Local Traces: An Over-Approximation of the Behavior of the Proteins in Rule-Based Models

Jérôme Feret, Kim Quyên Lý

Local traces: an over-approximation of the behaviour of the proteins in rule-based models

Jérôme Feret and Kim Quyên Lý

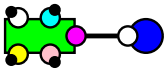
***Abstract*—Thanks to rule-based modelling languages, we can assemble large sets of mechanistic protein-protein interactions within integrated models. Our goal would be to understand how the behaviour of these systems emerges from these low-level interactions. Yet this is a quite long term challenge and it is desirable to offer intermediary levels of abstraction, so as to get a better understanding of the models and to increase our confidence within our mechanistic assumptions. To this extend, static analysis can be used to derive various abstractions of the semantics, each of them offering new perspectives on the models.**

We propose an abstract interpretation of the behaviour of each protein, in isolation. Given a model written in Kappa, this abstraction computes for each kind of proteins a transition system that describes which conformations this protein may take and how a protein may pass from one conformation to another one. Then, we use simplicial complexes to abstract away the interleaving order of the transformations between conformations that commute. As a result, we get a compact summary of the potential behaviour of each protein of the model.

I. Introduction

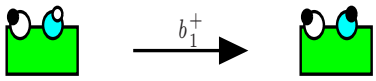
Systems biology aims at understanding how the collective behaviour of systems emerges from the interactions between individual proteins. At the individual level, proteins assemble with each other and activate one another, hence modifying dynamically their capabilities of interaction. At the population level, they receive, propagate, and integrate signals, which controls apoptosis (cell death), mitosis (cell duplication), specialisation and migration of the cell. When these systems fail, the cell life cycle is damaged, which may ultimately cause cancer. Describing and understanding these systems are very difficult due to their underlying complexity, that is mainly due to combinatorics, concurrency, wide time- and concentration-scale separation, and non-linear feedback loops.

Thanks to rule-based modelling languages, such as Kappa, one can model accurately the biochemical interactions between proteins involved for instance in signalling pathways, without abstracting away *a priori*, when they are available, the mechanistic details about these interactions. In Kappa, each compound is described as a site-graph. In a site graph, nodes are typed and each type of node is associated with a list of distinct interaction sites. Sites may be free or pair-wisely bound. Additionally some sites may carry a property, that may encode a level of activation. For instance, the following site-graph:

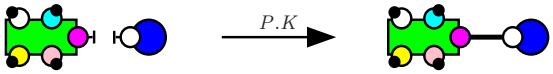


is made of two typed nodes (the type is denoted by the shape of the nodes). The node on the left has five distinct interaction sites. Four of them are phosphorylated, which is denoted by the black bullets. One site is bound to the unique interaction sites of the second node. The notions of node, site and property are abstract. Usually, each node type denotes a kind of proteins and each site denotes a particular address in the nucleotide chain. Interactions between proteins may fold and unfold them, which may hide or reveal interaction sites. In Kappa, the 3D structure of proteins is described only by site properties implicitly.

The evolution of the system is described by context-free rewrite rules. Each rule is defined as a pair of patterns. The left one denotes the condition to be satisfied to apply a rule, whereas the difference between the right one and the left one denotes the effect of applying a rule. In these patterns, only the sites that matter for the interaction are specified, which allows for a very compact description of these interactions. For instance, in the following rule:



the upper right site of a protein may be phosphorylated, provided that the upper left site of this protein is already phosphorylated. The states of the other sites of the protein are not specified because they do not change this interaction, nor the kinetic of this interaction. In the following other rule:



two proteins may bind to each other, provided that the first protein has its four sites phosphorylated. Each rule may be understood intensionally as a local transformation of a site graph, denoting the state of the system, or extensionally as a (finite or not) set of reactions, obtained by refining each rule by enumerating the potential context of applications, until getting fully specified connected components.

For example, one can describe faithfully the formation of dimers, scaffold proteins, and the phosphorylation of proteins on multiple sites in a very compact way. Yet understanding how the behaviour of the systems may emerge from these interactions remains a challenge. Moreover, when models become large, no matter they have been written by hand

This material is based upon works sponsored by the Defense Advanced Research Projects Agency (DARPA) and the U. S. Army Research Office under grant number W911NF-14-1-0367. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency, or the U. S. Department of Defense.

or automatically assembled from the literature, as suggested in [22], some automatic tools capable to understanding the content of the models and to checking that what is modelled matches with what the modeller has in mind, become crucial.

We use the abstract interpretation framework [5], [6] to derive automatic static analyses for Kappa models. Abstract interpretation has been introduced forty years ago as a mathematical framework to relate the behaviour of programs or models formally, at different levels of abstraction. Since then, abstract interpretation has been used not only to establish formal comparisons between abstraction techniques [4], but also to develop static analysers that abstract automatically the behaviour of programs or models [1], [11]. Abstract interpreters are now spreading across the industrial community (for instance, Microsoft, Google, Facebook, and The Mathworks have been developing their own abstract interpreters).

The abstract interpretation framework is based on the idea that the behaviour of a program or a model can be described formally as the least fix-point *lfp* $\mathbb{F}$ of a monotonic operator $\mathbb{F}$ over the elements of a so-called concrete domain $\mathcal{D}$. The concrete domain is usually the set $\wp(S)$ of the subsets of a given set S of elements. Then, an abstraction is a change of granularity in the description of the behaviour of the programs and the models, that can be formalised mathematically by various means, such as upper closure operators, ideals, Moore families, and Galois connections. Galois connections has become quickly the most popular way to describe an abstract interpretation. A change of observation level can be described by introducing a domain $\mathcal{D}^\sharp$ of properties of interest, that is ordered by a partial order $\sqsubseteq$. Each element $a^\sharp$ of the abstract domain $\mathcal{D}^\sharp$ is then related to the set of the concrete elements $\gamma(a^\sharp)$ that satisfy this abstract property, the so-defined function γ being monotonic. An abstract element $a^\sharp$ is called an abstraction of a given set a of concrete elements, if and only if, $a \subseteq \gamma(a^\sharp)$. A Galois connection is obtained when each subset $a \subseteq S$ has a most precise abstraction, i. e. for any subset $a \subseteq S$, there exists an abstract element $\alpha(a)$ such that, for any other abstraction $a^\sharp$ of the element a, we have $\alpha(a) \sqsubseteq a^\sharp$. In this case, any monotonic function $\mathbb{F}^\sharp$ over the abstract domain $\mathcal{D}^\sharp$ such that $[\alpha \circ \mathbb{F} \circ \gamma](a^\sharp) \sqsubseteq \mathbb{F}^\sharp(a^\sharp)$ for each element $a^\sharp \in \mathcal{D}^\sharp$, satisfies *lfp* $\mathbb{F} \subseteq \gamma($*lfp* $\mathbb{F}^\sharp)$; i. e. the behaviour of the program or the model can be computed in the abstract domain, and the result is a sound over-approximation. By soundness, we mean that we miss no concrete behaviour. Yet our result may be approximate when the inclusion is strict, meaning that, because of the over-approximation, our abstraction has introduced spurious behaviours. Besides, any monotonic function $\mathbb{F}^\sharp$ such that $[\alpha \circ \mathbb{F}](a) = [\mathbb{F}^\sharp \circ \alpha](a)$ for any set $a \subseteq S$ of concrete elements, satisfies $\alpha($*lfp* $\mathbb{F}) =$ *lfp* $\mathbb{F}^\sharp$. In such a case, the over-approximation provides as much information as possible to express at the abstract level and the approximation is exact. For instance, sign analysis consists in abstracting sets of non-zero natural numbers by the set of their signs. This abstraction is exact with respect to multiplication, since if we know the signs of two non-zero natural numbers, we know the sign of their product. But this abstraction is not exact with respect to addition, since we do not know the sign of a sum between a positive and a negative numbers.

Each static analysis offers new perspectives on models. Applications range from model debugging to the abstraction of complex properties offering new insights to investigate the system overall behaviour. In this paper, we propose to study the behaviour of each protein in isolation. Starting from a formal definition of the trace semantics, we collect the behaviour of each kind of proteins independently, and summarise the potential steps to reach these conformations within a transition system. When proteins have too many interaction sites, it is crucial to take benefit of the potential independence between some conformation changes in some protein states. Taking inspiration from simplicial complexes [12], we introduce the notion of macrotransition systems, in which the behaviour of different subsets of sites can be described independently, abstracting away the potential interleaving between their behaviour. The result is a scalable and convenient way to visualise both the different conformations that each protein may take and the causal relations among the different conformation changes.

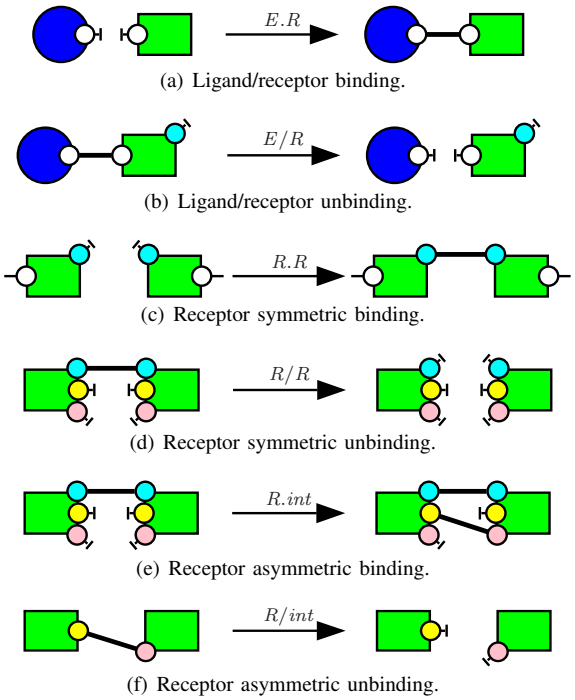
An open source implentation is available [2].

a) Related works.: The qualitative analysis that is proposed in [13], [8], captures all the conformations that an agent may take in a Kappa model. Here we go further and compute, for each agent, a transition system describing the causal relationships among its potential conformational changes.

