

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	Natural Computing	
Corresponding Author	FamilyName Particle Given Name Suffix Division Organization Address Phone Fax Email URL ORCID	Rensink Arend CS Department University of Twente Hallenweg 19, 7522, Enschede, The Netherlands arend.rensink@utwente.nl http://orcid.org/0000-0002-1714-6319
Author	FamilyName 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 http://orcid.org/0000-0002-7771-4154
Schedule	Received Revised Accepted	27 Jul 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		



1 Experimenting with reaction systems using graph transformation 2 and GROOVE

3 Roberto Bruni¹ · Arend Rensink²

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

6 Abstract

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

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

15 1 Introduction

17 We explore the capabilities of a state-of-the-art toolset
18 based on graph transformation to perform reachability,
19 causal analysis and verification of complex systems mod-
20 modelled using Reaction Systems.

21 Reaction systems (RSs) (Ehrenfeucht and Rozenberg
22 2007) are a computational model inspired by the func-
23 tioning of biochemical reactions within living cells. RSs
24 focus on the interaction of entities through a set of reac-
25 tions. Each reaction relies on some reactants, inhibitors,
26 and products to mimic two fundamental mechanisms found
27 in nature: facilitation and inhibition. At each time instant,
28 the next state of the system is determined by the products
29 of all enabled reactions plus some additional entities that
30 are possibly provided by the environment. Unlike tradi-
31 tional models of concurrency, like Petri nets, the theory of
32 RSs is based on three principles: *no permanency*: any entity

vanishes unless it is sustained by a reaction; *no competi-
tion*: an entity is either available for all reactions, or it is
not available at all; and *no counting*: the exact concentra-
tion level of available entities is ignored. Moreover, due to
the use of inhibitors, RS can exhibit non-monotonic
behaviour, in the sense that what can be done with fewer
resources is not necessarily replicable with more resources.

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

Since their introduction, RSs have been successfully
applied to the analysis of complex systems in many dif-
ferent fields (Azimi et al. 2014; Corolli et al. 2012; Azimi
2017; Okubo and Yokomori 2016; Ehrenfeucht et al.
2010, 2011). Recent applications concerned, e.g., with the
efficacy of medical treatments for comorbidities and with
the selection of the best environment to achieve some
desired phenomena (Bowles et al. 2024; Brodo et al.
2025a), led to experimenting with environments that
exhibit nondeterministic and recursive behaviour. As a
consequence, performing reachability and causal analysis
requires the exploration of large state spaces, for which the

A1 Arend Rensink
A2 arend.rensink@utwente.nl

A3 Roberto Bruni
A4 roberto.bruni@unipi.it

A5 ¹ CS Department, University of Pisa, Largo B. Pontecorvo, 3,
A6 56127 Pisa, Italy

A7 ² CS Department, University of Twente, Hallenweg 19,
A8 7522 Enschede, The Netherlands

59 prototype tool BioResolve (Brodo et al. 2021) struggled in
60 terms of memory consumption and response time.

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

76 Importantly, from a practical point of view, there are a
77 number of (academic) tools supporting the use of GT. The
78 research described here crucially relies on Ghamarian et al.
79 (2012), one of the most prominent tools in this area, which
80 was designed precisely to enable GT-based system analysis
81 of the kind described above. The features of GROOVE
82 that are essential for the purpose of this research are:

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

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

98 The core rule describes the simultaneous firing of all
99 **Reactions** of which no inhibitors and all reactants are
100 present; the firing results in the presence of all products.
101 Simultaneously, all currently present **Entity**s are removed.
102 In GROOVE syntax, the core rule is drawn as in Fig. 1. To
103 parse this, note that (in the GROOVE notation) the red

104 structure must be absent for the rule to apply; moreover,
105 green labels are added and blue ones deleted upon rule
106 application. The *present* flag signals whether an **Entity** is
107 considered to be currently present; hence, creating or
108 deleting that flag comes down to creating or deleting the
109 **Entity**. The \forall -nodes impose the desired quantification,
110 causing a single application of this rule to model the firing
111 of all enabled **Reactions**, even if there are thousands of
112 them. Similarly for the simultaneous deletion of all
113 **Entity**s.

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

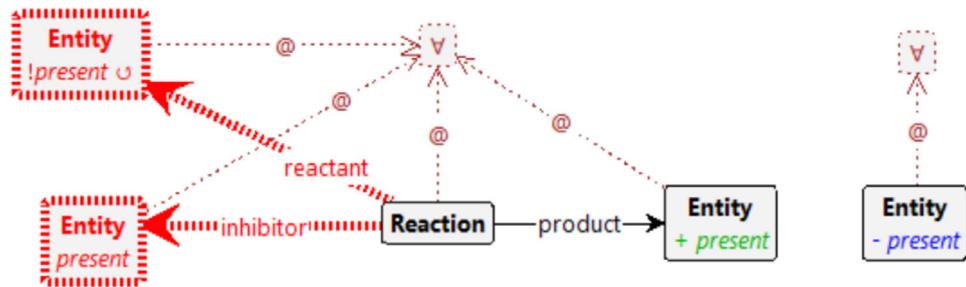
123 After a brief account of RSs (Sect. 2.1) and GROOVE
124 (Sect. 2.2), we present the overall rationale, blueprint and
125 key features of our approach in Sect. 4 through a simple
126 vending machine example (Sect. 3). The subsequent
127 experimentation in Sect. 5 is concerned with revisiting
128 existing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

2 Background

2.1 RSs with guarded contexts

148 First, we briefly account for the classical set theoretic
149 definition of Reaction Systems (RSs) (Ehrenfeucht and
150 Rozenberg 2007). Then, we focus on their process alge-
151 braic version (Brodo et al. 2021) and its further extension
152

1FL01¹ It should be noted that there is another, quite unrelated, way in
1FL02 which the concepts of Graph Transformation and **Reaction systems**
1FL03 may be combined, namely by extending the latter from pure entities
1FL04 (which are analogous to graph nodes) to entities with relations
1FL05 between them (which can be encoded through graph edges). In
1FL06 Kreowski and Rozenberg (2019) this has inspired a new methodology
1FL07 for Graph Transformation, there called *graph surfing*.

Fig. 1 Rule for reactions firing

153 with guarded contexts in Bowles et al. (2024) that are
154 supported by the RS analysis framework BioResolve.²

155 *RS basics.* A Reaction System is a pair $\mathcal{A} = (S, A)$,
156 where S is the finite set of *entities*, and A is a finite set of
157 *reactions* of the form $a = (R, I, P)$, with $R, I, P \subseteq S$ and
158 $R \cap I = \emptyset$. The sets R, I, P are the sets of *reactants*,
159 *inhibitors*, and *products*, respectively. Without loss of
160 generality, we admit the use of empty sets as reactants,
161 inhibitors, and products. Given the current state $W \subseteq S$, a
162 reaction $a = (R, I, P)$ is enabled in W if all its reactants are
163 present (i.e., $R \subseteq W$) and all its inhibitors absent (i.e.,
164 $W \cap I = \emptyset$). The *result* of the reaction a on the current
165 state W is P if a is enabled, and \emptyset otherwise. The result of
166 all reactions A on the current state W , is the union of the
167 results of all reactions. The no-permanency principle of
168 RSs dictates that entities disappear if not sustained by some
169 reaction. Thus, the current state $W = D \cup C$ is determined
170 by the result D of all reactions on the previous state,
171 together with some additional entities C that can be pro-
172 vided by the *context* at each step.

173 *Process algebraic RSs and guarded contexts.* Inspired
174 by Plotkin's Structural Operational Semantics approach
175 (Plotkin 2004) and process algebras such as CCS (Milner
176 1980), the key features of the process algebraic version of
177 RSs are compositionality and the ability to account for a
178 quite general notion of context (guarded, nondeterministic,
179 recursive) using a friendly syntax. This way, we derive a
180 Labelled Transition System (LTS) semantics for RSs by
181 means of inductive inference rules, where LTS states are
182 terms of an algebra, each transition defines a computation
183 step of the RS and its label records the entities involved in
184 that step.

185 *RS processes* are defined by the grammar below:

$P ::= [M]$
 $M ::= (R, I, P) \mid D \mid K \mid M|M$
 $K ::= \mathbf{0} \mid (R, I, C).K \mid K+K \mid X$

187 where R , I , P , C , and D are sets of entities (with
188 $R \cap I = \emptyset$) and X is a context identifier drawn from a

family of (possibly recursive) definitions $\Delta \triangleq \{X_j = K_j\}_{j \in J}$,
189 called the *environment*.

190 Roughly, a RS process P embeds a *mixture* process M
191 obtained as the parallel composition of some reactions
192 (R, I, P) , some available entities D (if any), and some
193 *context* process K . A context process K is either: the nil
194 context $\mathbf{0}$ that stops the computation; the guarded context
195 $(R, I, C).K$ that makes the entities C available to the reactions
196 if the reactants R are present and the inhibitors I are
197 absent, and then will behave as K at the next step; the non-
198 deterministic choice $K_1 + K_2$ that can behave as either K_1
199 or K_2 ; the context identifier X that behaves as K for
200 $X = K \in \Delta$. We write $C.K$ as a shorthand for the trivially
201 guarded process $(\emptyset, \emptyset, C).K$ and we assume the recursive
202 context $\mathbf{Emp} = \emptyset.\mathbf{Emp}$ is always defined.

203 We say that P and P' are structurally equivalent, written
204 $P \equiv P'$, when they denote the same term up to the laws of
205 Abelian monoids (unit, associativity and commutativity)
206 for parallel composition $\cdot|.$, with \emptyset as the unit, and the
207 laws of idempotent Abelian monoids for choice $\cdot + \cdot$, with
208 $\mathbf{0}$ as the unit. We also assume $D_1|D_2 \equiv D_1 \cup D_2$ for any
209 $D_1, D_2 \subseteq S$. Indexed sums and parallel compositions are
210 denoted, respectively, by $\sum_{i \in I} K_i$ and $\prod_{i \in I} M_i$.

211 The SOS semantics of RS processes is defined by the
212 SOS rules in Fig. 2. A transition label ℓ , written
213 $\langle\langle D \triangleright R', I', C \rangle\rangle \triangleright R, I, P$, records: the available entities D ;
214 the entities C provided by the guarded contexts, assuming
215 all entities in R' are present and those in I' are absent; the
216 set R of entities whose presence enables or disables some
217 reactions; the set I of entities whose absence enables or
218 disables some reactions; and the set P of reaction products.

219 The rules guarantee that, whenever $P \xrightarrow{\langle\langle D \triangleright R', I', C \rangle\rangle \triangleright R, I, P} P'$, it
220 holds that (R', I', C) is enabled in D and that (R, I, P) is
221 enabled in $W \triangleq (D \cup C)$.

