction system starts by having the context produce initial
 754 entities, and in this particular case study, the first move of
 755 the context is to select a patient profile; hence the initial state
 756 has $2^9 = 512$ outgoing transitions, whose target state corre-
 757 sponds to the chosen profile. Moreover, the rule `forbidden`
 758 tests for the presence of a **Forbidden** entity in a state. There-
 759 fore, a formula of the shape $\text{AX}(\bigwedge_i f_i \rightarrow \text{EF } \text{forbidden})$,
 760 where each of the f_i specifies the presence or absence of a

761 patient feature, specifies whether all patient profiles with that
 762 combination of features contain a potential risk.

763 Concretely, in the case study at hand, Bowles et al. (2024,
 764 Fig. 6) contains necessary and sufficient criteria for patient
 765 profiles to contain major, moderate and minor risks. The part
 766 of the table pertaining to major risks is reproduced in Fig.
 767 13. This should be read as: *precisely* in the combination of
 768 features where the green ones are present and the red ones
 769 absent, a major risk may occur.

770 In order to replicate these results, we can use CTL model
 771 checking. For instance, the relevant CTL property for the
 772 major risks is also shown in Fig. 13. Running the CTL model
 773 checker built into GROOVE, it reports that this is indeed sat-
 774 isfied, as are the corresponding characteristic properties for
 775 the moderate and minor risks. The following table reports the
 776 time taken for these checks (where the precision of our time
 777 measurement is such that the apparent difference between
 778 the model checking times is not significant):

Model generation:	2236 s
Major risk check:	67 s
Moderate risk check:	71 s
Minor risk check:	65 s

779 Note that the time taken for model generation is consistent
 780 with that reported in Fig. 11b.

781 Discussion.

782 Compared to the prior results in Bowles et al. (2024), the
 783 advantages of using GROOVE lie in performance and flexi-
 784 bility:

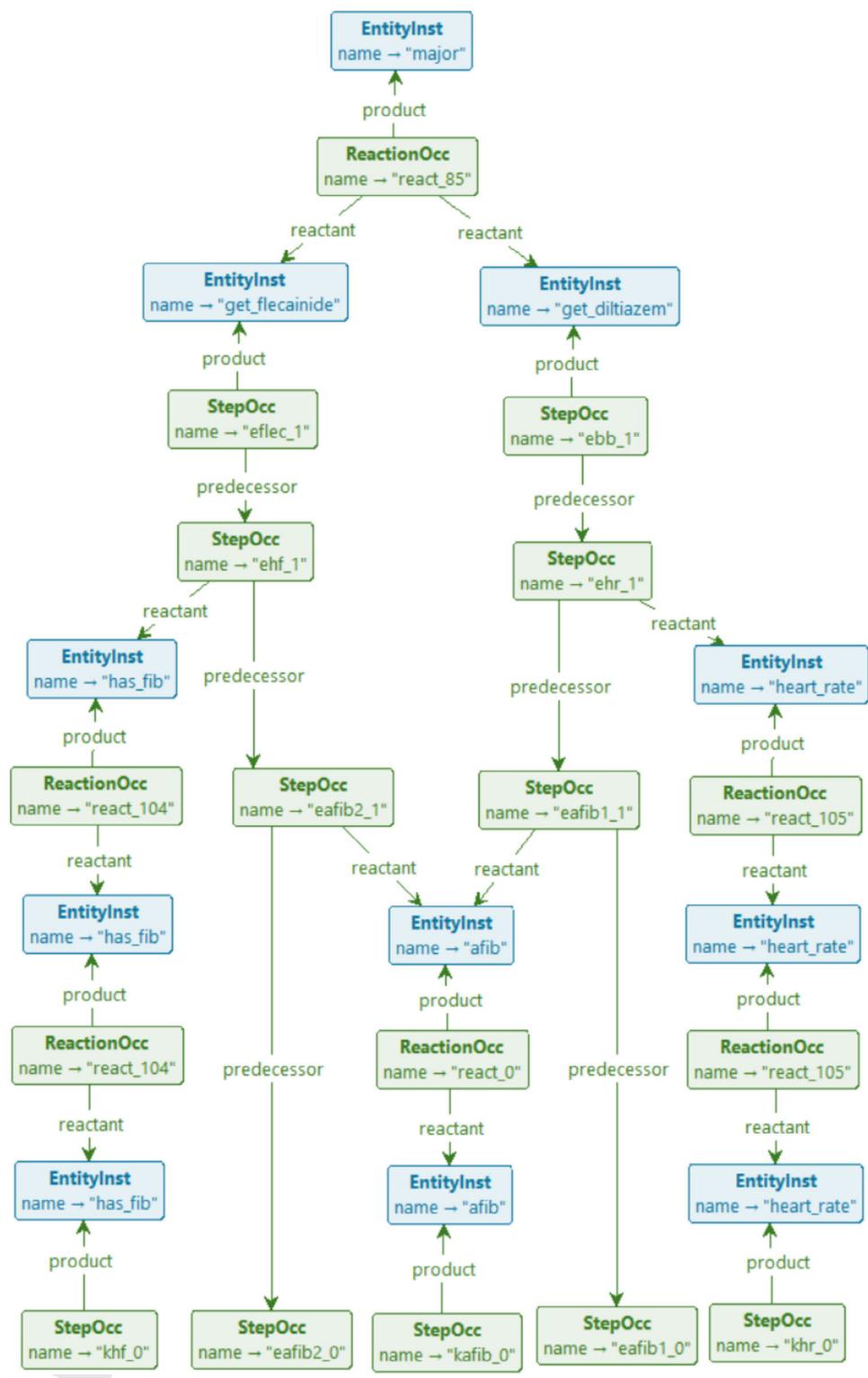
- 785 • The time needed to analyse the entire state space is around
 786 41 min for GROOVE; while not particularly fast, this still
 787 compares very favourably to the 5 h needed by BioRe-
 788 solve for LTS generation.
- 789 • Finding shortest paths to risk configurations and com-
 790 puting occurrence graphs is part of the core functionality
 791 of GROOVE—given, of course, suitable rule systems that
 792 encode the chosen notion of causal dependency.
- 793 • The CTL check can be used to immediately confirm the
 794 outcome of the slicing algorithm.

795 5.2 Protein signalling networks analysis

796 This case was studied in Ballis et al. (2024), where it was
 797 encoded into the Maude⁴ ecosystem (Clavel et al. 2007) to
 798 take advantage of their built-in LTL and CTL model checker
 799 facilities. It is based on a biological case study from der
 800 Heyde et al. (2014), aimed to identify the best drug treat-
 801 ment for three different breast cancer representative cell
 802 lines: BT474, SKBR3 and HCC1954. This is achieved by

803 ⁴ <https://maude.cs.illinois.edu>.

Fig. 12 Occurrence graph of a major risk



Patient profiles possibly leading to a "major" adverse reaction										
	afib	has_fib	heart_rate	consensus_acei	over75	below55	diabetes	doac_int	hyper	origin
1	TRUE		TRUE	TRUE	FALSE			FALSE	TRUE	
2	TRUE		TRUE	TRUE	FALSE		TRUE	FALSE		
3	TRUE		TRUE	TRUE	TRUE	FALSE		FALSE		
4	TRUE	TRUE		TRUE				FALSE	TRUE	
5	TRUE	TRUE							TRUE	
6	TRUE	TRUE	TRUE							

```

AX((afib      & heart_rate & consensus_acei & !over75          & !doac_int & hyper |
     afib      & heart_rate & consensus_acei & !over75          & diabetes & !doac_int |
     afib      & heart_rate & consensus_acei & over75 & !below55    & !doac_int |
     afib & has_fib           & consensus_acei          & !doac_int & hyper |
     afib & has_fib           & consensus_acei          & !doac_int & hyper |
     afib & has_fib & heart_rate          & !doac_int & hyper |
     <-> EF forbidden)

```

Fig. 13 CTL encoding of the major risk profiles found in Bowles et al., (2024, Fig. 6)

studying the behaviour of the protein signalling networks for the HER2-positive breast cancer subtype in the presence of different combinations of monoclonal antibody drugs. In a nutshell, Maude is a high-performance reflective language and system based on equational and rewriting logic specification. The encoding of RSs is made possible by setting up a specific rewrite theory, called **ccReact**, which is expressive enough to capture the relevant aspects of the protein signalling networks. The analysis conducted in Ballis et al. (2024) matches previous findings, and makes it possible to readily inspect new hypotheses.

Analysis goals. The goal of the analysis is to validate or refute some behavioural hypotheses of RSs.

Features of interest. Besides reachability analysis, mostly concerned with the possibility to reach certain attractors, the distinguishing feature of this case study is the possibility to model check RSs with guarded contexts against behavioural properties written in LTL and CTL.

Experimental set up. The technique in Ballis et al. (2024) starts directly from a RS specification, which is manually coded in **ccReact** and queried using Maude state exploration techniques and built-in model checkers. Likewise, here we can just exploit the direct translation of RSs (with guarded contexts) to GROOVE presented in the previous section, i.e., no preprocessing is necessary. The BioResolve specification is in Fig. 22 in the Appendix. The following properties have been experimented with:

1. searching for the the presence/absence of the attractor *akt* in steady states of the BT747 cell line, where the context [*k*, *ket*] is considered;
2. in order to observe the interactions when either erlotinib or pertuzumab are supplied, the context [{*e*, *egf*, *hrg*} . *korep*] is considered and Maude reports that there exists

at least one path where that treatment is successful, but not all paths avoid a steady state where *akt* is present;

3. using the context [*k*, *korept*], it is shown that, regardless the drug used, once *pdk1* is present, inevitably the steady state includes *akt*; and that *pdk1* never appears before *erbb1* is produced (which basically means that *pdk1* is a product of the activation of the *erbb1* receptor);
4. finally, using the context [*k*, *kge*], it is shown that by permanently providing the drug erlotinib and the stimulus (*egf* and *hrg*), the attractor *akt* is never produced. Moreover, Maude checks that the production of *akt* can be also inhibited by providing erlotinib only when receptors *erbb1* and *erbb2* are active.

Previous approach. **ccReact** allowed to perform reachability analysis directly exploiting the search command of Maude. The formal verification of temporal formulas has been made possible by relying on a general interface to different model checkers for Maude models, called the Unified Maude Model-Checking tool (umaudemc) Rubio et al. (2021). Some examples of verified temporal formulas are those expressing properties such as: *Does there exist at least one path where that treatment is successful? Do all paths prevent reaching a steady state in which a AKT is present?* GROOVE experimentation.

Like Maude, GROOVE has built-in model checking capabilities for both LTL and CTL properties; below, we show how to replicate the results of Ballis et al. (2024), for the four scenarios listed above.

The main challenge in replicating the results is that some of the properties to be checked are formulated in terms of *steady states* of the reaction system, which are essentially one-state attractors, that is, states in which the context and

reactions together reproduce exactly the entities of that state again. Though GROOVE detects such a loop as a matter of course, it is a structural property of the LTS and not a state property available for model checking. In order to be able to reason about steadiness, we have to remember *input entities*, i.e., entities that were present in the source state, and compare them to *present entities*, i.e., those that have been produced in the target state. Moreover, we should not accidentally mark the start state as steady even if it has neither inputs nor present entities. Figure 14 shows the additional rules that achieve this, together with the modified recipe defined by

```
881 recipe fire() {
882   try testStart; markInput; context; react;
883 }
```

The resulting steady condition is given (using quantifier syntax) in Fig. 14c.

