Scientific Annals of Computer Science vol. 23 (1), 2013, pp. 75–117
doi: 10.7561/SACS.2013.1.75

Modular Verification of Qualitative Pathway Models with Fairness

Peter Drábik[1], Andrea Maggiolo-Schettini[1], Paolo Milazzo[1], Giovanni Pardini[1]

Abstract

Modular verification is a technique used to face the state explosion problem often encountered in the verification of properties of complex systems such as concurrent interactive systems. The modular approach is based on the observation that properties of interest often concern a rather small portion of the system. As a consequence, reduced models can be constructed which approximate the overall system behaviour thus allowing more efficient verification.

Biochemical pathways can be seen as complex concurrent interactive systems. Consequently, verification of their properties is often computationally very expensive and could take advantage of the modular approach.

In this paper we develop a modular verification framework for biochemical pathways. We view biochemical pathways as concurrent systems of reactions competing for molecular resources. A modular verification technique could be based on reduced models containing only reactions involving molecular resources of interest.

For a proper description of the system behaviour we argue that it is essential to consider a suitable notion of fairness, which is a well-established notion in concurrency theory but novel in the field of pathway modelling. The fairness notion we consider forbids starvation of reactions, namely it ensures that a reaction that is enabled infinitely often cannot always occur to the detriment of another infinitely often enabled reaction causing the latter to never occur.

We prove the correctness of the approach and demonstrate it on the model of the EGF receptor-induced MAP kinase cascade by Schoeberl et al.

Keywords: Systems biology, cellular pathways, model checking, modular verification, model reduction, abstraction

[1]Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo, 3, 56127 Pisa, Italy. Email: peter.drabik@gmail.com, {maggiolo,milazzo,pardinig}@di.unipi.it

1 Introduction

A big challenge of current biology is understanding the principles and functioning of complex biological systems. Despite the great effort of molecular biologists investigating the functioning of cellular components and networks, we still cannot provide a detailed answer to the question "how a cell works?".

In the last decades, scientists have gathered an enormous amount of molecular level information. To uncover the principles of functioning of a biological system, just collecting data does not suffice. Actually, it is necessary to understand the functioning of parts and the way these interact in complex systems. The aim of *systems biology* is to build, on top of the data, the science that deals with principles of operation of biological systems. The comprehension of these principles is done by modelling and analysis exploiting mathematical means.

A typical scenario of modelling a biological system is as follows. To build a model that explains the behaviour of a real biological system, first a formalism needs to be chosen. Then a model of the system is created, simulation is performed, and the behaviour is observed. The model is validated by comparing the results with the real experiments. Simulation allows not only validation of laboratory experiments, but also prediction of behaviour under new conditions.

Depending on the considered simulation technique, simulation can give either the average system behaviour or a number of possible system behaviours. This may be insufficient when one is interested in analysing all the behaviours of a system. In these cases model checking may be of help. This technique permits the verification of properties (expressed as logical formulae) by exploring all the possible behaviours of a system. This analysis technique typically relies on a state space representation whose size, unfortunately, makes the analysis often intractable for realistic models. This is true in particular for systems of interest in systems biology (such as metabolic pathways, signalling pathways, and gene regulatory networks), which often consist of a huge number of components interacting in different ways, thus exhibiting very complex behaviours.

Many formalisms originally developed by computer scientists to model systems of interacting components have been applied to biology, also with extensions to allow more precise descriptions of the biological behaviours [2, 6, 10, 14, 35, 36]. Examples of well-established formal frameworks that can be used to model, simulate and model check descriptions of biological systems are [10, 24, 27].

Model checking techniques have traditionally suffered from the state explosion problem. Standard approaches to the solution of this problem are based on abstractions or similar model reduction techniques (e.g. [11]). Moreover, the use of Binary Decision Diagrams (BDDs) [12] to represent the state space (symbolic model checking) often allows significantly larger model to be treated [5].

A method for trying to avoid the state space explosion problem is to consider a decomposition of the system, and to apply a modular verification technique allowing global properties to be inferred from properties of the system components. This approach can be particularly efficient when the modelled systems consist of a high number of components, whereas properties of interest deal only with a rather small subset of them. This is often the case for properties of biological systems. Hence, for each property it would be useful to be able to isolate a minimal fragment of the model that is necessary for verifying such a property. If such a fragment can be obtained by working only on the syntax of the model, the application of a standard verification technique on the semantics of the fragment avoids the state explosion.

In previous work we developed a modular verification technique in which the system of interest is described by means of a general automata-based formalism, called sync-programs, suitable for qualitative description of a large class of biological systems [19, 20]. Sync-programs include a notion of synchronisation that enables the modelling of biological systems and support modular construction of models. The modular verification technique is based on property preservation and allows the verification of properties expressed in the temporal logic ACTL$^-$ [26] to be verified on fragments of models. In order to handle modelling and verification of more realistic biological scenarios, we have proposed a dynamic version of our formalism along with an extension of the modular verification framework [18].

The long-term aim of our research is the development of an efficient modular verification framework specifically designed for biochemical pathways, and of a pathway analysis tool based on such a framework. At the first stages of the development of the modular verification framework we faced some problems the solution of which required the definition of concepts related to the formal modelling of biochemical pathways that we believe could be interesting not only in the context of modular verification. In particular, we defined a notion of *fairness* for biochemical pathways and a notion of *molecular component* of a pathway. The former is a well-known concept in concurrency theory that could be useful to describe more accurately the

dynamics of a pathway (in a qualitative framework). The latter is a notion relating species involved in the same pathway such that two species are considered to be part of the same molecular component if they can be seen as different states of the same molecule. As far as we know, the adoption of a notion of fairness in the context of biology is new. On the other hand, the notion of molecular component has been often implicitly used (for instance in the modelling of biological systems by means of automata), but now we provide new insight on this notion.

In this paper we report preliminary results obtained during the development of the modular verification framework. Modular verification requires either adopting a modular notation for pathway modelling or finding a way to decompose a pathway, simply expressed as a set of biochemical reactions, into a number of modules. The approach that we choose to follow is in between these two alternatives. Actually, we assume the pathway to be expressed as a set of reactions in a "normal-form" satisfying some modularisation requirements, and then we define a modularisation procedure that allows modules to be inferred from reactions. In a recent paper [33] we have also defined a semi-automatic algorithm that allows any pathway to be transformed into normal-form. Hence, such an algorithm would allow our modular verification approach to be applied to any pathway.

Modules inferred from reactions will be molecular components, hence our modularisation procedure will allow us to consider a pathway not only as a set of reactions, but also as a set of entities interacting with each other (through reactions) and consequently changing state.

Once the molecular components of a pathway are identified, we can use them to decompose the verification of a global pathway property into the verification of a number of sub-properties related with groups of components. To this aim we define a *projection operation* that allows a model fragment describing the behaviour of a group of components to be obtained from a model describing the whole pathway. Such a projection operation is actually an abstraction function, since the behaviour of the group of components will be over-approximated (i.e. the model will include behaviours that are not present in the model of the whole pathway). By considering a suitable temporal logic for the specification of properties (namely ACTL^{-}, a fragment of the CTL logic consisting only of universally quantified formulae) we can prove that properties holding in model fragments obtained by projection also hold in the complete model of the pathway.

Nothing can be said of properties that do not hold in a model fragment.

They might be false in the model fragment since they were also false in the original model, or they can be false in the model fragment because some behaviours added by the projection operation violates them when they were true in the original model. In case a property turns out to be false in a model fragment obtained by projection, it is sometimes possible to assess whether the property really does not hold in the original model by verifying some stronger negative property. We will show some examples of this approach.

In order to verify properties of complete pathway models or of model fragments it is possible to translate them into the input language of an existing model checking tool. Specifically, we use the NuSMV model checker [9], which is a well-established and efficient instrument.

We demonstrate the modular verification approach on the model of the EGF receptor-induced MAP kinase cascade by Schoeberl et al. [37] and we discuss how we plan to continue the development of the approach to improve its efficiency.

Related Work

This paper is a revised extended version of [22]. The main improvement with respect to the previous version is a careful revision of the formal definition of the modular verification framework. In particular, we have revised the formal definition of the semantics both at the concrete (complete models) and abstract (projections) levels. Such a revision was aimed at avoiding unnecessary transitions as much as possible, and hence at reducing as much as possible the approximation in the abstract semantics. In addition, in order to improve presentation of concepts we changed the structure of the paper and added a number of examples. Finally, we included this section in which our approach is compared with related work.

In [32] a reduction methodology for logical models of regulatory networks is proposed. The *logical formalism* is a framework that allows regulatory networks to be described as graphs where nodes are genes and arcs are interactions between genes (promotion/inhibition). The behaviour of a network can then be described as a state transition graph in which states describe the expression level of each modelled gene and transitions describe a changes in such expression levels due to interactions. The proposed reduction methodology essentially consist in iteratively removing individual nodes by defining bypass interactions from the regulators to the targets of each of them. The logical formalism has also been translated into Petri nets in [8]. The translation encodes a node of a graph as a pair of places in the net, gene

interactions as net transitions and the expression level of the genes as the marking of the net. Model reductions are defined also at the net level and are aimed at removing unnecessary transitions.

Our modular verification framework is partially related with the approach based on logical modelling. Both approaches are based on formal qualitative discrete models of biological networks, and both aims at verifying behavioural properties. The approaches in [32, 8], however, are specific for gene regulatory networks, whereas our approach is more general since it aims at describing also reactions occurring between proteins, usually in the context of metabolic and signalling pathways. Moreover, both approaches face the problem of model reduction. The reduction approach we follow is similar, in principle, to the one in [32]. Namely, in both cases we have a model that is reduced by somehow removing parts of it describing some biological entities. However, the fact that the logical framework is specifically designed for regulatory networks makes the operation easier in that case (it suffices to remove one node from a graph and adapt the involved edges). In the case of our modular verification framework, removing parts from the model is more complex since the described biological entities (e.g. proteins) usually have a much more complex behaviour than genes (as they are seen in regulatory networks).

In [16, 17] biological networks are described and analysed by means of a discrete theoretical framework in which biological entities are modelled as agents that can change state depending on states of the other agents. The framework can be used to study different kinds of properties (asymptotic dynamics, causality properties, etc...) of different kinds of biological networks. In particular, in [17] modularity of interaction networks is considered by studying the conditions for module formation and by characterising the relations between the global behaviour of a network and the local behaviours of its components. The approach in [17] has aims similar to the ones of our approach. However, [17] focuses on asymptotic dynamics of networks, whereas our approach deals with behavioural properties expressed as modal logic formulae. Moreover, the nature of module in [17] is different from that of molecular component in our framework since the former is essentially a portion of the network, whereas the latter is the representation of an individual biological entity.

In [3] conditions are investigated for the preservation of the behaviour of a regulatory network when it is embedded into a larger network. Hence, when such conditions are proved for a subnetwork of a larger regulatory

network we obtain that any property of the subnetwork can be verified on the subnetwork model rather than on the (larger) model of the whole regulatory network. This makes the approach in [3] related with ours. However, our approach is more general for two reasons: (i) it is not specific for regulatory networks, and (ii) it is not limited to the cases of components (similar to subnetworks) whose behaviour is completely independent from the rest of the system. What really matters in our case is that only the properties to be verified (and not the whole component behaviour) does not depend on the behaviour of the rest of the system.

Feret et al. [13, 25] developed a reduction technique for pathway models based on an abstraction technique that groups together (fragments of) species representing different configurations of the same molecular complexes the behaviour of which is the same. Such an approach can significantly reduce the size of models by avoiding the combinatorial blow up of species. The approach proposed by Feret et al. is completely different from ours so that in principle it might be possible to combine the two approaches.

2 Modelling Notation for Biochemical Pathways

In biochemistry, metabolic pathways are networks of biochemical reactions occurring within a cell. The reactions are connected by their intermediates: products of one reaction are substrates for subsequent reactions. Reactions are influenced by catalysts and inhibitors, which are molecules (proteins) which can stimulate and block the occurrence of reactions, respectively. For the sake of simplicity we do not consider inhibitors in this paper, although they could be easily dealt with.

Given a set of species S, let us assume biochemical reactions constituting a pathway to have the following form:

$$r_1, \ldots, r_n \to p_1, \ldots, p_{n'} \ \{c_1, \ldots, c_m\}$$

where r_j, p_j and c_j, for suitable values of j, are all in S. We have that r_js are reactants, p_js are products and c_js are catalysts of the considered reaction. Given a reaction R we define $re(R) = \{r_1, \ldots, r_n\}$, $pro(R) = \{p_1, \ldots, p_{n'}\}$, and $cat(R) = \{c_1, \ldots, c_m\}$. We denote the set of species involved in reaction R as $species(R) = re(R) \cup pro(R) \cup cat(R)$. The set of all reactions over a given set of species S is denoted by $reactions(S)$. Finally, a *pathway* is a set of reactions, $P = \{R_1, \ldots, R_N\} \subseteq reactions(S)$. Given a pathway P, we can infer the set of species involved in it as $species(P) = \bigcup_{R \in P} species(R)$.

Note that, in the present paper, we use sets rather than multisets in reactions. This is done since, as discussed in the next section, we choose to describe pathways at a very high level of abstraction. Moreover, using sets allows us to simplify the presentation of our approach. Note that all of the concepts we will define could be defined by using multisets in place of simple sets and the results would hold even in such a case.