222 The rule (*Ent*) records the set of current entities D . By
223 rule (*Cxt*), a guarded context process $(R, I, C).K$ makes
224 available the entities in C if the reactants R are present in
225 the current state and the inhibitors I are absent, and then
226 reduces to K . Rules (*Suml*) and (*Sumr*) select a move of
227 either the left or the right context, resp., discarding the

$$\begin{array}{c}
\frac{}{D \xrightarrow{\langle\langle D \triangleright \emptyset, \emptyset, \emptyset \rangle \triangleright \emptyset, \emptyset, \emptyset \rangle} \emptyset} (Ent) \\
\frac{\text{[Talk]}}{\frac{K_1 \xrightarrow{\ell} K'_1}{K_1 + K_2 \xrightarrow{\ell} K'_1} (Suml)} \quad \frac{K_2 \xrightarrow{\ell} K'_2}{K_1 + K_2 \xrightarrow{\ell} K'_2} (Sumr) \quad \frac{X = K \in \Delta \quad K \xrightarrow{\ell} K'}{X \xrightarrow{\ell} K'} (Rec) \\
\\
\frac{(R, I, P) \xrightarrow{\langle\langle \emptyset \triangleright \emptyset, \emptyset, \emptyset \rangle \triangleright R, I, P \rangle} (R, I, P) | P}{(R, I, P)} (Pro) \quad \frac{\frac{J \subseteq I \quad Q \subseteq R \quad J \cup Q \neq \emptyset}{(R, I, P) \xrightarrow{\langle\langle \emptyset \triangleright \emptyset, \emptyset, \emptyset \rangle \triangleright J, Q, \emptyset} (R, I, P)}}{(Inh)} \\
\\
\frac{M_1 \xrightarrow{\ell_1} M'_1 \quad M_2 \xrightarrow{\ell_2} M'_2 \quad \ell_1 \frown \ell_2}{M_1 \mid M_2 \xrightarrow{\ell_1 \cup \ell_2} M'_1 \mid M'_2} (Par) \quad \frac{M \xrightarrow{\langle\langle D \triangleright R', I', C \rangle \triangleright R, I, P \rangle} M' \quad R' \subseteq D \quad R \subseteq D \cup C}{[M] \xrightarrow{\langle\langle D \triangleright R', I', C \rangle \triangleright R, I, P \rangle} [M']} (Sys)
\end{array}$$

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

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.

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 pre-fixed 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 preconditions 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.

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 $\mathcal{A} = (S, A)$, 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.

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

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

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

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

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

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

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

386 The complete encoding of the above RS in BioResolve
387 syntax is reported in Fig. 19 in the Appendix. Using the

388 BioResolve directive `main_do(digraph)`, we can
389 automatically generate the underlying LTS as in Fig. 4: the
390 initial state is in light blue, while there is also a “bad” state
391 in which the student is banging the machine, shown in light
392 coral. Note that, as the entity `am` is initially not present, it
393 means that the initial time is in the afternoon. In drawing
394 the transition system, we have used the representational
395 convention explained in Remark 1.

396 In this particular example, we wish to analyse why `bang`
397 is produced, i.e., what are its causes. By manual inspection
398 we can recover a trace starting from the initial state and
399 leading to the “bad” state, e.g., $\xrightarrow{\text{idle}} \xrightarrow{\text{ccoin:nomilk}} \xrightarrow{\text{high-}}$
400 $\xrightarrow{\text{light}}$ (highlighted in red in Fig. 4). From this trace, using the dynamic
401 slicing process described in Brodo et al. (2024a), we can
402 reconstruct that the production of `bang` was due to the
403 prior production of `anger`, which in turn was caused by the
404 student getting espresso instead of cappuccino, which is

Fig. 3 Core type graph for reaction systems

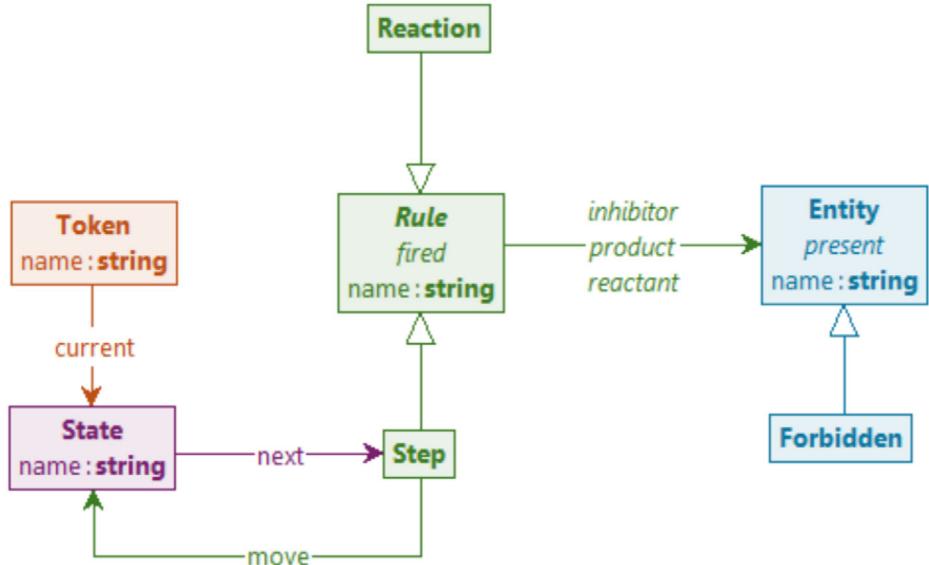
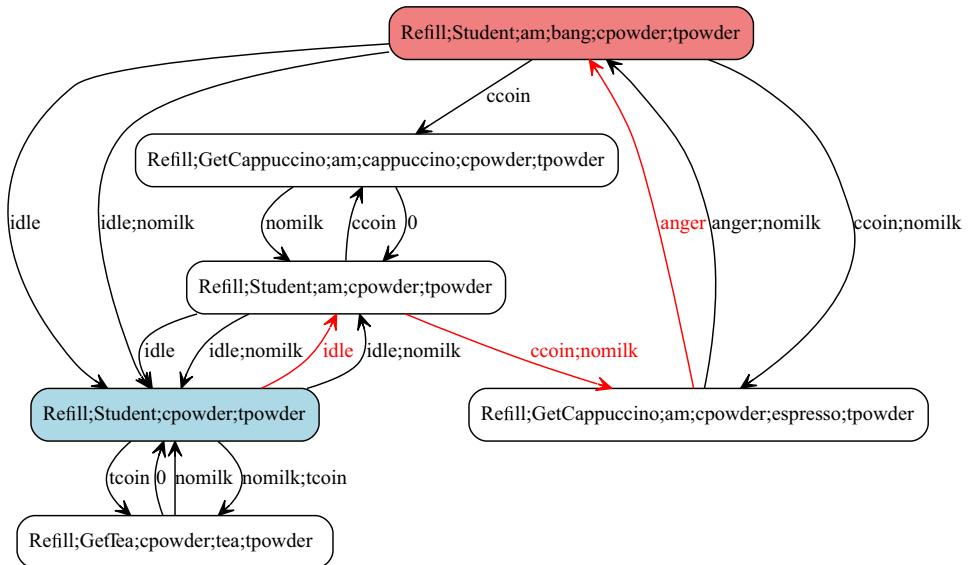


Fig. 4 LTS of the toy example.
For brevity we use the notation introduced in Remark 1, where node labels only account for the list of (separated by semicolons) current contexts and entities and transition labels carry the entities provided by the context



405 because there was no milk when a coin was inserted at
406 am.

407 **4 Encoding of RS in GT**

408 As an alternative to BioResolve, we investigate the use of
 409 graph transformation and GROOVE to generate the
 410 underlying LTS of a given Reaction System, on the basis of
 411 a start graph obtained by transformation from the RS
 412 specification. Because the start graph will typically include
 413 special **Entity** subtype, it comes together with an addi-
 414 tional type graph where those are specified. Depending on
 415 what one wants to analyse, the various strengths and
 416 capabilities of GROOVE then come into play.

417 For instance, one possibility is to use **GROOVE**'s
 418 model checking capabilities to check for temporal patterns
 419 of entity generation in the transition system. Another way
 420 to proceed is to focus on a given trace and build its *oc-*
 421 *currence graph*, which contains all the rule occurrences
 422 and entity instances present in that trace—analogous, in
 423 fact, to the way a Petri net process captures a particular
 424 behaviour. If the trace in question leads to a state in which

a **Forbidden** entity is present (such as the bang entity in our toy example), we can also *prune* the occurrence graph, again using graph transformation, to keep only those rule occurrences and entity instances that directly contributed to the existence of the forbidden entity.

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

Transform. The first step is a text-to-model transformation from a problem specification in BioResolve syntax into GROOVE syntax. This is achieved by running the `main_do(rs2gts)` directive of BioResolve, which produces two artefacts: firstly, an additional type graph, complementary to the one shown in Fig. 3, which specifies one subtype of **Entity** for each of the entities in the problem at hand (essentially for performance reasons: relying on dedicated types speeds up the matching step of GROOVE); and secondly (more importantly) a start graph in which the entire BioResolve system is encoded as suggested by Fig. 3. For the example system, the additional types as well as two self-explanatory fragments of the start graph are shown in Fig. 6.

Fig. 5 Reaction system exploration and analysis using GROOVE

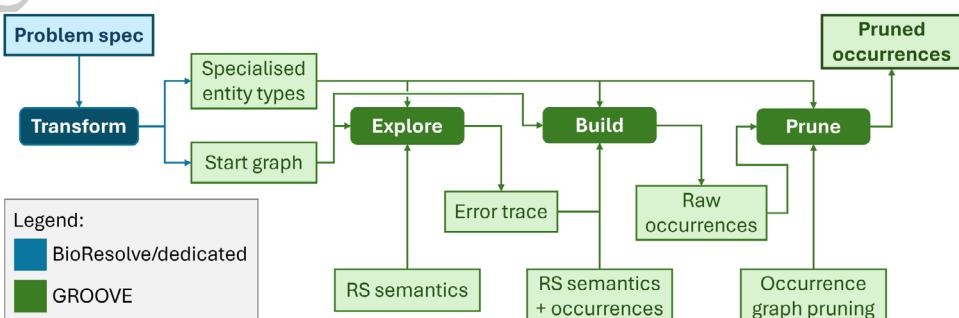
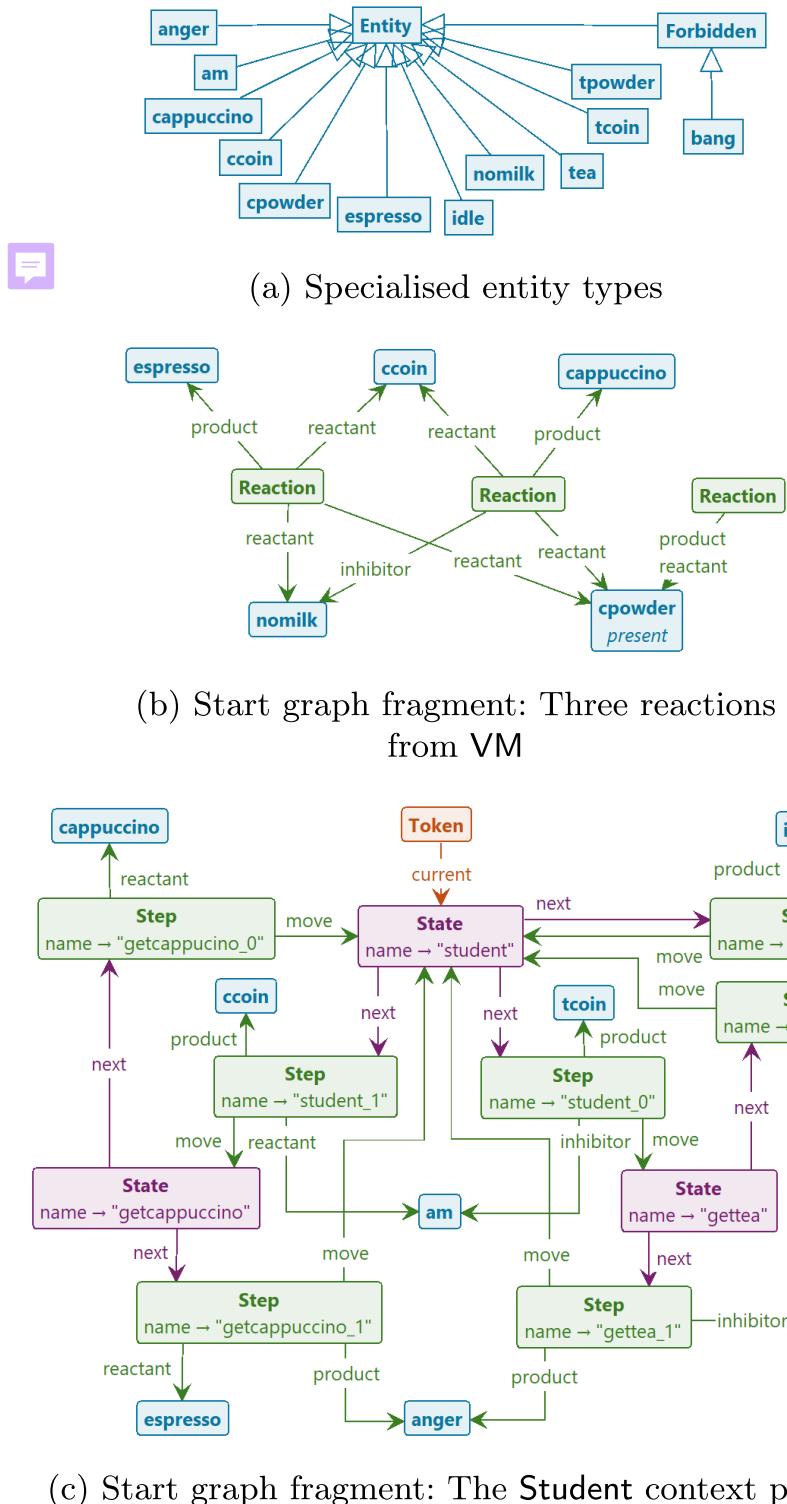