Causality plays an important role in the understanding and the verification of concurrent systems, as found in Systems Biology. Several frameworks are available to study and understand causality, and to reduce the combinatorial complexity of the models, by exploiting pair of commutative transitions. Partial order reduction is broadly used in model checking [15]. It consists in restricting the transitions of a concurrent system so as to force its computation to follow a canonical order for the interleaving of commutative transitions. Event structures [21] focus on the causal relations between events in a concurrent system. In [7], they provide a compact description of trace samples, in which the events which are not necessary, are discarded. It is worth noticing that these discarded events may have a kinetic impact. Application of event structures in static analysis can be found in [3]. Since they focus on accumulating the effect of causally related transformations, event structures somehow obfuscate the notion of states. Various notions of causal structures can be derived by abstracting the set of traces at different levels of granularity, as shown in [10]

Our notion of macrotransition systems is inspired by simplicial complexes. Simplicial complexes can be used for describing concurrent systems up to the interleaving order of commutative transitions [12]. They describe the state of the system as a point that moves along a geometrical object, in which commutative transitions are denoted by higher dimension faces. Our formalism offers a convenient compact abstraction of all the potential conformation changes of a protein, without discarding any transition.

This article is an extended version of [14]. In particular, it contains a detailed introduction to Kappa and to Abstract Interpretation (e.g. see Sect. I). We have included the definition for the composition between embeddings and spans of embeddings (e.g. see Figs. 7 and 8) to complete the description of



(a) Ligand/receptor binding.

(b) Ligand/receptor unbinding.

(c) Receptor symmetric binding.

(d) Receptor symmetric unbinding.

(e) Receptor asymmetric binding.

(f) Receptor asymmetric unbinding.

Fig. 1. Rules for the formation of dimers.

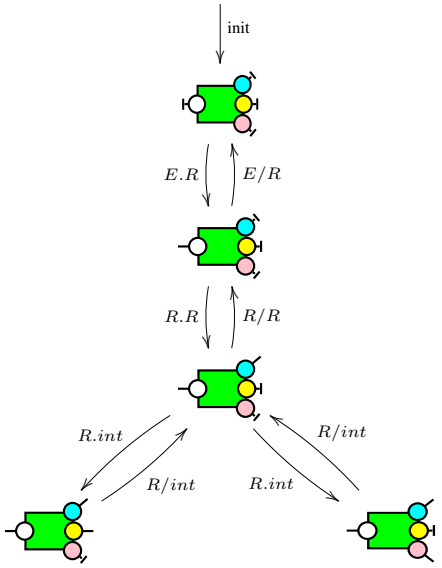


Fig. 2. The local transition system of membrane receptors.

the operational semantics of Kappa. We have added examples of traces, in the initial transition systems (e.g. see Fig. 11), in the local transition systems (e.g. see Fig. 14) and the macrotransition systems (e.g. see Fig. 17). Lastly, we have provided a benchmarks section (e.g. see Sect. VII) and detailed the detection of dead rules and potential non-weakly reversible transitions.

b) Outline. : In Sect. II, we introduce two case studies to motivate our framework. In Sect. III, we describe Kappa. In Sect. IV, we define its finite trace semantics, that we abstract in Sect. V by over-approximating the behaviour of each kind of agents thanks to *local* transition systems. In Sect. VI, we abstract away the interleaving order of the transitions that commute in these local transition systems. In Sect. VII, we discuss about the implementation and we experiment our tools on large scale models.

II. Case studies

So as to motivate our goal, we introduce two models.

The first model describes the formation of some dimers. Two kinds of proteins are involved: ligands and membrane receptors. When activated by ligands, receptors may form stable dimers, as shown in Fig. 1. We are interested in one particular binding site in ligand proteins, and in four binding sites in receptor proteins. Ligand proteins are depicted as circles, whereas receptor proteins are depicted as rectangles. Their binding sites are drawn as smaller circles. Some sites are connected pair-wisely. For the others, we use the symbol '$\dashv$' to specify a free site and the symbol '$-$' to specify a site that is bound to an unspecified site. By convention, the site alone on its side in a receptor protein is the one that may bind to a ligand protein; the three sites on the other side may bind to the other receptors (their order matters).

Let us now give more details about the interactions between these proteins. A ligand protein and a receptor protein may bind to each other provided that the sites that are dedicated to this binding are both free (e. g. see Fig. 1(a)), or detach from each other, provided that the receptor protein is not yet involved in a dimer (e. g. see Fig. 1(b)). Two activated receptor proteins may form a symmetric bond by connecting their respective top-most site (e. g. see Fig. 1(c)), or break this bond unless an asymmetric bond has been formed already (e. g. see Fig. 1(d)). To gain stability, a dimer with a symmetric link may form an asymmetric one by connecting one of its free site in the first receptor protein to the free site of the other kind in the second receptor protein (e. g. see Fig. 1(e)), or break this connection (e. g. see Fig. 1(f)).

Writing interaction rules may be error prone. Especially, which amount of information should be put in rules, is often not so clear. So as to gain confidence in our modelling process, we propose to compute, for each kind of proteins, a *local* transition system. The goal is to abstract the different conformations that each protein may take, and how a given protein may pass from one conformation to another. As an example, the local transition system for receptor proteins is given in Fig. 2. We claim that it provides a helpful summary of the effect of rules on the behaviour of each protein instance.

When proteins have too many interaction sites, we can no longer describe extensively their sets of potential conformations. Our second model deals with a protein with four phosphorylation sites and one binding site. The upper left (resp. lower left) site can be phosphorylated without any condition (e. g. see Figs. 3(a) and 3(e)). Then, the upper right (resp. lower right) site can get phosphorylated, if the upper left (resp. lower left) site is still phosphorylated (e. g. see Fig. 3(c) and 3(g)). When all sites are phosphorylated, the conformation of the protein changes which reveals the binding site. The protein may then bind to another kind of proteins (e. g. see Fig. 3(i)) or release this bond with no condition (e. g. see

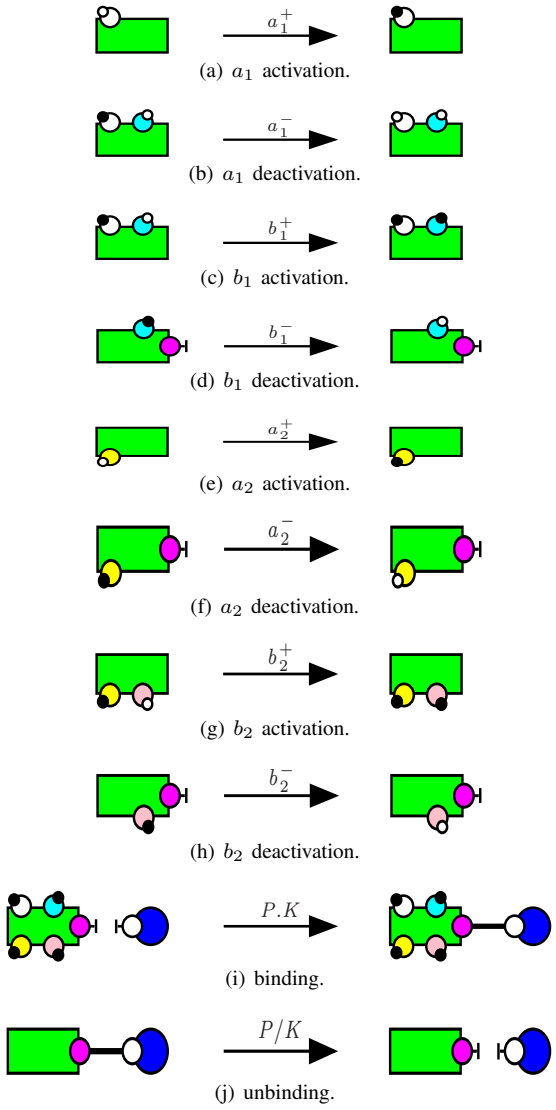


Fig. 3. Rules for the protein with four phosphorylation sites.

Fig. 3(j)). Phosphorylated sites can be dephosphorylated under the following conditions: as long as a protein is bound, none of its site can be dephosphorylated; and as long as the upper right site is phosphorylated, the upper left site cannot be dephosphorylated (e. g. see Figs. 3(b), 3(d), 3(f), and 3(h)).

We give in Fig. 4, the local transition system associated to this protein. We notice that this transition system is difficult to read because the protein may take too many conformations. This case study is taken from a more complex example in which a protein has ten sites that can be phosphorylated. In such a case, one cannot extract information from its transition system. Looking closer at the local transition system in Fig. 4, we can notice that in each protein instance the potential transformations of the states of both sites on the top *commute* with the potential transformations of those of both sites bellow. Thanks to this, we can describe the transition system between the different conformations of the protein in a more compact way (e. g. see Fig. 5). In this latter transition system, the behaviour of the pair of sites on the top and of the pair of sites on the bottom is described as two independent subprocesses.

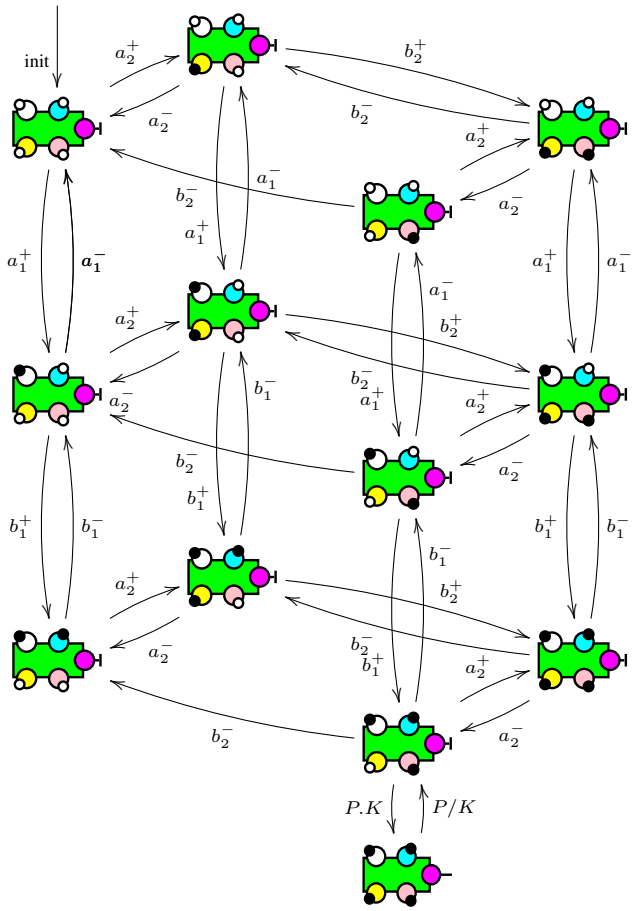


Fig. 4. Local transition system for the protein with four phosphorylation sites.