2.1 Semantics

In general, the dynamics of a pathway can be described at several different levels of abstraction. The most precise level consists of a quantitative description in which quantities (or concentrations) of species are taken into account, as well as reaction rates in either a deterministic or a stochastic framework. At a more abstract level reaction rates can be ignored. Ultimately, also quantities of species can be ignored by considering only their presence (or absence) in the considered biochemical solution. The less abstract description level is obviously the most precise, but also the most difficult to treat with formal analysis techniques. The more abstract levels are more suitable for the application of formal analysis techniques and are often precise enough to provide useful information on the role of the species and of the reactions involved in the pathway. We choose to adopt the most abstract description level, and hence we define a qualitative formal semantics of pathways in which species can only be either present or absent. This is a rather common choice, done for instance also in [15, 4].

Starting from the initial state representing a biochemical solution, the dynamics of the pathway is driven by the reactions. The occurrence of a reaction may cause the appearance of some new species in the biochemical solution. In this paper, we choose to interpret the effect of a reaction depending on whether it is catalysed or not. In particular, the application of a reaction always creates the products, but we choose reactants to be consumed only by catalysed reactions. In other words, a reaction without catalysts creates the products but does not consume the reactants, while a reaction with catalysts creates the products and consumes the reactants. We choose this interpretation since non-catalysed reactions usually reach a steady-state of dynamic equilibrium in which both reactants and products are present in the biochemical solution. On the other hand, a reaction favoured by catalysts usually tends to be performed as long as there are reactants. A consequence of this assumption is that a reversible reaction in which both directions are catalysed, which frequently occurs in biological

$$\frac{re(R) \subseteq \mathbf{s} \quad pro(R) \not\subseteq \mathbf{s} \quad cat(R) = \emptyset}{\mathbf{s} \xrightarrow{R} \mathbf{s} \cup pro(R)} \text{ (no-cat)}$$

$$\frac{re(R) \subseteq \mathbf{s} \quad re(R) \not\subseteq pro(R) \vee pro(R) \not\subseteq \mathbf{s} \quad \emptyset \neq cat(R) \subseteq \mathbf{s}}{\mathbf{s} \xrightarrow{R} (\mathbf{s} \setminus re(R)) \cup pro(R)} \text{ (cat)}$$

$$\frac{\forall R.\ \mathbf{s} \not\xrightarrow{R}}{\mathbf{s} \xrightarrow{\epsilon} \mathbf{s}} \text{ (deadlock)}$$

Figure 1: Inference rules of the semantics.

pathways, oscillates between two states. This is realistic in some cases, such as in the case of oscillatory behaviours, but not always. We leave a more detailed treatment of this aspect as future work.

Technically, the semantics is formalised as *Labelled Transition System* (LTS), where each state corresponds to a set of species, and transitions are labelled by the reaction which is applied. Formally, an LTS is a tuple $(S, \Sigma, \rightarrow)$, where S is the set of states, Σ is the set of labels, and $\rightarrow \subseteq S \times \Sigma \times S$ is the transition relation.

Definition 1 (Semantics)**.** *Let $P = \{R_1, \ldots, R_n\}$ be a pathway, and let $\epsilon \notin P$. The semantics of P is defined as the LTS $(\mathcal{P}(species(P)), P \cup \{\epsilon\}, \rightarrow)$, where $\rightarrow$ is the least transition relation satisfying the inference rules of Figure 1.*

Rules (no-cat) and (cat) formalise the dynamics of reactions in the absence and presence of catalysts, respectively. As regards non-catalysed reactions, they are applicable only if the reactants are present, and some products is missing. In this manner, since the resulting state is obtained from the starting state by adding the products, the application of reactions which would not change the state of the system are omitted. On the other hand, in order for a catalysed reaction to occur, all of its reactants and catalysts are required to be present. The resulting state is obtained by first removing the reactants (which are thus consumed) and then adding the products. Similarly to the previous case, only transitions causing a state change are considered, as ensured by condition $re(R) \not\subseteq pro(R) \vee pro(R) \not\subseteq \mathbf{s}$. Alternative combinations of catalysts that may enable the reaction should be modelled as different reactions having the same reactants and products. In

general, excluding transitions which do not change the state of the system is convenient for the verification as the size of the transition system is smaller but the set of properties that hold stays the same.

Finally, rule (deadlock) provides each state in which no reaction is enabled (denoted $\stackrel{R}{\not\rightarrow}$) with a self-looping transition with empty label, denoted ϵ. This rule was not present in [22], and its aim is simply to turn every finite path into an equivalent infinite one in order to simplify the definition of the modular verification methodology.

In the following, we denote the semantic function as LTS, namely $LTS(P) = (\mathcal{P}(species(P)), P \cup \{\epsilon\}, \rightarrow)$, for some pathway P. A *path* in $LTS(P)$ is an infinite sequence $\pi = \mathbf{s_0}, R_0, \mathbf{s_1}, R_1, \mathbf{s_2}, \ldots$ where for all $i \geq 0$, $\mathbf{s_i} \xrightarrow{R_i} \mathbf{s_{i+1}}$. We denote by π^k the suffix of π starting at $\mathbf{s_k}$, namely $\pi^k = \mathbf{s_k}, R_k, \mathbf{s_{k+1}}, R_{k+1}, \mathbf{s_{k+2}}, \ldots$.

Example 1. *Let P be a pathway consisting of the following reactions:*

$$
\begin{array}{lll}
P_1, P_2 \rightarrow C & (R_1) & P_1^*, M_2 \rightarrow D\ \{E_1\} \quad (R_5) \\
P_1 \rightarrow P_1^* & (R_2) & D \rightarrow P_1^*, M_2\ \{E_2\} \quad (R_6) \\
P_1^* \rightarrow P_1 & (R_3) & D \rightarrow\ \{C, M_1^*\} \quad (R_7) \\
M_1 \rightarrow M_1^*\ \{P_1^*\} & (R_4) &
\end{array}
$$

The fragment of $LTS(P)$ rooted at $\{P_1, P_2, M_1, M_2, E_1, E_2\}$ is shown in Figure 2. Note that no transition for reaction R_3 is present, since all states in which its reactants are present there are also its products (e.g. state $\mathbf{s_1}$).

2.2 Fairness

In order to describe the behaviour of a pathway more accurately we consider a notion of fairness. We motivate it by considering a quantitative system consisting of four reactions $A \xrightarrow{k_1} B\ \{\ D\ \}$, $B \xrightarrow{k_2} A\ \{\ D\ \}$, $A \xrightarrow{k_3} C\ \{\ D\ \}$ and $C \xrightarrow{k_4} A\ \{\ D\ \}$, where k_1, k_2, k_3 and k_4 are the reaction rates. By performing the qualitative abstraction, we get a pathway containing reactions $R_1 = A \rightarrow B\ \{D\}$ and $R_2 = B \rightarrow A\ \{D\}$, $R_3 = A \rightarrow C\ \{D\}$ and $R_4 = C \rightarrow A\ \{D\}$, whose semantics as defined above includes behaviours such as the one where R_3 never occurs. Such a behaviour is a qualitative abstraction which is not correct, since the standard quantitative dynamics ruled by the law of mass action would imply that both R_1 and R_3 occur with a frequency proportional to their kinetic rates. Actually, in a stochastic

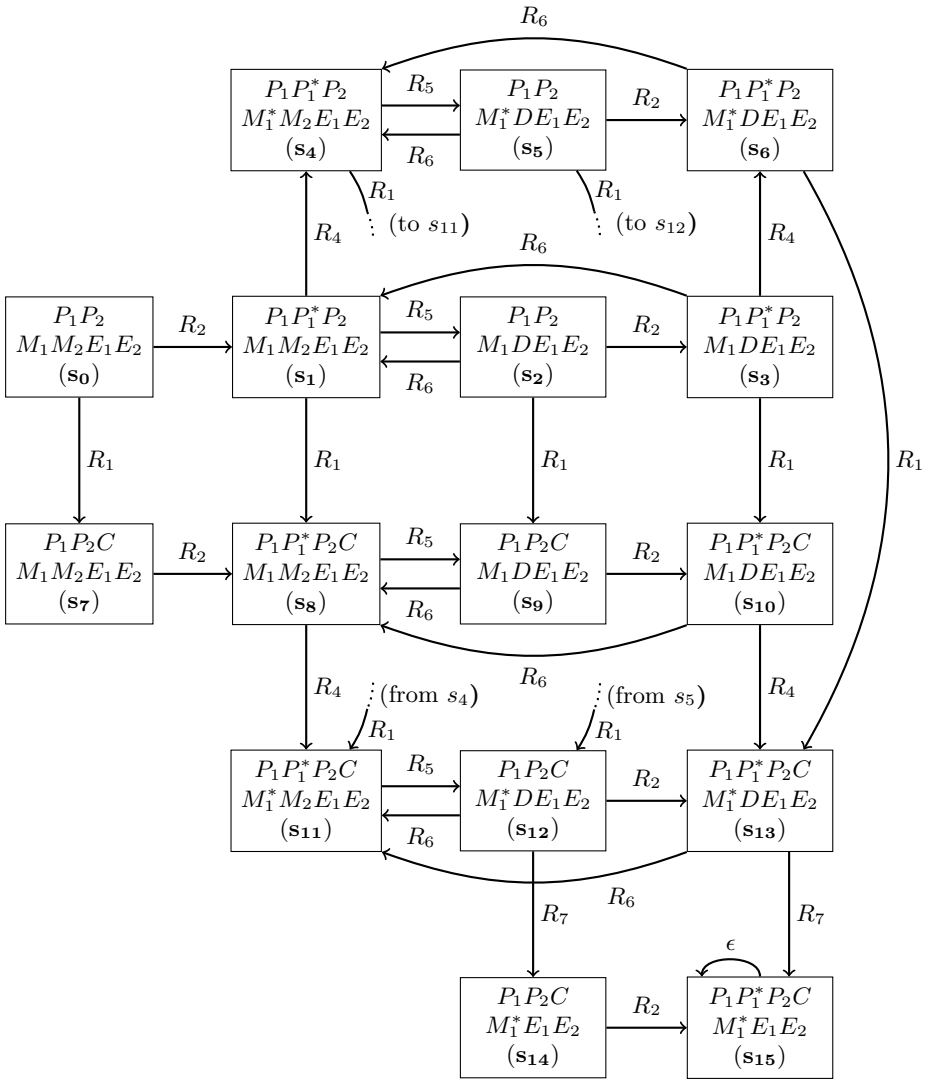

Figure 2: Representation of $LTS(P)$ rooted at $\mathbf{s_0} = \{P_1, P_2, M_1, M_2, E_1, E_2\}$, where P is as defined in Example 1. Set denotations (brackets and commas) are omitted in state representations, whereas each state is associated with a short name $\mathbf{s}_i$, for $i \in [0, 15]$.

$$
\begin{array}{lcl}
\pi \vDash_{LTL} s & \iff & \exists \mathbf{s}, R, \pi'.\ \pi = \mathbf{s}, R, \pi' \text{ and } s \in \mathbf{s} \\
\pi \vDash_{LTL} \neg f & \iff & \pi \nvDash_{LTL} f \\
\pi \vDash_{LTL} f \vee g & \iff & \pi \vDash_{LTL} f \text{ or } \pi \vDash_{LTL} g \\
\pi \vDash_{LTL} X\ g & \iff & \exists \mathbf{s}, R, \pi'.\ \pi = \mathbf{s}, R, \pi' \text{ and } \pi' \vDash_{LTL} g \\
\pi \vDash_{LTL} X_R\ g & \iff & \exists \mathbf{s}, \pi'.\ \pi = \mathbf{s}, R, \pi' \text{ and } \pi' \vDash_{LTL} g \\
\pi \vDash_{LTL} f\ U\ g & \iff & \exists k \geq 0.\ \pi^k \vDash_{LTL} g \text{ and} \\
 & & \forall 0 \leq j < k.\ \pi^j \vDash_{LTL} f
\end{array}
$$

Figure 3: Satisfaction relation for LTL formulae.

setting both R_1 and R_3 would occur infinitely often with probability 1. A correct qualitative abstraction of our system should therefore only include paths in which both R_1 and R_3 occur infinitely many times.

A concept from concurrency theory that allows to specify the correct behaviour is fairness, which stipulates that reactions should compete in a fair manner. We consider the well-known notion of strong fairness [23], also called compassion, which requires that if a reaction is enabled (ready to occur) infinitely many times, then it will occur infinitely many times.

Technically, fairness is specified by a *linear temporal logic* [34] (LTL) formula. For this reason, we briefly recall such a logic, instantiated to our setting, before formally defining fairness. In particular, we consider a variant of the LTL with action-specific next modality X_R, where R is a transition label, corresponding to a reaction in our setting. Given a finite set species S (c.f.r. atomic propositions from canonical definition), the syntax of LTL formulae is given by $f ::= s \mid \neg f \mid f \vee g \mid X\ g \mid X_R\ g \mid f\ U\ g$, where f, g are meta-variables denoting LTL formulae, $s \in S$, and R is a reaction. Given a pathway P, let us consider a Labelled Transition System $\mathcal{LTS}$ with states $\mathcal{P}(\mathit{species}(P))$ and labels $P \cup \{\epsilon\}$; the satisfaction relation $\vDash_{LTL}$ of an LTL formula f with respect to a path $\pi \in \mathcal{LTS}$, denoted $\pi \vDash_{LTL} f$, is defined in Figure 3. Additional logical operators can be defined, $\mathit{true} = s \vee \neg s$, $\mathit{false} = \neg \mathit{true}$, $f \wedge g = \neg(\neg f \vee \neg g)$ and $f \rightarrow g = \neg f \vee g$. Moreover, additional temporal operators *eventually* $F\ g = \mathit{true}\ U\ g$, and *globally* $G\ g = \neg F\ \neg g$, are usually defined.