- 886 1. Given the context $[k, ket]$, GROOVE confirms the status
887 of the following LTL properties:
 - 888 • $FG(steady \rightarrow akt)$ is not satisfied; GROOVE
889 produces a counter-example.
 - 890 • $G(erbb2 \rightarrow X(erbb2))$ is satisfied.
- 891 2. Given the context $\{e, egf, hrg\}.korep$, GROOVE
892 confirms the status of the following CTL properties:
 - 893 • $EF(steady \& !akt)$ is satisfied;
 - 894 • $AF(steady \& !akt)$ is not satisfied.
- 895 3. Given the context $[k, korept]$, GROOVE confirms the
896 status of the following LTL properties:
 - 897 • $G(pdk1 \rightarrow FG(steady \rightarrow akt))$ is satisfied;
 - 898 • $erbb1 R !pdk1$ is satisfied.
- 900 4. Given the context $[k, kge]$, GROOVE confirms the status
901 of the following CTL property:
 - 902 • $EG EF(steady \rightarrow !akt)$ is satisfied.

However, we want to point out that this property does not actually provide any useful guarantees, because the predicate *steady* (both in Ballis et al. (2024) and in our encoding explained above) only tests for *single-state* attractors. If the reaction system ends up in a multi-state loop, *steady* will never be satisfied and hence the implication $steady \rightarrow !akt$ is *always* satisfied, regardless of whether or not *akt* holds. Indeed, the state space of this scenario, visually reproduced in Fig. 15, has such a multi-state attractor (consisting of states *s15* and *s18*); in both of those states the predicate *akt* holds, yet the state space as a whole satisfies the CTL property above.

In replicating the results from Ballis et al. (2024), we have had to make a few adjustments. The LTL formulas reported in Ballis et al., (2024, Page 14) for scenarios 1 and 3 are actually *not* literally the ones above, but use the predicate *io-state* rather than *isSteady*. We believe that the use of *isSteady* (or, in our case *steady*) is more informative and probably the intended version.

As a final observation, we note that the numbers of states in all these scenarios is actually quite small. We have already shown the 6-state scenario 4 in Fig. 15; the size of the others is given by the following table.

Nr.	Context	States
1	$[k, ket]$	4
2	$\{e, egf, hrg\}.korep$	10
3	$[k, korept]$	32
4	$[k, kge]$	6

Discussion.

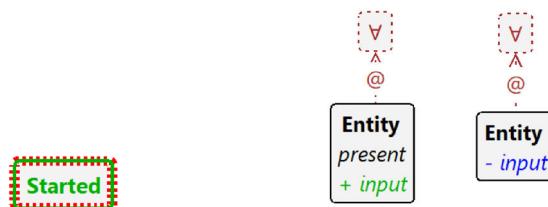
Compared to the prior results in Ballis et al. (2024), the advantages of using GROOVE lie in the combination of visual inspection and automatic model checking. Not only were we able to confirm the findings of the original paper using exactly the same encoding of reaction system as for the previous case, but the ability to inspect and visualise the state spaces also gives additional insights, such as the observation above that steadiness as formalised there does not actually capture the intended notion of being an attractor.

5.3 T cell differentiation analysis

The paper (Brodo et al. 2025b) (being the full and corrected version of Brodo et al. (2025a), see also Footnote 5) exploits Reaction Systems to analyse T cell differentiation in the immune system, a widely studied biological phenomenon. The starting point for the analysis is a Boolean network model; several of those are available as a Saez-Rodriguez et al. (2007); Thakar and Albert (2010); Puniya et al. (2018), among which the one in Puniya et al. (2018) was selected. The model encompasses reactions enabling T cells to manifest various phenotypes in response to environmental stimuli, and describes a realistic regulation system that is involved in many diseases (Lafaille 1998; Hirahara and Nakayama 2016; Meng et al. 2016).

The Boolean network model is graphically represented as shown in Fig. 16, where the 9 orange nodes represent different environmental stimuli that the T cell can receive; all the other nodes represent so-called *transcription factors* and have an associated Boolean update formula that specifies when they are triggered. T cells can manifest four phenotypes, represented by the four transcription factors *tbet*, *gata3*,

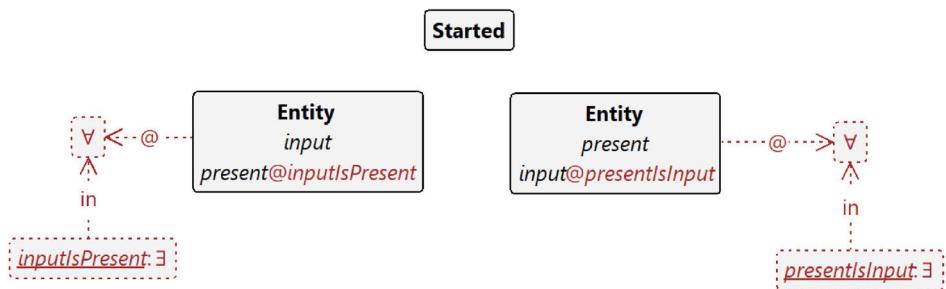
Fig. 14 Additional rules and condition for detecting steadiness



(a) Rule
testStart

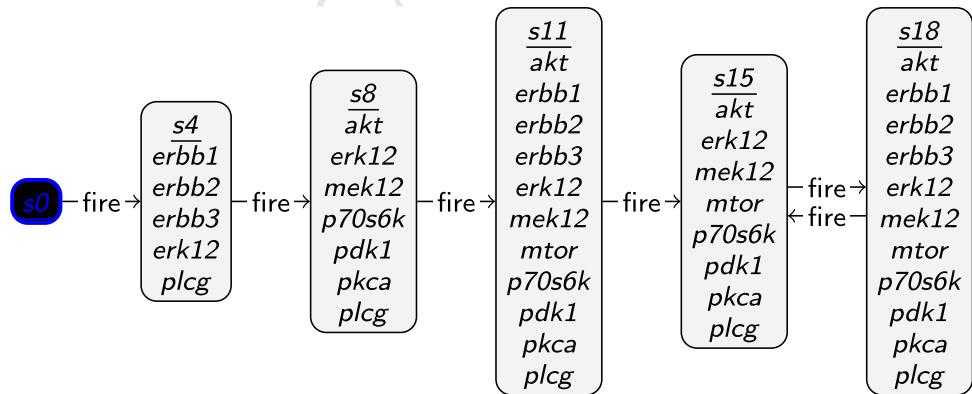
(b) Rule
markInput

Tests if simulation has started and the set of input entities equals the set of present entities



(c) Condition steady

Fig. 15 GTS for cancer scenario 4



957 rorgt and foxp3, respectively. There exists experimental
958 and computational evidence that a T cell can manifest
959 more than one phenotype (Luckheeram et al. 2012; Puniya
960 et al. 2018).

Analysis goals. The reachability analysis must take into account the different combinations of phenotypes that a T cell can express, called a *target* (hence, $2^4 = 16$ targets overall). For example, for the target containing the combination of transcription factors {tbet, gata3}, we must select an attractor that includes at least one state in which tbet is present, at least one state (possibly the same) in which gata3 is present, and no state in which either rorat or

foxp3 are present. The causal analysis aims to collect the combinations of environmental stimuli that are responsible for leading to that target.

Features of interest. We have selected this case study because it shows the applicability of our method to Boolean networks models, like those available in the public database on the CellCollective platform (Helikar et al. 2012). For these models, the most relevant viewpoints are often reachability (e.g., *which phenotypes are reachable?*) and causality (*what is the effect of environmental conditions?*) analyses. Their corresponding RSs always use a special kind of nondeterministic persistent context, where at the beginning of the experiment

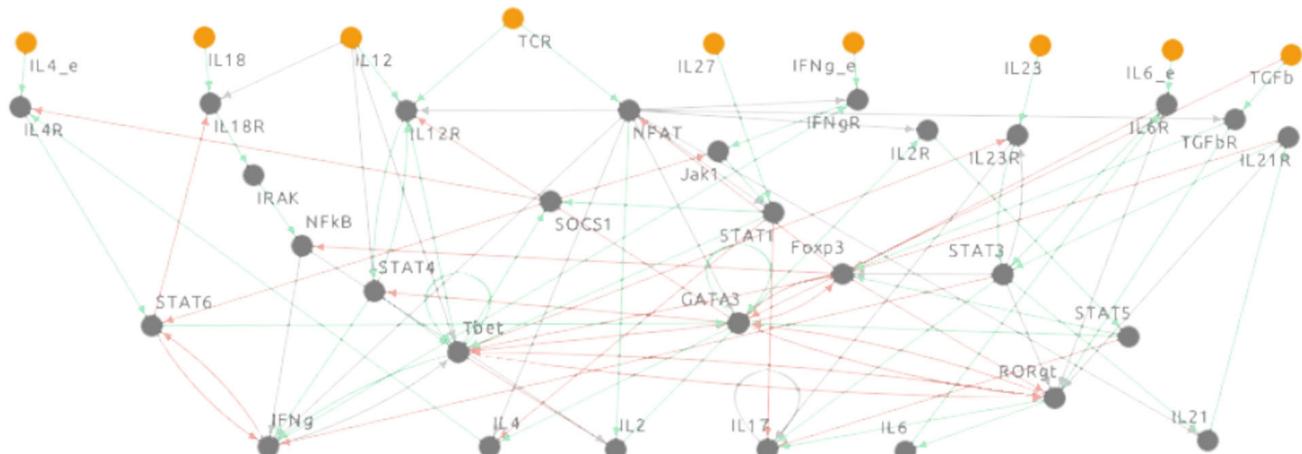


Fig. 16 Graphical representation of the Boolean network model of T cell differentiation from Puniya et al. (2018)

a subset of external stimuli is chosen and then provided at each subsequent step, inevitably causing the RS to end up in a loop (called an attractor).

Experimental set up.

The translation from Boolean networks to RS consists in turning every update formula into disjunctive normal form. Then, every clause of the disjunction produces a reaction in which (i) reactants are the positive atoms, (ii) inhibitors are the negated atoms and (iii) the updated variable forms a singleton product. The translation from Boolean network to BioResolve syntax is done using the directive `main_do(bn2rs)`. For the readers' convenience, all update formulas and the resulting reactions are reported, respectively, in Fig. 22 and in Fig. 23 in the Appendix. For example, the update formula for `IL12R` is

$$\begin{aligned} & (\text{IL12\&NFAT}) \mid (\text{STAT4\&\neg GATA3}) \\ & \mid \text{Tbet} \mid (\text{TCR\&\neg GATA3}) \end{aligned}$$

which yields the four reactions⁵

$$\begin{aligned} & ((\text{il12}, \text{nfat}), \emptyset, \{\text{i112r}\}) \quad ((\text{stat4}), \{\text{gata3}\}, \{\text{i112r}\}) \\ & ((\text{tbet}), \emptyset, \{\text{i112r}\}) \quad ((\text{tcr}), \{\text{gata3}\}, \{\text{i112r}\}) \end{aligned}$$

The RS context can choose any combination of environmental stimuli that will then persist, i.e., for each possible stimulus s we define the context processes $X_s \triangleq \{s\} \cdot X_s$ and then take the context $\prod_s (X_s + \text{Emp})$. The resulting LTS has

an initial branching into 2^9 different states, because there are 9 possible stimuli to be considered. Subsequently, each of the 2^9 states originates a deterministic computation, leading to some attractors.

Previous approach. The paper (Brodo et al. 2025b) presents a toolchain (BioResolve, SWI-Prolog, Python and Python-to-Prolog binding facilitated by the swiplserver Python package) to study the Boolean network model. Roughly, after translating the Boolean network model to RS specifications the whole LTS is constructed according to any combination of persistent stimuli that can be provided by the context. BioResolve returns the LTS as a graph in dot format, which is then loaded by a Python script. Then, attractors related with a target of interest are identified by looking for cycles in the LTS, and a slicing algorithm performs some form of causal analysis, to simplify each computation trace by preserving only the relevant causes of those target entities. The generation of the LTS is often the bottleneck of the approach, both in terms of time (Prolog performance), but also in terms of space, because BioResolve can require to allocate a large stack limit size to succeed.

GROOVE experimentation.

The capabilities of GROOVE called upon for this case study are very similar to those in Sect. 5.1; the main difference lies in the specific interest in attractors. Indeed, in contrast to the situation for comorbidities, here after the initial selection of a profile, the context does not cause any more nondeterminism; hence every profile eventually ends up in such an attractor.