This description is inspired by simplicial complexes [12]. It describes independent processes modulo the interleaving order of their execution.

III. Kappa

In this section, we introduce formally the syntax and the operational semantics of Kappa. We focus on the single push-out (SPO) semantics.

A. Signature

Firstly we define the signature of a model.

Definition 1 (signature): The signature of a model is a tuple $\Sigma = (\Sigma_{ag}, \Sigma_{site}, \Sigma_{int}, \Sigma^{int}_{ag\text{-}st}, \Sigma^{lnk}_{ag\text{-}st})$ *where:*

1) Σ_{ag} *is a finite set of agent types,*
2) Σ_{site} *is a finite set of site identifiers,*
3) Σ_{int} *is a finite set of internal state identifiers,*
4) $\Sigma^{lnk}_{ag\text{-}st} : \Sigma_{ag} \to \wp(\Sigma_{site})$ *and* $\Sigma^{int}_{ag\text{-}st} : \Sigma_{ag} \to \wp(\Sigma_{site})$ *are two site maps.*

Agent types in Σ_{ag} denote agents of interest, as kinds of proteins for instance. A site identifier in Σ_{site} represents an identified locus for a capability of interactions. Each agent type $A \in \Sigma_{ag}$ is associated with a set of sites which may bear an internal state $\Sigma^{int}_{ag\text{-}st}(A)$ and a set of sites which may be

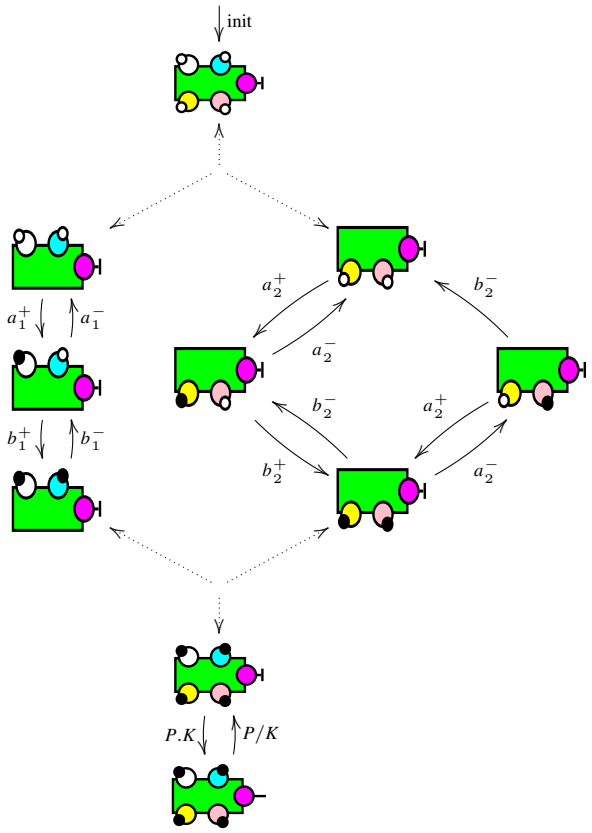


Fig. 5. Local transition system for the protein with four phosphorylation sites.

linked $\Sigma^{lnk}_{ag\text{-}st}(A)$. We assume without any loss of generality that $\Sigma^{lnk}_{ag\text{-}st}(A) \cap \Sigma^{int}_{ag\text{-}st}(A) = \emptyset$, for any $A \in \Sigma_{ag}$ and we write $\Sigma_{ag\text{-}st}(A)$ for the set of sites $\Sigma^{lnk}_{ag\text{-}st}(A) \uplus \Sigma^{int}_{ag\text{-}st}(A)$.

Example 1 (running example): We define the signature for the model in the second case study as the tuple $(\Sigma_{ag}, \Sigma_{site}, \Sigma_{int}, \Sigma^{int}_{ag\text{-}st}, \Sigma^{lnk}_{ag\text{-}st})$ *where:*

1) $\Sigma_{ag} := \{P, K\}$;
2) $\Sigma_{site} := \{a_1, a_2, b_1, b_2, x\}$;
3) $\Sigma_{int} := \{\circ, \bullet\}$;
4) $\Sigma^{int}_{ag\text{-}st} := [P \mapsto \{a_1, a_2, b_1, b_2\}, K \mapsto \emptyset]$;
5) $\Sigma^{lnk}_{ag\text{-}st} := [P \mapsto \{x\}, K \mapsto \{x\}]$.

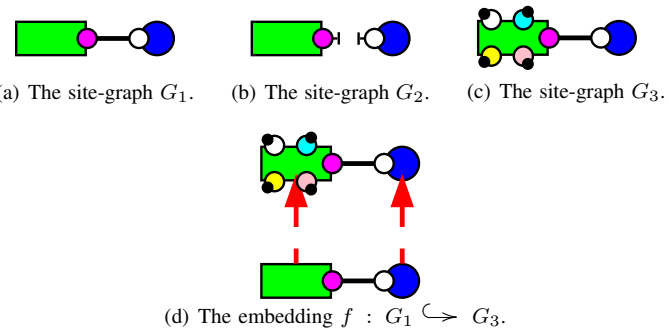
The agent type P denotes the first kind of proteins and the agent type K denotes the second one; the site identifier x denotes the binding site (both in P and K), and the site identifiers a_1, a_2, b_1, b_2 denote the top left, bottom left, top right, and lower right sites in the protein P respectively. □

B. Site-graphs

Site-graphs describe both patterns and chemical mixtures. Their nodes are typed agents with some sites which may bear internal states and binding states.

Definition 2 (site-graph): A site-graph is defined as a tuple $G = (\mathcal{A}, type, \mathcal{S}, \mathcal{L}, p\kappa)$ *where:*

1) $\mathcal{A} \subseteq \mathbb{N}$ *is a finite set of agents,*
2) $type : \mathcal{A} \to \Sigma_{ag}$ *is a function mapping each agent to its type,*



(a) The site-graph G_1. (b) The site-graph G_2. (c) The site-graph G_3.

(d) The embedding $f : G_1 \hookrightarrow G_3$.

Fig. 6. Three site-graphs G_1, G_2, and G_3, and an embedding f from the site-graph G_1 into the site-graph G_3.

3) $\mathcal{S}$ *is a set of sites satisfying the following property:*

$$\mathcal{S} \subseteq \{(n,i) \mid n \in \mathcal{A}, i \in \Sigma_{ag\text{-}st}(type(n))\},$$

4) $\mathcal{L}$ *maps the set* $\{(n,i) \in \mathcal{S} \mid i \in \Sigma^{lnk}_{ag\text{-}st}(type(n))\}$ *to the set* $\{(n,i) \in \mathcal{S} \mid i \in \Sigma^{lnk}_{ag\text{-}st}(type(n))\} \cup \{\dashv, -\}$, *such that for any two sites* $(n,i), (n',i') \in \mathcal{S}$, *we have* $(n',i') = \mathcal{L}(n,i)$ *if and only if* $(n,i) = \mathcal{L}(n',i')$;
5) *and* $p\kappa$ *maps the set* $\{(n,i) \in \mathcal{S} \mid i \in \Sigma^{int}_{ag\text{-}st}(type(n))\}$ *to the set* Σ_{int}.

A site $(n,i) \in \mathcal{S}$ such that $i \in \Sigma^{int}_{ag\text{-}st}(type(n))$ is called a property site, whereas a site $(n,i) \in \mathcal{S}$ such that $i \in \Sigma^{lnk}_{ag\text{-}st}(type(n))$ is called a binding site. Whenever $\mathcal{L}(n,i) = \dashv$, the binding site (n,i) is free. Various levels of information can be given about the sites that are bound. Whenever $\mathcal{L}(n,i) = -$, the binding site (n,i) is bound to an unspecified site. Whenever $\mathcal{L}(n,i) = (n',i')$ (and hence $\mathcal{L}(n',i') = (n,i)$), the sites (n,i) and (n',i') are bound together.

For a site-graph G, we write as $\mathcal{A}_G$ its set of agents, $type_G$ its typing function, $\mathcal{S}_G$ its set of sites, and $\mathcal{L}_G$ its set of links.

A mixture is a site-graph in which the state of each site in each agent is documented. Formally, a site-graph G is a chemical mixture, if and only if, the set $\mathcal{S}_G$ is equal to the set $\{(n,i) \mid n \in \mathcal{A}_G,\ i \in \Sigma_{ag\text{-}st}(type_G(n))\}$.

Example 2 (running example): We consider three site-graphs G_1, G_2, and G_3 as follows:

1) a) $\mathcal{A}_{G_1} = \{1, 2\}$,
   b) $type_{G_1} = [1 \mapsto P,\ 2 \mapsto K]$,
   c) $\mathcal{S}_{G_1} = \{(1,x), (2,x)\}$,
   d) $\mathcal{L}_{G_1} = [(1,x) \mapsto (2,x), (2,x) \mapsto (1,x)]$,
   e) $p\kappa_{G_1} = []$;
2) a) $\mathcal{A}_{G_2} = \{1, 2\}$,
   b) $type_{G_2} = type_{G_1}$,
   c) $\mathcal{S}_{G_2} = \mathcal{S}_{G_1}$,
   d) $\mathcal{L}_{G_2} = [(1,x) \mapsto \dashv, (2,x) \mapsto \dashv]$,
   e) $p\kappa_{G_2} = []$;
3) a) $\mathcal{A}_{G_3} = \{1, 2\}$,
   b) $type_{G_3} = type_{G_1}$,
   c) $\mathcal{S}_{G_3} = \{(1,x), (1,a_1), (1,a_2), (1,b_1), (1,b_2), (2,x)\}$,
   d) $\mathcal{L}_{G_3} = [(1,x) \mapsto (2,x), (2,x) \mapsto (1,x)]$,
   e) $p\kappa_{G_3} = [a_1 \mapsto \bullet, a_2 \mapsto \bullet, b_1 \mapsto \bullet, b_2 \mapsto \bullet,]$.