Fig. 6 Graph representation of running example

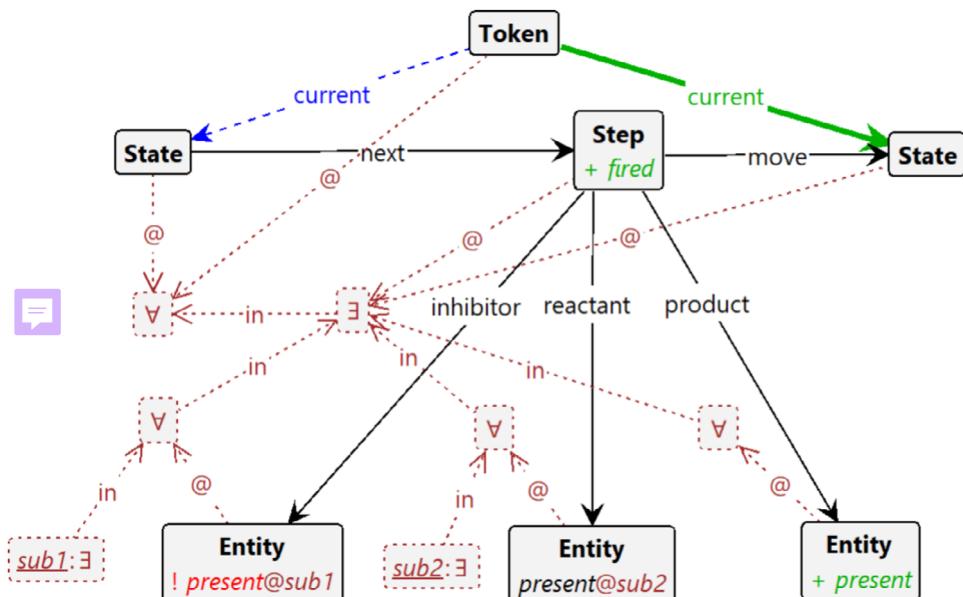


447 We claim that this transformation is semantics-preserving; Appendix 1 gives a sketch of the argument. A
 448 fully formal statement and proof of semantic correspondence,
 449 however, is outside of the scope of this paper.
 450

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

451
452
453
454

Fig. 7 Rule for context firing



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

```
recipe fire() {
    context; react;
}
```

where a “*recipe*” is the keyword for a transaction wrapping the body. With this in place, the GUI-based version of GROOVE produces the transition system displayed in Fig. 8 (which can also be exported to a range of standard formats), which is easily (visually) checked to be essentially 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:

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.

485

486

487

488

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

Fig. 8 GROOVE LTS of the toy example

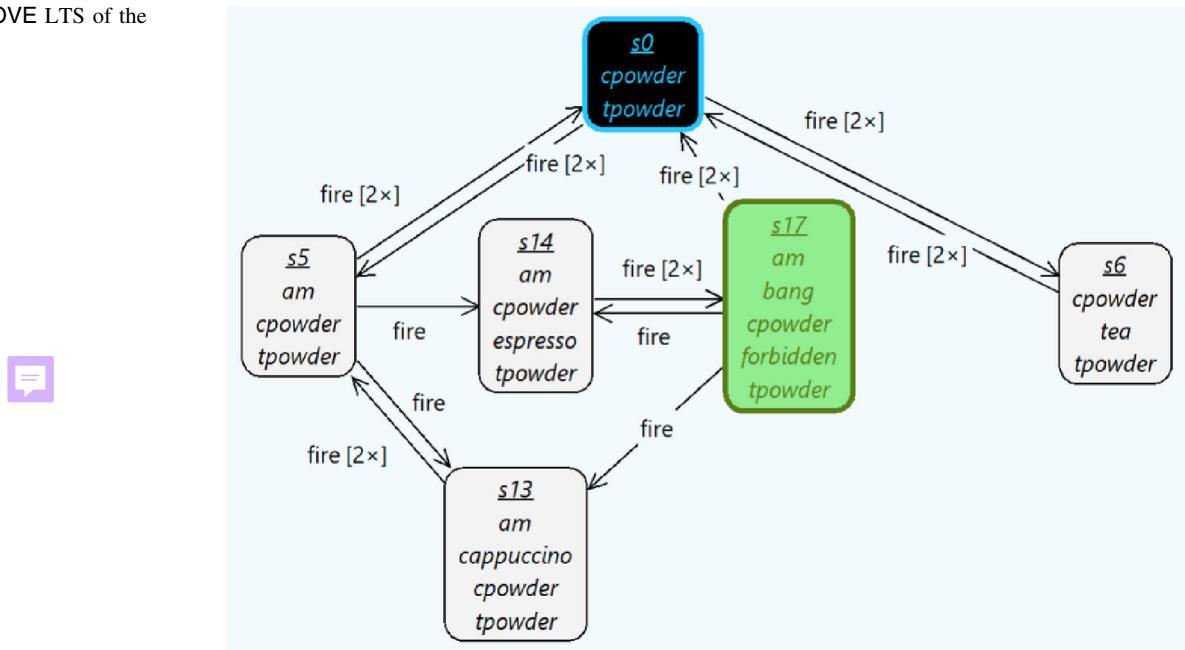
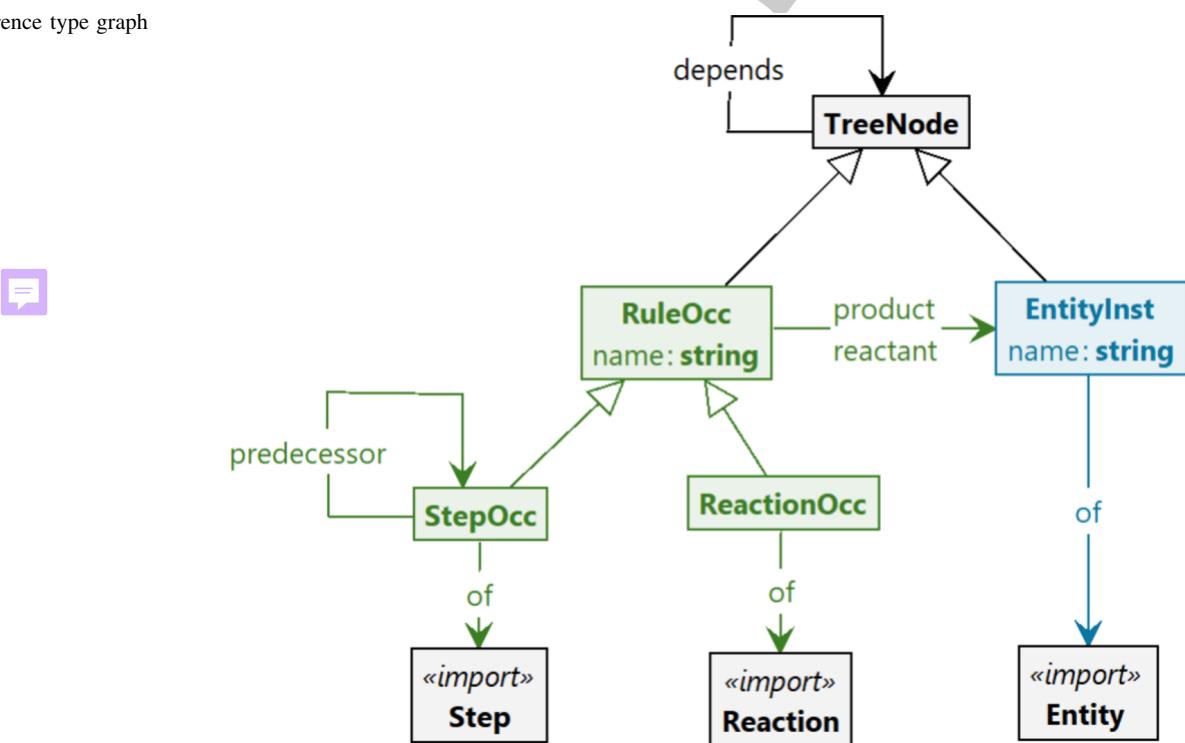


Fig. 9 Occurrence type graph



```

context;
fired("student_2");
fired("refill_1");
react;
context;
fired("student_1");
fired("refill_0");
react;
context;
fired("refill_1");
fired("getcappuccino_1");
react;

```

517

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

522 **Build.** The purpose of this phase is to build an occurrence
 523 graph that explains how **Forbidden** was produced,
 524 by collecting its (transitive) dependencies (Brodo et al.
 525 2024b). Concretely, we record the following dependencies:

- 526 • From each non-initial **Entity** instance to the **Rule**
 527 occurrence of which it is the **product**;
- 528 • From each **Rule** occurrence to all its reactant **Entity**
 529 instances;
- 530 • From each **Step** occurrence to all directly preceding
 531 **Step** occurrences.

532 Figure 9 shows the occurrence type graph.

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

543 Note that all this is restricted to *positive* dependencies.
 544 In fact, we have constructed our example so that there are
 545 no inhibitors in the **Rules** that fire in the trace above; if we
 546 would rely on an entity `milk` that inhibits the production of
 547 `espresso`, rather than on `nomilk` as a reactant, the occur-
 548 rence graph for `bang` would not include `milk`; and likewise
 549 if we would use `cappuccino` as an inhibitor for `anger`
 550 rather than `espresso` as a reactant for it. The representa-
 551 tion of negative dependencies is a research question in its
 552 own, and is outside the scope of this paper (although a
 553 possibility to capture negative dependencies is to focus the

analysis on the Positive RS that results from the transfor-
 554 mation defined in Brodo et al. 2024b).