A trace ending in a loop, sometimes called a “lollipop”, is in fact precisely what LTL properties are checked over; hence an LTL property violation takes the form of a lollipop. This means that we can find attractors with specific properties by formulating their non-existence in LTL, and then finding a counter-example through model checking. For instance, the

⁵ The specification analysed in Brodo et al. (2025b), on which this paper is based, repairs a minor typo in the original conference version (Brodo et al. 2025a). Specifically, the product set of the reaction `(stat4,gata3,i112r)` was mistakenly written as `i12r`, omitting the digit 1. Unfortunately, since `i12r` was also a valid entity, the error was difficult to detect. Though the mistake mildly affected the original results, it turns out that for the analysis reported in this paper there is no difference at all.:

1039 following formulas deny the reachability of an attractor in
 1040 which *tbet* and *gata3* are expressed:

- 1041 • $\neg G(F \text{ gata3} \wedge F \text{ tbet})$ (separate expression)
 1042 • $\neg GF(\text{gata3} \wedge \text{tbet})$ (simultaneous expression)

1043 Using the start graph derived from BioResolve using the
 1044 process outlined in Fig. 5, both of these formulas yield counterexamples, meaning that *tbet* and *gata3* can in fact be
 1045 (recurrently) expressed simultaneously. Using a variation of
 1046 the process outlined in Sect. 4, we can once more visualise a
 1047 trace leading to such a recurrent state. The variation lies in the
 1048 fact that, this time, we do not want to show the causal history
 1049 of a *single* forbidden entity, but rather of the combination of
 1050 two distinct entities. Fortunately, this is just a matter of cre-
 1051 ating another rule, *gata3-tbet*, which applies precisely when
 1052 *gata3* & *tbet* holds. On this basis we can go through the
 1053 steps outlined in Fig. 5, resulting in the occurrence graph
 1054 displayed in Fig. 17.

1055 This complements the observation embodied in Brodo et
 1056 al., (2025b, Fig. 7) that for the combination of *tbet* and
 1057 *gata3*, the context has to provide the stimuli *ifnge* (pro-
 1058 duced here by the **StepOcc** named *x81_0*), *tcr* (repeatedly
 1059 produced by **StepOccs** named *x91_0*) and *i14e* (also
 1060 repeatedly produced, by **StepOccs** named *x51_0*). In more
 1061 detail, we see that *tbet* derives, in a linear sequence
 1062 of four **ReactionOccs**, from *ifnge* and *tcr*, whereas
 1063 *gata3* derives, in 3 successive combinations of simulta-
 1064 neous **ReactionOccs**, from *tcr* and *i14e*. Moreover, the
 1065 genes produced along the way are precisely the ones reported
 1066 in Brodo et al., (2025b, Fig. 8(a)), using the slicing algorithm
 1067 of that paper, as being relevant for the expression of *tbet*
 1068 and *gata3*.

1070 Besides the production of such occurrence graphs for spe-
 1071 cific cases, GROOVE can also be used once more to directly
 1072 confirm the findings of Brodo et al., (2025b, Figs. 7 and 8), by
 1073 expressing them as CTL formulas similar to the one reported
 1074 in Fig. 13. In this context, it is relevant to report that, in con-
 1075 trast to BioResolve, where (as reported above) time and space
 1076 performance were a bottleneck in the analysis of this model,
 1077 GROOVE can fully explore the state space in approximately
 1078 3 s.

1079 Discussion

1080 Compared to the results in Brodo et al. (2025b), the ad-
 1081 vantages of GROOVE are threefold (reiterating the observations
 1082 made for the previous two cases):

- 1083 • LTL model checking allows to express, in a flexible man-
 1084 ner, the scenarios one wants to investigate;
 1085 • The occurrence graph visualisation offers analysis pos-
 1086 sibilities beyond the outcome of the slicing algorithm;

- 1087 • The performance of GROOVE is an order of magnitude
 1088 better than that of BioResolve

6 Comparison with existing tools

1089 The approach presented in this paper can provide several
 1090 advantages over existing tools in the literature.⁶

- 1091 • **brsim**⁷ (Basic Reaction System Simulator, written in
 1092 Haskell and distributed under the terms of GNU GPLv3
 1093 license) Azimi et al. (2015) was the first RS simulator to
 1094 be made publicly available. Given the reactions of the RS
 1095 and a context sequence, **brsim** is capable of computing
 1096 the resulting sequence and generating additional anno-
 1097 tations for each computation step, such as the enabled
 1098 reactions. Alternatively, **brsim** can be executed in an
 1099 interactive mode, allowing the user to manually provide
 1100 the context to be used at each step.
- 1101 • WebRSim⁸ is a basic RS simulator that makes all func-
 1102 tionalities of **brsim** available through a friendly web
 1103 interface (Ivanov et al. 2018);
- 1104 • HERESY⁹ is a Highly Efficient REaction SYstem GPU-
 1105 based simulator, developed using CUDA (Nobile et al.
 1106 2017). It features a user-friendly GUI and is designed to
 1107 exploit the high degree of parallelism offered by modern
 1108 GPUs to handle very large-scale RSs simulations.
- 1109 • cl-rs¹⁰ is an optimised Common Lisp simulator for
 1110 RSs (Ferretti et al. 2020) that can exhibit performances
 1111 comparable with the GPU-based simulator HERESY. This
 1112 is achieved by discarding all reactions that cannot pro-
 1113 duce effects and by encoding RS evolution in terms of
 1114 matrix–vector multiplications and vector additions.
- 1115 • BioResolve¹¹ is a Prolog interpreter for Reaction Sys-
 1116 tem analysis, first proposed in Brodo et al. (2021) and
 1117 later extended in a series of papers to deal with enhanced
 1118 features, like delays, duration, monitoring, slicing and
 1119 guarded contexts (Brodo et al. 2023, 2024a; Bowles et al.
 1120 2024). Many capabilities of BioResolve have been dis-
 1121 cussed at length in the previous sections.
- 1122 • ccReact¹² is an interacting language for Reaction Sys-
 1123 tems based on Maude 3.2.1 (Ballis et al. 2024), whose
 1124 key features have been illustrated in Sect. 5.2.

6 See, e.g., the list of Reaction Systems Computer Environments at <https://www.reactionsystems.org/about-reaction-systems>.

7 Available at <https://github.com/scolobb/brsim/>.

8 Available at <https://github.com/scolobb/brsim>.

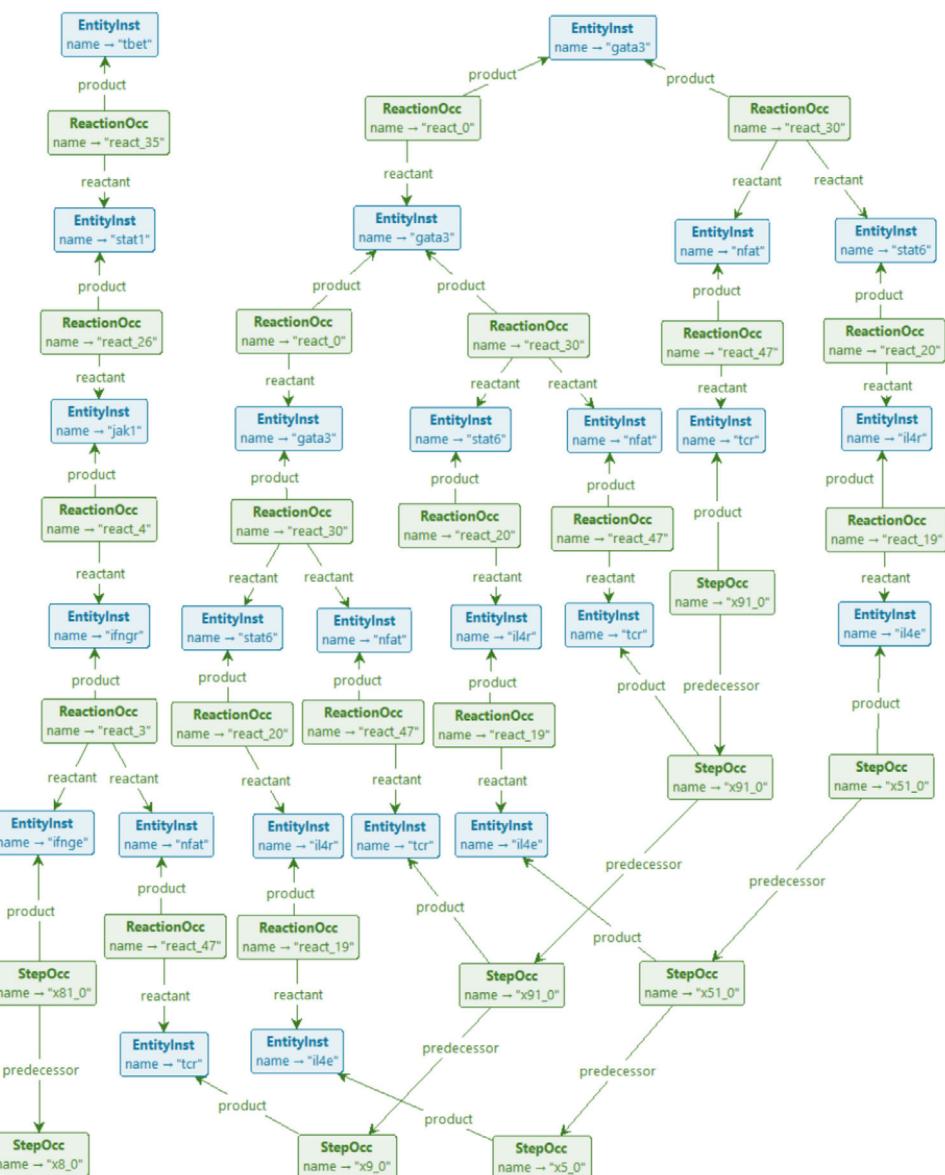
9 Available at <https://github.com/aresio/HERESY>.

10 Available at <https://github.com/mnzuLuca/cl-rs>.

11 Available at <https://www.di.unipi.it/~bruni/LTSRS/>.

12 Available at <https://depot.lipn.univ-paris13.fr/olarte/reaction-systems-maude>.

Fig. 17 Occurrence graph for the simultaneous expression of gata3 and tbet



- ReactICS¹³ is a Reaction Systems Verification Toolkit that consists of two main modules for model checking temporal properties expressed in logical languages tailored to Reaction Systems: one that exploits binary decision diagrams (BDD) and bounded model checking; the other that translates verification problems in rsLTL (Meski et al. 2015) into satisfiability modulo theories (SMT) (Meski et al. 2024).

Modelling capabilities. The GROOVE-based method supports a rich and expressive encoding of RSs, including the most recent features such as the handling of *guarded*, *recursive*, and *nondeterministic contexts*. Among the other tools, such features are only supported by BioResolve, which,

however, relies on a Prolog back-end that limits scalability and requires external scripting for improving the performance of many analyses whenever large state generation and exploration is necessitated. Tools such as HERESY, WebR-Sim, and cl-rs provide lightweight RS simulators but are limited to basic semantics, lacking support for more advanced interactions with the context or advanced verification features. ccReact supports temporal logic model checking (LTL/CTL), but not recursive contexts. Moreover, the encoding of RSs in ccReact is manual and less suited to visual inspection or dynamic causal analysis. In addition to (bounded) model checking of custom temporal logics specifically designed for Reaction Systems, ReactICS also supports context automata (a slightly less general notion of contexts than the one considered here), reactions with concentration

¹³ Available at <https://github.com/arturmeski/reactics/>.

1154 levels, parameter and reaction synthesis, as well as parameter
 1155 optimisation.