The three site-graphs G_1, G_2, and G_3 are graphically described in Figs. 6(a), 6(b), and 6(c). Among these three site-graphs, we notice that only G_3 is a chemical mixture. □

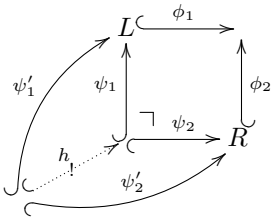
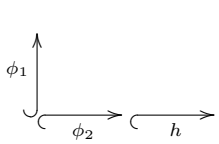
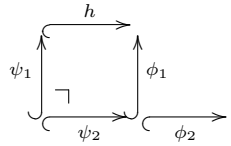


Fig. 7. Pull-back of a cospan $L \overset{\phi_1}{\hookrightarrow} \overset{\phi_2}{\hookleftarrow} R$.



(a) Composition between a span of embeddings $\overset{\phi_1}{\hookleftarrow} \overset{\phi_2}{\hookrightarrow}$ and an embedding h.



(b) Composition between an embedding h and a span of embeddings $\overset{\phi_1}{\hookleftarrow} \overset{\phi_2}{\hookrightarrow}$.

Fig. 8. Composition of embeddings and spans (and conversely).

C. Embeddings

Two site-graphs may be related by structure-preserving injective functions, which are called embeddings. The notion of embedding is defined as follows:

Definition 3 (embedding): An embedding $h : G \hookrightarrow H$ *from the site-graph G into the site-graph H is a function of agents $h : \mathcal{A}_G \to \mathcal{A}_H$ satisfying, for all agent identifiers m, $n, n' \in \mathcal{A}_G$, for all site identifiers $i \in \Sigma_{ag\text{-}st}(type_G(n))$, $i' \in \Sigma_{ag\text{-}st}(type_G(n'))$, and for all internal state identifier $\iota \in \Sigma_{int}$:*

1) *if $m \neq n$, then $h(m) \neq h(n)$;*
2) *$type_G(n) = type_H(h(n))$;*
3) *if $(n, i) \in \mathcal{S}_G$, then $(h(n), i) \in \mathcal{S}_H$;*
4) *if $\mathcal{L}(n, i) = (n', i')$, then $\mathcal{L}(h(n), i) = (h(n'), i')$;*
5) *if $\mathcal{L}(n, i) = \dashv$, then $\mathcal{L}(h(n), i) = \dashv$;*
6) *if $\mathcal{L}(n, i) = -$, then $\mathcal{L}(h(n), i) \in \{-\} \cup \mathcal{S}_H$;*
7) *if $p\kappa(n, i) = \iota$, then $p\kappa(h(n), i) = \iota$.*

Example 3 (running example): An embedding from the site-graph G_1 into the site-graph G_3 is shown in Fig. 6(d). This embedding is defined by mapping the agent 1 *of the site-graph G_1 to the agent* 1 *of the site-graph G_3 and the agent* 2 *of the site-graph G_1 to the agent* 2 *of the site-graph G_3.* □

Given three site-graphs E, F, and G, two embeddings from the site-graph E into the site-graph F, and from the site-graph F into the site-graph G respectively, compose in the usual way (and form an embedding from the site-graph E into the site-graph G). Moreover, the site-graphs E and F are said isomorphic, if and only if, there exist an embedding from the site-graph E into the site-graph F and an embedding from the site-graph F into the site-graph E. An embedding between two isomorphic site-graphs, is called an isomorphism. Given three site-graphs L, R, and D, a couple of embeddings respectively from the site-graph D into the site-graph L, and from the site-graph D and into the site-graph R, is called a span of embeddings between the site-graphs L and R. Besides, a couple of embeddings respectively from the site-graph L into the site-graph D, and from the site-graph R into the site-graph D is called a cospan of embeddings between the site-graphs L and R.

Any cospan (ϕ_1, ϕ_2) of embeddings between two site-graphs L and R is associated with a span (ψ_1, ψ_2) of embeddings between the site-graphs L and R, that is defined up to isomorphism by both following properties (e. g. see Fig. 7):

1) $\phi_1\psi_1 = \phi_2\psi_2$, and
2) for any span (ψ_1', ψ_2') of embeddings such that $\phi_1\psi_1' = \phi_2\psi_2'$, there exists a unique embedding h such that:
   a) $\psi_1' = \psi_1 h$
   b) and $\psi_2' = \psi_2 h$.

The span (ψ_1, ψ_2) is called the pull-back of the cospan (ϕ_1, ϕ_2). Intuitively, a cospan between the site-graphs L and R formalises a matching relation between the agents of the site-graph L and the agents of the site-graph R. The cospan is a proof that the site-graphs L and R can be glued together by identifying the agents which are matched this way. Then, the pull-back encodes the biggest common site-graphs that may complete this initial gluing.

As shown in Fig. 8(a), a span (ϕ_1, ϕ_2) and an embedding h compose, provided that the embeddings ϕ_2 and h compose as well. In such a case, the composition is defined as the span $(\phi_1, h\phi_2)$. Moreover, as shown in Fig. 8(b) an embedding h and a span (ϕ_1, ϕ_2) compose, provided that the embeddings h and ϕ_1 have the same target. In such a case, the composition is defined up to isomorphism as the span of embeddings $(\psi_1, \phi_2\psi_2)$ where the pair of embeddings (ψ_1, ψ_2) is a pull-back of the cospan (h, ϕ_1).

D. Rules

Transformations between site-graphs are described by rules (e. g. see Figs. 1 and 3). For the sake of simplicity, we only define a fragment of Kappa. We assume that rule can break and create bonds between sites, and can change the internal states of sites, but we consider neither agent degradation, nor agent creation.

These requirements are formalised as follows:

Definition 4 (rule): A rule *is a span of embeddings $L \overset{h_L}{\hookleftarrow} D \overset{h_R}{\hookrightarrow} R$ such that:*

1) *$\mathcal{A}_D = \mathcal{A}_L$ and $\mathcal{A}_D = \mathcal{A}_R$;*
2) *for all agents $n \in \mathcal{A}_D$, $h_L(n) = n$ and $h_R(n) = n$;*
3) *$\mathcal{S}_D = \mathcal{S}_L$ and $\mathcal{S}_D = \mathcal{S}_R$;*
4) *for all sites $(n, i) \in \mathcal{S}_D$,*
   if $\mathcal{L}_R(n, i) = -$, then $\mathcal{L}_L(n, i) = -$.

Since we do consider neither agent creation, nor agent degradation, we can assume that the agents on the left hand side and on the right hand side of a rule are the same (constraint 1) and that both embeddings preserve agent identifiers (constraint 2). In a rule, sites cannot be removed or added (constraint 3). When the binding state of a site is modified, the new state has to be fully specified (constraint 4).

A rule $L \hookleftarrow D \hookrightarrow R$ is usually denoted as $L \rightarrowtriangle R$ (leaving the two embeddings and the common region implicit).

Rules can be applied to site-graphs via an embedding thanks to a push-out construction.

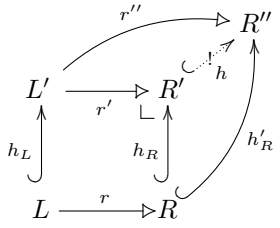


Fig. 9. Push-out of a span $L' \xleftarrow{h_L} L \xrightarrow{r} R$ made of an embedding h_L and a rule r.

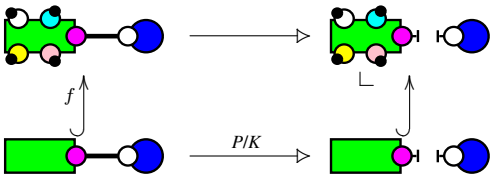


Fig. 10. Applying the rule $P/K : G_1 \rightarrowtriangle G_2$ along the embedding f.

Definition 5 (rule refinement [7]): Let r be a rule $L \rightarrowtriangle R$, L' be a site-graph, and h_L be an embedding from the site-graph L into the site-graph L'. Then, there exists a rule r' between the site-graph L' and a site-graph R' and an embedding h_R from the site-graph R into the site-graph R' such that both following properties are satisfied:

1) $h_R r = r' h_L$;
2) *for all rule r'' between the site-graph L' and a site-graph R'' and all embedding h'_R from R into R'' such that: $h'_R r = r'' h_L$, there exists a unique embedding h from R' into R'' such that $r'' = hr'$ and $h'_R = hh_R$.*

With these notations, there is a transition from the state L' into the state R' via a computation step with the label (r, h_L), and we write $L' \xrightarrow{(r,h_L)} R'$. Moreover, with the same notations, whenever the site-graph L' is a chemical mixture, the site-graph R' is a chemical mixture as well.

In Def. 5, the compositions between rules and embeddings are obtained as in Fig. 8 by interpreting each rule as a span of embeddings.

Example 4 (running example): The embedding f from the left hand side of the rule P/K into the site-graph G_3 induces a computation step as described in Fig. 10. □

In Kappa, a model $\mathcal{M}$ (over a given signature Σ) is defined as a pair $(G_0, \mathcal{R})$ where G_0 is a chemical mixture and $\mathcal{R}$ is a set of rules. The chemical mixture G_0 denotes the initial state. Since we focus only on qualitative properties, we do not associate kinetic rates with rules.

IV. Trace semantics

In this section, we define the semantics of a model (written in Kappa) as the set of the traces that is induced by the underlying transition system.

A. Transition systems

We fix $\mathbb{Q}$ a set of *states* and $\mathbb{L}$ a set of *labels*.