Definition 2 (Fairness)**.** *Let P be a pathway. A path π in $LTS(P)$ is* fair

iff it satisfies the following LTL formula:

$$\Phi = \bigwedge_{R \in P} (GF\ \mathit{enabled}(R)) \rightarrow (GF\ \mathit{occurs}(R))$$

where

$$\mathit{enabled}(R) = \begin{cases} (\bigwedge_{r \in re(R)} r) \wedge (\bigwedge_{c \in cat(R)} c) & \text{if } cat(R) \neq \emptyset,\ re(R) \not\subseteq pro(R); \\ (\bigwedge_{r \in re(R)} r) \wedge (\bigvee_{p \in pro(R)} \neg p) \wedge (\bigwedge_{c \in cat(R)} c) & \text{otherwise;} \end{cases}$$

$$\mathit{occurs}(R) = X_R\ true.$$

The definition of fairness requires each reaction which is enabled infinitely often in the path ($GF\ \mathit{enabled}(R)$) to also occur infinitely often ($GF\ \mathit{occurs}(R)$). This prevents a component which can progress indefinitely to block the application of reactions belonging to other components.

Given a reaction R, formula $\mathit{occurs}(R)$ is satisfiable only if there is a transition exiting from the state labelled by R, denoting the application of such a reaction. As regards formula $\mathit{enabled}(R)$, according to the format of the reaction R (i.e. either catalysed or non-catalysed) it is satisfiable only if the reaction is applicable in the state. First case deals with catalysed reactions for which there is at least one reactant which is not also a product. In this case, the reaction is enabled if both reactants and catalysts are present, irregardless of the products already present in the state (since the state is modified by removing the reactants). The second case deals with both cases in which either the reaction is not catalysed, or it is catalysed and reactants are a subset of the products; in both cases, apart reactants and catalysts which need to be present, there must also be at least one product missing.

Example 2. *Let P be the pathway defined in Example 1, and $LTS(P)$ be its semantics. According to the definition of fairness, the following path is fair:*

$$\pi_1 = \mathbf{s_0}, R_2, \mathbf{s_1}, R_5, \mathbf{s_2}, R_2, \mathbf{s_3}, R_1, \mathbf{s_{10}}, R_4, \mathbf{s_{13}}, R_7, \mathbf{s_{15}}, \epsilon, \mathbf{s_{15}}, \epsilon, \ldots .$$

On the other hand, the following path in which the system loops between states $\mathbf{s_{s_1}}$ and $\mathbf{s_{s_1}}$ is not fair:

$$\pi_2 = \mathbf{s_0}, R_2, \mathbf{s_1}, R_5, \mathbf{s_2}, R_6, \mathbf{s_1}, R_5, \mathbf{s_2}, R_6, \mathbf{s_1}, R_5, \mathbf{s_2}, R_6, \ldots .$$

Indeed, in π_2 reaction R_2 is infinitely often enabled (every time the system is in state $\mathbf{s_2}$) but it occurs only once. In other words reaction R_6, that competes with R_2 for application in $\mathbf{s_2}$, always wins the competition thus causing starvation of R_2.

3 Identification of Molecular Components

The approach to modular verification of pathways that we propose is built upon the concept of *molecular component*. The main observation is that pathways are usually composed of a few basic biological entities which interact; for example, a protein can be involved in a series of transformations, starting from its initial synthesised form, which can then be activated and later become part of different complexes.

A typical aspect of a biochemical pathway is that the process described involves mainly a few chains of reactions, in which the occurrence of a reaction produces some intermediate molecule which is then transformed subsequently by other reactions. Intuitively, this allows us to regard the intermediate molecular species which appear in the pathway as different "states" or "configurations" of the same initial biological entity, and thus the reactions as a synchronised state change of a set of such basic entities. According to this view, in the modelling of biochemical pathways we can consider a notion of *molecular component* [22, 21] that is the formal counterpart of the notion of biological entity. A molecular component thus groups together all the species mentioned in the model which correspond to different states of the same biological entity.

In the previous section we have introduced the syntax of the modelling formalism of biochemical pathways, in which a reaction is allowed to have a different number of reactants and products. In order to identify the components of a pathway, we assume that each reaction has the same number of reactants and products. Moreover, we assume a positional correspondence between the reactants and the products, in particular we assume that product p_j is the result of the transformation of reactant r_j by the reaction.

The idea behind this assumption on the form of reactions is that reactions of cellular pathways very often represent bindings (and unbindings) of well-defined macromolecules, such as proteins and genes, to form (or to break) complexes either with other macromolecules or with small molecules such as ions and nutrients. Also conformational changes are common, in which a protein (or a complex constituted by a few proteins) changes its own "state". If we consider a complex not as a single species, but as a set of species, one for each molecule involved in it, we have that all of the mentioned kinds of reaction turn out to have as many product as reactants.

In general, pathways are not expressed using this kind of "normal-form" reactions, with positional correspondence between reactants and products. We argue that, under conditions often found in practice, a pathway can be

rewritten in normal form by making explicit the molecular species involved in each reaction, thus allowing its decomposition into components. The aspect of translating a generic pathway model into a normal-form pathway, has been addressed in [33], in which we proposed a semi-automatic algorithm for component identification in SBML pathway models. In [30] the algorithm has been tested on a vast array of real pathway models expressed by using the SBML notation [28], namely on all of the 436 models available in the BioModels database [29] under the category "curated models". The results of the test showed that in most cases the algorithm has been able to transform the pathways into normal form (and also to infer molecular components) in a fully automatic way.

Example 3. *Let P be the pathway defined in Example 1. The corresponding pathway in "normal-form" P' consists of the following reactions:*

$$P_1, P_2 \to C_{P_1}, C_{P_2} \quad (R'_1) \qquad P_1^*, M_2 \to D_{P_1^*}, D_{M_2}\ \{E_1\} \quad (R'_5)$$
$$P_1 \to P_1^* \quad (R'_2) \qquad D_{P_1^*}, D_{M_2} \to P_1^*, M_2\ \{E_2\} \quad (R'_6)$$
$$P_1^* \to P_1 \quad (R'_3) \qquad D_{P_1^*}, D_{M_2} \to Z_{P_1^*}, Z_{M_2}\ \{M_1^*, C_{P_1}, C_{P_2}\} \quad (R'_7)$$
$$M_1 \to M_1^*\ \{P_1^*\} \quad (R'_4)$$

where species C representing the complex obtained by the binding of P_1 with P_2 has been replaced by the two species C_{P_1} and C_{P_2} representing the bound states of P_1 and P_2, respectively. Similarly, complex D has been replaced (three times) by $D_{P_1^*}$ and D_{M_2}. Moreover, dummy species $Z_{P_1^*}$ and Z_{M_2} have been introduced to represent the degraded states of $D_{P_1^*}$ and D_{M_2}, respectively.

3.1 Component Inference from Normal-Form Pathways

We present an algorithm that given a pathway P infers the components appearing in it, by returning a partition of the species into sets, each corresponding to a different component. We assume that the pathway P consists of normal-form reactions $r_1, \ldots, r_n \to p_1, \ldots, p_n\ \{c_1, \ldots, c_m\}$ in which there is a one-to-one correspondence between reactants and products.

We illustrate the intuitive idea on an example. Each reaction can be seen as a synchronisation of components. For example reaction $r_1, r_2 \to p_1, p_2\ \{c\}$ can be interpreted as a synchronisation of three components: one that changes its state from r_1 into p_1, another component that changes its state from r_2 into p_2, and a component which participates passively and stays in a state c. Since we suppose that only one reaction takes place at a time in the whole system, the states of all the components do not change other than

Algorithm 1 Algorithm to partition species into different components

Let $map : S \mapsto Z$ be a total injective mapping
for all R in P **do**
  let $R = r_1, \ldots, r_n \to p_1, \ldots, p_n \ \{c_1, \ldots, c_m\}$
  for all $j \in \{1, \ldots, n\}$ **do**

$$map := \begin{cases} x \mapsto map(r_j) & \text{if } map(x) = map(p_j) \\ x \mapsto map(x) & \text{if } map(x) \neq map(p_j) \end{cases}$$

  end for
end for
return *map*

those involved in the reaction in the way we described. From the example we can see that species r_1 and p_1 belong to the same component. Similarly r_2 belongs to the component that contains p_2, while c is from a separate one.

The algorithm to infer components from a normal-form pathway is given in Algorithm 1. It assumes an infinite set of component names Z. Initially, each species is assumed to belong to a different component, then the algorithm refines this assumption by iterating over the reactions constituting P. For each reaction in the pathway, the algorithm updates the mapping by unifying the species assigned to the i-th reactant with the i-th product, for all $i \in \{1, \ldots, n\}$. The result of the algorithm is a mapping *map* assigning each species to its component.

In the following, we denote by $comp(P)$ the set of components of a given pathway P; formally, it is defined as the image of mapping *map*. Using the same notation, we denote by $comp(R) = comp(\{R\})$ the components of a given reaction R.

Example 4. *Let P' be the pathway defined in Example 3. The set of components $comp(P')$ computed by the algorithm is as follows:*

$$comp(P') = \big\{ \{P_1, C_{P_1}, P_1^*, D_{P_1^*}, Z_{P_1^*}\}, \{P_2, C_{P_2}\}, \{M_1, M_1^*\}, \{M_2, D_{M_2}, Z_{M_2}\}, \{E_1\}, \{E_2\} \big\}$$

Six components are inferred from reactions. Each component is the set of states of a different entity involved in the pathway.

A *component interaction graph* can be drawn which visualises the components of a pathway and their interactions. It is a directed graph in which vertices are system components (elements of $comp(P)$) and edges connect

components that are involved together in a reaction. If two components are both involved as reactants (and consequently products), the edge connecting them will not be oriented. If one of the two is involved as reactant and the other as catalyst, then the edge will start from the vertex representing the latter to the vertex representing the former. There is no edge between vertices representing components involved in the same reactions only as catalysts. An example of component interaction graph will be shown in Section 5.3.

4 Modular Verification

In this section we define a modular verification technique for pathway models. We proceed by defining the projection of a pathway with the help of the identified components. Such a projection can be seen as an abstraction, giving rise to abstract pathways. We prove that a successful verification of a property in the abstraction implies its truth in the original model.

4.1 Abstract Pathways: Syntax, Semantics and Fairness

We are interested in analysing only a portion of the entire pathway, in particular a portion induced by only a subset of all components. Let $I = \mathit{comp}(P)$ and $J \subseteq I$, we define the projection of a pathway P onto J as the abstract pathway $P{\restriction}J$. In the following, by abusing the notation, we assume function *species* to be also defined on sets of components J as $\mathit{species}(J) = \bigcup_{x \in J} \mathit{species}(x)$. Moreover, we denote as $u{\restriction}J$ the projection of a set of species u over a set of components J, which is formally defined as $u{\restriction}J = u \cap \mathit{species}(J)$.

Definition 3. *Let P be a pathway. The* abstract pathway $P{\restriction}J$*, obtained by abstracting w.r.t. a set of components $J \subseteq \mathit{comp}(P)$, is defined as $P{\restriction}J = (PR_c, PR_{nc}, AR_c, AR_{nc})$, where:*

$$PR_c = \{R \in P \mid \mathit{comp}(R) \subseteq J,\ \mathit{cat}(R) \neq \emptyset\}$$
$$PR_{nc} = \{R \in P \mid \mathit{comp}(R) \subseteq J,\ \mathit{cat}(R) = \emptyset\}$$
$$AR_c = \bigcup_{R \in P} \left\{ R{\restriction}J \;\middle|\; \mathit{comp}(R) \backslash J \neq \mathit{comp}(R),\ \mathit{re}(R){\restriction}J \neq \emptyset,\ \mathit{cat}(R) \neq \emptyset \right\}$$
$$AR_{nc} = \bigcup_{R \in P} \left\{ R{\restriction}J \;\middle|\; \mathit{comp}(R) \backslash J \neq \mathit{comp}(R),\ \mathit{re}(R){\restriction}J \neq \emptyset,\ \mathit{cat}(R) = \emptyset \right\}$$

where $R{\restriction}J = \mathit{re}(R){\restriction}J \rightarrow \mathit{pro}(R){\restriction}J\ \{\mathit{cat}(R){\restriction}J\}$.

An abstract pathway consists of four sets of reactions, namely PR_c, PR_{nc}, AR_c, AR_{nc}. If we consider sets $PR = PR_c \cup PR_{nc}$ and $AR = AR_c \cup AR_{nc}$, then reactions are distinguished between unmodified reactions in PR, and abstract reactions in AR. In particular, PR contains reactions which deal only with components inside J, while AR contains projections of reactions that influence components both inside and outside J. On the other hand, reactions which affect only components outside of J are excluded from the abstract pathway.