1156 *Performance and scalability.* The ability of GROOVE to
 1157 explore large state spaces efficiently is central to our method.
 1158 Through configurable exploration strategies and a rule-based
 1159 control mechanism, GROOVE handles complex RS instances
 1160 that involve thousands of reachable configurations. Our
 1161 experiments demonstrate a substantial improvement in anal-
 1162 ysis time compared to BioResolve, often reducing execution
 1163 time by an order of magnitude. Furthermore, the performance
 1164 of BioResolve is strongly influenced by the nature of Prolog
 1165 evaluation strategies, which can lead to excessive memory
 1166 and time consumption in large case studies. With respect to
 1167 **ccReact**, the paper (Ballis et al. 2024) on which we based our
 1168 experiments does not provide information on performance,
 1169 and indeed the system studied there is so small that no use-
 1170 ful comparison could be made on that basis. Since Maude,
 1171 underlying **ccReact**, is a long-standing and mature tool, it
 1172 would be interesting to investigate this in more detail; how-
 1173 ever, we leave this to future work. In contrast, other tools
 1174 either consider linear executions only and do not scale to
 1175 large models or lack optimisation strategies necessary for
 1176 handling non-trivial state spaces.

1177 Contrary to GROOVE, which is a general purpose, explicit-
 1178 state tool, not optimised for Reaction Systems, ReactICS can
 1179 take advantage of symbolic representations to abstract much
 1180 larger state-spaces. For instance, for the mutual exclusion
 1181 protocol reported in Meski et al. (2015), Nobile et al. (2017),
 1182 ReactICS can cope with 37 to 54 processes (depending on the
 1183 precise analysis) in a matter of hours; in contrast, in compar-
 1184 able time GROOVE can exhaustively generate the state space of
 1185 up to 13 processes. Even though the model checking queries
 1186 in ReactICS probably do not require full state space explo-
 1187 ration, and hence the comparison is skewed, the dedicated
 1188 methods of ReactICS clearly pay off.

1189 *Causal analysis and verification.* A key distinguishing fea-
 1190 ture of our approach is the ability to perform graph-based
 1191 *causal slicing*. By automatically generating and pruning
 1192 *occurrence graphs*, GROOVE provides detailed and visual
 1193 explanations of how specific states, such as those involving
 1194 undesirable or forbidden entities, are reached. This form of
 1195 causal reasoning is not available in high-performance tools
 1196 such as HERESY, WebRSim, cl-rs, or ReactICS and is
 1197 only partially addressed in **ccReact**, where the focus is pri-
 1198 marily on reachability and temporal properties. GROOVE’s
 1199 integrated support for CTL and LTL model checking further
 1200 extends its applicability to behavioural verification, enabling
 1201 the specification and validation of complex temporal proper-
 1202 ties.

1203 *Summary.* In conclusion, the combination of expressive mod-
 1204 elling, efficient state space exploration, and integrated causal
 1205 analysis makes GROOVE a powerful and versatile full-fledged
 1206 platform for the study of Reaction Systems. It not only gen-

1207 eralises and extends existing tools, but also opens the door
 1208 to new forms of analysis that were previously impractical or
 1209 unsupported.

7 Conclusion and future work

1210 In this work, we have demonstrated how Reaction Systems
 1211 can be effectively encoded and analysed within the GROOVE
 1212 framework, so to reuse the expressiveness and efficiency
 1213 of graph transformation techniques. By exploiting quanti-
 1214 fied rules, the encoding consists of a direct translation from
 1215 RS specification to a typed graph, which is made auto-
 1216 matic in BioResolve. Then, GROOVE enables both exhaustive
 1217 state space exploration, the extraction of causal information
 1218 through occurrence graphs and property-based verification
 1219 based on model checking.

1220 We have used GROOVE to revisit several case studies from
 1221 the literature and our experimental results, although prelim-
 1222 inary, are promising: GROOVE not only supports complex
 1223 RS features such as guarded, nondeterministic and recur-
 1224 sive contexts but also significantly improves performance
 1225 and flexibility compared to existing RS tools. Moreover, the
 1226 use of GROOVE’s recipes and model-checking capabilities
 1227 opens the door to sophisticated analyses that were previously
 1228 impractical.

1229 A number of interesting avenues for future research
 1230 remain open, among which we mention the possibility to
 1231 extend the methodology to support quantitative RS variants
 1232 with durations or weights (Brodo et al. 2023); investi-
 1233 gate alternative notions of causality and their representation
 1234 in graph-based semantics, like dependencies drawn from
 1235 inhibitors rather than reactants¹⁴; apply the GROOVE toolset
 1236 to further biological case studies, like those available in the
 1237 CellCollective public repository (Helikar et al. 2012), possi-
 1238 bly exploiting an automated pipeline for analysis.

1239 Overall, the results show that graph transformation, and
 1240 GROOVE in particular, provide a robust and scalable founda-
 1241 tion for the specification, execution, and analysis of Reaction
 1242 Systems.

1243 One option that we have ignored throughout the paper
 1244 deserves a brief mention here. The slicing algorithms
 1245 reported in (Bowles et al. 2024; Brodo et al. 2025a) on the
 1246 basis of BioResolve are actually implemented as stand-alone
 1247 scripts that operate on a .dot-formatted LTS. Given that
 1248 GROOVE can also produce LTSs as .dot files, an alterna-
 1249 tive way to benefit from its superior performance might be
 1250 merely to replace BioResolve in the tool chain used previ-
 1251 ously. For this to be possible, the LTS labelling information

¹⁴ In this respect, we could, e.g., exploit the semantic-preserving trans-
 formation from RSs to Positive RSs proposed in Brodo et al. (2024b).

produced by GROOVE should conform to the requirements of the slicing algorithm, following the principles outlined in Remark 1. Though that is currently not the case (compare GROOVE's Fig. 8 to BioResolve's Fig. 4), we believe that this requires only a minor adjustment of the rule system.

From the perspective of GROOVE development, carrying out the experiments described in this paper has led to many small improvements as well as inspiration for new features. For instance, the various strategies for "ordinary" state space exploration (DFS- or BFS based, bounded or not, conditional or not) on the one hand and the model checking capabilities on the other are not well-integrated. Queries such as "count the number of states of max depth n where a given CTL property holds" or "find all prefixes of length n of paths satisfying a given LTL property," which would enhance the capabilities for analysing graph-based models such as the ones studied here, currently cannot be posed in a straightforward manner. Another useful extension would be the ability to automatically chain different transformations, such as the three steps of explore-build-prune in Fig. 5, which currently have to be invoked separately. We plan to investigate these extensions in the future.

Supplementary information

For replication of the GROOVE experiments, we provide an archive with supplementary material, described in Sect. C.5.

Appendix A Semantic correspondence

In this paper, we claim, but do not prove, that the GROOVE encoding correctly captures the operational Reaction Systems semantics recalled in Sect. 2.1. A complete formal proof is outside the scope of this paper, which focusses on experimental results; however, in this section we provide a sketch of such a proof.

As described in Sect. 4, each GROOVE state is a graph consisting of a fixed part (which is the same in every state) and a variable part. The fixed part contains an encoding of (i) the reaction system A itself, with the entities S as **Entity**-typed nodes, and (ii) the automata of the context processes, with individual processes corresponding to **Token**-typed nodes and states as **State**-typed nodes.

The variable part consisting of *present* flags on **Entity** nodes and a current-edge from every **Token** to a **State**. As observed in Remark 1, any RS process $[M]$ can be written

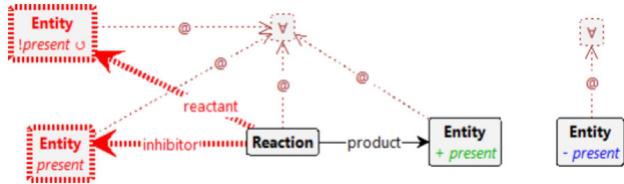


Fig. 18 Rule for reaction firing

in the form $[Rs \mid Ks \mid D]$ where: (i) $Rs = \prod_i (R_i, I_i, P_i)$ is the parallel composition of all reactions in the system, and never changes along the computation; (ii) $Ks = \prod_j K_j$ is the parallel composition of all contexts, and (iii) D is the set of currently present entities.

A GROOVE state G is equivalent to an RS process $[Rs \mid Ks \mid D]$ if: (i) an entity e is present in D if and only if its **Entity** node is flagged as *present* in G , and (ii) there is a one-to-one correspondence of the available contexts K_j in the RS process and the **Token**-nodes in G , such that the current state of that **Token** is the start state of the automaton for K_j .

We claim that this equivalence establishes a bisimulation between the GROOVE and RS state spaces. (It is not an isomorphism, because the RS semantics for K "consumes" the process (rules $(Cxt)-(Rec)$ in Fig. 2) whereas G keeps the automaton intact, merely moving the current pointer.)

To prove this claim, we have to show correspondence of the transitions. Concretely, to any transition carrying the label $\langle\langle D \triangleright R', I', C \rangle\rangle \triangleright R, I, P$, there corresponds a single application of the recipe **fire**, which in turn consists of applications of rules **context** (see Fig. 7) followed by **react** (see Fig. 18).

Both of these rules are universally quantified. Roughly speaking, referring to Fig. 2, context simultaneously encodes all sub-transitions of the form

$$\begin{array}{c} D \xrightarrow{\langle\langle D \triangleright \emptyset, \emptyset, \emptyset \rangle\rangle \triangleright \emptyset, \emptyset, \emptyset} D' \quad (\text{rule } (Ent)) \\ K \xrightarrow{\langle\langle \emptyset \triangleright R', I', C \rangle\rangle \triangleright \emptyset, \emptyset, \emptyset} K' \quad (\text{rules } (Cxt)-(Rec)) \end{array}$$

as well as their composition and filtering (rules $(Par)-(Sys)$). **react** in turns simultaneously encodes all sub-transitions of the form

$$(R, I, P) \xrightarrow{\langle\langle \emptyset \triangleright \emptyset, \emptyset, \emptyset \rangle\rangle \triangleright R', I', P'} M' \quad (\text{rules } (Pro)-(Inh))$$

Table 1 State space generation for mutual exclusion

Case #	States #	Ratio	Time s	Ratio	Memory MB	Ratio
2	11		0.2		1	
3	27	2.5	0.4	1.9	2	2.2
4	63	2.3	0.7	2.0	4	2.1
5	143	2.3	2.0	2.8	13	3.0
6	319	2.2	5.6	2.7	34	2.6
7	703	2.2	17.5	3.2	96	2.8
8	1535	2.2	56.6	3.2	262	2.7
9	3327	2.2	175.0	3.1	700	2.7
10	7167	2.2	595.3	3.4	1929	2.8
11	15,359	2.1	1890.3	3.2	4764	2.5
12	32767	2.1	5146.1	2.7	8588	1.8
13	69631	2.1	15,647.3	3.0	13,640	1.6

1325 together with their composition with the context-transitions
 1326 (rule (*Par*)) and filtering (rule (*Sys*)). (The *fired*-flags on the
 1327 **Step**-nodes, added by context and removed again by react,
 1328 play no role in this correspondence: for the purpose of the
 1329 equivalence of the GROOVE and RS semantics, they might
 1330 be omitted entirely.)

Appendix B Mutual exclusion

1332 Table 1 shows the GROOVE performance in generating the
 1333 full state space of the mutual exclusion example of Meski
 1334 et al. (2015), Nobile et al. (2017), discussed in Sect. 6.

```

myentities([cpowder,tpowder]). % initial set D0

myreactions([
    react([idle],[am],[am]),
    react([am],[idle],[am]),
    react([ccoin,cpowder],[nomilk],[cappuccino]),
    react([ccoin,cpowder,nomilk],[],[espresso]),
    react([tcoin,tpowder],[],[tea]),
    react([cpowder],[],[cpowder]),
    react([tpowder],[],[tpowder]),
    react([anger],[],[bang]) ]).

mycontext("refill,student"). % context processes

myenvironment([
    refill = ({nomilk}.refill
               + {}.{refill}),
    student = (?{},{}.{am},{tcoin}?.gettea
               + ?{am},{}.{ccoin}?.getcappuccino
               + {idle}.student),
    gettea = (?{tea},{}.,{}?.student
               + ?{},{}.{anger}?.student),
    getcappuccino = (?{cappuccino},{}.,{}?.student
                     + ?{espresso},{}.,{}.{anger}?.student) ]).
  