A transition is defined as follows:

Definition 6 (transitions): A transition *is a triple (q, λ, q') in the set $\mathbb{Q} \times \mathbb{L} \times \mathbb{Q}$.*

In Kappa, states are chemical mixtures which can be written according to the signature Σ, whereas transition labels are pairs composed of a rule and an embedding from the left hand side of this rule into a chemical mixture.

A transition system is given by a set of initial states and a set of transitions, as formalised in the following definition:

Definition 7 (transition system): A transition system *is a pair $(\mathcal{Q}_0, T)$ satisfying both following conditions:*

1) $\mathcal{Q}_0 \subseteq \mathbb{Q}$;
2) $T \subseteq \mathbb{Q} \times \mathbb{L} \times \mathbb{Q}$.

We denote by $\mathbb{T}_{\mathbb{Q},\mathbb{L}}$ the set of all the transition systems over the set of states $\mathbb{Q}$ and the set of transition labels $\mathbb{L}$.

Transition systems can be ordered by the relation $\sqsubseteq$ that is defined as $(Q_0, T) \sqsubseteq (Q'_0, T')$ if and only if $Q_0 \subseteq Q'_0$ and $T \subseteq T'$. The pair $(\mathbb{T}_{\mathbb{Q},\mathbb{L}}, \sqsubseteq)$ forms a complete lattice, i. e. every family $(T_i)_{i \in I}$ of transition systems has a least upper bound, that we denote by $\sqcup\{T_i \mid i \in I\}$.

By definition, both following properties are satisfied:

1) for each element $i \in I$, $T_i \sqsubseteq \sqcup\{T_i \mid i \in I\}$;
2) for each transition system $Y \in \mathbb{T}_{\mathbb{Q},\mathbb{L}}$ such that $T_i \sqsubseteq Y$ for each element $i \in I$, we have $\sqcup\{T_i \mid i \in I\} \subseteq Y$.

Example 5 (Kappa): Each model $\mathcal{M} := (G_0, \mathcal{R})$ in Kappa is associated with the transition system $(\mathcal{Q}_0, T)$ that is defined as follows:

1) $\mathcal{Q}_0 = \{G_0\}$;
2) *T is the set of the transitions $(L', (r, h_L), R')$ such that $L' \xrightarrow{(r,h_L)} R'$ as defined in Def. 5.*

□

B. Traces

Each transition system induces a set of traces. We focus on finite traces which are made of an initial state followed by a (potentially empty) finite sequence of transitions. In a trace, each transition starts from the state the previous transition had ended in. This is formalised in the following definition:

Definition 8 (finite traces): A finite trace *is defined as a pair $\tau = (q'_0, (q_i, \lambda_i, q'_i)_{1 \leq i \leq p})$, where:*

1) *q'_0 is a state (in $\mathbb{Q}$),*
2) *$(q_i, \lambda_i, q'_i)_{1 \leq i \leq p}$ is a family of transitions (in $\mathbb{Q} \times \mathbb{L} \times \mathbb{Q}$) such that $q'_{i-1} = q_i$ for each integer i between 1 and p.*

We denote as $\mathcal{T}^+$ the set of all the traces.

With the notations in Def. 8, we call the state q'_0 (resp. q'_p) the initial (resp. the final) state of the trace τ and we denote it as $\mathrm{fst}(\tau)$, (resp. as $\mathrm{last}(\tau)$). When a trace is made of a single state, we write it as q instead of $(q, ())$. Besides, any trace $\tau' := (q'', (q_{p+i}, \lambda_{p+i}, q'_{p+i})_{1 \leq i \leq p'})$ such that $\mathrm{last}(\tau) = \mathrm{fst}(\tau')$ (i. e. $q'_p = q''$), can be concatenated to the trace τ: we write $\tau \,.\, \tau'$ for the finite trace $(q'_0, (q_i, \lambda_i, q'_i)_{1 \leq i \leq p+p'})$. Also, any transition $t := (q_{p+1}, \lambda_{p+1}, q'_{p+1})$ such that the state q_{p+1} is equal to the final state q'_p of the trace τ, can be inserted at the end of the trace τ: we write $\tau \frown (q_{p+1}, \lambda_{p+1}, q'_{p+1})$ for the finite trace $(q'_0, (q_i, \lambda_i, q'_i)_{1 \leq i \leq p+1})$.

Example 6 (running example): In Fig 11, we give an example of a trace for the second case study. □

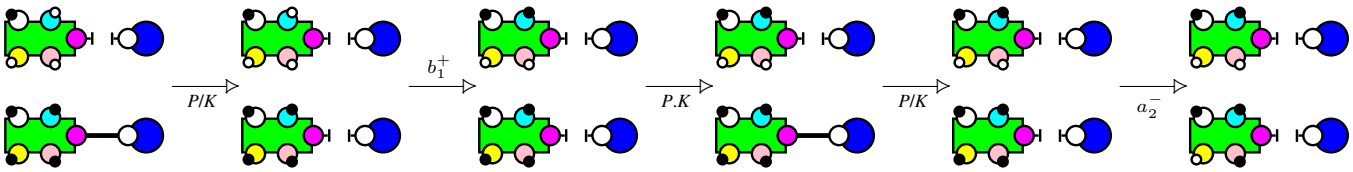


Fig. 11. A trace.

A transition system $(\mathcal{Q}_0, T)$ induces a set of traces $\gamma_{\mathbb{Q},\mathbb{L}}(\mathcal{Q}_0, T)$, that is defined as the set of the traces $(q'_0, (q_i, \lambda_i, q'_i)_{1\leq i\leq p})$ such that:

1) the state q'_0 belongs to the set $\mathcal{Q}_0$;
2) and for each integer i between 1 and p, the transition (q_i, λ_i, q'_i) belongs to the set T.

Indeed, the set of traces $\gamma_{\mathbb{Q},\mathbb{L}}(\mathcal{Q}_0, T)$ is the least fix-point of the operator $\mathbb{F}_{\mathcal{Q}_0,T}$ that is defined as follows:

$$\begin{cases} \wp(\mathcal{T}^+) \to \wp(\mathcal{T}^+) \\ X \qquad \mapsto \{q \mid q \in \mathcal{Q}_0\} \cup \left\{ \tau \frown (q, \lambda, q') \;\middle|\; \begin{array}{l} \tau \in X \wedge \\ \text{last}(\tau) = q \wedge \\ (q, \lambda, q') \in T \end{array} \right\}. \end{cases}$$

The operator $\mathbb{F}_{\mathcal{Q}_0,T}$ is monotonic (i. e. for every two sets of traces $X, Y \in \wp(\mathcal{T}^+)$ such that $X \subseteq Y$, we have $\mathbb{F}_{\mathcal{Q}_0,T}(X) \subseteq \mathbb{F}_{\mathcal{Q}_0,T}(Y)$). By [20], it follows that $\mathbb{F}_{\mathcal{Q}_0,T}$ has a fix-point that is included in any other fix-point. This least fix-point, *lfp* $\mathbb{F}_{\mathcal{Q}_0,T}$, is defined by the following conditions:

1) *lfp* $\mathbb{F}_{\mathcal{Q}_0,T} = \mathbb{F}_{\mathcal{Q}_0,T}($*lfp* $\mathbb{F}_{\mathcal{Q}_0,T})$;
2) for each set $X' \subseteq \wp(\mathcal{T}^+)$ of traces, we have: *lfp* $\mathbb{F}_{\mathcal{Q}_0,T} \subseteq X'$ whenever $X' = \mathbb{F}_{\mathcal{Q}_0,T}(X')$.

Conversely, a set $X \subseteq \mathcal{T}^+$ of finite traces can be abstracted by the transition system $\alpha_{\mathbb{Q},\mathbb{L}}(X)$ that is defined as the pair $(\mathcal{Q}_0, T)$ with:

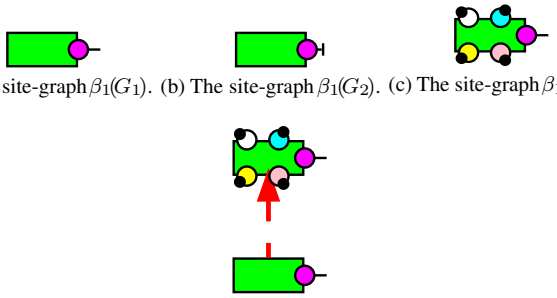
1) $\mathcal{Q}_0 = \{\text{fst}(\tau) \mid \tau \in X\}$;
2) $T = \{t_i \mid \exists (q_0, (t_i)_{1\leq i\leq p}) \in X, i \in [\![1, p]\!]\}$.

The pair $(\alpha_{\mathbb{Q},\mathbb{L}}, \gamma_{\mathbb{Q},\mathbb{L}})$ forms a Galois connection between the complete lattices $(\wp(\mathcal{T}^+), \subseteq)$ and $(\mathbb{T}_{\mathbb{Q},\mathbb{L}}, \sqsubseteq)$. This means that for any set $X \subseteq \mathcal{T}^+$ of finite traces and any transition system $(\mathcal{Q}'_0, T')$, we have $\alpha_{\mathbb{Q},\mathbb{L}}(X) \sqsubseteq (\mathcal{Q}'_0, T')$, if and only if, $X \subseteq \gamma_{\mathbb{Q},\mathbb{L}}(\mathcal{Q}'_0, T')$. It follows (e. g. see [6]), that:

1) the functions $\alpha_{\mathbb{Q},\mathbb{L}}$ and $\gamma_{\mathbb{Q},\mathbb{L}}$ are both monotonic;
2) the function $\alpha_{\mathbb{Q},\mathbb{L}}$ maps each set of finite traces to the smallest (for $\sqsubseteq$) transition system which induces this set of traces, i. e. $\alpha_{\mathbb{Q},\mathbb{L}}(X)$ is the best abstraction of the set of traces X as a transition system.

V. Local transition systems

In Sect. IV, we have associated each model with a transition system describing the set of the finite traces of this model. However, such transition systems are usually too large to be computed, or even if they could, they are too complex to help understanding the behaviour of the models. This is the reason why we propose to simplify these transition systems by focusing on the behaviour of each protein independently, abstracting away which proteins are bound together. This abstraction has already been applied in [13], [8], to infer the relationships among the state of sites in protein instances. Here we extend this static analysis to traces.