Moreover, both sets of reactions PR and AR are further classified between catalysed reactions (PR_c and AR_c) and non-catalysed reactions (PR_{nc} and AR_{nc}). Note that reactions are classified as catalysed/non-catalysed according to their form in the original pathway, rather than their resulting form in the abstract pathway. This aspect is actually important only for reactions in AR, where projecting a catalysed reaction R over J may cause its abstract version to have an empty set of catalysts if catalysts are only from components not in J.

Example 5. *Let P' be the pathway defined in Example 3, and let J be the following subset of $comp(P')$:*

$$\left\{\{P_1, C_{P_1}, P_1^*, D_{P_1^*}, Z_{P_1^*}\}, \{M_2, D_{M_2}, Z_{M_2}\}, \{E_1\}\right\}.$$

The projection of P' onto J, $P'{\uparrow}J$, is given by the following sets of reactions:

$$PR_c = \{R_5''\} \quad PR_{nc} = \{R_2'', R_3''\} \quad AR_c = \{R_6'', R_7''\} \quad AR_{nc} = \{R_1''\}$$

where reactions are defined as follows:

$$\begin{aligned}
P_1 &\to C_{P_1} && (R_1'')\\
P_1 &\to P_1^* && (R_2'')\\
P_1^* &\to P_1 && (R_3'')\\
P_1^*, M_2 &\to D_{P_1^*}, D_{M_2}\ \{E_1\} && (R_5'')\\
D_{P_1^*}, D_{M_2} &\to P_1^*, M_2 && (R_6'')\\
D_{P_1^*}, D_{M_2} &\to Z_{P_1^*}, Z_{M_2}\ \{C_{P_1}\} && (R_7'')
\end{aligned}$$

Note that: (i) reaction R_4' has no corresponding reaction in $P'{\uparrow}J$; (ii) although reactions R_6'' does not include any catalyst, it is in AR_c since the corresponding reaction R_6' in P' included one catalyst.

$$\frac{R \in PR_{nc} \cup AR_{nc} \quad re(R) \subseteq \mathbf{s} \quad pro(R) \not\subseteq \mathbf{s}}{\mathbf{s} \xrightarrow{R}_{\alpha} \mathbf{s} \cup pro(R)} \text{ (no-cat)}$$

$$\frac{R \in PR_c \cup AR_c \quad re(R) \subseteq \mathbf{s} \quad cat(R) \subseteq \mathbf{s} \quad re(R) \not\subseteq pro(R) \vee pro(R) \not\subseteq \mathbf{s}}{\mathbf{s} \xrightarrow{R}_{\alpha} (\mathbf{s} \backslash re(R)) \cup pro(R)} \text{ (cat)}$$

$$\frac{R \in AR_c \cup AR_{nc} \quad \mathbf{s} \xrightarrow{R}_{\alpha} \mathbf{s}'}{\mathbf{s} \xrightarrow{\epsilon}_{\alpha} \mathbf{s}} \text{ (self-loop)}$$

$$\frac{\forall R \in PR_c \cup PR_{nc} \cup AR_c \cup AR_{nc}.\ \mathbf{s} \xrightarrow{R}\!\!\!\!\!\!/\ }{\mathbf{s} \xrightarrow{\epsilon}_{\alpha} \mathbf{s}} \text{ (deadlock)}$$

Figure 4: Inference rules of the abstract semantics.

Definition 4 (Abstract semantics)**.** *Let* $P_\alpha = (PR_c, PR_{nc}, AR_c, AR_{nc})$ *be an abstract pathway, and* $\overline{P_\alpha} = PR_c \cup PR_{nc} \cup AR_c \cup AR_{nc}$*. The* abstract semantics *of* P_α *is defined as the LTS* $(\mathcal{P}(species(\overline{P_\alpha})), \overline{P_\alpha} \cup \{\epsilon\}, \rightarrow_\alpha)$*, where* $\rightarrow_\alpha$ *is the least transition relation satisfying the inference rules of Figure 4.*

In the following, the abstract semantic function is denoted LTS_α, i.e. $LTS_\alpha(P_\alpha) = (\mathcal{P}(species(\overline{P_\alpha})), \overline{P_\alpha} \cup \{\epsilon\}, \rightarrow_\alpha)$. Note that a pathway is essentially a special case of abstract pathway, since the semantics of P is equivalent (isomorphic) to that of $P{\restriction}comp(P)$.

Rules (no-cat), (cat), and (deadlock) are analogous to the rules with the same name from semantics of Definition 1. Note that rule (no-cat) deals with the application of non-catalysed reactions from the sets PR_{nc} and AR_{nc}, rule (cat) deals with catalysed reactions from PR_c and AR_c, and rule (deadlock) considers any reaction in $PR \cup AR$. The abstract semantics also provides rule (self-loop) which allows deriving a self-loop transition for projected reactions in AR. The purpose of this transition is to model the case in which a projected reaction $R{\restriction}J$ is applicable in the abstract state $\mathbf{s}$ (i.e. both projected reactants $re(R){\restriction}J$ and projected catalysts $cat(R){\restriction}J$ are present in $\mathbf{s}$), but for which the corresponding unprojected reaction would not be applicable in the original state because some reactant/catalysts from unobservable components are missing.

Example 6. *Let* $P'{\restriction}J$ *be as in Example 5. The fragment of the abstract*

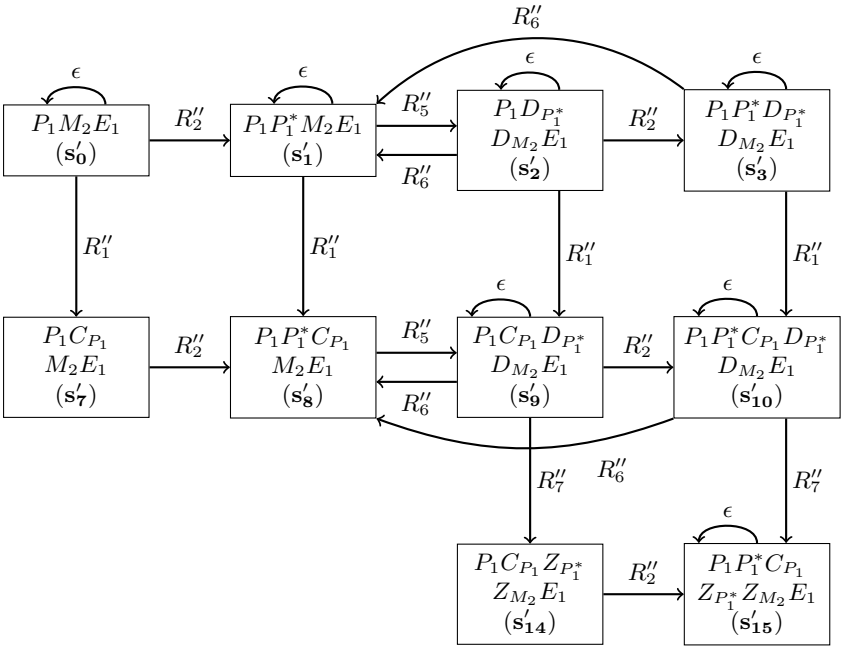

Figure 5: Representation of $LTS_\alpha(P'{\restriction}J)$ rooted at $\mathbf{s'_0} = \{P_1, M_2, E_1\}$, where $P'{\restriction}J$ is as defined in Example 5. Set denotations (brackets and commas) are omitted in state representations, whereas each state is associated with a short name $\mathbf{s'_i}$, where i is as in Figure 2.

semantics of $P'{\restriction}J$, denoted $LTS_\alpha(P'{\restriction}J)$, rooted at $\{P_1, M_2, E_1\}$ is shown in Figure 5.

With respect to the semantics of the original pathway P (see Figure 2) the abstract semantics there are less states. In the abstract semantics, however, there are a few more transitions with label ϵ than in the semantics of P. These are in states $\mathbf{s'_0}$, $\mathbf{s'_1}$, $\mathbf{s'_2}$, $\mathbf{s'_3}$, $\mathbf{s'_9}$ and $\mathbf{s'_{10}}$. In those states there are enabled reactions from either AR_c or AR_{nc} (e.g. reaction R''_1) that have been modified by projection and hence depend by some component not in J that in principle may be blocked in a state that never allows these reactions to occur. The ϵ transition models such a situation.

Abstract fairness

In the case of abstract pathways fairness constraints can be applied only to reactions in PR. In fact, note that, given the state $\mathbf{s}$, each applicable reaction in AR allows deriving two transitions: (i) $\mathbf{s} \xrightarrow{R}_{\alpha} \mathbf{s}'$ (using either rules (cat)/(no-cat), for some $\mathbf{s}'$) describing the application of the reaction, and (ii) $\mathbf{s} \xrightarrow{\epsilon}_{\alpha} \mathbf{s}$, (using rule (self-loop)) describing the non-application of the reaction. Since each applicable reaction in AR allows deriving these two transitions, it might happen that one of the two is always preferred, capturing the situation in which the corresponding unprojected reaction is either always enabled or always disabled. In other words, this describes the case in which some reactants or catalysts from components which have been projected out are not present in the original state.

The formal definition of abstract fairness, which follows, is analogous to Definition 2. As we have anticipated, the only difference is that it deals only with unmodified reactions included in PR.

Definition 5 (Abstract fairness)**.** *Let $P_\alpha = (PR_c, PR_{nc}, AR_c, AR_{nc})$ be an abstract pathway. A path π in $LTS_\alpha(P_\alpha)$ is* fair *iff it satisfies the following LTL formula:*

$$\Phi_\alpha = \bigwedge_{R \in PR_c \cup PR_{nc}} (GF\ enabled(R)) \rightarrow (GF\ occurs(R))$$

where

$$enabled(R) = \begin{cases} (\bigwedge_{r \in re(R)} r) \wedge (\bigwedge_{c \in cat(R)} c) & \text{if } R \in PR_c,\ re(R) \not\subseteq pro(R); \\ (\bigwedge_{r \in re(R)} r) \wedge (\bigvee_{p \in pro(R)} \neg p) \wedge (\bigwedge_{c \in cat(R)} c) & \text{otherwise;} \end{cases}$$

$$occurs(R) = X_R\ true.$$

Example 7. *Let us consider the semantics of $P' \restriction J$ described in Example 6 and shown in Figure 5. According to the definition of abstract fairness we have that the following paths are fair:*

$$\pi'_1 = \mathbf{s'_0}, R''_2, \mathbf{s'_1}, R''_5, \mathbf{s'_2}, R''_2, \mathbf{s'_3}, R''_1, \mathbf{s'_{10}}, R''_7, \mathbf{s'_{15}}, \epsilon, \mathbf{s'_{15}}, \epsilon, \dots ,$$

$$\pi'_2 = \mathbf{s'_0}, R''_2, \mathbf{s'_1}, R''_5, \mathbf{s'_2}, R''_2, \mathbf{s'_3}, \epsilon, \mathbf{s'_3}, \epsilon, \mathbf{s'_3}, \epsilon, \dots .$$

On the other hand, the following paths are not fair:

$$\pi'_3 = \mathbf{s'_0}, R''_2, \mathbf{s'_1}, R''_5, \mathbf{s'_2}, R''_6, \mathbf{s'_1}, R''_5, \mathbf{s'_2}, R''_6, \mathbf{s'_1}, R''_5, \mathbf{s'_2}, R''_6, \dots ,$$

$$\pi'_4 = \mathbf{s'_0}, R''_2, \mathbf{s'_1}, R''_5, \mathbf{s'_2}, \epsilon, \mathbf{s'_2}, \epsilon, \mathbf{s'_2}, \epsilon, \ldots .$$

Paths π'_1 and π'_3 are analogous to paths π_1 and π_2 in Example 2. Path π'_2 is fair since both reactions enabled in $\mathbf{s'_3}$ are not subject to the fairness condition since they are in AR. Finally, path π'_4 is not fair since reaction R''_2, that is enabled in $\mathbf{s'_2}$, is in PR, and hence the fairness condition applies to it.

4.2 Logic for Specifying Properties

Properties of pathways are specified in temporal logic with species as atomic propositions. The logic we consider is a fragment of the Computation Tree Logic CTL. Following Attie and Emerson [1], we assume the logic ACTL$^-$ for specification of properties. ACTL is the "universal fragment" of CTL which results from CTL by restricting negation to propositions and eliminating the existential path quantifier and ACTL$^-$ is ACTL without the AX modality.

Definition 6. *The* syntax of ACTL$^-$ *is defined inductively as follows:*

- *The constants true and false are formulae. s and $\neg s$ are formulae for any atomic proposition s, where the set of atomic propositions AP are the set of all species S.*
- *If f, g are formulae, then so are $f \wedge g$ and $f \vee g$.*
- *If f, g are formulae, then so are $A[f\ U\ g]$ and $A[f\ U_w\ g]$.*

Given a set of components J, we define the logic ACTL^-_J to be ACTL$^-$ where the atomic propositions are drawn from $AP_J = species(J)$. We define the following abbreviations in ACTL$^-$: $AF\ f \equiv A[true\ U\ f]$ and $AG\ f \equiv A[f\ U_w\ false]$.