```

Fig. 19 BioResolve implementation of the vending machine RS from Sect. 3. The question marks ? are used to delimit guarded prefixes in context processes

Appendix C Auxiliary material

C.1 Auxiliary material for the toy running example

The BioResolve specification for the toy running example about the interaction between the student and the vending machine is reported in Fig. 19. The corresponding RS has been described in Sect. 3 and it has been used to illustrate some key features of the GROOVE encoding in Sect. 4.

Author Proof

Feats \triangleq $\{(\{\text{hyper}\}, \emptyset, \{\text{hyper}\}) | (\{\text{afib}\}, \emptyset, \{\text{afib}\}) | (\{\text{has_fib}\}, \emptyset, \{\text{has_fib}\}) | (\{\text{heart_rate}\}, \emptyset, \{\text{heart_rate}\}) | (\{\text{consensus_acei}\}, \emptyset, \{\text{consensus_acei}\})$
 $| (\{\text{over75}\}, \emptyset, \{\text{over75}\}) | (\{\text{below55}\}, \emptyset, \{\text{below55}\}) | (\{\text{diabete}\}, \emptyset, \{\text{diabete}\}) | (\{\text{origin}\}, \emptyset, \{\text{origin}\})$
 $| (\{\text{doac_int}\}, \emptyset, \{\text{doac_int}\}) | (\{\text{hyper}\}, \emptyset, \{\text{diseases}\}) | (\{\text{diabete}\}, \emptyset, \{\text{diseases}\})$

Drugs \triangleq $\{(\{\text{get_diltiazem}\}, \{\text{stop_cbb}\}, \{\text{diltiazem, cbb}\}) | (\{\text{diltiazem}\}, \{\text{stop_cbb}\}, \{\text{diltiazem, cbb}\}) | (\{\text{get_verapamil}\}, \{\text{stop_cbb}\}, \{\text{verapamil, cbb}\})$
 $| (\{\text{verapamil}\}, \{\text{stop_cbb}\}, \{\text{verapamil, cbb}\}) | (\{\text{diltiazem, verapamil}\}, \{\text{stop_cbb}\}, \{\text{alert_dup}\}) | (\{\text{get_propranolol}\}, \{\text{stop_nsbb}\}, \{\text{propranolol, nsbb}\})$
 $| (\{\text{propranolol}\}, \{\text{stop_nsbb}\}, \{\text{propranolol, nsbb}\}) | (\{\text{get_carvedilol}\}, \{\text{stop_nsbb}\}, \{\text{carvedilol, nsbb}\}) | (\{\text{carvedilol}\}, \{\text{stop_nsbb}\}, \{\text{carvedilol, nsbb}\})$
 $| (\{\text{propranolol, carvedilol}\}, \{\text{stop_nsbb}\}, \{\text{alert_dup}\}) | (\{\text{get_bisoprolol}\}, \{\text{stop_sbb}\}, \{\text{bisoprolol, sbb}\}) | (\{\text{bisoprolol}\}, \{\text{stop_sbb}\}, \{\text{bisoprolol, sbb}\})$
 $| (\{\text{get_atenolol}\}, \{\text{stop_sbb}\}, \{\text{atenolol, sbb}\}) | (\{\text{atenolol}\}, \{\text{stop_sbb}\}, \{\text{atenolol, sbb}\}) | (\{\text{bisoprolol, atenolol}\}, \{\text{stop_sbb}\}, \{\text{alert_dup}\})$
 $| (\{\text{get_flecainide}\}, \{\text{stop_flec}\}, \{\text{flecainide}\}) | (\{\text{flecainide}\}, \{\text{stop_flec}\}, \{\text{flecainide}\}) | (\{\text{get_warfarin}\}, \{\text{stop_warf}\}, \{\text{warfarin}\})$
 $| (\{\text{warfarin}\}, \{\text{stop_warf}\}, \{\text{warfarin}\}) | (\{\text{get_apixbaban}\}, \{\text{stop_doac}\}, \{\text{apixbaban, doac}\}) | (\{\text{apixbaban}\}, \{\text{stop_doac}\}, \{\text{apixaban, doac}\})$
 $| (\{\text{get_dabigatran}\}, \{\text{stop_doac}\}, \{\text{dabigatran, doac}\}) | (\{\text{dabigatran}\}, \{\text{stop_doac}\}, \{\text{dabigatran, doac}\}) | (\{\text{apixbaban, dabigatran}\}, \{\text{stop_doac}\}, \{\text{alert_dup}\})$
 $| (\{\text{get_vkant}\}, \{\text{stop_vkant}\}, \{\text{vkant}\}) | (\{\text{vkant}\}, \{\text{stop_vkant}\}, \{\text{vkant}\}) | (\{\text{get_benazepril}\}, \{\text{stop_acei}\}, \{\text{benazepril, acei}\})$
 $| (\{\text{benazepril}\}, \{\text{stop_acei}\}, \{\text{benazepril, acei}\}) | (\{\text{get_captoril}\}, \{\text{stop_acei}\}, \{\text{captoril, acei}\}) | (\{\text{captoril}\}, \{\text{stop_acei}\}, \{\text{captoril, acei}\})$
 $| (\{\text{benazepril, captoril}\}, \{\text{stop_acei}\}, \{\text{alert_dup}\}) | (\{\text{get_olmesortan}\}, \{\text{stop_arb}\}, \{\text{olmesortan, arb}\}) | (\{\text{olmesortan}\}, \{\text{stop_arb}\}, \{\text{olmesortan, arb}\})$
 $| (\{\text{get_irbesartan}\}, \{\text{stop_arb}\}, \{\text{irbesartan, arb}\}) | (\{\text{irbesartan}\}, \{\text{stop_arb}\}, \{\text{irbesartan, arb}\}) | (\{\text{olmesortan, irbesartan}\}, \{\text{stop_arb}\}, \{\text{alert_dup}\})$
 $| (\{\text{get_indapamide}\}, \{\text{stop_td}\}, \{\text{indapamide, td}\}) | (\{\text{indapamide}\}, \{\text{stop_td}\}, \{\text{indapamide, td}\}) | (\{\text{get_chlorothiazide}\}, \{\text{stop_td}\}, \{\text{chlorothiazide, td}\})$
 $| (\{\text{chlorothiazide}\}, \{\text{stop_td}\}, \{\text{chlorothiazide, td}\}) | (\{\text{indapamide, chlorothiazide}\}, \{\text{stop_td}\}, \{\text{alert_dup}\}) | (\{\text{doac}\}, \{\text{doac_ok, doac_fail}\}, \{\text{doac_test}\})$
 $| (\{\text{doac_ok}\}, \{\text{doac_fail}\}, \{\text{doac_ok}\}) | (\{\text{doac_fail}\}, \{\text{doac_ok}\}, \{\text{doac_fail}\}) | (\{\text{doac}\}, \{\text{doac_fail, stop_doac}\}, \{\text{doac_danger}\})$
 $| (\{\text{doac}\}, \{\text{doac_danger, stop_doac}\}, \{\text{danger}\})$

ADR \triangleq $\{(\{\text{get_apixbaban, get_diltiazem}\}, \emptyset, \{\text{moderate}\}) | (\{\text{get_apixbaban, diltiazem}\}, \emptyset, \{\text{moderate}\}) | (\{\text{apixbaban, get_diltiazem}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{apixbaban, diltiazem}\}, \emptyset, \{\text{moderate}\}) | (\{\text{get_apixbaban, get_verapamil}\}, \emptyset, \{\text{moderate}\}) | (\{\text{get_apixbaban, verapamil}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{apixbaban, get_verapamil}\}, \emptyset, \{\text{moderate}\}) | (\{\text{apixbaban, verapamil}\}, \emptyset, \{\text{moderate}\}) | (\{\text{get_dabigatran, get_diltiazem}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{get_dabigatran, diltiazem}\}, \emptyset, \{\text{moderate}\}) | (\{\text{dabigatran, get_diltiazem}\}, \emptyset, \{\text{moderate}\}) | (\{\text{dabigatran, diltiazem}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{get_dabigatran, get_verapamil}\}, \emptyset, \{\text{major}\}) | (\{\text{get_dabigatran, verapamil}\}, \emptyset, \{\text{major}\}) | (\{\text{dabigatran, get_verapamil}\}, \emptyset, \{\text{major}\})$
 $| (\{\text{dabigatran, verapamil}\}, \emptyset, \{\text{major}\}) | (\{\text{get_dabigatran, get_carvedilol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{get_dabigatran, carvedilol}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{dabigatran, get_carvedilol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{dabigatran, carvedilol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{get_warfarin, get_benazepril}\}, \emptyset, \{\text{minor}\})$
 $| (\{\text{get_warfarin, benazepril}\}, \emptyset, \{\text{minor}\}) | (\{\text{warfarin, get_benazepril}\}, \emptyset, \{\text{minor}\}) | (\{\text{warfarin, benazepril}\}, \emptyset, \{\text{minor}\})$
 $| (\{\text{get_warfarin, get_indapamide}\}, \emptyset, \{\text{minor}\}) | (\{\text{get_warfarin, indapamide}\}, \emptyset, \{\text{minor}\}) | (\{\text{warfarin, get_indapamide}\}, \emptyset, \{\text{minor}\})$
 $| (\{\text{warfarin, indapamide}\}, \emptyset, \{\text{minor}\}) | (\{\text{get_warfarin, get_chlorothiazide}\}, \emptyset, \{\text{minor}\}) | (\{\text{get_warfarin, chlorothiazide}\}, \emptyset, \{\text{minor}\})$
 $| (\{\text{warfarin, get_chlorothiazide}\}, \emptyset, \{\text{minor}\}) | (\{\text{warfarin, chlorothiazide}\}, \emptyset, \{\text{minor}\}) | (\{\text{get_warfarin, get_propranolol}\}, \emptyset, \{\text{minor}\})$
 $| (\{\text{get_warfarin, propranolol}\}, \emptyset, \{\text{minor}\}) | (\{\text{warfarin, get_propranolol}\}, \emptyset, \{\text{minor}\}) | (\{\text{warfarin, propranolol}\}, \emptyset, \{\text{minor}\})$
 $| (\{\text{get_flecainide, get_diltiazem}\}, \emptyset, \{\text{major}\}) | (\{\text{get_flecainide, diltiazem}\}, \emptyset, \{\text{major}\}) | (\{\text{flecainide, get_diltiazem}\}, \emptyset, \{\text{major}\})$
 $| (\{\text{flecainide, diltiazem}\}, \emptyset, \{\text{major}\}) | (\{\text{get_flecainide, get_verapamil}\}, \emptyset, \{\text{major}\}) | (\{\text{get_flecainide, verapamil}\}, \emptyset, \{\text{major}\})$
 $| (\{\text{flecainide, get_verapamil}\}, \emptyset, \{\text{major}\}) | (\{\text{flecainide, verapamil}\}, \emptyset, \{\text{major}\}) | (\{\text{get_flecainide, get_bisoprolol}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{get_flecainide, bisoprolol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{flecainide, get_bisoprolol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{flecainide, bisoprolol}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{get_flecainide, get_atenolol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{get_flecainide, atenolol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{flecainide, get_atenolol}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{flecainide, atenolol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{get_flecainide, get_propranolol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{get_flecainide, propranolol}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{flecainide, get_propranolol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{flecainide, propranolol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{get_flecainide, get_carvedilol}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{get_flecainide, carvedilol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{flecainide, get_carvedilol}\}, \emptyset, \{\text{moderate}\}) | (\{\text{flecainide, carvedilol}\}, \emptyset, \{\text{moderate}\})$
 $| (\{\text{major}\}, \emptyset, \{\text{major}\}) | (\{\text{moderate}\}, \emptyset, \{\text{moderate}\}) | (\{\text{minor}\}, \emptyset, \{\text{minor}\}) | (\{\text{alert_dup}\}, \emptyset, \{\text{alert_dup}\}) | (\{\text{danger}\}, \emptyset, \{\text{danger}\})$

Fig. 20 Reactions for the comorbidity case study in Sect. 5.1

1342 C.2 Auxiliary material for the comorbidity case study

1343 The RS specification for the comorbidity case study presented in Bowles et al. (2024) is reported in Fig. 20 (set of
1344 reactions) and Fig. 21 (context processes definitions), where
1345 we assume the initial state is $D_0 = \emptyset$. The corresponding
1346 experimentation with GROOVE has been discussed in Sect.
1347 5.1.