555 As indicated in Fig. 5, the occurrence graph is produced
 556 by another GROOVE rule system, using the same start
 557 graph but driven by the control program previously created
 558 in the explore phase, which encodes (as we have seen) the
 559 trace to the undesirable state. The occurrence graph
 560 semantics consists of rules with the same names (**react**,
 561 **context** and **fired**), but different functionality: in partic-
 562 ular, rather than manipulating *present* flags, the **react** rule
 563 now creates **RuleOcc**- and **EntityInst**-nodes together
 564 with their dependencies. This is a non-trivial procedure that
 565 in fact itself requires several successive stages. Though the
 566 details of these stages are not of sufficient interest to
 567 include in this paper, we want to point out that breaking
 568 down a single rule (**react**, in this case) into multiple stages
 569 would seem to contradict the tenet of algebraic graph
 570 transformation that a rule embodies a single, atomic change
 571 to a graph. This contradiction is solved by relying once
 572 more on GROOVE recipes: in the occurrence graph
 573 semantics, **react** is actually not a rule but a recipe, defined
 574 as

```

recipe react() {
    entities-age;
    react-produce;
    merge;
}

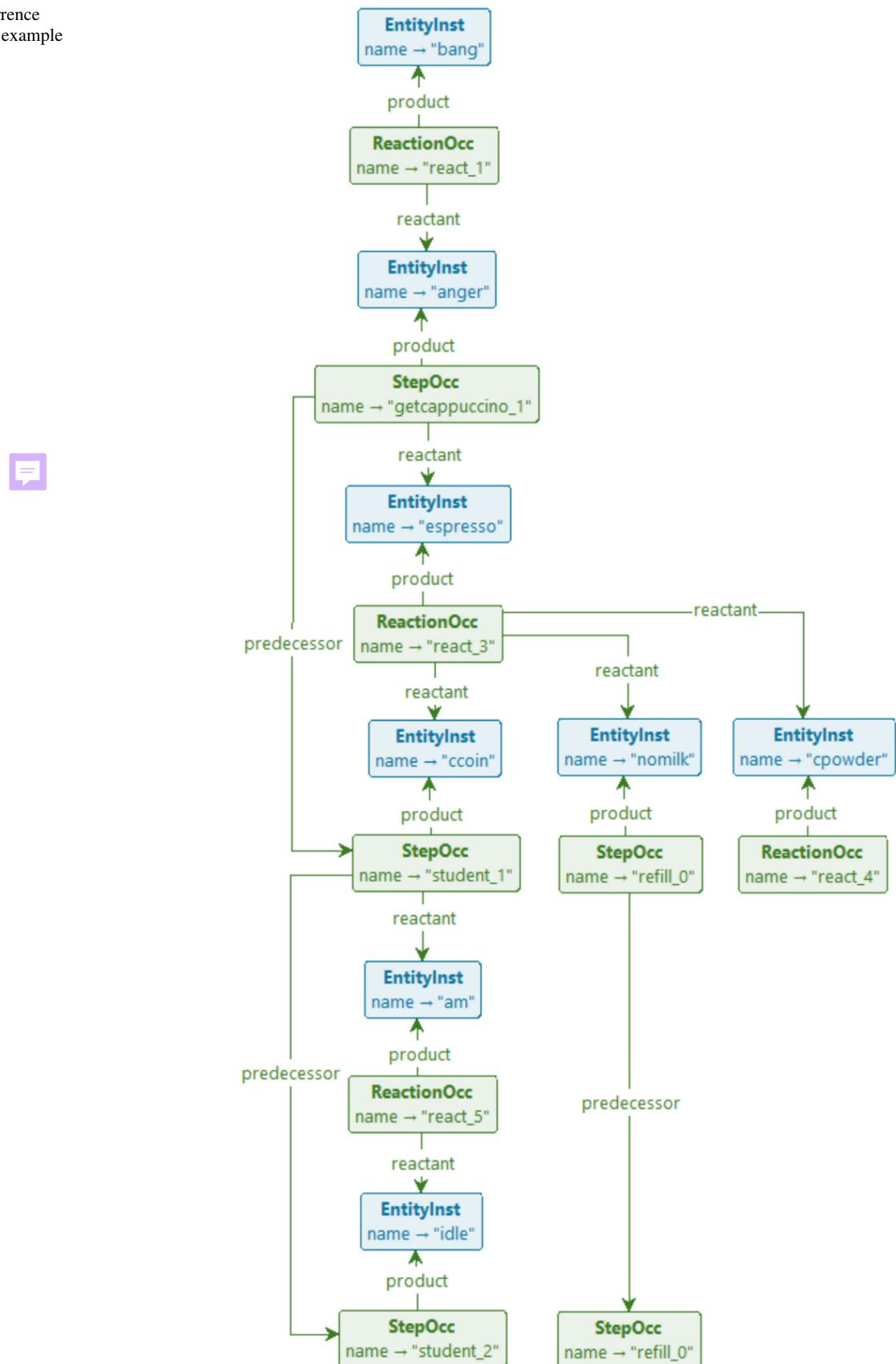
```

575 of which the three atomic steps perform the necessary
 576 bookkeeping to correctly produce the occurrence graph.

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

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

Fig. 10 Pruned occurrence graph of the running example

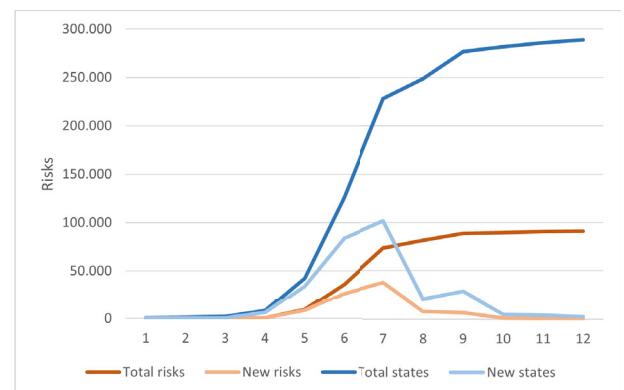


Measurement	Search method	Count	Time (s)
Total states	BFS	309 798	3 368
Total states	DFS	309 798	2 464
Major risks	DFS	91 113	2 447
Moderate risks	DFS	97 805	1 534
Maj&Mod risks	DFS	61 976	2 727
Minor risks	DFS	24	2 218

(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

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 “Auxiliary material for the GROOVE experiments” section.

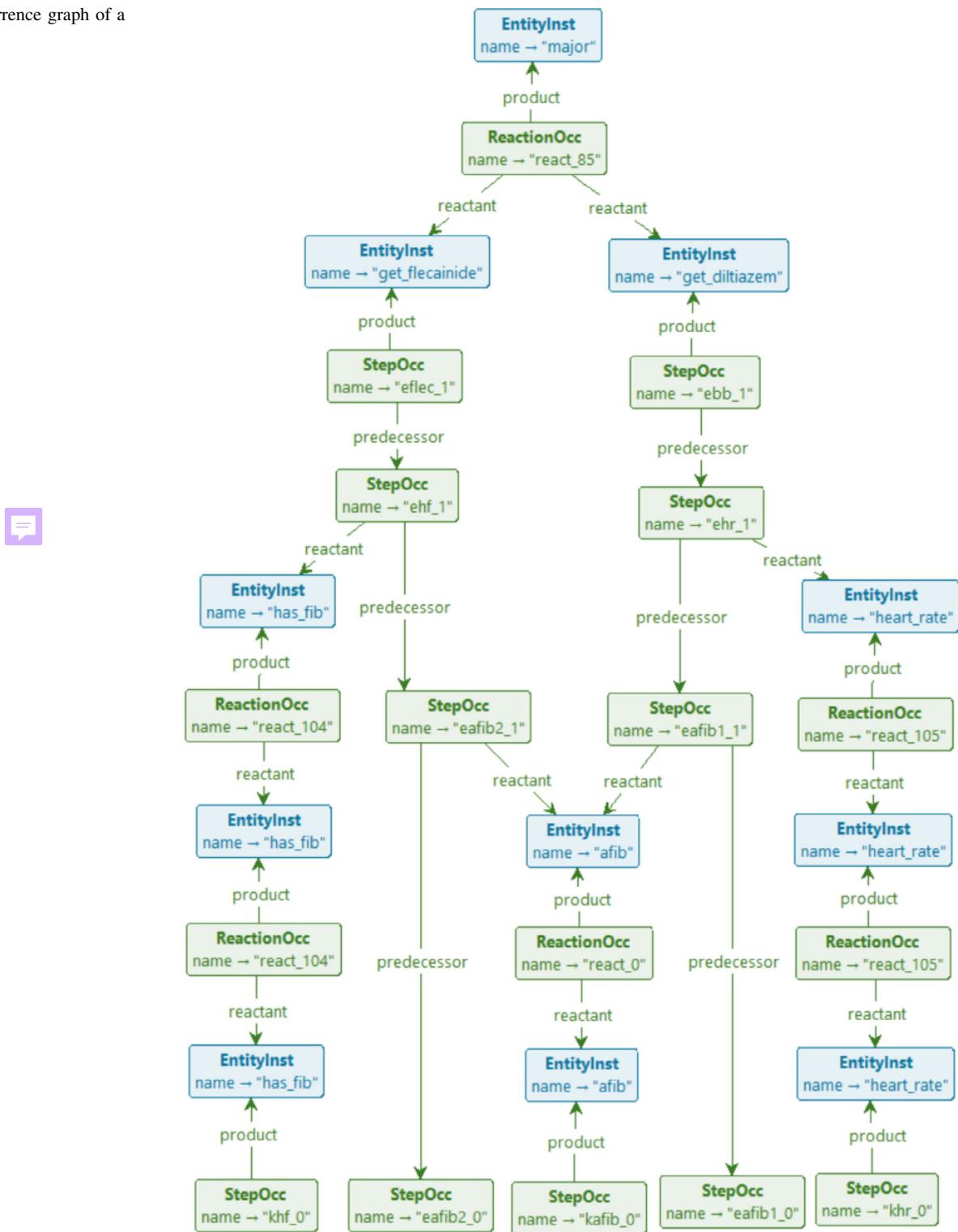
5.1 Comorbidity treatment analysis

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.

Fig. 12 Occurrence graph of a major risk



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

665 For each drug d that appears in the therapies, we con-
 666 sider three corresponding entities get_d , $stop_d$ and d : the
 667 first represents the prescription of d by the doctor, the
 668 second the removal of d from the current treatment and the
 669 third the intake of the drug by the patient. For handling



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)

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 $K_{f_1} | \dots | K_{f_n}$. 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?

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 the total number of configurations containing a **Forbidden** (i.e., major or minor) entity, and there may very well be entities whose presence or absence does not causally contribute to that **Forbidden** entity — in other words, which would not appear in its occurrence

graph. Configurations counted as separate risks may well reduce to the same causation. To analyse this further, one would have to construct (and prune) the occurrence graphs for all risk configurations, and compare them on that basis. Though this is beyond the scope of this paper, such an analysis is in principle straightforward to carry out in **GROOVE**— it is a matter of combining the three steps in Fig. 5 into a single rule system.