Properties expressible by ACTL$^-$ formulae represent a significant class of properties investigated in the systems biology literature as identified in [31], such as properties concerning exclusion (*"It is not possible for a state* $\mathbf{s}$ *to occur"*), necessary consequence (*"If a state* $\mathbf{s_1}$ *occurs, then it is necessarily followed by a state* $\mathbf{s_2}$"), and necessary persistence (*"A state* $\mathbf{s}$ *must persist indefinitely"*). On the other hand, properties as occurrence, possible consequence, sequence and possible persistence are of inherently existential nature, and are not expressible in ACTL$^-$.

We define the semantics of ACTL$^-$ formulae on a generic labelled transition system $\mathcal{LTS}$, where each state corresponds to a subset of the possible species S, and transitions are labelled by reactions. This requires

$$
\begin{array}{lcl}
\mathcal{LTS}, \mathbf{s} \vDash_\phi \mathit{true} & & \\
\mathcal{LTS}, \mathbf{s} \nvDash_\phi \mathit{false} & & \\
\mathcal{LTS}, \mathbf{s} \vDash_\phi s & \iff & s \in \mathbf{s} \\
\mathcal{LTS}, \mathbf{s} \vDash_\phi \neg s & \iff & s \notin \mathbf{s} \\
\mathcal{LTS}, \mathbf{s} \vDash_\phi f \wedge g & \iff & \mathcal{LTS}, \mathbf{s} \vDash_\phi f \text{ and } \mathcal{LTS}, \mathbf{s} \vDash_\phi g \\
\mathcal{LTS}, \mathbf{s} \vDash_\phi f \vee g & \iff & \mathcal{LTS}, \mathbf{s} \vDash_\phi f \text{ or } \mathcal{LTS}, \mathbf{s} \vDash_\phi g \\
\mathcal{LTS}, \mathbf{s} \vDash_\phi Af & \iff & \text{for all paths } \pi \text{ starting from } \mathbf{s} \text{ such that} \\
 & & \pi \vDash_{LTL} \phi \text{ it holds } \mathcal{LTS}, \pi \vDash_\phi f \\
\mathcal{LTS}, \pi \vDash_\phi f \ U \ g & \iff & \pi = \mathbf{s_0}, R_0, \mathbf{s_1}, R_1, \ldots \\
 & & \exists k \geq 0.\ \mathcal{LTS}, \mathbf{s_k} \vDash_\phi g \text{ and} \\
 & & \forall 0 \leq j < k.\ \mathcal{LTS}, \mathbf{s_j} \vDash_\phi f \\
\mathcal{LTS}, \pi \vDash_\phi f \ U_w \ g & \iff & \pi = \mathbf{s_0}, R_0, \mathbf{s_1}, R_1, \ldots \\
 & & \forall k \geq 0. \text{ if } \mathcal{LTS}, \mathbf{s_k} \nvDash_\phi f \text{ then} \\
 & & \exists 0 \leq j \leq k.\ \mathcal{LTS}, \mathbf{s_j} \vDash_\phi g
\end{array}
$$

Figure 6: Satisfaction relation for ACTL$^-$ formulae.

defining the satisfaction relation $\vDash_\phi$ over both states and paths, where ϕ is an LTL formula specifying the fairness constraint. In particular, $\mathcal{LTS}, f \vDash_\phi \mathbf{s}$ means that f is satisfied by state $\mathbf{s}$ of $\mathcal{LTS}$, while $\mathcal{LTS}, f \vDash_\phi \pi$ means that f is satisfied by path π of $\mathcal{LTS}$. In both cases we consider only fair paths, i.e. path satisfying ϕ.

Definition 7. *Let $\mathcal{LTS} = (\mathcal{P}(S), P, \rightarrow)$ be an LTS, with S being a finite set of species and $P \subseteq reactions(S)$, and ϕ an LTL formula specifying the fairness constraint. The satisfaction relation $\vDash_\phi$ is inductively defined as in Figure 6.*

Example 8. *Let us consider $LTS_\alpha(P' \restriction J)$ as in Example 6 with initial state $\mathbf{s'_0}$. It holds*

$$LTS_\alpha(P' \restriction J), \mathbf{s'_0} \vDash_\phi AF\ (P_1^* \wedge \neg M_2)$$

since every fair path terminates with a ϵ loop in either $\mathbf{s'_3}$, $\mathbf{s'_{10}}$, or $\mathbf{s'_{15}}$, and in all of these states $(P_1^* \wedge \neg M_2)$ holds.

By the theorem to be proved in the next section we have that property $AF\ (P_1^* \wedge \neg M_2)$ also holds in P' from Example 3. Consequently, the property also holds in the original pathway P defined in Example 1.

4.3 Modular Verification Theorems

Now we prove that in order to verify an ACTL^-_J property for a pathway P and a set of components J, it is enough to verify the same property in the abstract semantics of the abstract pathway $P{\uparrow}J$. The principle behind property preservation is that each path in the semantics of the modelled pathway must have a corresponding abstract path in the abstract semantics of a model obtained by projection. This, combined with the fact that ACTL^- properties are universally quantified (namely describe properties that have to be satisfied by all paths) ensures that if an ACTL^- property holds in the abstract semantics of the projection, then it will also hold in the semantics of the original model. In fact, for the components considered in a projection the semantics of the original model will contain essentially a subset of the paths of the projected model.

First we define the path projection, which from a path in semantics of a pathway with the set of components I removes transitions made by components outside of portion $J \subseteq I$ and restricts the rest of transitions onto J.

Definition 8. *Let* $\pi = \mathbf{s_0}, R_0, \mathbf{s_1}, R_1, \mathbf{s_2}, \ldots$.

$$\pi{\lceil}J = \begin{cases} (\mathbf{s_0}{\lceil}J, \epsilon)^\infty & \text{if } \forall i \geq 0.\ \mathbf{s_i}{\lceil}J = \mathbf{s_{i+1}}{\lceil}J \\ \mathbf{s_k}{\lceil}J, R_k{\uparrow}J, \pi^{k+1}{\lceil}J & \text{if } \exists k \geq 0.\ \mathbf{s_k}{\lceil}J \neq \mathbf{s_{k+1}}{\lceil}J \\ & \text{and } \forall 0 \leq i < k.\ \mathbf{s_i}{\lceil}J = \mathbf{s_{i+1}}{\lceil}J \end{cases}$$

where $(\mathbf{s}, \epsilon)^\infty$ *denotes the path such that* $(\mathbf{s}, \epsilon)^\infty = \mathbf{s}, \epsilon, (\mathbf{s}, \epsilon)^\infty$, *and* $\mathbf{s}{\lceil}J = \mathbf{s} \cap \mathit{species}(J)$.

Now we are in the position to present the crucial result, which states that a fair path in the semantics of a pathway P is projected into an abstract fair path in the abstract semantics of the abstract pathway $P{\uparrow}J$. It is split into four lemmas. Lemma 1 states that if there exists a transition in the semantics of P describing the occurrence of a reaction that changes the portion of the state specified by J, then a corresponding transition exists in the abstract semantics of $P{\uparrow}J$. Lemma 2 considers fair paths in the semantics of a pathway and states that, if from a certain point onwards the portion of the state induced by J never changes, then there is a corresponding transition labelled by ϵ in the abstract semantics. Lemmas 1 and 2 are used to prove Lemma 3, which shows that, given a fair path π in the semantics, its projection $\pi{\lceil}J$ according to Definition 8 is a path in the abstract semantics

of $P\restriction J$. Finally, Lemma 4 states that the abstraction of a fair path is also fair with respect to the abstract fairness.

Lemma 1. *Let* $R \neq \epsilon$. *If* $\mathbf{s} \xrightarrow{R} \mathbf{s}'$ *and* $\mathbf{s}\lceil J \neq \mathbf{s}'\lceil J$ *then* $\mathbf{s}\lceil J \xrightarrow{R\restriction J}_{\alpha} \mathbf{s}'\lceil J$.

Proof. The proof is by induction on the inference rules of Definition 1.
Case (no-cat). We assume transition $\mathbf{s} \xrightarrow{R} \mathbf{s} \cup pro(R)$, such that $\mathbf{s}\lceil J \neq (\mathbf{s} \cup pro(R))\lceil J$. Hence $\mathbf{s}\lceil J \neq \mathbf{s}\lceil J \cup pro(R)\lceil J$, which implies $pro(R)\lceil J \not\subseteq \mathbf{s}\lceil J$ and $pro(R\restriction J) \not\subseteq \mathbf{s}\lceil J$. Moreover, as regards the reactants, $re(R) \subseteq \mathbf{s}$ implies $re(R)\lceil J \subseteq \mathbf{s}\lceil J$, thus $re(R\restriction J) \subseteq \mathbf{s}\lceil J$.

In order to show that transition $\mathbf{s}\lceil J \xrightarrow{R\restriction J}_{\alpha} \mathbf{s}'\lceil J$ with $\mathbf{s}' = \mathbf{s} \cup pro(R)$ can always be derived, we consider the different forms of rule R.

1. If either (i) $comp(R) \subseteq J$; or (ii) $comp(R)\backslash J \neq comp(R)$ and $re(R\restriction J) \neq \emptyset$ then, in both cases, $R\restriction J \in PR_{nc} \cup AR_{nc}$. For the first case, it holds that $R\restriction J = R$. For the second case, note that by the assumed positional correspondence between reactants and products and by the definition of projection we have that $re(R\restriction J) \neq \emptyset$ iff $pro(R\restriction J) \neq \emptyset$. Hence, if $re(R\restriction J)$ was empty we would have also $pro(R\restriction J)$ empty and hence $\mathbf{s}\lceil J = \mathbf{s}'\lceil J$. Rule premises are already satisfied, thus transition $\mathbf{s}\lceil J \xrightarrow{R\restriction J}_{\alpha} \mathbf{s}\lceil J \cup pro(R\restriction J)$ can be derived, where the target state is such that $\mathbf{s}\lceil J \cup pro(R\restriction J) = \mathbf{s}\lceil J \cup pro(R)\lceil J = (\mathbf{s} \cup pro(R))\lceil J = \mathbf{s}'\lceil J$.

2. The case $comp(R) \cap J = \emptyset$ cannot occur, since it would imply that $pro(R)\lceil J = \emptyset$, which is absurd since $pro(R)\lceil J \not\subseteq \mathbf{s}\lceil J$.

Case (cat). Let us assume transition $\mathbf{s} \xrightarrow{R} (\mathbf{s}\backslash re(R)) \cup pro(R)$ can be derived by using rule (cat), with $\mathbf{s}\lceil J \neq ((\mathbf{s}\backslash re(R)) \cup pro(R))\lceil J$. Similarly to the previous case, premises of the rule imply both $re(R\restriction J) \subseteq \mathbf{s}\lceil J$ and $cat(R\restriction J) \subseteq \mathbf{s}\lceil J$. Moreover, $(\mathbf{s}\backslash re(R) \cup pro(R))\lceil J = (\mathbf{s}\backslash re(R))\lceil J \cup pro(R)\lceil J = (\mathbf{s}\lceil J)\backslash re(R\restriction J) \cup pro(R\restriction J)$. Therefore $\mathbf{s}\lceil J \neq (\mathbf{s}\lceil J)\backslash re(R\restriction J) \cup pro(R\restriction J)$, which implies that either (i) $re(R\restriction J) \not\subseteq pro(R\restriction J)$, since $re(R\restriction J) \subseteq \mathbf{s}\lceil J$, or (ii) $pro(R\restriction J) \not\subseteq \mathbf{s}\lceil J$.

Finally, we consider the different forms of reaction R.

1. If either $comp(R) \subseteq J$ or $comp(R)\backslash J \neq comp(R)$ then, in both cases, $R\restriction J \in PR_c \cup AR_c$, allowing transition $\mathbf{s}\lceil J \xrightarrow{R\restriction J}_{\alpha} (\mathbf{s}\lceil J)\backslash re(R\restriction J) \cup pro(R\restriction J)$ to be derived, where the target state is s.t. $(\mathbf{s}\lceil J)\backslash re(R\restriction J) \cup pro(R\restriction J) = (\mathbf{s}\backslash re(R) \cup pro(R))\lceil J$.

2. The case $comp(R) \cap J = \emptyset$ cannot occur, since it would imply both $re(R)\lceil J = \emptyset$ and $pro(R)\lceil J = \emptyset$, which is absurd since we assumed that either $re(R)\lceil J \not\subseteq pro(R)\lceil J$ or $pro(R)\lceil J \not\subseteq \mathbf{s}\lceil J$. □

Lemma 2. *Let $\pi = \mathbf{s_0}, R_0, \mathbf{s_1}, R_1, \ldots \in LTS(P)$ be such that $\pi \vDash_{LTL} \Phi$. If there exists $k \geq 0$ such that $\forall i \geq k.\ \mathbf{s_i}\lceil J = \mathbf{s_{i+1}}\lceil J$ then $\mathbf{s_k}\lceil J \xrightarrow{\epsilon}_\alpha \mathbf{s_k}\lceil J$.*

Proof. Let $k \geq 0$ be such that $\forall i \geq k.\ \mathbf{s_i}\lceil J = \mathbf{s_{i+1}}\lceil J$. We distinguish three possible cases, according to the form of the path π, as detailed in the following.