$\text{eafib1} \triangleq (\emptyset, \{\text{afib}\}, \emptyset).\text{eafib1} + (\{\text{afib}\}, \emptyset, \emptyset).\text{ehr}$
 $\text{ehr} \triangleq (\emptyset, \{\text{heart_rate}\}, \emptyset).\text{ehr} + (\{\text{heart_rate}\}, \emptyset, \emptyset).\text{ebb}$
 $\text{ebb} \triangleq \emptyset.\text{ebb} + \text{e_cbb} + \text{e_nsbb} + \text{e_sbb}$
 $\text{e_cbb} \triangleq (\emptyset, \{\text{verapamil}\}, \{\text{get_diltiazem}\}).\text{empty} + (\emptyset, \{\text{diltiazem}\}, \{\text{get_verapamil}\}).\text{empty}$
 $\text{e_nsbb} \triangleq (\emptyset, \{\text{carvedilol}\}, \{\text{get_propranolol}\}).\text{empty} + (\emptyset, \{\text{propranolol}\}, \{\text{get_carvedilol}\}).\text{empty}$
 $\text{e_sbb} \triangleq (\emptyset, \{\text{atenolol}\}, \{\text{get_bisoprolol}\}).\text{empty} + (\emptyset, \{\text{bisoprolol}\}, \{\text{get_atenolol}\}).\text{empty}$
 $\text{eafib2} \triangleq (\emptyset, \{\text{afib}\}, \emptyset).\text{eafib2} + (\{\text{afib}\}, \emptyset, \emptyset).\text{ehf}$
 $\text{ehf} \triangleq (\emptyset, \{\text{has_fib}\}, \emptyset).\text{ehf} + (\{\text{has_fib}\}, \emptyset, \emptyset).\text{eflec}$
 $\text{eflec} \triangleq \emptyset.\text{eflec} + \text{e_flec}$
 $\text{e_flec} \triangleq \{\text{get_flecainide}\}.\text{empty}$
 $\text{eafib3} \triangleq (\emptyset, \{\text{afib}\}, \emptyset).\text{eafib3} + (\{\text{afib}\}, \emptyset, \emptyset).\text{econs}$
 $\text{econs} \triangleq (\emptyset, \{\text{heart_rate}, \text{has_fib}\}, \emptyset).\text{econs} + (\emptyset, \{\text{consensus_acei}\}, \emptyset).\text{econs} + (\{\text{consensus_acei}, \text{heart_rate}\}, \emptyset, \emptyset).\text{estroke} + (\{\text{consensus_acei}, \text{has_fib}\}, \emptyset, \emptyset).\text{estroke}$
 $\text{estroke} \triangleq (\emptyset, \{\text{diseases}, \text{over75}\}, \emptyset).\text{ewarf} + (\{\text{over75}\}, \{\text{doac_fail}, \text{doac_int}\}, \emptyset).\text{edoac} + (\{\text{diseases}\}, \{\text{doac_fail}, \text{doac_int}\}, \emptyset).\text{edoac} + (\{\text{over75}, \text{doac_fail}\}, \emptyset, \emptyset).\text{evkant}$
 $+ (\{\text{over75}, \text{doac_int}\}, \emptyset, \emptyset).\text{evkant} + (\{\text{diseases}\}, \{\text{doac_fail}, \emptyset, \emptyset\}).\text{evkant} + (\{\text{diseases}\}, \{\text{doac_int}, \emptyset, \emptyset\}).\text{evkant}$
 $\text{ewarf} \triangleq \emptyset.\text{ewarf} + \text{e_warf}$
 $\text{e_warf} \triangleq \{\text{get_warfarin}\}.\text{empty}$
 $\text{edoac} \triangleq \emptyset.\text{edoac} + \text{e_doac}$
 $\text{e_doac} \triangleq (\emptyset, \{\text{dabigatran}\}, \{\text{get_apixaban}\}).\text{e_doacfai} + (\emptyset, \{\text{apixaban}\}, \{\text{get_dabigatran}\}).\text{e_doacfai}$
 $\text{e_doacfai} \triangleq (\{\text{doac_fail}\}, \emptyset, \{\text{stop_doac}\}).\text{evkant} + (\emptyset, \{\text{doac_fail}\}, \emptyset).\text{e_doacfai}$
 $\text{evkant} \triangleq \emptyset.\text{evkant} + \text{e_vkant}$
 $\text{e_vkant} \triangleq \{\text{get_vkant}\}.\text{empty}$
 $\text{ghyper} \triangleq (\emptyset, \{\text{hyper}\}, \emptyset).\text{ghyper} + (\{\text{hyper}\}, \emptyset, \emptyset).\text{g1}$
 $\text{g1} \triangleq (\{\text{diabete}\}, \emptyset, \emptyset).\text{g2} + (\{\text{below55}\}, \{\text{diabete}, \text{origin}\}, \emptyset).\text{g2} + (\emptyset, \{\text{below55}, \text{diabete}\}, \emptyset).\text{g3} + (\{\text{origin}\}, \{\text{diabete}\}, \emptyset).\text{g3}$
 $\text{g2} \triangleq \emptyset.\text{g2} + (\emptyset, \{\text{captopril}\}, \{\text{get_benazepril}\}).\text{g4} + (\emptyset, \{\text{benazepril}\}, \{\text{get_captopril}\}).\text{g4} + (\emptyset, \{\text{irbesartan}\}, \{\text{get_olmesortan}\}).\text{g5}$
 $+ (\emptyset, \{\text{olmesortan}\}, \{\text{get_irbesartan}\}).\text{g5}$
 $\text{g3} \triangleq \emptyset.\text{g3} + (\emptyset, \{\text{verapamil}\}, \{\text{get_diltiazem}\}).\text{g6} + (\emptyset, \{\text{diltiazem}\}, \{\text{get_verapamil}\}).\text{g6}$
 $\text{g4} \triangleq \emptyset.\text{g4} + (\emptyset, \{\text{verapamil}\}, \{\text{get_diltiazem}\}).\text{g7} + (\emptyset, \{\text{diltiazem}\}, \{\text{get_verapamil}\}).\text{g7} + (\emptyset, \{\text{chlorothiazide}\}, \{\text{get_indapamide}\}).\text{g8}$
 $+ (\emptyset, \{\text{indapamide}\}, \{\text{get_chlorothiazide}\}).\text{g8}$
 $\text{g5} \triangleq \emptyset.\text{g5} + (\emptyset, \{\text{verapamil}\}, \{\text{get_diltiazem}\}).\text{g9} + (\emptyset, \{\text{diltiazem}\}, \{\text{get_verapamil}\}).\text{g9} + (\emptyset, \{\text{chlorothiazide}\}, \{\text{get_indapamide}\}).\text{g10}$
 $+ (\emptyset, \{\text{indapamide}\}, \{\text{get_chlorothiazide}\}).\text{g10}$
 $\text{g6} \triangleq \emptyset.\text{g6} + (\emptyset, \{\text{captopril}\}, \{\text{get_benazepril}\}).\text{g7} + (\emptyset, \{\text{benazepril}\}, \{\text{get_captopril}\}).\text{g7} + (\emptyset, \{\text{irbesartan}\}, \{\text{get_olmesortan}\}).\text{g9}$
 $+ (\emptyset, \{\text{olmesortan}\}, \{\text{get_irbesartan}\}).\text{g9} + (\emptyset, \{\text{chlorothiazide}\}, \{\text{get_indapamide}\}).\text{g11} + (\emptyset, \{\text{indapamide}\}, \{\text{get_chlorothiazide}\}).\text{g11}$
 $\text{g7} \triangleq \emptyset.\text{g7} + (\emptyset, \{\text{irbesartan}\}, \{\text{get_olmesortan}\}).\text{etd} + (\emptyset, \{\text{olmesortan}\}, \{\text{get_irbesartan}\}).\text{etd} + (\emptyset, \{\text{chlorothiazide}\}, \{\text{get_indapamide}\}).\text{earb}$
 $+ (\emptyset, \{\text{indapamide}\}, \{\text{get_chlorothiazide}\}).\text{earb}$
 $\text{g8} \triangleq \emptyset.\text{g8} + (\emptyset, \{\text{irbesartan}\}, \{\text{get_olmesortan}\}).\text{ecbb} + (\emptyset, \{\text{olmesortan}\}, \{\text{get_irbesartan}\}).\text{ecbb} + (\emptyset, \{\text{verapamil}\}, \{\text{get_diltiazem}\}).\text{earb}$
 $+ (\emptyset, \{\text{diltiazem}\}, \{\text{get_verapamil}\}).\text{earb}$
 $\text{g9} \triangleq \emptyset.\text{g9} + (\emptyset, \{\text{captopril}\}, \{\text{get_benazepril}\}).\text{etd} + (\emptyset, \{\text{benazepril}\}, \{\text{get_captopril}\}).\text{etd} + (\emptyset, \{\text{chlorothiazide}\}, \{\text{get_indapamide}\}).\text{eacei}$
 $+ (\emptyset, \{\text{indapamide}\}, \{\text{get_chlorothiazide}\}).\text{eacei}$
 $\text{g10} \triangleq \emptyset.\text{g10} + (\emptyset, \{\text{captopril}\}, \{\text{get_benazepril}\}).\text{ecbb} + (\emptyset, \{\text{benazepril}\}, \{\text{get_captopril}\}).\text{ecbb} + (\emptyset, \{\text{chlorothiazide}\}, \{\text{get_indapamide}\}).\text{eacei}$
 $+ (\emptyset, \{\text{indapamide}\}, \{\text{get_chlorothiazide}\}).\text{eacei}$
 $\text{g11} \triangleq \emptyset.\text{g11} + (\emptyset, \{\text{captopril}\}, \{\text{get_benazepril}\}).\text{earb} + (\emptyset, \{\text{benazepril}\}, \{\text{get_captopril}\}).\text{earb} + (\emptyset, \{\text{irbesartan}\}, \{\text{get_olmesortan}\}).\text{eacei}$
 $+ (\emptyset, \{\text{olmesortan}\}, \{\text{get_irbesartan}\}).\text{eacei}$
 $\text{ecbb} \triangleq \emptyset.\text{ecbb} + \text{e_cbb}$
 $\text{eacei} \triangleq \emptyset.\text{eacei} + \text{e_acei}$
 $\text{e_acei} \triangleq (\emptyset, \{\text{captopril}\}, \{\text{get_benazepril}\}).\text{empty} + (\emptyset, \{\text{benazepril}\}, \{\text{get_captopril}\}).\text{empty}$
 $\text{earb} \triangleq \emptyset.\text{earb} + \text{e_arb}$
 $\text{e_arb} \triangleq (\emptyset, \{\text{irbesartan}\}, \{\text{get_olmesortan}\}).\text{empty} + (\emptyset, \{\text{olmesortan}\}, \{\text{get_irbesartan}\}).\text{empty}$
 $\text{etd} \triangleq \emptyset.\text{etd} + \text{e_td}$
 $\text{e_td} \triangleq (\emptyset, \{\text{chlorothiazide}\}, \{\text{get_indapamide}\}).\text{empty} + (\emptyset, \{\text{indapamide}\}, \{\text{get_chlorothiazide}\}).\text{empty}$
 $\text{k_doac} \triangleq (\{\text{doac_test}\}, \emptyset, \{\text{doac_ok}\}).\text{empty} + (\{\text{doac_test}\}, \emptyset, \{\text{doac_fail}\}).\text{empty} + (\emptyset, \{\text{doac_test}\}, \emptyset).\text{k_doac}$
 $\text{empty} \triangleq \emptyset.\text{empty}$
 $\text{kafib} \triangleq \{\text{afib}\}.\text{empty} + \text{empty}$
 $\text{khf} \triangleq \{\text{has_fib}\}.\text{empty} + \text{empty}$
 $\text{khr} \triangleq \{\text{heart_rate}\}.\text{empty} + \text{empty}$
 $\text{kcons} \triangleq \{\text{consensus_acei}\}.\text{empty} + \text{empty}$
 $\text{kage} \triangleq \{\text{over75}\}.\text{empty} + \{\text{below55}\}.\text{empty} + \text{empty}$
 $\text{kdiabete} \triangleq \{\text{diabete}\}.\text{empty} + \text{empty}$
 $\text{kdoacint} \triangleq \{\text{doac_int}\}.\text{empty} + \text{empty}$
 $\text{khyper} \triangleq \{\text{hyper}\}.\text{empty} + \text{empty}$
 $\text{korigin} \triangleq \{\text{origin}\}.\text{empty} + \text{empty}$