The line of Fig. 11a headed “Maj&Mod risks” reports the number of configurations at which *both* a major and a moderate entity appear during the same step; hence, these are counted as both major and moderate risks (partially explaining their high numbers).

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

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

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

We can also use **GROOVE** to replicate the findings of Bowles et al. (2024, Fig. 6) in terms of the relation between patient profiles and risks, using model checking. Recall that the reaction system starts by having the context produce initial entities, and in this particular case study, the first move of the context is to select a patient profile; hence the initial state has $2^9 = 512$ outgoing transitions, whose target state corresponds to the chosen profile. Moreover, the rule forbidden tests for the presence of a **Forbidden** entity in a state. Therefore, a formula of the shape $\text{AX}(\bigwedge_i f_i \rightarrow \text{EF } \text{forbidden})$, where each of the f_i specifies the presence or absence of a patient feature, specifies whether all patient profiles with that combination of features contain a potential risk.

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

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

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

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

Discussion. Compared to the prior results in Bowles et al. (2024), the advantages of using **GROOVE** lie in performance and flexibility:

- The time needed to analyse the entire state space is around 41 min for **GROOVE**; while not particularly fast, this still compares very favourably to the 5 h needed by **BioResolve** for LTS generation.
- Finding shortest paths to risk configurations and computing occurrence graphs is part of the core functionality of **GROOVE**— given, of course, suitable rule systems that encode the chosen notion of causal dependency.
- The CTL check can be used to immediately confirm the outcome of the slicing algorithm.

5.2 Protein signalling networks analysis

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

⁴ <https://maude.cs.illinois.edu>.

Fig. 14 Additional rules and condition for detecting steadiness

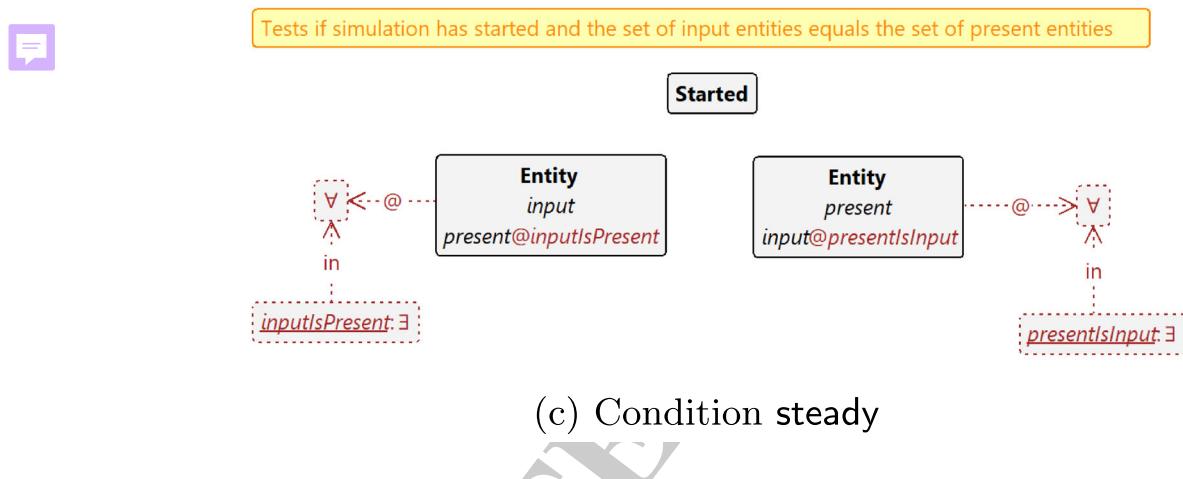
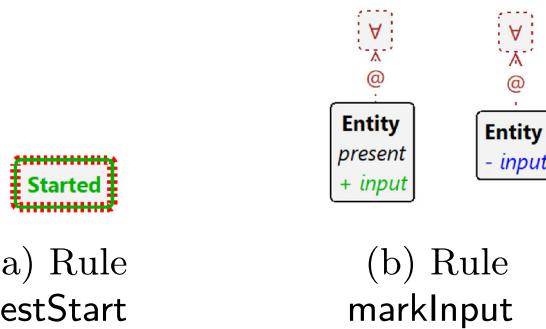
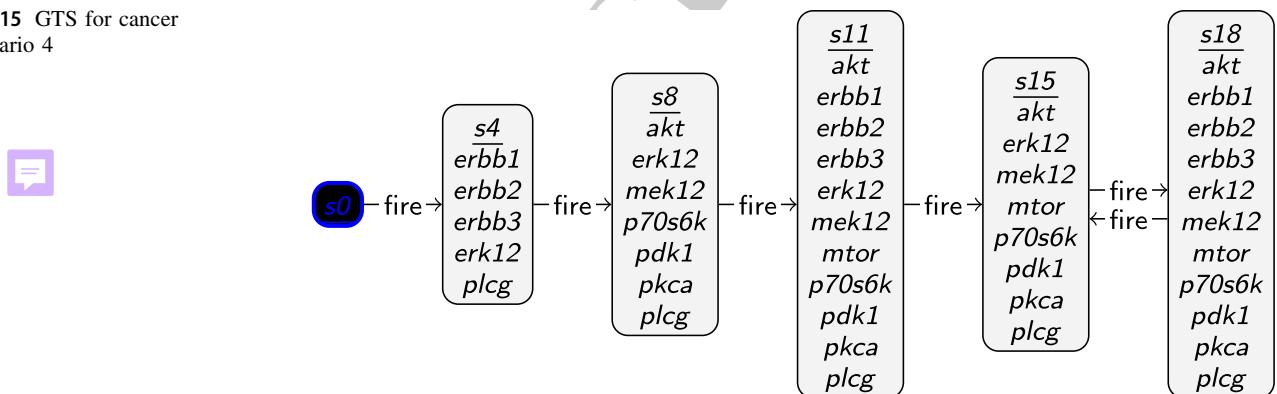


Fig. 15 GTS for cancer scenario 4



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

Analysis goals. The goal of the analysis is to validate or refute some behavioural hypotheses of RSs. 842
 843

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. 844
 845
 846
 847
 848

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 849
 850
 851
 852
 853

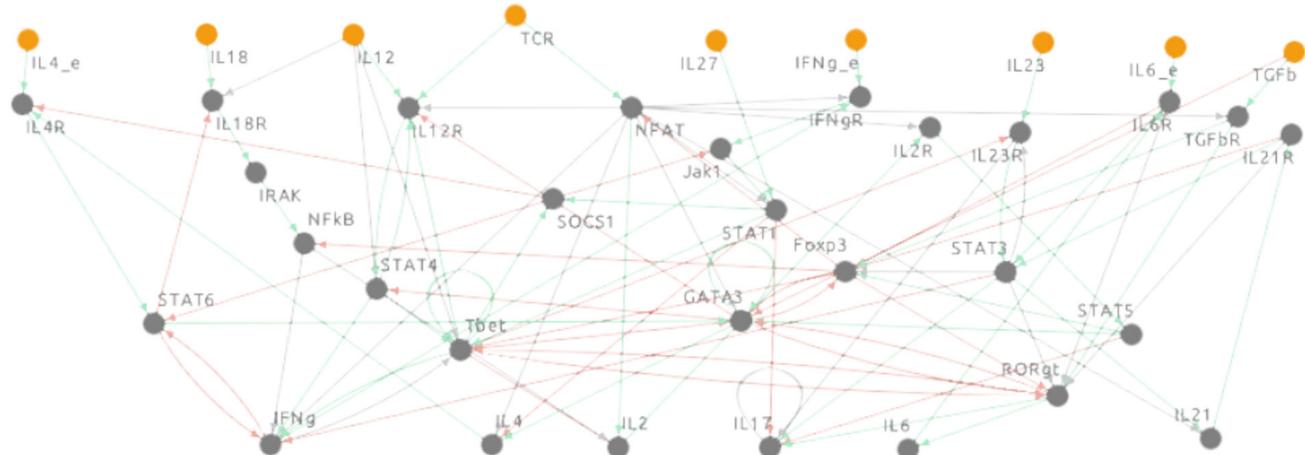


Fig. 16 Graphical representation of the Boolean network model of T cell differentiation from Puniya et al. (2018)

544 (with guarded contexts) to **GROOVE** presented in the
545 previous section, i.e., no preprocessing is necessary. The
546 **BioResolve** specification is in Fig. 22 in the Appendix.
547 The following properties have been experimented with:

- 548 1. searching for the presence/absence of the attractor
549 akt in steady states of the BT747 cell line, where the
550 context $[k, ket]$ is considered;
- 551 2. in order to observe the interactions when either
552 erlotinib or pertuzumab are supplied, the context
553 $\{\{e, egf, hrg\}.korep\}$ is considered and Maude
554 reports that there exists at least one path where that
555 treatment is successful, but not all paths avoid a steady
556 state where akt is present;
- 557 3. using the context $[k, korept]$, it is shown that,
558 regardless the drug used, once pdk1 is present,
559 inevitably the steady state includes akt; and that
560 pdk1 never appears before erbB1 is produced (which
561 basically means that pdk1 is a product of the
562 activation of the erbB1 receptor);
- 563 4. finally, using the context $[k, kge]$, it is shown that by
564 permanently providing the drug erlotinib and the
565 stimulus (egf and hrg), the attractor akt is never
566 produced. Moreover, Maude checks that the production
567 of akt can be also inhibited by providing erlotinib
568 only when receptors erbB1 and erbB2 are active.

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

577 paths prevent reaching a steady state in which a AKT is
578 present?

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

583 The main challenge in replicating the results is that
584 some of the properties to be checked are formulated in
585 terms of *steady states* of the reaction system, which are
586 essentially one-state attractors, that is, states in which the
587 context and reactions together reproduce exactly the enti-
588 ties of that state again. Though **GROOVE** detects such a
589 loop as a matter of course, it is a structural property of the
590 LTS and not a state property available for model checking.
591 In order to be able to reason about steadiness, we have to
592 remember *input entities*, i.e., entities that were present in
593 the source state, and compare them to *present entities*, i.e.,
594 those that have been produced in the target state. Moreover,
595 we should not accidentally mark the start state as steady
596 even if it has neither inputs nor present entities. Figure 14
597 shows the additional rules that achieve this, together with
598 the modified recipe defined by

```
599 recipe fire() {
600   try testStart; markInput; context; react;
601 }
```

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

- 604 1. Given the context $[k, ket]$, **GROOVE** confirms the
605 status of the following LTL properties:
606
 - 607 • FG(steady = \exists akt) is not satisfied; **GROOVE**
608 produces a counter-example.
 - 609 • G(erbB2 = \exists X(erbB2)) is satisfied.
- 610 2. Given the context $\{\{e, egf, hrg\}.korep\}$, **GROOVE**
611 confirms the status of the following CTL properties:
612

- 923 • EF(steady&!akt) is satisfied;
 924 • AF(steady&!akt) is not satisfied.
- 925 3. Given the context [k, korept], GROOVE confirms
 926 the status of the following LTL properties:
 927 • G(pdk1 —> FG(steady —> akt)) is satisfied;
 928 • erbb1 R !pdk1 is satisfied.
- 929 4. Given the context [k, kge], GROOVE confirms the
 930 status of the following CTL property:
 931 • EGEF(steady —> !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 —> !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