(a) The site-graph $\beta_1(G_1)$. (b) The site-graph $\beta_1(G_2)$. (c) The site-graph $\beta_1(G_3)$.

(d) The embedding $\beta_1(f)$ from the site-graph $\beta_1(G_1)$ and the site-graph $\beta_1(G_3)$.

Fig. 12. Abstraction of the site-graphs G_1, G_2, G_3, and of the embedding f. The abstraction of the embedding f from the site-graph G_1 to G_2 is indeed an embedding from the abstraction of the site-graph G_1 to the abstraction of the site-graph G_2.

A. Best abstraction

Firstly we explain how to track the behaviour of each protein independently while forgetting about the bonds between pairs of binding sites. For any agent identifier $n \in \mathbb{N}$, we denote by β_n the function that, when applied to a site-graph G containing an agent with identifier n:

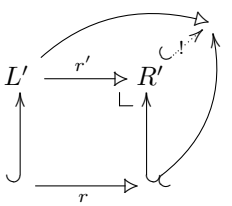
1) replaces any bond between two sites by two occurrences of the symbol '$-$';
2) restricts the site-graph G to the agent with identifier n;
3) renames the identifier n with 1.

Definition 9 (site-graph restriction): The site-graph $\beta_n(G)$ is defined by:

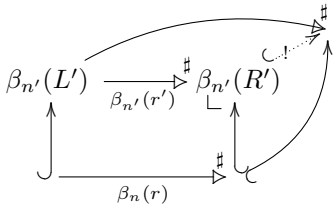
1) $\mathcal{A}_{\beta_n(G)} = \{1\}$;
2) $type_{\beta_n(G)} = [1 \mapsto type_G(n)]$;
3) $\mathcal{S}_{\beta_n(G)} := \{(1, i) \mid (n, i) \in \mathcal{S}_G\}$;
4) the function $\mathcal{L}_{\beta_n(G)}$ maps each site $(1, i)$ such that $i \in \Sigma^{lnk}_{ag\text{-}st}(type_G(n))$ and $(n, i) \in \mathcal{S}_G$, to the symbol '$\dashv$' whenever $\mathcal{L}_G(n, i) = \dashv$, and to the symbol '$-$' otherwise;
5) the function $p\kappa_{\beta_n(G)}$ maps each site $(1, i)$ such that $i \in \Sigma^{int}_{ag\text{-}st}(type_G(n))$ and $(n, i) \in \mathcal{S}_G$, to the state $p\kappa_G(n, i)$.

Example 7 (running example): The restriction of the site-graphs G_1, G_2, and G_3 to their agent with identifier 1, are depicted in Figs. 12(a), 12(b), and 12(c). Formally, they are defined as follows:

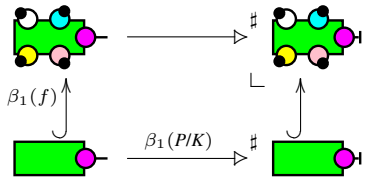
1) a) $\mathcal{A}_{\beta_1(G_1)} = \{1\}$,
b) $type_{\beta_1(G_1)} = [1 \mapsto P]$,
c) $\mathcal{S}_{\beta_1(G_1)} = \{(1, x)\}$,
d) $\mathcal{L}_{\beta_1(G_1)} = [(1, x) \mapsto -]$,



(a) Push-out in the concrete.



(b) Push-out in the abstract.



(c) Applying the abstraction of the rule P/K along the embedding $\beta(f)$.

Fig. 13. The abstraction of a push-out (in the concrete) is a push-out (in the abstract), and application of rule (in the abstract).

e) $p\kappa_{\beta_1(G_1)} = []$;

2) *a)* $\mathcal{A}_{\beta_1(G_2)} = \{1\}$,
b) $type_{\beta_1(G_2)} = [1 \mapsto P]$,
c) $\mathcal{S}_{\beta_1(G_2)} = \mathcal{S}_{G_1}$,
d) $\mathcal{L}_{\beta_1(G_2)} = [(1,x) \mapsto \dashv]$,
e) $p\kappa_{\beta_1(G_2)} = []$;

3) *a)* $\mathcal{A}_{\beta_1(G_3)} = \{1\}$,
b) $type_{\beta_1(G_3)} = [1 \mapsto P]$,
c) $\mathcal{S}_{\beta_1(G_3)} = \{(1,x),(1,a_1),(1,a_2),(1,b_1),(1,b_2)\}$,
d) $\mathcal{L}_{\beta_1(G_3)} = [(1,x) \mapsto -]$,
e) $p\kappa_{\beta_1(G_3)} = [a_1 \mapsto \bullet, a_2 \mapsto \bullet, b_1 \mapsto \bullet, b_2 \mapsto \bullet,]$.

□

For each agent type A, we define the set of the local views $\mathbb{Q}_A$ for the agents of type A, as the set of the site-graphs G with only one agent of type A and identifier 1, that documents the state of all the sites in $\Sigma_{ag\text{-}st}(A)$ and such that each binding site is either free or bound to an unspecified site. Formally, the latter two constraints means that $\mathcal{S}_G = \{(1,i) \mid i \in \Sigma_{ag\text{-}st}(A)\}$ and that for every $i \in \Sigma^{lnk}_{ag\text{-}st}(A)$, $\mathcal{L}(1,i) \in \{\dashv, -\}$.

Embeddings are compatible with site-graph restriction. If f induces an embedding from a site-graph G into a site-graph H and $n \in \mathcal{A}_G$ is an agent identifier of an agent of the site-graph G, then the function $[1 \mapsto 1]$ induces an embedding from the site-graph $\beta_n(G)$ into the site graph $\beta_{f(n)}(H)$.

Example 8 (running example): We notice that the function $[1 \mapsto 1]$ *induces an embedding from the site-graph* $\beta_1(G_1)$ *into the site-graph* $\beta_1(G_3)$*, as depicted in Fig. 12(d).* □

We define the set of the local transition labels $\mathbb{L}_A$ as the set of pairs (r,n) where r is a rule and n is an agent identifier. Intuitively, the local transition label (r,n) denotes the fact that a rule r is applied along an embedding that matches the agent with identifier n on the left hand side of the rule r, to the local view to which we want to apply the rule. Given a rule $r : L \hookleftarrow D \hookrightarrow R$ and an identifier $n \in \mathcal{A}_L$ of an agent on the left hand side L of the rule r, we define the abstraction $\beta_n(r)$ of the rule r as the span of embeddings $\beta_n(L) \hookleftarrow \beta_n(D) \hookrightarrow \beta_n(R)$. In this span, both embeddings are induced by the function $[1 \mapsto 1]$. The span of embeddings $\beta_n(r)$ is not a rule in general, yet it can be applied along every embedding from its left hand side $\beta_n(L)$ into a site-graph L' thanks to a push-out construction (e. g. see Fig. 13). Indeed, given a pushout in the concrete (e. g. see Fig. 13(a)) in which the left embedding h_L maps the agent with identifier n on the left hand side of the rule to the agent with identifier n' in the site-graph L', we can apply the abstraction β_n to the rule r, and the abstraction $\beta_{n'}$ to the rule r', as a result, we get a push-out (the left and right embeddings are both induced by the function $[1 \mapsto 1]$). We obtain an abstract computation step that we write $L' \xrightarrow{(r,n)}{}^{\sharp} R'$.

Example 9 (running example): The application of the abstraction $\beta_1(P/K)$ *of the rule* P/K *can be applied to the site-graph* $\beta_1(G_1)$ *(e.g. see Fig. 13(c)).* □

In an abstract computation step $q^\sharp \xrightarrow{(r,n)}{}^{\sharp} q^{\sharp\prime}$ if $q^\sharp$ is a local view in $\mathbb{Q}_A$, then $q^{\sharp\prime}$ is a local view in $\mathbb{Q}_A$ as well.

The application of the function β_n can be lifted to traces. Given a trace

$$\tau := (q'_0, (q_i, (r_i, f_i), q'_i)_{1 \le i \le p})$$

in $\mathcal{T}^+$ and an agent identifier $n \in \mathcal{A}_{q'_0}$, we define the local trace $\beta_n(\tau)$ as the trace:

$$(\beta_n(q'_0), (\beta_n(q_{\sigma(i)}), (r_{\sigma(i)}, n_i), \beta_n(q'_{\sigma(i)}))_{1 \le i \le p'})$$

where:

1) $\sigma_1, \ldots, \sigma_{p'}$ is the sequence (in increasing order) of the integers i between 1 and p such that $\beta_n(q_i) \neq \beta_n(q_{i+1})$;
2) and for each integer i between 1 and p', the integer n_i is the unique identifier such that the embedding f_{σ_i} maps the agent with identifier n_i on the left hand side of the rule r_{σ_i} into the agent with identifier n in the chemical mixture q_{σ_i} (such an integer always exists, otherwise the abstract state would not have been modified).

Now we combine our abstractions to over-approximate the behaviour of each agent as an independent *local* transition system. The abstraction $\alpha_\pi(X)$ of a set of traces $X \subseteq \mathcal{T}^+$ is defined as the function mapping each agent type $A \in \Sigma_{ag}$ into the following local transition system:

$$\alpha_{\mathbb{Q}_A,\mathbb{L}_A}\left(\left\{\beta_n(\tau) \;\middle|\; \begin{array}{l}\tau \in X,\ n \in \mathcal{A}_{\mathrm{fst}(\tau)} \\ \text{such that } type_{\mathrm{fst}(\tau)}(n) = A\end{array}\right\}\right)$$

(i. e. the best over-approximation, as a transition system, of the set of the local traces that can be associated with an agent of type A; in this formula, the expression $type_{\mathrm{fst}(\tau)}(n)$ denotes the type of the agent with identifier n in the site-graphs all along the trace τ, since the agent with identifier n in each state of the trace is the same, we pick it in the initial state).