Fig. 21 Context process definitions for the comorbidity case study in Sect. 5.1. The initial context is given by the parallel composition of therapies $\text{eafib1} | \text{eafib2} | \text{eafib3} | \text{ghyper}$ and the parallel composition of features $\text{kafib} | \text{khf} | \text{khr} | \text{kcons} | \text{kage} | \text{kdiabete} | \text{kdoacint} | \text{khyper} | \text{korigin} | \text{k_doac}$

1349
1350

C.3 Auxiliary material for the protein signaling networks case study

1351
1352
1353
1354

The BioResolve specification for the protein signaling networks case study presented in Ballis et al. (2024) is reported in Fig. 22. The corresponding experimentation with GROOVE has been discussed in Sect. 5.2.

1355
1356

C.4 Auxiliary material for the T cell differentiation case study

1357
1358
1359
1360
1361
1362

The BioResolve specification derived from the Boolean network model (available at Puniya (2024), see Fig. 23) of the T cell differentiation case study from Puniya et al. (2018), and exploited in Brodo et al. (2025a) is reported in Fig. 24. The corresponding experimentation with GROOVE has been discussed in Sect. 5.3.

```

myentities([]).

myreactions([
    react([akt],[],[akt]),
    react([erbB3],[],[akt]),
    react([mtor],[],[akt]),
    react([pdk1],[],[akt]),
    react([erbB1],[e,p],[erbB1]),
    react([egf],[e,p],[erbB1]),
    react([plcg],[e,p],[erbB1]),
    react([erbB2],[e,t,p],[erbB2]),
    react([egf],[e,t,p],[erbB2]),
    react([erbB3],[e,t,p],[erbB2]),
    react([erbB3],[e,p],[erbB3]),
    react([hrg],[e,p],[erbB3]),
    react([erk12],[],[erk12]),
    react([egf],[],[erk12]),
    react([p],[],[erk12]),
    react([mek12],[],[erk12]),
    react([mek12],[],[mek12]),
    react([erbB1],[],[mek12]),
    react([erbB2],[],[mek12]),
    react([erbB3],[],[mek12]),
    react([mtor],[],[mtor]),
    react([p],[],[mtor]),
    react([akt],[],[mtor]),
    react([p70s6k],[],[p70s6k]),
    react([akt],[],[p70s6k]),
    react([mtor],[],[p70s6k]),
    react([erk12],[],[p70s6k]),
    react([pdk1],[],[pdk1]),
    react([erbB1],[],[pdk1]),
    react([erbB2],[],[pdk1]),
    react([erbB3],[],[pdk1]),
    react([mek12],[],[pdk1]),
    react([pkca],[],[pkca]),
    react([plcg],[],[pkca]),
    react([plcg],[],[plcg]),
    react([egf],[],[plcg]),
    react([erbB1],[],[plcg]),
    react([erbB2],[],[plcg]),
    react([erbB3],[],[plcg]) ]).

myenvironment([
    k = {egf,hrg}.k,
    ket = {e,t}.ket,
    korep = {e}.korep + {p}.korep,
    korept = {e}.korept + {p}.korept + {t}.korept,
    kge = (?{erbB1},{},e).kge
        + ?{erbB2},{},e.kge
        + ?{},erbB1,erbB2,??.kge
    ]).

```

Jak1 = IFNgR and not SOCS1
IL21 = STAT3 and NFAT
IL18R = IL18 and IL12 and not STAT6
SOCS1 = STAT1 or Tbet
IL6 = RORgt
STAT5 = IL2R
IL17 = (RORgt and not STAT1)
or (STAT3 and IL17 and IL23R and not STAT1 and not STAT5)
STAT4 = IL12R and IL12 and not GATA3
IFNgR = (IFNg_e and NFAT) or (IFNg and NFAT)
STAT6 = IL4R and not IFNg and not SOCS1
GATA3 = (STAT6 and NFAT and not TGFB and not RORgt and not Foxp3
and not Tbet) or (GATA3 and not Tbet)
or (STAT5 and not TGFB and not RORgt and not Foxp3 and not Tbet)
IL4 = GATA3 and NFAT and not STAT1
NFkB = IRAK and not Foxp3
IL2 = NFAT and NFkB and not Tbet
IL23R = (IL23 and STAT3 and not Tbet) or STAT3
Tbet = (STAT4 and not RORgt and not Foxp3)
or (STAT1 and not RORgt and not Foxp3)
or (Tbet and not IL12 and not IFNg and not RORgt and not Foxp3)
TGFB = TGFB and NFAT
RORgt = TGFB and ((STAT3 and IL21R) or (STAT3 and IL6R)) and not Tbet
and not GATA3 and not Foxp3
IL6R = IL6 or IL6_e
IL21R = IL21
Foxp3 = (TGFB and not (IL6R and STAT3) and not IL21R and not GATA3)
or (STAT5 and not (IL6R and STAT3) and not IL21R and not GATA3)
IRAK = IL18R
IL12R = (IL12 and NFAT) or (STAT4 and not GATA3) or Tbet or (TCR and not GATA3)
IL2R = IL2 and NFAT
STAT3 = IL21R or IL23R or IL6R
IFNg = NFkB or (STAT4 and NFkB and NFAT and not STAT3 and not STAT6)
or (Tbet and not STAT3)
NFAT = TCR and not Foxp3
STAT1 = (IL27 and NFAT) or Jak1
IL4R = (IL4 and not SOCS1) or IL4_e

Fig. 23 Boolean updates of the T Cell differentiation model from Puniya et al. (2018), available at Puniya (2024)

C.5 Auxiliary material for the GROOVE experiments

1363

To replicate the GROOVE experiments reported in Sects. 4 (for the toy running example) and 5, we have included the following supplementary resources with this submission:

1364

1365

1366

- The rule systems described in Sect. 4;
- The start graphs derived from the BioResolve specifications in this appendix (C.1–C.4);
- Instructions for calling the GROOVE generator so as to reproduce all the exploration runs, occurrence graphs and model checking results (using GROOVE version 7.4.3).

1367

1368

1369

1370

1371

1372

Fig. 22 BioResolve implementation of the protein signaling network case study from Sect. 5.2

```

myreactions([
    react([stat5],[gata3,i121r,i16r],[foxp3]),
    react([stat5],[gata3,i121r,stat3],[foxp3]),
    react([tgfb], [gata3,i121r,i16r],[foxp3]),
    react([tgfb], [gata3,i121r,stat3],[foxp3]),
    react([gata3],[tbet],[gata3]),
    react([nfat,stat6],[foxp3,rorgt,tbet,tgfb],[gata3]),
    react([stat5],[foxp3,rorgt,tbet,tgfb],[gata3]),
    react([nfat,nfk,stat4],[stat3,stat6],[ifng]),
    react([nfkb],[],[ifng]),
    react([tbet],[stat3],[ifng]),
    react([ifng,nfat],[],[ifngr]),
    react([ifnge,nfat],[],[ifngr]),
    react([i112,nfat],[],[i112r]),
    react([stat4],[gata3],[i112r]),
    react([tbet],[],[i112r]),
    react([tcr],[gata3],[i112r]),
    react([i117,i123r,stat3],[stat1,stat5],[i117]),
    react([rorgt],[stat1],[i117]),
    react([i112,i118],[stat6],[i118r]),
    react([nfat,nfk],[tbet],[i12]),
    react([nfat,stat3],[],[i121]),
    react([i121],[],[i121r]),
    react([i123,stat3],[tbet],[i123r]),
    react([stat3],[],[i123r]),
    react([i12,nfat],[],[i12r]),
    react([gata3,nfat],[stat1],[i14]),
    react([i14],[socs1],[i14r]),
    react([i14e],[],[i14r]),
    react([rorgt],[],[i16]),
    react([i16],[],[i16r]),
    react([i16e],[],[i16r]),
    react([i118r],[],[irak]),
    react([ifngr],[socs1],[jak1]),
    react([tcr],[foxp3],[nfat]),
    react([irak],[foxp3],[nfkb]),
    react([i121r,stat3,tgfb],[foxp3,gata3,tbet],[rorgt]),
    react([i16r,stat3,tgfb],[foxp3,gata3,tbet],[rorgt]),
    react([stat1],[],[socs1]),
    react([tbet],[],[socs1]),
    react([i127,nfat],[],[stat1]),
    react([jak1],[],[stat1]),
    react([i121r],[],[stat3]),
    react([i123r],[],[stat3]),
    react([i16r],[],[stat3]),
    react([i112,i112r],[gata3],[stat4]),
    react([i12r],[],[stat5]),
    react([i14r],[ifng,socs1],[stat6]),
    react([stat1],[foxp3,rorgt],[tbet]),
    react([stat4],[foxp3,rorgt],[tbet]),
    react([tbet],[foxp3,ifng,i112,rorgt],[tbet]),
    react([nfat,tgfb],[],[tgfb]) ]).

```

Fig. 24 BioResolve implementation of the T cell case study from Section 5.3.

Acknowledgements We thank Paolo Milazzo for the technical help in double checking the coherence of large state space generated by Python scripts based on BioResolve and those generated by GROOVE.

Author Contributions Both authors contributed equally to the paper, in writing and reviewing all sections.

Funding Research partially supported by the PRIN PNRR 2022 project *Resource Awareness in Programming* (RAP, P2022HXNSC), by the project *SEcurity and RIghts In the CyberSpace* (SERICS, PE00000014 - CUP H73C2200089001), under the National Recovery and Resilience Plan (NRRP) funded by the European Union - NextGenerationEU, by the INdAM-GNCS Projects *Reversibilità In Sistemi Concorrenti: analisi Quantitative e Funzionali* (RISICO, CUP E53C22001930001) and *Modelli e Analisi per sistemi reversibili e quantistici* (MARQ, CUP E53C24001950001), and by the University of Pisa PRA project *Formal methods for the healthcare domain based on spatial information* (FM4HD, PRA_2022_99).

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

Declarations

Ethical approval Not applicable.

Conflict of interest The authors declare no Conflict of interest.

Code availability GROOVE is an open-source tool, available at <https://github.com/nl-utwente-groove/code>. For replication of the GROOVE experiments reported in this paper, we provide supplementary material described in Sect. C.5.