966 *Discussion.* Compared to the prior results in Ballis et al.
 967 (2024), the advantages of using GROOVE lie in the
 968 combination of visual inspection and automatic model
 969 checking. Not only were we able to confirm the findings of
 970 the original paper using exactly the same encoding of
 971 reaction system as for the previous case, but the ability to
 972 inspect and visualise the state spaces also gives additional

973 insights, such as the observation above that steadiness as
 974 formalised there does not actually capture the intended
 975 notion of being an attractor. 976

5.3 T cell differentiation analysis

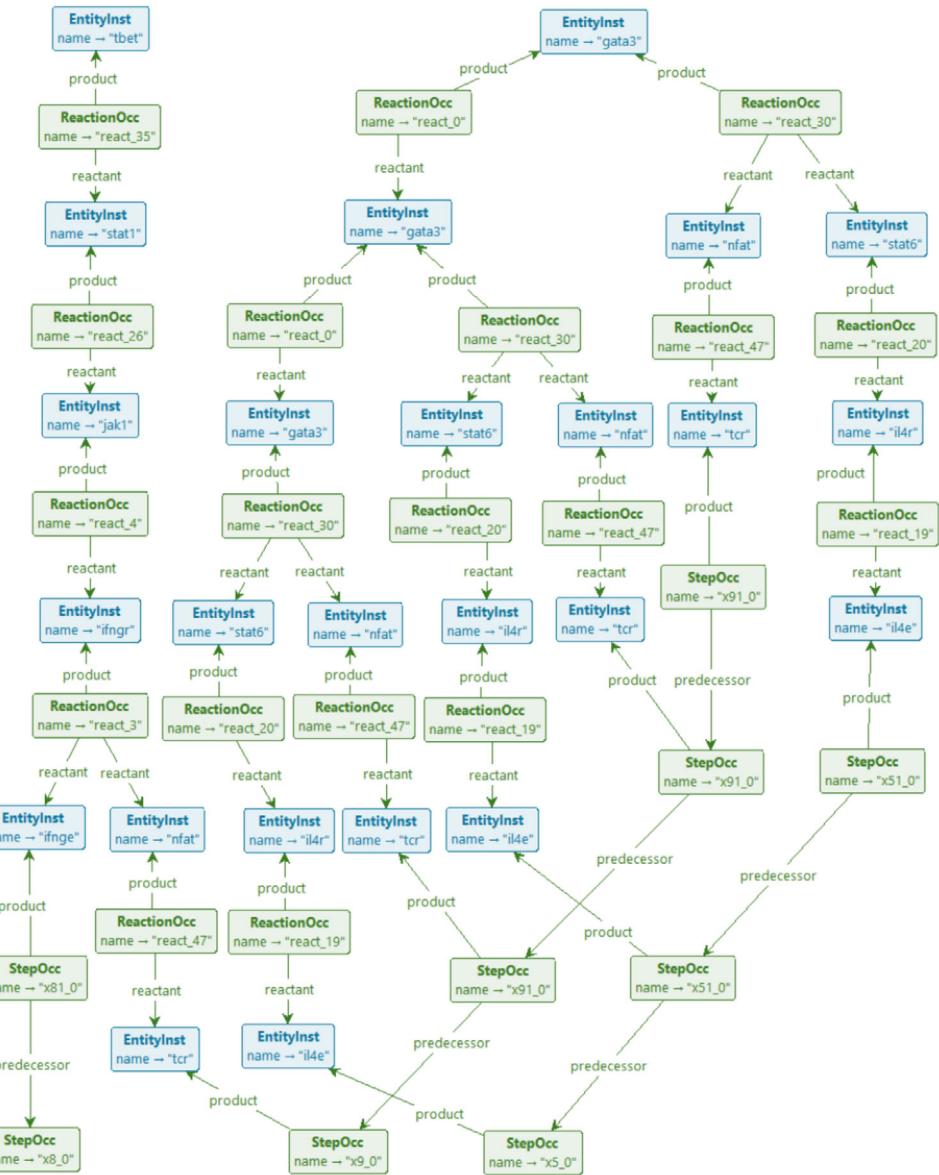
The paper Brodo et al. (2025b) (being the full and corrected version of Brodo et al. (2025a), see also Footnote 5) exploits Reaction Systems to analyse T cell differentiation in the immune system, a widely studied biological phenomenon. The starting point for the analysis is a Boolean network model; several of those are available as a Saez-Rodriguez et al. (2007), Thakar and Albert (2010) and 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`, `rorgt` and `foxp3`, respectively. There exists experimental and computational evidence that a T cell can manifest more than one phenotype (Luckheeram et al. 2012; Puniya 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 `rorgt` 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 a subset of external stimuli is chosen and then provided at each subsequent

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



1024 step, inevitably causing the RS to end up in a loop (called
1025 an attractor).

1026 *Experimental set up.* The translation from Boolean
1027 networks to RS consists in turning every update formula
1028 into disjunctive normal form. Then, every clause of the
1029 disjunction produces a reaction in which (i) reactants are
1030 the positive atoms, (ii) inhibitors are the negated atoms and
1031 (iii) the updated variable forms a singleton product. The
1032 translation from Boolean network to BioResolve syntax is
1033 done using the directive `main_do(bn2rs)`. For the
1034 readers' convenience, all update formulas and the resulting
1035 reactions are reported, respectively, in Fig. 22 and in
1036 Fig. 24 in the Appendix. For example, the update formula
1037 for IL12R is

(IL12&NFAT) | (STAT4& \neg GATA3) | Tbet | (TCR& \neg GATA3)

which yields the four reactions⁵

$\{il12, nfat\}, \emptyset, \{il12r\}$	$\{\text{stat4}\}, \{gata3\}, \{il12r\}$
$\{\text{tbet}\}, \emptyset, \{il12r\}$	$\{\text{tcr}\}, \{gata3\}, \{il12r\}$

The RS context can choose any combination of environmental stimuli that will then persist, i.e., for each possible



1039

1041
1042

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,il12r) was mistakenly written as il2r, omitting the digit 1. Unfortunately, since il2r 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.

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

- $\neg G(F gata3 \& F tbet)$ (separate expression)
- $\neg GF(gata3 \& tbet)$ (simultaneous expression)

Using the start graph derived from **BioResolve** using the process outlined in Fig. 5, both of these formulas yield counterexamples, meaning that $tbet$ and $gata3$ can in fact be (recurrently) expressed simultaneously. Using a variation of the process outlined in Sect. 4, we can once more visualise a trace leading to such a recurrent state. The variation lies in the fact that, this time, we do not want to show the causal history of a *single* forbidden entity, but rather of the combination of two distinct entities. Fortunately, this is just a matter of creating another rule, $gata3$

$\neg tbet$, which applies precisely when $gata3 \& tbet$ holds. On this basis we can go through the steps outlined in Fig. 5, resulting in the occurrence graph displayed in Fig. 17.

This complements the observation embodied in Brodo et al. (2025b, Fig. 7) that for the combination of $tbet$ and $gata3$, the context has to provide the stimuli $ifnge$ (produced here by the **StepOcc** named $x81_0$), tcr (repeatedly produced by **StepOccs** named $x91_0$) and $il4e$ (also repeatedly produced, by **StepOccs** named $x51_0$). In more detail, we see that $tbet$ derives, in a linear sequence of four **ReactionOccs**, from $ifnge$ and tcr , whereas $gata3$ derives, in 3 successive combinations of simultaneous **ReactionOccs**, from tcr and $il4e$. Moreover, the genes produced along the way are precisely the ones reported in Brodo et al. (2025b, Fig. 8(a)), using the slicing algorithm of that paper, as being relevant for the expression of $tbet$ and $gata3$.

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

Discussion Compared to the results in Brodo et al. (2025b), the advantages of **GROOVE** are threefold (reiterating the observations made for the previous two cases):

- LTL model checking allows to express, in a flexible manner, the scenarios one wants to investigate;
- The occurrence graph visualisation offers analysis possibilities beyond the outcome of the slicing algorithm;
- The performance of **GROOVE** is an order of magnitude better than that of **BioResolve**.

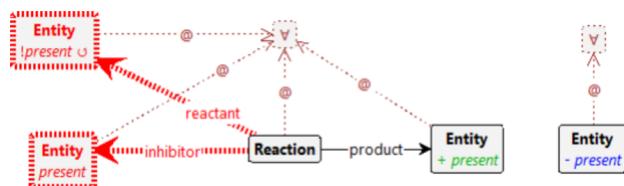


Fig. 18 Rule for reaction firing

Table 1 State space generation for mutual exclusion

Case #	States #		Time s		Memory MB	
	#	Ratio	Ratio		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	32,767	2.1	5146.1	2.7	8588	1.8
13	69,631	2.1	15,647.3	3.0	13,640	1.6



1130

6 Comparison with existing tools

1131 The approach presented in this paper can provide several
1132 advantages over existing tools in the literature.⁶

- 1133 • **brsim**⁷ (Basic Reaction System Simulator, written in
1134 Haskell and distributed under the terms of GNU GPLv3
1135 license) (Azimi et al. 2015) was the first RS simulator
1136 to be made publicly available. Given the reactions of
1137 the RS and a context sequence, brsim is capable of
1138 computing the resulting sequence and generating additional
1139 annotations for each computation step, such as
1140 the enabled reactions. Alternatively, brsim can be
1141 executed in an interactive mode, allowing the user to
1142 manually provide the context to be used at each step.
- 1143 • **WebRSim**⁸ is a basic RS simulator that makes all
1144 functionalities of brsim available through a friendly
1145 web interface (Ivanov et al. 2018);
- 1146 • **HERESY**⁹ is a Highly Efficient REaction SYstem
1147 GPU-based simulator, developed using CUDA (Nobile
1148 et al. 2017). It features a user-friendly GUI and is
1149 designed to exploit the high degree of parallelism
1150 offered by modern GPUs to handle very large-scale RSs
1151 simulations.
- 1152 • **c1-rs**¹⁰ is an optimised Common Lisp simulator for
1153 RSs (Ferretti et al. 2020) that can exhibit performances
1154 comparable with the GPU-based simulator HERESY.