Example 10 (running example): The trace in Fig. 11 describes a potential evolution for the state of four proteins. We give the four corresponding local traces in Fig. 14. □

Conversely, the concretization $\gamma_\pi(Y)$ of a function between agent types and local transition systems, is defined as the set of the traces $\tau \in \mathcal{T}^+$ such that for each identifier n of an agent in the initial state $\mathrm{fst}(\tau)$ of the trace τ, the local trace $\beta_n(\tau)$ belongs to the set $\gamma_{\mathbb{Q}_A,\mathbb{L}_A}(Y(type_{\mathrm{fst}(\tau)}(n)))$. Again, the

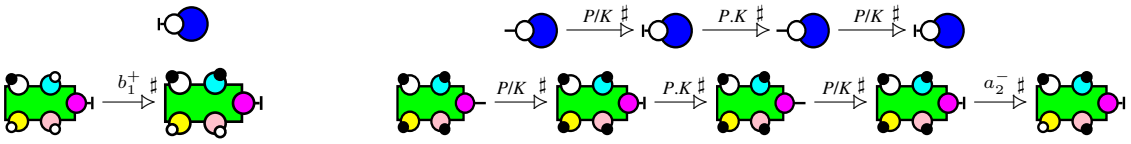


Fig. 14. The four local traces of the trace in Fig. 11

expression $type_{\mathrm{fst}(\tau)}(n)$ denotes the type of the agent with identifier n in the site-graphs all along the trace τ.

By [6], the function $\alpha_\pi \circ \mathbb{F}_{\mathcal{Q}_0,T} \circ \gamma_\pi$ is the best abstract counterpart to the function $\mathbb{F}_{\mathcal{Q}_0,T}$. That function is monotonic and its least fix-point satisfies the inclusion: $lfp\ \mathbb{F}_{\mathcal{Q}_0,T} \subseteq \gamma_\pi(lfp\ \alpha_\pi \circ \mathbb{F}_{\mathcal{Q}_0,T} \circ \gamma_\pi)$. The least fix-point on the right hand side of the inclusion can be computed in a finite number of iterations since the domain of $\alpha_\pi \circ \mathbb{F}_{\mathcal{Q}_0,T} \circ \gamma_\pi$ is finite.

B. Efficient over-approximation

However computing these iterations is quite cumbersome because it intertwines the computation of the local transition systems that are associated to each agent type. We propose to desynchronise these computations. To do this, we introduce, for each agent type $A \in \Sigma_{ag}$, the function $\mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}$ that maps any local transition system $(\mathcal{Q}_0^\sharp, T^\sharp) \in \mathbb{T}_{\mathbb{Q}_A,\mathbb{L}_A}$ to the local transition system $(\mathcal{Q}_0^{\sharp\prime}, T^{\sharp\prime}) \in \mathbb{T}_{\mathbb{Q}_A,\mathbb{L}_A}$ where:

1) $\mathcal{Q}_0^{\sharp\prime} = \mathcal{Q}_0^\sharp \cup \left\{ \beta_n(q_0) \;\middle|\; \begin{array}{l} q_0 \in \mathcal{Q}_0,\ n \in \mathcal{A}_{q_0}, \\ type_{q_0}(n) = A \end{array} \right\}$;
2) and $T^{\sharp\prime}$ is the union between the set $T^\sharp$ and the set of the transitions $(q^\sharp, (r,n), q^{\sharp\prime})$ such that the local view $q^\sharp$ is reachable in the transition system $(\mathcal{Q}_0^\sharp, T^\sharp)$ and such that $q^\sharp \xrightarrow{r,n}{}^\sharp q^{\sharp\prime}$.

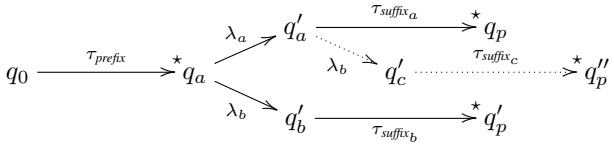
For any function Y mapping each agent type $A \in \Sigma_{ag}$ to a local transition system over the states $\mathbb{Q}_A$ and the transition labels $\mathbb{L}_A$, we have $[\alpha_\pi \circ \mathbb{F}_{\mathcal{Q}_0,T} \circ \gamma_\pi](Y) \sqsubseteq \mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}(Y)$. Moreover, for each agent type $A \in \Sigma_{ag}$, the function $\mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}$ is monotonic. By [20], for each agent type $A \in \Sigma_{ag}$, the function $\mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}$ has a least fix-point. By [6], the inclusion $lfp\ \mathbb{F}_{\mathcal{Q}_0,T} \subseteq \gamma_\pi([A \in \Sigma_{ag} \mapsto lfp\ \mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}(A)])$ is satisfied. This ensures the soundness of our approach: the concretization of the result of our analysis is a superset of the set of the traces of the model.

Example 11 (running example): The local transition system for receptor proteins in the first case study (e. g. see Fig. 1 on page 3) is given in Fig. 2 on page 2. The local transition for the protein of type P in the second case study (e. g. see Fig. 3 on page 4) is given in Fig. 4 on page 4.

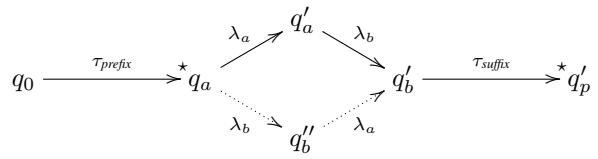
We have derived an abstraction of the finite trace semantics, as a family of local transition systems describing the behaviour of each particular type of agent, in isolation. Each local transition system can be computed independently iteratively.

VI. Macrotransition systems

In the second case study (e. g. see Fig. 4), the local transition system for the protein P is not practically visualisable due to the number of potential conformations. We propose to identify which transitions commute. Then, by abstracting the interleaving order of commutative transitions, we get a more compact symbolic description of each local transition system.



(a) Concurrent transitions.



(b) Consecutive transitions.

Fig. 15. Pairs of commutative transitions.

A. Commutating events

Let us firstly define the notion of pairs of commutative transitions in a given set of traces $X \subseteq \mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$. We denote by T_X the set of the transitions that occur in X, i. e. $T_X := \{(q_i, \lambda_i, q'_i) \mid (q_0, (q_i, \lambda_i, q'_i))_{1 \leq i \leq p} \in X, 1 \leq i \leq p\}$. For the sake of simplicity and since this is the case in Kappa, we assume that every transition that occurs in a trace τ in the set X, is fully defined by the state it is starting from and the label of the transition. Formally, this means that for every two transitions $(q_1, \lambda_1, q'_1), (q_2, \lambda_2, q'_2) \in T_X$, the states q'_1 and q'_2 are equal whenever the pairs (q_1, λ_1) and (q_2, λ_2) are equal. In other words, the label is used as a proof which fully identifies how to apply a computation step to the current state.

Definition 10 (commutative transitions): We say that the transitions with labels $\lambda_a \in \mathbb{L}$ and $\lambda_b \in \mathbb{L}$ commute in the set of trace $X \subseteq \mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$, if and only if, both following properties are satisfied:

1) *for every three traces τ_{prefix}, τ_{suffix_a} and τ_{suffix_b} in the set $\mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$, and every three states q_a, q'_a, and q'_b in the set $\mathbb{Q}$, such that $(q_a, \lambda_a, q'_a) \in T_X$, $(q_a, \lambda_b, q'_b) \in T_X$, $last(\tau_{prefix}) = q_a$, $q'_a = fst(\tau_{suffix_a})$, and $q'_b = fst(\tau_{suffix_b})$, if both traces $\tau_{prefix} \frown (q_a, \lambda_a, q'_a) \,.\, \tau_{suffix_a}$ and $\tau_{prefix} \frown (q_b, \lambda_b, q'_b) \,.\, \tau_{suffix_b}$ belong to the set X (e.g. see Fig. 15(a)), then there exist a state q'_c in the set $\mathbb{Q}$ and a trace τ_{suffix_c} in the set $\mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$ with $(q'_a, \lambda_b, q'_c) \in T_X$ and $fst(\tau_{suffix_c}) = q'_c$ such that the trace $\tau_{prefix} \frown (q_a, \lambda_a, q'_a) \frown (q'_a, \lambda_b, q'_c) \,.\, \tau_{suffix_c}$ belongs to the X as well.*
2) *for every two traces τ_{prefix} and τ_{suffix} in the set $\mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$, and every three states q_a, q'_a, and q'_b in the set $\mathbb{Q}$, satisfying the conditions $(q_a, \lambda_a, q'_a) \in T_X$, $(q'_a, \lambda_b, q'_b) \in T_X$, $last(\tau_{prefix}) = q_a$, and $q'_b = fst(\tau_{suffix})$, if the trace $\tau_{prefix} \frown (q_a, \lambda_a, q'_a) \frown (q'_a, \lambda_b, q'_b) \,.\, \tau_{suffix}$ belongs to the set X (e.g. see Fig. 15(b)), then there exists a state q''_b in the set $\mathbb{Q}$ with $(q_a, \lambda_b, q''_b) \in T_X$ and $(q''_b, \lambda_a, q'_b) \in T_X$ such*

that the trace $\tau_{prefix} \frown (q_a, \lambda_b, q''_b) \frown (q''_b, \lambda_a, q'_b) \,.\, \tau_{suffix}$ *belongs to the* X *as well.*

In Def. 10, the first property entails that whenever two transitions that commute are enabled after a given prefix of trace (they necessarily start from the same state), each transition may be followed by the other one provided that the latter transition now starts from the ending state of the former one (e. g. see Fig. 15(a)), whereas the second property entails that two consecutive transitions that commute may always be performed in the reverse order, modulo the fact that the intermediary state may be different (e. g. see Fig. 15(b)).

It is worth noticing that both conditions are necessary to define commuting events properly as shown is the following two counter-examples.

Example 12 (counter-examples): Let us consider two distinct transitions labels. For the sake of simplicity, we denote each trace as a sequence of transitions labels, discarding the states.

We consider both following sets of traces:

1) $X_1 := \{\stackrel{\lambda_1}{\rightarrow}; \stackrel{\lambda_2}{\rightarrow}\}$;
2) $X_2 := \{\stackrel{\lambda_1}{\rightarrow}\stackrel{\lambda_2}{\rightarrow}\}$.