Case $\forall i \geq k.\ R_i = \epsilon$. This implies that $\forall R.\ \mathbf{s_i} \overset{R}{\not\rightarrow}$. As regards the abstract pathway, no reaction in $R \in PR_c \cup PR_{nc}$ is applicable in $\mathbf{s_i}\lceil J$, otherwise transition $\mathbf{s_i} \xrightarrow{R} \mathbf{s'}$, for some $\mathbf{s'}$, could be derived, contradicting the hypothesis.

Therefore, if there is at least one abstract reaction $R\restriction J \in AR_c \cup AR_{nc}$ applicable in $\mathbf{s_i}\lceil J$ (i.e. a transition using either rule (no-cat)/(cat) for $R\restriction J$ can be derived) then a corresponding transition $\mathbf{s_i}\lceil J \xrightarrow{\epsilon}_\alpha \mathbf{s_i}\lceil J$ can be also derived by using rule (self-loop). On the other hand, if no abstract reaction $R\restriction J \in AR_c \cup AR_{nc}$ is applicable in $\mathbf{s_i}\lceil J$, then transition $\mathbf{s_i}\lceil J \xrightarrow{\epsilon}_\alpha \mathbf{s_i}\lceil J$ can be derived by using rule (deadlock).

Case $\forall i \geq k.\ R_i \neq \epsilon$. Reactions R_i, $\forall i \geq k$, cannot be used to derive transitions in the semantics of the abstract pathway since they do not change $\mathbf{s_i}\lceil J$. Let us suppose there exists a reaction $R\restriction J \in AR_c \cup AR_{nc}$ which is applicable in state $\mathbf{s_k}\lceil J$, i.e. $\mathbf{s_k}\lceil J \xrightarrow{R\restriction J}_\alpha \mathbf{s'}$ for some $\mathbf{s'} \neq \mathbf{s_k}\lceil J$, using either rule (no-cat)/(cat). Therefore rule (self-loop) can also be applied, allowing transition $\mathbf{s_k}\lceil J \xrightarrow{\epsilon}_\alpha \mathbf{s_k}\lceil J$ to be derived.

On the other hand, let us assume that no reaction $R\restriction J \in AR_c \cup AR_{nc}$ is applicable in state $\mathbf{s_k}\lceil J$, therefore rule (self-loop) cannot be applied. Let us suppose, by absurd, that also rule (deadlock) cannot be applied, namely $\exists R \in PR_c \cup PR_{nc}.\ \mathbf{s_k}\lceil J \xrightarrow{R}_\alpha \mathbf{s'}$, for some $\mathbf{s'} \neq \mathbf{s_k}\lceil J$. Since $\forall i \geq k.\ \mathbf{s_i}\lceil J = \mathbf{s_k}\lceil J$, this implies that $\forall i \geq k.\ \mathbf{s_i}\lceil J \xrightarrow{R} \mathbf{s'}$. Because reaction R is the same as in the original pathway, this also entails that $\forall i \geq k.\ \exists \mathbf{s''}.\ \mathbf{s_i} \xrightarrow{R} \mathbf{s''}$, namely reaction R is enabled infinitely often in path π. According to the fairness constraint, such a reaction R need to occur in the path, i.e. there must exist $h \geq k$ such that $\mathbf{s_h} \xrightarrow{R} \mathbf{s_{h+1}}$ with $\mathbf{s_h} \neq \mathbf{s_{h+1}}$. Because $R \in PR_c \cup PR_{nc}$, this implies that $\mathbf{s_h}\lceil J \neq \mathbf{s_{h+1}}\lceil J$, which is a contradiction since we assumed that $\forall i \geq k.\ \mathbf{s_i}\lceil J = \mathbf{s_{i+1}}\lceil J$.

Case $\exists h > k.\ (\forall k \leq i < h.\ R_i \neq \epsilon) \wedge (\forall j \geq h.\ R_j = \epsilon)$. Note

that, by definition of semantics of pathways, it is not possible to have a transition with label R, for some reaction R, after a transition with label ϵ. The first case can be applied to subpath π' starting from $\mathbf{s_h}$, i.e. $\pi' = \mathbf{s_h}, \mathbf{s_{h+1}}, \ldots$. This implies that $\mathbf{s_h}\lceil J \xrightarrow{\epsilon}_\alpha \mathbf{s_h}\lceil J$, and thus $\mathbf{s_k}\lceil J \xrightarrow{\epsilon}_\alpha \mathbf{s_k}\lceil J$ since $\mathbf{s_k}\lceil J = \mathbf{s_{k+1}}\lceil J = \cdots = \mathbf{s_h}\lceil J$. □

Lemma 3. *$\pi \in LTS(P)$ with $\pi \vDash_{LTL} \Phi$ implies $\pi\lceil J \in LTS_\alpha(P\restriction J)$*

Proof. Let $\pi = \mathbf{s_0}, R_0, \mathbf{s_1}, R_1, \mathbf{s_2}, \ldots$. The proof is by cases on the definition of path projection $\pi\lceil J$ from Definition 8.

Case $\forall i \geq 0.\ \mathbf{s_i}\lceil J = \mathbf{s_{i+1}}\lceil J$. According to Lemma 2, transition $\mathbf{s_0}\lceil J \xrightarrow{\epsilon}_\alpha \mathbf{s_0}\lceil J$ is present in the abstract semantics, thus $(\mathbf{s_0}\lceil J, \epsilon)^\infty \in LTS_\alpha(P\restriction J)$.

Case $\exists k \geq 0.\ \mathbf{s_k}\lceil J \neq \mathbf{s_{k+1}}\lceil J$ and $\forall 0 \leq i < k.\ \mathbf{s_i}\lceil J = \mathbf{s_{i+1}}\lceil J$. Let us consider transition $\mathbf{s_k} \xrightarrow{R_k}_\alpha \mathbf{s_{k+1}}$; according to the semantics, condition $\mathbf{s_k}\lceil J \neq \mathbf{s_{k+1}}$ implies that $R_k \neq \epsilon$. Therefore, it follows from Lemma 1 that $\mathbf{s_k}\lceil J \xrightarrow{R\restriction J}_\alpha \mathbf{s_{k+1}}\lceil J$. Let us assume that $\pi^{k+1}\lceil J \in LTS_\alpha(P\restriction J)$. It is easy to see that, according to Definition 8, the starting state of $\pi^{k+1}\lceil J$ is $\mathbf{s_{k+1}}\lceil J$. Therefore, $\pi\lceil J \in LTS_\alpha(P\restriction J)$. □

Lemma 4. *$\pi \in LTS(P)$ with $\pi \vDash_{LTL} \Phi$ implies $\pi\lceil J \vDash_{LTL} \Phi_\alpha$.*

Proof. Suppose that $\pi \vDash_{LTL} \Phi$, i.e. $\pi \vDash_{LTL} \bigwedge_{R\in P}(GF\ enabled(R)) \rightarrow (GF\ occurs(R))$. Let $P\restriction J = (PR_c, PR_{nc}, AR_c, AR_{nc})$. We want to prove that $\pi\lceil J \vDash_{LTL} \Phi_\alpha$, that is $\pi\lceil J \vDash_{LTL} \bigwedge_{R\in PR_c\cup PR_{nc}}(GF\ enabled(R)) \rightarrow (GF\ occurs(R))$. It is easy to see that, for any reaction R from $PR_c \cup PR_{nc}$, it holds that, for all $i \geq 0$:

1. $\pi^i \vDash_{LTL} enabled(R)$ implies $\pi^i\lceil J \vDash_{LTL} enabled(R)$;
2. $\pi^i \vDash_{LTL} occurs(R)$ implies $\pi^i\lceil J \vDash_{LTL} occurs(R)$.

The proof is concluded by noting that, since π^i is a suffix of π, then, by Definition 8, $\pi^i\lceil J$ is a suffix of $\pi\lceil J$. □

We conclude this section with the property preservation theorem, which states that a successful verification of a property in the abstraction implies its truth in the original model.

Theorem 1. *For a pathway P, $J \subseteq comp(P)$, and f an $ACTL^-_J$ formula we have $LTS_\alpha(P\restriction J), \mathbf{s}\lceil J \vDash_{\Phi_\alpha} f$ implies $LTS(P), \mathbf{s} \vDash_\Phi f$.*

Proof. By induction on the structure of f.

$f = s$. By definition of state projection, since $s \in species(J)$, then $s \in \mathbf{s}\lceil J$ iff $s \in \mathbf{s}$, i.e. $LTS_\alpha(P\restriction J), \mathbf{s}\lceil J \vDash_{\Phi_\alpha} s$ iff $LTS(P), \mathbf{s} \vDash_\Phi s$. Analogously for $f = \neg s$.

$f = g \wedge h$. By induction hypothesis.

$f = g \vee h$. By induction hypothesis.

$f = A[g\ U\ h]$. Given a fair path π in $LTS(P)$ starting from $\mathbf{s}$, according to Lemmas 3 and 4 it holds that $\pi\lceil J$ is a fair path in $LTS_\alpha(P\restriction J)$. By definition of path projection, $\pi\lceil J$ starts from $\mathbf{s}\lceil J$. Since $LTS_\alpha(P\restriction J), \mathbf{s}\lceil J \vDash_{\Phi_\alpha} f$, it holds $LTS_\alpha(P\restriction J), \pi\lceil J \vDash_{\Phi_\alpha} g\ U\ h$. As a consequence, $\pi\lceil J = \mathbf{s_0}, R_0, \mathbf{s_1}, R_1, \ldots$ and $\exists k \geq 0.\ LTS_\alpha(P\restriction J), \mathbf{s_k} \vDash_{\Phi_\alpha} h$ and $\forall 0 \leq j < k.\ LTS_\alpha(P\restriction J), \mathbf{s_j} \vDash_{\Phi_\alpha} g$. By definition of path projection, $\pi = \mathbf{s_0}', R_0', \mathbf{s_1}', R_1', \ldots$, and there exists $k' \geq k$ such that $\forall 0 \leq i < k'.\ \exists 0 \leq j < k.\ \mathbf{s_i}'\lceil J = \mathbf{s_j}$, and $\mathbf{s_{k'}}'\lceil J = \mathbf{s_k}$. Therefore, $LTS(P), \pi \vDash_\Phi g\ U\ h$ and hence $LTS(P), \mathbf{s} \vDash_\Phi f$.

$f = A[g\ U_w\ h]$. Given a fair path π in $LTS(P)$ starting from $\mathbf{s}$, according to Lemmas 3 and 4 it holds that $\pi\lceil J$ is a fair path in $LTS_\alpha(P\restriction J)$. By definition of path projection, $\pi\lceil J$ starts from $\mathbf{s}\lceil J$. Since $LTS_\alpha(P\restriction J), \mathbf{s}\lceil J \vDash_{\Phi_\alpha} f$, it holds $LTS_\alpha(P\restriction J), \pi\lceil J \vDash_{\Phi_\alpha} g\ U_w\ h$. This implies that either $LTS_\alpha(P\restriction J), \pi\lceil J \vDash_{\Phi_\alpha} g\ U\ h$ or $LTS_\alpha(P\restriction J), \pi\lceil J \vDash_{\Phi_\alpha} G\ g$. In the first case the proof is the same as for $f = A[g\ U\ h]$. In the second case we have that $\pi\lceil J = \mathbf{s_0}, R_0, \mathbf{s_1}, R_1, \ldots$ and $\forall k \geq 0.\ LTS_\alpha(P\restriction J), \mathbf{s_k} \vDash_{\Phi_\alpha} g$. By definition of path projection, $\pi = \mathbf{s_0}', R_0', \mathbf{s_1}', R_1', \ldots$, and $\forall k' \geq 0.\ \exists k \geq 0.\ \mathbf{s_{k'}}'\lceil J = \mathbf{s_k}$. Therefore, $LTS(P), \pi \vDash_\Phi G\ g$ and hence $LTS(P), \mathbf{s} \vDash_\Phi f$. □

5 Modelling the EGF Receptor-Induced MAP Kinase Cascade

We apply our modular verification approach to a well-established computational model of the EGF signalling pathway. We consider the model of the MAP kinase cascade activated by surface and internalised EGF receptors, proposed by Schoeberl et al. in [37]. This model includes a detailed description of the reactions that involve active EGF receptors and several effectors named GAP, ShC, SOS, Grb2, RasGDP/GTP and Raf. Moreover, the model describes the activity of internalised receptors, namely receptors that are no longer located on the cell membrane, but on a vesicle obtained by endocytosis and floating in the cytoplasm. Such internalised receptors continue to interact with effectors and to contribute to the pathway functioning, but actually the pathway can be seen as composed by two almost identical

branches: the first consisting of the reactions stimulated by receptors on the cell membrane and the second consisting of reactions stimulated by internalised receptors.

A diagram representing all of the reactions of the pathway considered in the model is shown in Figure 7. In the figure, each species is identified by a short name, and also by a number (in black) in the interval $[1 - 87]$. The number of species which occur more than once is shown only for one of their occurrences. Arrows represent reactions, which are also associated with an identifier (in grey), for a total of 102 reactions. The two branches of the pathway are partially combined in the figure. In particular, the representation of most of the species is combined with the representation of its internalised counterpart. In such cases, the number between brackets identifies the internalised species. The same holds for reactions: in many cases an arrow denotes both a reaction stimulated by receptors in the cell membrane and the corresponding reaction stimulated by internalised receptors.