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/c1-rs>.

```

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

myreactions([
    react([idle],[am],[am]), % list of reactions
    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("[ % context definitions
    refill = ({nomilk}.refill
        + {}.{refill}),
    student = (?{am},{}.{tcoin}?.gettea
        + ?{am},{}.{ccoin}?.getcappuccino
        + {idle}.student),
    gettea = (?{tea},{}.{tea}?.student
        + ?{tea},{}.{anger}?.student),
    getcappuccino = (?{cappuccino},{}.{tea}?.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

This is achieved by discarding all reactions that cannot produce effects and by encoding RS evolution in terms of matrix–vector multiplications and vector additions. 1155

- **BioResolve**¹¹ is a Prolog interpreter for Reaction System analysis, first proposed in Brodo et al. (2021) 1156 and later extended in a series of papers to deal with 1157 enhanced features, like delays, duration, monitoring, 1158 slicing and guarded contexts (Brodo et al. 2023, 2024a; 1159 Bowles et al. 2024). Many capabilities of BioResolve 1160 have been discussed at length in the previous sections. 1161
- **ccReact**¹² is an interacting language for Reaction 1162 Systems based on Maude 3.2.1 (Ballis et al. 2024), 1163 whose key features have been illustrated in Sect. 5.2. 1164
- **ReactlCS**¹³ is a Reaction Systems Verification Toolkit 1165 that consists of two main modules for model checking 1166 temporal properties expressed in logical languages 1167 tailored to Reaction Systems: one that exploits binary 1168 decision diagrams (BDD) and bounded model checking; 1169 the other that translates verification problems in 1170 rsLTL (Meski et al. 2015) into satisfiability modulo 1171 theories (SMT) (Meski et al. 2024). 1172

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, 1173

11 Available at <https://www.di.unipi.it/~bruni/LTSRS/>. 11FL01

12 Available at <https://depot.lipn.univ-paris13.fr/olarte/reaction-systems-maude>. 12FL01

13 Available at <https://github.com/arturmeski/reactics/>. 13FL01

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

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

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

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

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, WebRSim, and c1-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 levels, parameter and reaction synthesis, as well as parameter optimisation.

Performance and scalability. The ability of GROOVE to explore large state spaces efficiently is central to our method. Through configurable exploration strategies and a

rule-based control mechanism, GROOVE handles complex RS instances that involve thousands of reachable configurations. Our experiments demonstrate a substantial improvement in analysis time compared to BioResolve, often reducing execution time by an order of magnitude. Furthermore, the performance of BioResolve is strongly influenced by the nature of Prolog evaluation strategies, which can lead to excessive memory and time consumption in large case studies. With respect to **ccReact**, the paper (Ballis et al. 2024) on which we based our experiments does not provide information on performance, and indeed the system studied there is so small that no useful comparison could be made on that basis. Since Maude, underlying **ccReact**, is a long-standing and mature tool, it would be interesting to investigate this in more detail; however, we leave this to future work. In contrast, other tools either consider linear executions only and do not scale to large models or lack optimisation strategies necessary for handling non-trivial state spaces.

Contrary to GROOVE, which is a general purpose, explicit-state tool, not optimised for Reaction Systems,

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



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 $\text{kafib} | \text{khf} | \text{khr} | \text{kcons} | \text{kage} | \text{kdiabete} | \text{kdoacint} | \text{khyper} | \text{korigin} | \text{k_doac}$

1223 ReactICS can take advantage of symbolic representations
1224 to abstract much larger state-spaces. For instance, for the
1225 mutual exclusion protocol reported in Meski et al. (2015)
1226 and Nobile et al. (2017), ReactICS can cope with 37–54
1227 processes (depending on the precise analysis) in a matter of
1228 hours; in contrast, in comparable time GROOVE can
1229 exhaustively generate the state space of up to 13 processes.
1230 Even though the model checking queries in ReactICS
1231 probably do not require full state space exploration, and

1232 hence the comparison is skewed, the dedicated methods of
1233 ReactICS clearly pay off.

1234 *Causal analysis and verification.* A key distinguishing
1235 feature of our approach is the ability to perform graph-
1236 based *causal slicing*. By automatically generating and
1237 pruning *occurrence graphs*, GROOVE provides detailed
1238 and visual explanations of how specific states, such as
1239 those involving undesirable or forbidden entities, are
1240 reached. This form of causal reasoning is not available in

high-performance tools such as HERESY, WebRSim, cl-rs, or ReactICS and is only partially addressed in ccReact, where the focus is primarily on reachability and temporal properties. GROOVE's integrated support for CTL and LTL model checking further extends its applicability to behavioural verification, enabling the specification and validation of complex temporal properties.

Summary. In conclusion, the combination of expressive modelling, efficient state space exploration, and integrated causal analysis makes GROOVE a powerful and versatile full-fledged platform for the study of Reaction Systems. It not only generalises and extends existing tools, but also opens the door to new forms of analysis that were previously impractical or unsupported.

7 Conclusion and future work

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

We have used GROOVE to revisit several case studies from the literature and our experimental results, although preliminary, are promising: GROOVE not only supports complex RS features such as guarded, nondeterministic and recursive contexts but also significantly improves performance and flexibility compared to existing RS tools. Moreover, the use of GROOVE's recipes and model-checking capabilities opens the door to sophisticated analyses that were previously impractical.

A number of interesting avenues for future research remain open, among which we mention the possibility to extend the methodology to support quantitative RS variants with durations or weights (Brodo et al. 2023); investigate alternative notions of causality and their representation in graph-based semantics, like dependencies drawn from inhibitors rather than reactants;¹⁴ apply the GROOVE toolset to further biological case studies, like those available in the CellCollective public repository (Helikar et al. 2012), possibly exploiting an automated pipeline for analysis.

Overall, the results show that graph transformation, and GROOVE in particular, provide a robust and scalable foundation for the specification, execution, and analysis of Reaction Systems.

One option that we have ignored throughout the paper deserves a brief mention here. The slicing algorithms reported in Bowles et al. (2024) and Brodo et al. (2025a) on the basis of BioResolve are actually implemented as stand-alone scripts that operate on a .dot-formatted LTS.

```
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],[],[mek12]),
    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} ]").
```

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

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

```

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)
TGFbR = TGFb and NFAT
RORgt = TGFbR 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 = (TGFbR 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)

Given that GROOVE can also produce LTSs as .dot files, an alternative way to benefit from its superior performance might be merely to replace BioResolve in the tool chain used previously. For this to be possible, the LTS labelling information 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.

8 Supplementary information

For replication of the GROOVE experiments, we provide an archive with supplementary material, described in “Auxiliary material for the GROOVE experiments” section.

Appendix 1: 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 \mathcal{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.

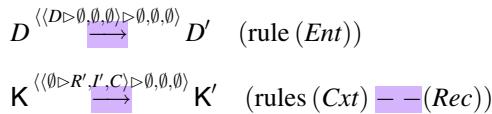
The variable part consisting of *present* flags on **Entity** nodes and a *current-edge* from every **Token** to a **State**. As observed in Remark 1, any RS process $[M]$ can be written in the form $[Rs \mid Ks \mid D]$ where: (i) $Rs = \prod_i (R_i, I_i, P_i)$ is the parallel composition of all reactions 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 .

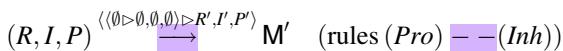
We claim that this equivalence establishes a bisimulation between the GROOVE and RS state spaces. (It is not an isomorphism, because the RS semantics for K “consumes” the process (rules $(Cxt)-(Rec)$ in Fig. 2) whereas G keeps the automaton intact, merely moving the current pointer.)

To prove this claim, we have to show correspondence of the transitions. Concretely, to any transition carrying the label $\langle \langle D \triangleright R', I', C \rangle \triangleright R, I, P \rangle$, 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).

1369 Both of these rules are universally quantified. Roughly
 1370 speaking, referring to Fig. 2, context simultaneously
 1371 encodes all sub-transitions of the form



1373 as well as their composition and filtering (rules (*Par*)—
 1374 (*Sys*)). **react** in turns simultaneously encodes all sub-tran-
 1375 sitions of the form



1377 together with their composition with the **context**-transi-
 1378 tions (rule (*Par*)) and filtering (rule (*Sys*)). (The *fired-flags*
 1379 on the **Step**-nodes, added by **context** and removed again
 1380 by **react**, play no role in this correspondence: for the
 1381 purpose of the equivalence of the GROOVE and RS
 1382 semantics, they might be omitted entirely.)

1383 Appendix 2: Mutual exclusion

1384 Table 1 shows the GROOVE performance in generating
 1385 the full state space of the mutual exclusion example of
 1386 Meski et al. (2015); Nobile et al. (2017), discussed in
 1387 “Comparison with existing tools” section.

1388 Appendix 3: Auxiliary material

1389 Auxiliary material for the toy running example

1390 The BioResolve specification for the toy running example
 1391 about the interaction between the student and the vending
 1392 machine is reported in Fig. 19. The corresponding RS has
 1393 been described in “Running example: a toyvending
 1394 machine” section and it has been used to illustrate some
 1395 key features of the GROOVE encoding in “Encoding of
 1396 RS in GT” section.