References

- Azimi S (2017) Steady states of constrained reaction systems. *Theor Comput Sci* 701((C)):20–26. <https://doi.org/10.1016/j.tcs.2017.03.047>
- Azimi S, Iancu B, Petre I (2014) Reaction system models for the heat shock response. *Fund Inform* 131(3–4):299–312. <https://doi.org/10.3233/FI-2014-1016>
- Azimi S, Gratić C, Ivanov S, Petre I (2015) Dependency graphs and mass conservation in reaction systems. *Theor Comput Sci* 598:23–39. <https://doi.org/10.1016/j.tcs.2015.02.014>
- Ballis D, Brodo L, Falaschi M, Olarte C (2024) Process calculi and rewriting techniques for analyzing reaction systems. In: Gori R, Milazzo P, Tribastone M (eds) Computational methods in systems biology—22nd international conference, CMSB 2024, Pisa, Italy, September 16–18, 2024, Proceedings. Lecture notes in computer science, vol 14971. Springer, pp 1–18. https://doi.org/10.1007/978-3-031-71671-3_1
- Bowles JKF, Caminati MB (2017) A flexible approach for finding optimal paths with minimal conflicts. In: Proceedings of ICFEM 2017. LNCS, vol 10610. Springer, pp 209–225. https://doi.org/10.1007/978-3-319-68690-5_13
- Bowles J, Brodo L, Bruni R, Falaschi M, Gori R, Milazzo P (2024) Enhancing reaction systems with guards for analysing comorbidity treatment strategies. In: Gori R, Milazzo P, Tribastone M (eds) Computational methods in systems biology—22nd international conference, CMSB 2024, Pisa, Italy, September 16–18, 2024, Proceedings. Lecture notes in computer science, vol 14971. Springer, pp 27–44. https://doi.org/10.1007/978-3-031-71671-3_3

- 1427 Brodo L, Bruni R, Falaschi M (2021) A logical and graphical framework
1428 for reaction systems. *Theor Comput Sci* 875:1–27. <https://doi.org/10.1016/J.TCS.2021.03.024> 1429
- 1430 Brodo L, Bruni R, Falaschi M, Gori R, Levi F, Milazzo P (2023) Quantitative extensions of reaction systems based on SOS semantics.
1431 *Neural Comput Appl* 35(9):6335–6359. <https://doi.org/10.1007/S00521-022-07935-6> 1432
- 1433 Brodo L, Bruni R, Falaschi M (2024a) A framework for monitored
1434 dynamic slicing of reaction systems. *Nat Comput* 23(2):217–234.
1435 <https://doi.org/10.1007/S11047-024-09976-3> 1436
- 1437 Brodo L, Bruni R, Falaschi M, Gori R, Milazzo P, Montagna V,
1438 Pulieri P (2024b) Causal analysis of positive reaction systems.
1439 *Int J Softw Tools Technol Transf* 26(4):509–526. <https://doi.org/10.1007/S10009-024-00757-Y> 1440
- 1441 Brodo L, Bruni R, Falaschi M, Gori R, Milazzo P (2025a) Attractor
1442 and slicing analysis of a t cell differentiation model based on
1443 reaction systems. In: From data to models and back—11th interna-
1444 tional symposium, DataMod 2023, Eindhoven, The Netherlands,
1445 November 6–7, 2023, Proceedings. Lecture notes in computer sci-
1446 ence, vol 14618. Springer, pp 69–89. https://doi.org/10.1007/978-3-031-87217-4_4 1447
- 1448 Brodo L, Bruni R, Falaschi M, Gori R, Milazzo P (2025b) Slicing anal-
1449 yses for negative dependencies in reaction systems modeling gene
1450 regulatory networks. *Natural Computing*. Under review; will hope-
1451 fully appear in same volume as this paper 1452
- 1452 Clavel M, Durán F, Eker S, Lincoln P, Martí-Oliet N, Meseguer J, Talcott
1453 CL (eds.) (2007) All about Maude—a high-performance logical
1454 framework, how to specify, program and verify systems in rewriting
1455 logic. *Lecture notes in computer science*, vol 4350. Springer.
1456 <https://doi.org/10.1007/978-3-540-71999-1> 1457
- 1457 Corollini L, Maj C, Marini F, Besozzi D, Mauri G (2012) An excursion in
1458 reaction systems: from computer science to biology. *Theor Comput
1459 Sci* 454:95–108. <https://doi.org/10.1016/j.tcs.2012.04.003> 1460
- 1460 Ehrenfeucht A, Rozenberg G (2007) Reaction systems. *Fundam Inform*
1461 75(1–4):263–280 1462
- 1462 Ehrenfeucht A, Main MG, Rozenberg G (2010) Combinatorics of life
1463 and death for reaction systems. *Int J Found Comput Sci* 21(3):345–
1464 356. <https://doi.org/10.1142/S0129054110007295> 1465
- 1465 Ehrenfeucht A, Main MG, Rozenberg G (2011) Functions defined by
1466 reaction systems. *Int J Found Comput Sci* 22(1):167–178. <https://doi.org/10.1142/S0129054111007927> 1467
- 1467 Ehrlig H, Ehrlig K, Prange U, Taentzer G (2006) Fundamentals of alge-
1468 braic graph transformation. *Monographs in theoretical computer
1469 science. An EATCS series*. Springer. <https://doi.org/10.1007/3-540-31188-2> 1470
- 1470 Feder G, Eccles M, Grol R, Griffiths C, Grimshaw J (1999) Using
1471 clinical guidelines. *BMJ* 318(7185):728–730. <https://doi.org/10.1136/bmj.318.7185.728> 1472
- 1472 Ferretti C, Leporati A, Manzoni L, Porreca AE (2020) The many roads to
1473 the simulation of reaction systems. *Fundam Inform* 171(1–4):175–
1474 188. <https://doi.org/10.3233/FI-2020-1878> 1475
- 1475 Ghamarian AH, Mol M, Rensink A, Zambon E, Zimakova M (2012)
1476 Modelling and analysis using GROOVE. *Int J Softw Tools Technol
1477 Transf* 14(1):15–40. <https://doi.org/10.1007/S10009-011-0186-X> 1478
- 1478 Heckel R, Taentzer G (2020) Graph transformation for software
1479 engineers—with applications to model-based development and
1480 domain-specific language engineering. Springer. <https://doi.org/10.1007/978-3-030-43916-3> 1481
- 1481 Helikar T, Kowal B, McClenathan S, Bruckner M, Rowley T, Madra-
1482 hovim A, Wicks B, Shrestha M, Limbu K, Rogers JA (2012) The cell
1483 collective: toward an open and collaborative approach to systems
1484 biology. *BMC Syst Biol* 6(1):1–14. <https://doi.org/10.1186/1752-0509-6-96> 1485
- 1485 Heyde SV, Bender C, Henjes F (2014) Boolean ErbB network recon-
1486 structions and perturbation simulations reveal individual drug
1487 response in different breast cancer cell lines. *BMC Syst Biol* 8:75.
1488 <https://doi.org/10.1186/1752-0509-8-75> 1489
- 1489 Hirahara K, Nakayama T (2016) CD4+ T-cell subsets in inflammatory
1490 diseases: beyond the Th 1/Th 2 paradigm. *Int Immunopharmacol* 28(4):163–
1491 171. <https://doi.org/10.1093/intimm/dxw006> 1492
- 1492 Hughes LD, McMurdie MET, Guthrie B (2013) Guidelines for people
1493 not for diseases: the challenges of applying UK clinical guidelines
1494 to people with multimorbidity. *Age Ageing* 42:62–69. <https://doi.org/10.1093/ageing/afs100> 1495
- 1495 Ivanov S, Rogojin V, Azimi S, Petre I (2018) WEBRSIM: a web-based
1496 reaction systems simulator. In: Díaz CG, Riscos-Núñez A, Paun
1497 G, Rozenberg G, Salomaa A (eds) Enjoying natural computing—
1498 essays dedicated to Mario de Jesús Pérez-Jiménez on the occasion
1499 of his 70th birthday. *Lecture notes in computer science*, vol 11270.
1500 Springer, pp 170–181. https://doi.org/10.1007/978-3-030-00265-7_14 1501
- 1501 Kreowski H, Rozenberg G (2019) Graph transformation through graph
1502 surfing in reaction systems. *J Log Algebraic Methods Program*.
1503 <https://doi.org/10.1016/J.JLAMP.2019.100481> 1504
- 1504 Lafaille JJ (1998) The role of helper T cell subsets in autoimmune
1505 diseases. *Cytokine Growth Factor Rev* 9(2):139–151. [https://doi.org/10.1016/s1359-6101\(98\)00009-4](https://doi.org/10.1016/s1359-6101(98)00009-4) 1506
- 1506 Luckheram RV, Zhou R, Verma AD, Xia B (2012) CD4+ T cells:
1507 differentiation and functions. *Clin Dev Immunol*. <https://doi.org/10.1155/2012/925135> 1508
- 1508 Meng X, Yang J, Dong M, Zhang K, Tu E, Gao Q, Chen W, Zhang
1509 C, Zhang Y (2016) Regulatory T cells in cardiovascular diseases.
1510 *Nat Rev Cardiol* 13(3):167–179. <https://doi.org/10.1038/nrcardio.2015.169> 1511
- 1511 Meski A, Penczek W, Rozenberg G (2015) Model checking temporal
1512 properties of reaction systems. *Inf Sci* 313:22–42. <https://doi.org/10.1016/J.IINS.2015.03.048> 1513
- 1513 Meski A, Koutny M, Mikulski L, Penczek W (2024) Reaction mining
1514 for reaction systems. *Nat Comput* 23(2):323–343. <https://doi.org/10.1007/S11047-024-09989-Y> 1515
- 1515 Milner R (1980) A calculus of communicating systems. In: LNCS, vol
1516 92. Springer 1517
- 1517 Nobile MS, Porreca AE, Spolaor S, Manzoni L, Cazzaniga P, Mauri
1518 G, Besozzi D (2017) Efficient simulation of reaction systems
1519 on graphics processing units. *Fundam Inform* 154(1–4):307–321.
1520 <https://doi.org/10.3233/FI-2017-1568> 1521
- 1521 Okubo F, Yokomori T (2016) The computational capability of chemical
1522 reaction automata. *Nat Comput* 15(2):215–224. <https://doi.org/10.1007/s11047-015-9504-7> 1523
- 1523 Plotkin GD (2004) A structural approach to operational semantics. *J
1524 Log Algebraic Methods Program* 60–61:17–139 1525
- 1525 Puniya BL (2024) CD4+ T cell Differentiation model webpage on the
1526 CellCollective platform. Accessed: 18 March. <https://research.cellcollective.org/?dashboard=true#module/6678:1/cd4-t-cell-differentiation/1> 1527
- 1527 Puniya BL, Todd RG, Mohammed A, Brown DM, Barberis M, Helikar T
1528 (2018) A mechanistic computational model reveals that plasticity
1529 of CD4+ T cell differentiation is a function of cytokine composition
1530 and dosage. *Front Physiol* 9:878. <https://doi.org/10.3389/fphys.2018.00878> 1531
- 1531 Rensink A (2024) The GROOVE tool set, version 7.0.1. Available at
1532 <https://github.com/nl-utwente-groove/code> 1533
- 1533 Rubio R, Martí-Oliet N, Pita I, Verdejo A (2021) Strategies, model
1534 checking and branching-time properties in Maude. *J Log Algebraic
1535 Methods Program* 123:100700. <https://doi.org/10.1016/J.JLAMP.2021.100700> 1536
- 1536 Saez-Rodriguez J, Simeoni L, Lindquist JA, Hemenway R, Bommhardt
1537 U, Arndt B, Haus U-U, Weismantel R, Gilles ED, Klamt S (2007)
1538 A logical model provides insights into T cell receptor signaling.
1539 *PLoS Comput Biol* 3(8):163. <https://doi.org/10.1371/journal.pcbi.0030163> 1540
- 1540

- 1558 Thakar J, Albert R (2010) Boolean models of within-host immune inter-
1559 actions. *Curr Opin Microbiol* 13(3):377–381. [https://doi.org/10.](https://doi.org/10.1016/j.mib.2010.04.003)
1560 [1016/j.mib.2010.04.003](https://doi.org/10.1016/j.mib.2010.04.003)
- 1561 Woolf SH, Grol R, Hutchinson A, Eccles M, Grimshaw J (1999) Poten-
1562 tial benefits, limitations, and harms of clinical guidelines. *BMJ*
1563 318(7182):527–530. <https://doi.org/10.1136/bmj.318.7182.527>

1564 **Publisher's Note** Springer Nature remains neutral with regard to juris-
1565 dictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.