The set of reactions constituting the pathway can be trivially reconstructed from the diagram in Figure 7. The only non-trivial aspect is related with the presence in the diagram of some reactions in which one reactants is actually acting as a catalyst. For instance, this happens in the case of the reactions involving Raf* and MEK, in which Raf* initially binds MEK and then releases it phosphorylated. We describe these two reactions in the diagram with the following single catalysed reaction:

$$\text{MEK} \rightarrow \text{MEK-P } \{\text{Raf}^*\}$$

Other species acting as catalysts are MEK-PP, Phosphatase1, Phosphatase2 and Phosphatase3. By applying the same transformation also to the reactions they are involved in we obtain a pathway constituted by 80 reactions, which constitutes the starting point for the application of the techniques presented in the paper. We call this pathway P_{EGF}.

We recall that fairness requires that a reaction that is infinitely often enabled is also infinitely often performed. This prevents starvation situations to happen among reactions. In the case of P_{EGF} the two branches of the pathway include reactions that could be involved in infinite loops (e.g. the reactions involving MEK and ERK). This means that the semantics of the pathway includes behaviours in which only one branch executes forever even if the other is constantly enabled. Such unrealistic behaviours are excluded by the adoption of fairness.

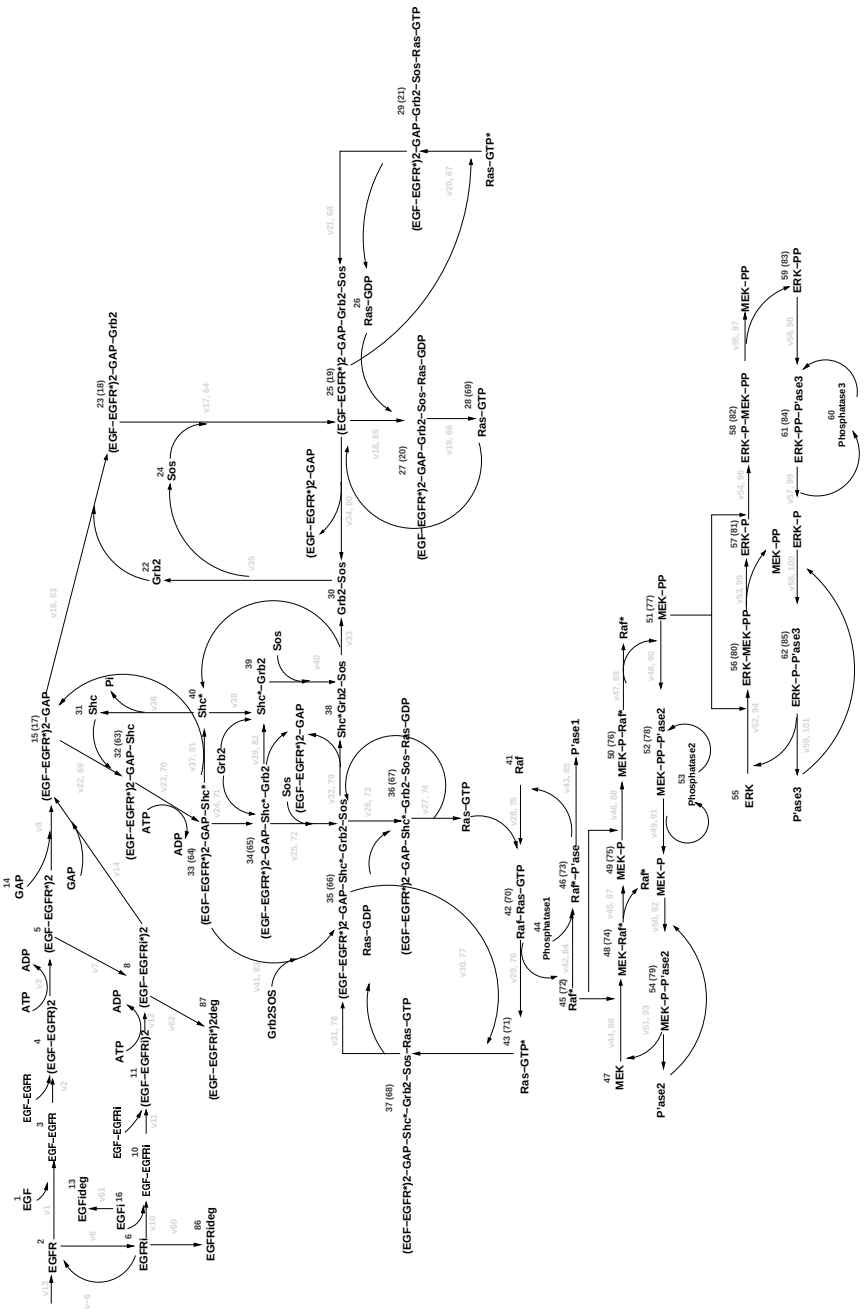

Figure 7: Scheme of the EGF receptor-induced MAP kinase cascade [37].

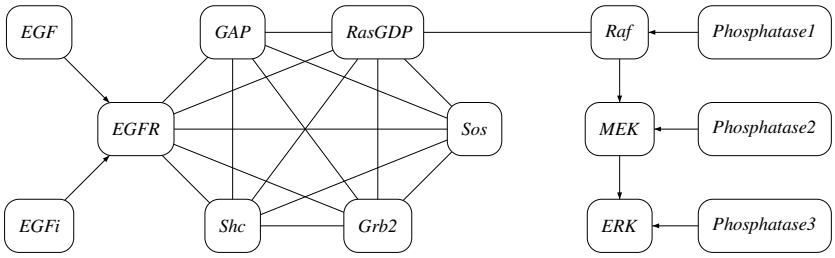

Figure 8: Component interaction graph of P_{EGF}.

5.1 Initial State

We adopt a semi-automatic heuristic procedure to find an initial state of the pathway. The idea is the following: for each species s in $species(P)$, if there is no reaction creating it (i.e. if $s \notin \bigcup_{R \in P} pro(R)$) then in the initial state s is present. This means that species that cannot be produced are assumed to be present in the initial state. Otherwise their presence in the model would not be meaningful. Subsequently, we resort again to the partitioning of species according to components to find other species to be inserted. In particular, we find those components containing no species present in the previous phase. These components must contain loops, hence we choose manually some of their species to insert. All other species are assumed absent.

5.2 The Model

The model P_{EGF} is made up of 143 species and 80 reactions. It is in the correct form assumed in Section 3.1 and no preprocessing is needed. After performing the components identification procedure, 14 components are identified. On Figure 8 we can see the component interaction graph of P_{EGF}. Each node of the graph is labelled by the intuitive name of the component that we have chosen.

Visually, we can do some simple observations on the component interaction graph. We can identify enzymes like Phosphatase1, Phosphatase2 and Phosphatase3. We can see the first part of the pathway corresponding to the EGF receptor and its interaction with effectors, and its connection to the MAP kinase cascade through the component RasGDP.

5.3 Experiments

In this section we exploit the NuSMV model checker to perform some experiments on the model of the EGF pathway. NuSMV includes model checking algorithms that allow fairness constraints to be taken into account. We rely on such algorithms to manage fairness constraints introduced in this paper. Moreover, in order to carry out the projection and encode the resulting abstract pathway in the NuSMV format we have developed a tool (available upon request).

The first experiment is aimed at showing how modular verification could be applied to verify a global property of the pathway, namely that the final product of the pathway is always produced. This can be done in a modular way by proving sub-properties in three different model fragments obtained by projection.

Subsequently, a number of experiments are performed with the aim of showing how the molecular components we identified in the pathway can be used to better understand the pathway dynamics. In particular, we check whether there are some molecular components that are not really necessary to obtain the final product of the pathway. This will be done by applying model checking on models in which molecular components are selectively disabled by setting their initial states to false. Also in this case the modular verification approach is adopted.

In this case study modular verification allows properties to be verified faster than on the complete model. However, modular verification is still not significantly more efficient than verification on the complete model. This is due to the projection operation we are considering at the moment, which is rather rough. In Section 6 we discuss why this modular verification is a promising approach for the analysis of pathways, and how we plan to improve the approach to make it substantially more efficient.

To run the experiment we used NuSMV 2.5.4 on a workstation equipped with an Intel i5 CPU 2.80 Ghz, with 8GB RAM and running Ubuntu GNU/Linux. In order to make verification faster NuSMV was executed in batch mode by enabling dynamic reordering of BDD variables and by disabling the generation of counterexamples.

5.3.1 Modular Verification of a Global Property

The final product of the MAP kinase cascade activated by surface and internalised EGF receptors is species ERK-PP. Since surface and internalised

receptors activate two different branches of the pathway, we denote by ERK-PP the product of the branch activated by the surface receptors and by ERK-PPi the product of the branch activated by the internalised receptors.

The property to be verified is

$$AF(\text{ERK-PP} \vee \text{ERK-PPi}) \tag{1}$$

The property holds in the complete model and its verification required 260 seconds. By looking at the diagram in Figure 7 we noticed that the pathway could be partitioned in three parts, with two species acting as "gates". These two species are (EGF-EGFR*)2-GAP and Raf*. Hence, we decided to try to apply modular verification by splitting property 1 into the following three sub-properties:

$$AF((\text{EGF-EGFR}^*)\text{2-GAP}) \tag{2}$$
$$AG((\text{EGF-EGFR}^*)\text{2-GAP} \rightarrow (AF\ \text{Raf}^*)) \tag{3}$$
$$AG(\text{Raf}^* \rightarrow AF(\text{ERK-PP} \vee \text{ERK-PPi})) \tag{4}$$

Property (2) states that in all paths of the system a state in which species (EGF-EGFR*)2-GAP is present is eventually reached. Property (3) states that whenever a state is reached in which species (EGF-EGFR*)2-GAP is present, then a state in which Raf* is present is eventually reached. Finally, property (4) states that whenever a state is reached in which species Raf* is present, then a state in which either ERK-PP or ERK-PPi is present is eventually reached. It is easy to see that the conjunction of (2), (3) and (4) implies (1).

We considered three projections of the complete model to be used to verify properties (2), (3) and (4), respectively. In particular, from the component interaction graph of the model (shown in Figure 8) we extracted the following subsets to be used for projections:

- in order to verify (2) we considered the subset J_1 consisting of components *EGF*, *EGFi*, *EGFR* and *GAP*;
- in order to verify (3) we considered the subset J_2 consisting of components *EGFR*, *GAP*, *Shc*, *RasGDP*, *Grb2* and *Sos*;
- in order to verify (4) we considered the subset J_3 consisting of components *RasGDP*, *Raf*, *MEK*, *ERK*, *Phosphatase1*, *Phosphatase2* and *Phosphatase3*.

We obtained that (2), (3) and (4) hold in the abstract semantics of the abstract pathways $P{\uparrow}J_1$, $P{\uparrow}J_2$ and $P{\uparrow}J_3$, respectively. Moreover, model checking required less than three seconds for (2), 213 seconds for (3) and less than one second for (4). Overall, modular verification required 217 seconds, that is 43 seconds less than verification on the complete model. Note that the time needed to perform pathway projection over a set of components is negligible.

5.3.2 Reasoning on Molecular Components

As it can be seen in the component interaction graph and in the diagram in Figure 7, some molecular components are involved in complex interactions. This is true in particular for components *EGFR*, *GAP*, *RasGDP*, *Sos*, *Shc* and *Grb2* which form a clique in the component interaction graph. We are interested in understanding whether all of these components are really necessary in order to obtain the final products of the pathway. The idea is to test whether the final species are produced when the components of interest are assumed one by one as disabled. Molecular components *EGFR* and *RasGDP* are for sure necessary since they connect the clique with the other molecular components of the pathway. Consequently, we focus our analysis on *GAP*, *Sos*, *Shc* and *Grb2*.

In order to disable a molecular component we consider as absent all of its species in the initial state of the systems. Hence, we consider a set of four (complete) models, each with one of the four components under study disabled. On each model we try to verify property (1): if the property does not hold, then the component that is disabled in such a model is necessary for the pathway; on the other hand, if the property holds, then the component turns out to be not necessary since the products of the pathway can be obtained even without it. The same tests can be also done in a modular way by decomposing the pathway and the property as in Section 5.3.1.

In Table 1 we summarise the property verification results and compare verification times obtained by model checking the complete models and by following the modular approach. The first row of data in the table reports verification results in which no component is disabled (as in Section 5.3.1). The other results show that *Shc* is not a necessary component, whereas all of the other three are. As previously, the time required by modular verification is smaller than the one required by model checking the complete model. This is true in particular in the case in which *GAP* is disabled since property (2), the verification of which is very fast, turns out to be false.

	Verification complete model			Modular Verification			
Disabled component	**Property**	**Result**	**Time**	**Property**	**Result**	**Time**	**Total time**
none	(1)	true	260s	(2)	true	3s	217s
				(3)	true	213s	
				(4)	true	1s	
GAP	(1)	false	252s	(2),(5)	false,true	2s	2s
Sos	(1)	false	253s	(2)	true	3s	210s
				(3),(6)	false,true	207s	
Shc	(1)	true	252s	(2)	true	3s	212s
				(3)	true	208s	
				(4)	true	1s	
Grb2	(1)	false	253s	(2)	true	3s	211s
				(3),(6)	false,true	208s	

Table 1: Model checking results and comparison of verification times. (Time needed for constructing reduced models is negligible.)