1397 Auxiliary material for the comorbidity case study

1398 The RS specification for the comorbidity case study pre-
 1399 sented in Bowles et al. (2024) is reported in Fig. 20 (set of
 1400 reactions) and Fig. 21 (context processes definitions),
 1401 where we assume the initial state is $D_0 = \emptyset$. The corre-
 1402 sponding experimentation with GROOVE has been dis-
 1403 cussed in Sect. 5.1.

```
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,nfkb,stat4],[stat3,stat6],[ifng]),  
    react([nfkb],[],[ifng]),  
    react([tbet],[stat3],[ifng]),  
    react([ifng,nfat],[],[ifngr]),  
    react([ifnge,nfat],[],[ifngr]),  
    react([i12,nfat],[],[i12r]),  
    react([stat4],[gata3],[i12r]),  
    react([tbet],[],[i12r]),  
    react([tcr],[gata3],[i12r]),  
    react([i17,i123r,stat3],[stat1,stat5],[i117]),  
    react([rorgt],[stat1],[i117]),  
    react([i12,i118],[stat6],[i118r]),  
    react([nfat,nfkb],[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 Sect. 5.3

Auxiliary material for the protein signaling networks case study

1404
1405

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.

1406
1407
1408
1409

Auxiliary material for the T cell differentiation case study

1410
1411

The BioResolve specification derived from the Boolean network model (available at Puniya 2024, see Fig. 23) of

1412
1413

1414 the T cell differentiation case study from Puniya et al.
 1415 (2018), and exploited in Brodo et al. (2025a) is reported in
 1416 Fig. 24. The corresponding experimentation with
 1417 GROOVE has been discussed in Sect. 5.3.

1418 Auxiliary material for the GROOVE experiments

1419 To replicate the GROOVE experiments reported in Sect. 4
 1420 (for the toy running example) and Sect. 5, we have included
 1421 the following supplementary resources with this
 1422 submission:

- 1423 • The rule systems described in Sect. 4;
- 1424 • The start graphs derived from the BioResolve specifi-
 1425 cations in this Appendix (“Auxiliary material for the
 1426 toy running example” section–“Auxiliary material for
 1427 the T cell differentiation case study” section);
- 1428 • Instructions for calling the GROOVE generator so as to
 1429 reproduce all the exploration runs, occurrence graphs
 1430 and model checking results (using GROOVE version
 1431 7.4.3).

1432 **Acknowledgements** We thank Paolo Milazzo for the technical help in
 1433 double checking the coherence of large state space generated by
 1434 Python scripts based on BioResolve and those generated by
 1435 GROOVE.

1436 **Author contributions** Both authors contributed equally to the paper,
 1437 in writing and reviewing all sections.

1438 **Funding** Research partially supported by the PRIN PNRR 2022
 1439 project *Resource Awareness in Programming* (RAP, P2022HXNSC),
 1440 by the project *SEcurity and RIghts In the CyberSpace* (SERICS,
 1441 PE00000014 - CUP H73C2200089001), under the National Recovery
 1442 and Resilience Plan (NRRP) funded by the European Union -
 1443 NextGenerationEU, by the INdAM-GNCS Projects *Reversibilità In
 1444 Sistemi Concorrenti: analisi Quantitative e Funzionali* (RISICO,
 1445 CUP E53C22001930001) and *Modelli e Analisi per sistemi reversibili
 1446 e quantistici* (MARQ, CUP E53C24001950001), and by the University
 1447 of Pisa PRA project *Formal methods for the healthcare domain
 1448 based on spatial information* (FM4HD, PRA_2022_99).

1449 **Data availability** No datasets were generated or analysed during the
 1450 current study.

1451 **Code availability** GROOVE is an open-source tool, available at
 1452 <https://github.com/nl-utwente-groove/code>. For replication of the
 1453 GROOVE experiments reported in this paper, we provide supple-
 1454 mentary material described in “Auxiliary material for the GROOVE
 1455 experiments” section.

1456 Declarations

1457 **Conflict of interest** The authors declare no Conflict of interest.

1458 **Ethical approval** Not applicable.

References

- 1459 Azimi S (2017) Steady states of constrained reaction systems. Theor
 1460 Comput Sci 701(C):20–26. <https://doi.org/10.1016/j.tcs.2017.03.047>
- 1461 Azimi S, Iancu B, Petre I (2014) Reaction system models for the heat
 1462 shock response. Fundam Inform 131(3–4):299–312. <https://doi.org/10.3233/FI-2014-1016>
- 1463 Azimi S, Gratia C, Ivanov S, Petre I (2015) Dependency graphs and
 1464 mass conservation in reaction systems. Theor Comput Sci 598:23–39. <https://doi.org/10.1016/j.tcs.2015.02.014>
- 1465 Ballis D, Brodo L, Falaschi M, Olarte C (2024) Process calculi and
 1466 rewriting techniques for analyzing reaction systems. In: Gori R,
 1467 Milazzo P, Tribastone M (eds) Computational methods in
 1468 systems biology—22nd International Conference, CMSB 2024,
 1469 Pisa, Italy, September 16–18, 2024, Proceedings. Lecture Notes
 1470 in Computer Science, vol 14971. Springer, pp 1–18. https://doi.org/10.1007/978-3-031-71671-3_1
- 1471 Bowles JKF, Caminati MB (2017) A flexible approach for finding
 1472 optimal paths with minimal conflicts. In: Proceedings of ICFEM
 1473 2017. LNCS, vol 10610. Springer, pp 209–225. https://doi.org/10.1007/978-3-319-68690-5_13
- 1474 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
- 1475 Brodo L, Bruni R, Falaschi M (2021) A logical and graphical
 1476 framework for reaction systems. Theor Comput Sci 875:1–27. <https://doi.org/10.1016/J.TCS.2021.03.024>
- 1477 Brodo L, Bruni R, Falaschi M, Gori R, Levi F, Milazzo P (2023) Quantitative extensions of reaction systems based on SOS
 1478 semantics. Neural Comput Appl 35(9):6335–6359. <https://doi.org/10.1007/S00521-022-07935-6>
- 1479 Brodo L, Bruni R, Falaschi M (2024a) A framework for monitored
 1480 dynamic slicing of reaction systems. Nat Comput 23(2):217–234. <https://doi.org/10.1007/S11047-024-09976-3>
- 1481 Brodo L, Bruni R, Falaschi M, Gori R, Milazzo P, Montagna V,
 1482 Pulieri P (2024b) Causal analysis of positive reaction systems. Int J Softw Tools Technol Transf 26(4):509–526. <https://doi.org/10.1007/S10009-024-00757-Y>
- 1483 Brodo L, Bruni R, Falaschi M, Gori R, Milazzo P (2025a) Attractor
 1484 and slicing analysis of a T cell differentiation model based on
 1485 reaction systems. In: From data to models and back—11th
 1486 International Symposium, DataMod 2023, Eindhoven, The
 1487 Netherlands, November 6–7, 2023, Proceedings. Lecture Notes
 1488 in Computer Science, vol 14618. Springer, pp 69–89. https://doi.org/10.1007/978-3-031-87217-4_4
- 1489 Brodo L, Bruni R, Falaschi M, Gori R, Milazzo P (2025b) Slicing
 1490 analyses for negative dependencies in reaction systems modeling
 1491 gene regulatory networks. Natural Comput. Under review; will
 1492 hopefully appear in same volume as this paper
- 1493 Clavel M, Durán F, Eker S, Lincoln P, Martí-Oliet N, Meseguer J,
 1494 Talcott CL (eds) (2007) All about Maude—a high-performance
 1495 logical framework, how to specify, program and verify systems
 1496 in rewriting logic. Lecture notes in computer science, vol 4350.
 1497 Springer. <https://doi.org/10.1007/978-3-540-71999-1>
- 1498 Corolli L, Maj C, Marini F, Besozzi D, Mauri G (2012) An excursion
 1499 in reaction systems: From computer science to biology. Theor
 1500 Comput Sci 454:95–108. <https://doi.org/10.1016/j.tcs.2012.04.003>

- 1522 Ehrenfeucht A, Rozenberg G (2007) Reaction systems. Fundam Inform 75(1–4):263–280
 1523
 1524 Ehrenfeucht A, Main MG, Rozenberg G (2010) Combinatorics of life and death for reaction systems. Int J Found Comput Sci 21(3):345–356. <https://doi.org/10.1142/S0129054110007295>
 1525 Ehrenfeucht A, Main MG, Rozenberg G (2011) Functions defined by reaction systems. Int J Found Comput Sci 22(1):167–178. <https://doi.org/10.1142/S0129054111007927>
 1526 Ehrig H, Ehrig K, Prange U, Taentzer G (2006) Fundamentals of algebraic graph transformation. In: Monographs in theoretical computer science. An EATCS Series. Springer. <https://doi.org/10.1007/3-540-31188-2>
 1527 Feder G, Eccles M, Grol R, Griffiths C, Grimshaw J (1999) Using clinical guidelines. BMJ 318(7185):728–730. <https://doi.org/10.1136/bmj.318.7185.728>
 1528 Ferretti C, Leporati A, Manzoni L, Porreca AE (2020) The many roads to the simulation of reaction systems. Fundam Inform 171(1–4):175–188. <https://doi.org/10.3233/FI-2020-1878>
 1529 Ghamaran AH, Mol M, Rensink A, Zambon E, Zimakova M (2012) Modelling and analysis using GROOVE. Int J Softw Tools Technol Transf 14(1):15–40. <https://doi.org/10.1007/S10009-011-0186-X>
 1530 Heckel R, Taentzer G (2020) Graph transformation for software engineers—with applications to model-based development and domain-specific language engineering. Springer. <https://doi.org/10.1007/978-3-030-43916-3>
 1531 Helikar T, Kowal B, McClenathan S, Bruckner M, Rowley T, Madrahimov A, Wicks B, Shrestha M, Limbu K, Rogers JA (2012) The cell collective: toward an open and collaborative approach to systems biology. BMC Syst Biol 6(1):1–14. <https://doi.org/10.1186/1752-0509-6-96>
 1532 Heyde SV, Bender C, Henjes F (2014) Boolean ErbB network reconstructions and perturbation simulations reveal individual drug response in different breast cancer cell lines. BMC Syst Biol 8:75. <https://doi.org/10.1186/1752-0509-8-75>
 1533 Hirahara K, Nakayama T (2016) CD4+ T-cell subsets in inflammatory diseases: beyond the T h 1/T h 2 paradigm. Int Immunopharmacol 28(4):163–171. <https://doi.org/10.1093/intimm/dxw006>
 1534 Hughes LD, McMurdo MET, Guthrie B (2013) Guidelines for people not for diseases: the challenges of applying UK clinical guidelines to people with multimorbidity. Age Ageing 42:62–69. <https://doi.org/10.1093/ageing/afs100>
 1535 Ivanov S, Rogojin V, Azimi S, Petre I (2018) WEBRSIM: a web-based reaction systems simulator. In: Díaz CG, Riscos-Núñez A, Paun G, Rozenberg G, Salomaa A (eds) Enjoying natural computing—essays dedicated to Mario de Jesús Pérez-Jiménez on the occasion of his 70th birthday. Lecture notes in computer science, vol 11270. Springer, pp 170–181. https://doi.org/10.1007/978-3-030-00265-7_14
 1536 Kreowski H, Rozenberg G (2019) Graph transformation through graph surfing in reaction systems. J Log Algebraic Methods Program 109:100481. <https://doi.org/10.1016/J.JLAMP.2019.100481>
 1537 Lafaille JJ (1998) The role of helper T cell subsets in autoimmune 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)
 1538 Luckheeram RV, Zhou R, Verma AD, Xia B (2012) CD4+ T cells: differentiation and functions. Clin Dev Immunol 2012:925135. <https://doi.org/10.1155/2012/925135>
- Meng X, Yang J, Dong M, Zhang K, Tu E, Gao Q, Chen W, Zhang C, Zhang Y (2016) Regulatory T cells in cardiovascular diseases. Nat Rev Cardiol 13(3):167–179. <https://doi.org/10.1038/nrcardio.2015.169>
 1539 Meski A, Penczek W, Rozenberg G (2015) Model checking temporal properties of reaction systems. Inf Sci 313:22–42. <https://doi.org/10.1016/J.IINS.2015.03.048>
 1540 Meski A, Koutny M, Mikulski L, Penczek W (2024) Reaction mining for reaction systems. Nat Comput 23(2):323–343. <https://doi.org/10.1007/S11047-024-09989-Y>
 1541 Milner R (1980) A calculus of communicating systems. LNCS, vol 92. Springer
 1542 Nobile MS, Porreca AE, Spolaor S, Manzoni L, Cazzaniga P, Mauri G, Besozzi D (2017) Efficient simulation of reaction systems on graphics processing units. Fundam Inform 154(1–4):307–321. <https://doi.org/10.3233/FI-2017-1568>
 1543 Okubo F, Yokomori T (2016) The computational capability of chemical reaction automata. Nat Comput 15(2):215–224. <https://doi.org/10.1007/s11047-015-9504-7>
 1544 Plotkin GD (2004) A structural approach to operational semantics. J Log Algebraic Methods Program 60–61:17–139
 1545 Puniya BL (2024) CD4+ T cell Differentiation model webpage on the CellCollective platform. <https://research.cellcollective.org/?dashboard=true#module=6678;1/cd4-t-cell-differentiation/1>. Accessed 18 Mar 2024
 1546 Puniya BL, Todd RG, Mohammed A, Brown DM, Barberis M, Helikar T (2018) A mechanistic computational model reveals that plasticity of CD4+ T cell differentiation is a function of cytokine composition and dosage. Front Physiol 9:878. <https://doi.org/10.3389/fphys.2018.00878>
 1547 Rensink A (2024) The GROOVE tool set, version 7.0.1. <https://github.com/nl-twente-groove/code>
 1548 Rubio R, Martí-Oliet N, Pita I, Verdejo A (2021) Strategies, model checking and branching-time properties in Maude. J Log Algebraic Methods Program 123:100700. <https://doi.org/10.1016/J.JLAMP.2021.100700>
 1549 Saez-Rodriguez J, Simeoni L, Lindquist JA, Hemenway R, Bommhardt U, Arndt B, Haus U-U, Weismantel R, Gilles ED, Klamt S (2007) A logical model provides insights into T cell receptor signaling. PLoS Comput Biol 3(8):163. <https://doi.org/10.1371/journal.pcbi.0030163>
 1550 Thakar J, Albert R (2010) Boolean models of within-host immune interactions. Curr Opin Microbiol 13(3):377–381. <https://doi.org/10.1016/j.mib.2010.04.003>
 1551 Woolf SH, Grol R, Hutchinson A, Eccles M, Grimshaw J (1999) Potential benefits, limitations, and harms of clinical guidelines. BMJ 318(7182):527–530. <https://doi.org/10.1136/bmj.318.7182.527>
- Publisher's Note** Springer Nature remains neutral with regard to jurisdictional 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.