Note that in the case of modular verification of the models in which *GAP*, *Sos* and *Grb2* were disabled we needed to verify some additional properties. In particular, in the case of *GAP* we have that property (2) does not hold in the abstract semantics of $P{\restriction}J_1$, and in the cases of *Sos* and *Grb2* property (3) does not hold in the abstract semantics of $P{\restriction}J_2$. We remark that our modular verification approach guarantees only that properties proved to hold in a model fragment also hold in the complete model. Nothing can be said, instead, of properties that does not hold in the model fragments. In order to avoid applying model checking on the complete model to check whether these properties hold there, we consider some new properties whose satisfaction in suitable model fragments implies that properties (2) and (3) actually do not hold. In order to prove that (2) is actually false when *GAP* is disabled we consider the following property:

$$AG(\neg(\text{EGF-EGFR}^*)\text{2-GAP}) \tag{5}$$

In order to prove that (3) is actually false when either *Sos* or *Grb2* is disabled we consider the following property:

$$AG(\neg\text{Raf}^*) \tag{6}$$

Note that it is convenient to verify properties (5) and (6) together with (2) and (3), respectively. This avoids spending twice the time needed by the model checker to construct the data structure necessary to perform the verification. In the case of our experiments the construction of such data structures takes usually the 98%-99% of the verification time. Times reported in Table 1 are based on this optimisation.

6 Discussion and Conclusions

In this paper we presented preliminary results in the development of a modular verification framework for biochemical pathways. We defined a modelling notation for pathways associated with a formal semantics and a notion of fairness that allows the dynamics to be accurately described by avoiding starvation situations among reactions. Moreover, we investigated a notion of molecular component of a pathway and we provided a methodology to infer molecular components from pathways the reactions of which satisfy some assumptions. Molecular components were then used by a projection operation that allows abstract pathways modelling an over-approximation

of the behaviour of a group of components to be obtained from a pathway model. The fact that a property expressed by means of the ACTL$^-$ logic holds in an abstract pathway was shown to imply that they hold also in the complete pathway model. This preservation is at the basis of the modular verification approach which was demonstrated on a well-established model of the EGF pathway.

The results of experiments given in Section 5.3 show that our modular verification approach allows properties to be verified in a shorter time than in the case of verification of the complete pathway model. However, in most of the cases the time saved was relatively small ($\sim 15\%$). We believe that the cause of this limited gain in efficiency is due to the projection operation we are considering at the moment, which is still somewhat rough. Our plan to improve efficiency is to improve the projection operation by including in it minimisation of components. In particular, projection could be used to abstract from some of the model components as it happens now, but also to minimise the remaining components. Minimisation should be aimed at removing from the model all the reactions describing internal changes of the considered components. Indeed, removing internal changes do not affect satisfiability of properties (if the state reached after an internal change is actually not mentioned in the considered property). Minimisation can be obtained by translating a component and its reactions into a finite state automaton, and then by applying a standard minimisation algorithm on it. Minimisation could also be aimed at replacing sequences of reactions with single reactions when they involve more than one component but do not present any branching opportunity in between the sequence (i.e. no other reaction is applicable to the same species in between the sequence). These minimisation operations would allow, for example, to reduce the size of the model of the components constituting the clique in the component interaction graph in Figure 8. Indeed, most of the reactions involving *GAP*, *Shc*, *Sos* and *Grb2* could be probably removed from the model. This would allow for a significant improvement in modular verification efficiency.

In the case study, the choice of splitting the global property into three sub-properties has been crucial since it allowed our modular verification methodology to be successfully applied. In general, finding a set of sub-properties suitable for modular verification when possible can be a difficult task. Most real pathways, however, when observed at a very abstract level, show a sequential dynamics in which from initial substrates a sequence of intermediate species are obtained until the final product is produced. During

this roughly sequential process there might be some components involved only at the beginning, others only in intermediate stages, and others only at the end. The component interaction graph can help the modeller to identify which molecular components are involved in which stage, and consequently which species are "checkpoints" between subsequent stages. In turn, this can suggest how to split the global property into a set of sub-properties in a way similar to the one we followed, namely by testing with each sub-property the reachability of states in which checkpoint species for stage n are present by assuming that checkpoint species for stage $n-1$ have already been produced. We believe that this approach could also be automatised, although it will not be successful for all pathways and for all properties. Indeed, this approach is actually an instantiation of the *assume-guarantee* paradigm for which automatisations have been proposed [7].

Moreover, we believe that with more complex and realistic pathways splitting properties into sub-properties will be less needed since often the properties will not be as "global" as in our case study, and there will probably be many components whose contribution to the satisfaction of such properties is limited. In addition, the definition of a finer projection operation could allow property splitting to become less and less necessary.

References

[1] Paul C. Attie and E. Allen Emerson. Synthesis of concurrent systems with many similar processes. *ACM Transactions on Programming Languages and Systems*, 20(1):51–115, 1998. doi:10.1145/271510.271519.

[2] Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo, and Angelo Troina. A calculus of looping sequences for modelling microbiological systems. *Fundamenta Informaticae*, 72(1-3):21–35, 2006.

[3] Gilles Bernot and Fariza Tahi. Behaviour preservation of a biological regulatory network when embedded into a larger network. *Fundamenta Informaticae*, 91(3):463–485, 2009. doi:10.3233/FI-2009-0052.

[4] Chiara Bodei, Andrea Bracciali, and Davide Chiarugi. On deducing causality in metabolic networks. *BMC Bioinformatics*, 9(Suppl 4):S8, 2008. doi:10.1186/1471-2105-9-S4-S8.

[5] Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992. doi:10.1016/0890-5401(92)90017-A.

[6] Luca Cardelli. Brane calculi. In Vincent Danos and Vincent Schachter, editors, *Computational Methods in Systems Biology*, volume 3082 of *Lecture Notes in Computer Science*, pages 257–278. Springer Berlin Heidelberg, 2005. doi:10.1007/978-3-540-25974-9_24.

[7] Sagar Chaki, Edmund Clarke, Nishant Sinha, and Prasanna Thati. Automated assume-guarantee reasoning for simulation conformance. In Kousha Etessami and SriramK. Rajamani, editors, *Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, pages 534–547. Springer Berlin Heidelberg, 2005. doi:10.1007/11513988_51.

[8] Claudine Chaouiya, Elisabeth Remy, and Denis Thieffry. Petri net modelling of biological regulatory networks. *Journal of Discrete Algorithms*, 6(2):165–177, 2008. doi:10.1016/j.jda.2007.06.003.

[9] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV version 2: An opensource tool for symbolic model checking. In *Proc. International Conference on Computer-Aided Verification (CAV 2002)*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364, Copenhagen, Denmark, 2002. Springer Berlin Heidelberg. doi:10.1007/3-540-45657-0_29.

[10] Federica Ciocchetta and Jane Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410(33-34):3065–3084, 2009. doi:10.1016/j.tcs.2009.02.037.

[11] Edmund M. Clarke, Orna Grumberg, and David E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994. doi:10.1145/186025.186051.

[12] Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, 1999.

[13] Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, and Jean Krivine. Abstracting the differential semantics of rule-based models:

exact and automated model reduction. In *Logic in Computer Science (LICS), 2010 25th Annual IEEE Symposium on*, pages 362–381. IEEE, 2010. doi:10.1109/LICS.2010.44.

[14] Vincent Danos and Cosimo Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004. doi:10.1016/j.tcs.2004.03.065.

[15] Maria I Davidich and Stefan Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. *PLoS One*, 3(2):e1672, 2008. doi:10.1371/journal.pone.0001672.

[16] Franck Delaplace, Hanna Klaudel, and Amandine Cartier-Michaud. Discrete causal model view of biological networks. In *Proceedings of the 8th International Conference on Computational Methods in Systems Biology*, CMSB '10, pages 4–13, New York, NY, USA, 2010. ACM. doi:10.1145/1839764.1839767.

[17] Franck Delaplace, Hanna Klaudel, Tarek Melliti, and Sylvain Sené. Analysis of modular organisation of interaction networks based on asymptotic dynamics. In David Gilbert and Monika Heiner, editors, *Computational Methods in Systems Biology*, Lecture Notes in Computer Science, pages 148–165. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-33636-2_10.

[18] Peter Drábik, Andrea Maggiolo-Schettini, and Paolo Milazzo. Dynamic sync-programs for modular verification of biological systems. In *2nd Int. Workshop on Non-Classical Models of Automata and applications (NCMA'10)*, volume 263, Jena, Germany, 2010. Austrian Computer Society.

[19] Peter Drábik, Andrea Maggiolo-Schettini, and Paolo Milazzo. Modular verification of interactive systems with an application to biology. *Electronic Notes in Theoretical Computer Science*, 268:61–75, 2010. doi:10.1016/j.entcs.2010.12.006.

[20] Peter Drábik, Andrea Maggiolo-Schettini, and Paolo Milazzo. Modular verification of interactive systems with an application to biology. *Scientific Annals of Computer Science*, 21:39–72, 2011.

[21] Peter Drábik, Andrea Maggiolo-Schettini, and Paolo Milazzo. On conditions for modular verification in systems of synchronising components. *Fundamenta Informaticae*, 120(3-4):259–274, 2012. doi:10.3233/FI-2012-761.

[22] Peter Drábik, Andrea Maggiolo-Schettini, and Paolo Milazzo. Towards modular verification of pathways: fairness and assumptions. In Gabriel Ciobanu, editor, Proceedings 6th Workshop on *Membrane Computing and Biologically Inspired Process Calculi,* Newcastle, UK, 8th September 2012, volume 100 of *Electronic Proceedings in Theoretical Computer Science*, pages 63–81. Open Publishing Association, 2012. doi:10.4204/EPTCS.100.5.

[23] E. Allen Emerson and Chin-Laung Lei. Modalities for model checking: branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987. doi:10.1016/0167-6423(87)90036-0.

[24] François Fages, Sylvain Soliman, and Nathalie Chabrier-Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4(2):64–73, 2004. doi:10.4024/2040402.jbpc.04.02.

[25] Jérôme Feret, Vincent Danos, Jean Krivine, Russ Harmer, and Walter Fontana. Internal coarse-graining of molecular systems. *Proceedings of the National Academy of Sciences*, 106(16):6453–6458, 2009. doi:10.1073/pnas.0809908106.

[26] Orna Grumberg and David E Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(3):843–871, 1994. doi:10.1145/177492.177725.

[27] John Heath, Marta Kwiatkowska, Gethin Norman, David Parker, and Oksana Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 391(3):239–257, 2008. doi:10.1016/j.tcs.2007.11.013.

[28] M. Hucka, A. Finney, H.M. Sauro, H. Bolouri, J.C. Doyle, H. Kitano, and the rest of the SBML Forum. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003. doi:10.1093/bioinformatics/btg015.

[29] C. Li, M. Donizelli, N. Rodriguez, H. Dharuri, L. Endler, V. Chelliah, L. Li, E. He, A. Henry, M.I. Stefan, J.L. Snoep, M. Hucka, N. Le Novère, and C. Laibe. BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, 4:92, 2010. doi:10.1186/1752-0509-4-92.

[30] Andrea Maggiolo-Schettini, Paolo Milazzo, and Giovanni Pardini. Application of a semi-automatic algorithm for identification of molecular components in SBML models. In *Proceedings of Wivace 2013, Italian Workshop on Artificial Life and Evolutionary Computation, (Milan, Italy, July 1-2, 2013)*, Electronic Proceedings in Theoretical Computer Science, 2013. To appear.

[31] Pedro T. Monteiro, Delphine Ropers, Radu Mateescu, Ana T. Freitas, and Hidde de Jong. Temporal logic patterns for querying dynamic models of cellular interaction networks. *Bioinformatics*, 24(16):i227–i233, 2008. doi:10.1093/bioinformatics/btn275.

[32] Aurélien Naldi, Elisabeth Remy, Denis Thieffry, and Claudine Chaouiya. Dynamically consistent reduction of logical regulatory graphs. *Theoretical Computer Science*, 412(21):2207–2218, 2011. doi:10.1016/j.tcs.2010.10.021.

[33] Giovanni Pardini, Paolo Milazzo, and Andrea Maggiolo-Schettini. An algorithm for the identification of components in biochemical pathways. In *Proceedings of CS2Bio 2013, 4th International Workshop on Interactions between Computer Science and Biology, (Florence, Italy, June 6, 2013)*, Electronic Notes in Theoretical Computer Science, 2013. To appear.

[34] Amir Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13(1):45–60, 1981. doi:10.1016/0304-3975(81)90110-9.

[35] Corrado Priami, Aviv Regev, Ehud Shapiro, and William Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80(1):25–31, 2001. doi:10.1016/S0020-0190(01)00214-9.

[36] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Shapiro. Bioambients: an abstraction for biological com-

partments. *Theoretical Computer Science*, 325(1):141–167, 2004. doi:10.1016/j.tcs.2004.03.061.

[37] Birgit Schoeberl, Claudia Eichler-Jonsson, Ernst Dieter Gilles, and Gertraud Muller. Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors. *Nature Biotechnology*, 20(4):370–375, 2002. doi:10.1038/nbt0402-370.

© Scientific Annals of Computer Science 2013