The set X_1 *contains two traces made of a single transition each, whereas the set* X_2 *contains a single trace made of two consecutive transitions. The transitions with labels* λ_1 *and* λ_2 *commute neither in the set of traces* X_1*, nor in the set of traces* X_2*.*

The first condition in Def. 10 ensures that the transitions with labels λ_1 *and* λ_2 *do not commute in the set of traces* X_1*, whereas the second condition in Def. 10 ensures that the transitions with labels* λ_1 *and* λ_2 *do not commute in the set of traces* X_2*.* □

We define a closure operator that complete a set of traces to ensure that some pairs of transitions commute.

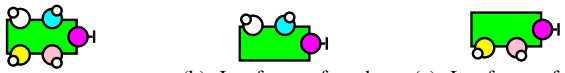
Definition 11 (commutation closure): Given $\mathcal{C} \in \wp(\mathbb{L}^2)$ *a set of pairs of transition labels, we define the operator* $\rho_{\mathcal{C}}$ *which maps each set of traces* $X \subseteq \mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$*, into the smallest set of traces* $\rho_{\mathcal{C}}(X) \subseteq \mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$ *that contains the set* X *and in which for every pair of transition labels* (λ_a, λ_b) *in* $\mathcal{C}$ *the transitions with labels* λ_a *and* λ_b *commute in the set of traces* $\rho_{\mathcal{C}}(X)$*.*

The function $\rho_{\mathcal{C}}$ is an upper closure operator, i. e. it satisfies the following three properties:

1) $\rho_{\mathcal{C}}$ is monotonic (i. e. for every subsets $X, Y \in \mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$ such that $X \subseteq Y$, we have: $\rho_{\mathcal{C}}(X) \subseteq \rho_{\mathcal{C}}(Y)$;
2) $\rho_{\mathcal{C}}$ is idempotent (i. e. $\rho_{\mathcal{C}} \circ \rho_{\mathcal{C}} = \rho_{\mathcal{C}}$);
3) $\rho_{\mathcal{C}}$ is extensive (i. e. for every subset $X \subseteq \mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$ of traces, we have: $X \subseteq \rho_{\mathcal{C}}(X)$).

We notice that the fix-points of the upper closure $\rho_{\mathcal{C}}$ are the sets of the traces $X \subseteq \mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$ such that every pair of transitions with their pair of labels in the set $\mathcal{C}$ commutes.

Let us consider a transition system $(\mathcal{Q}_0, T)$. We assume that we are given a set $\mathcal{C}$ of pairs of transition labels such that for every pair $(\lambda_1, \lambda_2) \in \mathcal{C}$, the transitions with labels λ_1 and λ_2 commute in the set of traces $\gamma_{\mathbb{Q},\mathbb{L}}(\mathcal{Q}_0, T)$ that is induced by the transition system $(\mathcal{Q}_0, T)$. We can use the upper closure $\rho_{\mathcal{C}}$ to accelerate the computation of the set $\gamma_{\mathbb{Q},\mathbb{L}}(\mathcal{Q}_0, T)$. The function $\rho_{\mathcal{C}} \circ \mathbb{F}_{\mathcal{Q}_0,T}$ is monotonic and satisfies the condition: $\mathbb{F}_{\mathcal{Q}_0,T}(X) \subseteq [\rho_{\mathcal{C}} \circ \mathbb{F}_{\mathcal{Q}_0,T}](X)$ for every set of traces $X \subseteq \mathbb{T}^+_{\mathbb{Q},\mathbb{L}}$. It follows by [20], that the function $\rho_{\mathcal{C}} \circ \mathbb{F}_{\mathcal{Q}_0,T}$ has a least fix-point, and by [6], that the condition: $\mathit{lfp}\ \mathbb{F}_{\mathcal{Q}_0,T} \subseteq \mathit{lfp}\ [\rho_{\mathcal{C}} \circ \mathbb{F}_{\mathcal{Q}_0,T}]$ is satisfied. Since, additionally, each pair of transitions in the set $\mathcal{C}$ commutes in the set of traces $\gamma_{\mathbb{Q},\mathbb{L}}(\mathcal{Q}_0, T)$ (i. e. $\rho_{\mathcal{C}}(\mathit{lfp}\ \mathbb{F}_{\mathcal{Q}_0,T}) = \mathit{lfp}\ \mathbb{F}_{\mathcal{Q}_0,T}$), we conclude that the property: $\mathit{lfp}\ \mathbb{F}_{\mathcal{Q}_0,T} = \mathit{lfp}\ [\rho_{\mathcal{C}} \circ \mathbb{F}_{\mathcal{Q}_0,T}]$ is satisfied.



(a) A local view. (b) Its frame for the transition label $(a_1^+, 1)$. (c) Its frame for the transition label $(a_2^+, 1)$.

Fig. 16. A local view and its frames.

In Kappa, pairs of commutative local transitions can be identified syntactically. Let us consider an agent type $A \in \Sigma_{ag}$. Given the label $(r, n) \in \mathbb{L}_A$ of a local transition, we denote as $\textsc{test}(r, n)$ the set of the site identifiers i such that the site (n, i) occurs in the domain of the rule r, and as $\textsc{mod}(r, n)$ the set of the site identifiers i that carry a different (binding or internal) state in the left hand side and in the right hand side of the rule r. Then, for any two labels λ_1, $\lambda_2 \in \mathbb{L}_A$ of local transitions such that both $\textsc{mod}(\lambda_1) \cap \textsc{test}(\lambda_2) = \emptyset$ and $\textsc{mod}(\lambda_2) \cap \textsc{test}(\lambda_1) = \emptyset$, any pair of local transitions with the labels λ_1 and λ_2 commutes [18].

Example 13 (running example): In the rules Fig. 3 on page 4, we have:

- $\textsc{test}(a_1^+, 1) = \{a_1\}$, $\textsc{mod}(a_1^+, 1) = \{a_1\}$;
- $\textsc{test}(a_2^+, 1) = \{a_2\}$, $\textsc{mod}(a_2^+, 1) = \{a_2\}$.
- $\textsc{test}(b_1^+, 1) = \{a_1, b_1\}$, $\textsc{mod}(b_1^+, 1) = \{b_1\}$;
- $\textsc{test}(b_2^+, 1) = \{a_2, b_2\}$, $\textsc{mod}(b_2^+, 1) = \{b_2\}$;
- $\textsc{test}(P/K, 1) = \{a_1, a_2, b_1, b_2, x\}$,
- $\textsc{mod}(P/K, 1) = \{x\}$.

As a consequence, every pair of local transition with respective labels λ_1 *and* λ_2 *such that* λ_1 *belongs to the set* $\{(a_1^+, 1), (b_1^+, 1)\}$ *and* λ_2 *belongs to the set* $\{(a_2^+, 1), (b_2^+, 1)\}$ *commutes. But for every label* λ *in the set* $\{(a_1^+, 1), (b_1^+, 1), (a_2^+, 1), (b_2^+, 1)\}$ *there may be pair of local transitions with labels* λ *and* $(P/K, 1)$ *respectively that do not commute.* □

Given a local view $v \in \mathbb{Q}_A$, we consider the set $\Lambda(\lambda, v)$ as the set of all the transitions labels λ' such that $\textsc{mod}(\lambda) \cap \textsc{test}(\lambda') = \emptyset$, $\textsc{test}(\lambda) \cap \textsc{mod}(\lambda') = \emptyset$ and there exists a local transition starting from the view v and with the label λ'. We define the site-graph $\textsc{frame}_\lambda(v)$ as the site-graph that is obtained by removing from the site-graph v any site that belongs to the set $\bigcup\{\textsc{mod}(\lambda') \mid \lambda' \in \Lambda(\lambda, v) \setminus \{\lambda\}\}$. Intuitively, $\textsc{frame}_\lambda(v)$ is the restriction of the local view v to the sites that cannot be modified by local transitions that commute with a local transition with the label λ.

Example 14 (running example): For instance, in our second case study, the local view that is depicted in Fig. 16(a) has the following frames:

- $\textsc{frame}_{(a_1^+,1)}(v) = \{a^1, b^1, x\}$
- $\textsc{frame}_{(a_2^+,1)}(v) = \{a^2, b^2, x\}$;

which corresponds to the site-graphs that are given in Fig. 16(b) and 16(c) respectively. □

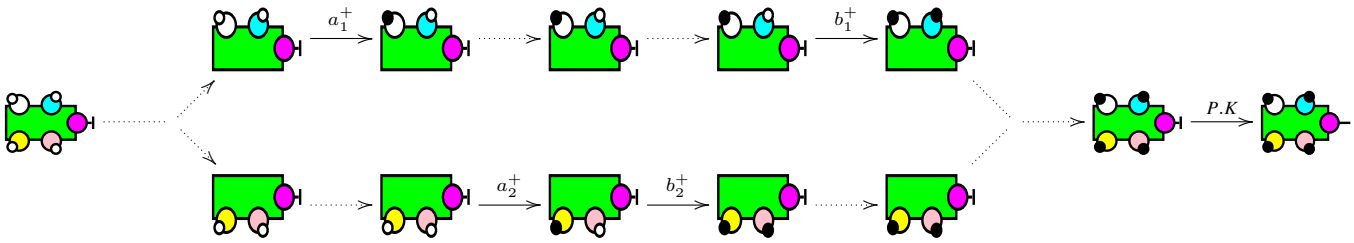


Fig. 17. An example of a trace induced by the macrotransition system of Fig. 5.

B. Macrotransition systems

We are left to provide a compact data-structure to represent transition systems with pairs of commutative transitions. We propose to use macrostates and macrotransitions. A *macrostate* is a symbolic representation of a set of (micro)states that behave similarly, and *macrotransitions* are transitions between macrostates, that denote some transitions between the corresponding microstates. Let us assume that we are given a set of macrostates $\mathbb{Q}^\sharp$. Each macrostate $q^\sharp \in \mathbb{Q}^\sharp$ denotes a set of microstates $\Gamma_{\mathbb{Q},\mathbb{Q}^\sharp}(q^\sharp) \subseteq \mathbb{Q}$.

A macrotransition system is defined as follows:

Definition 12 (macrotransition system): A macrotransition system is a pair $(\mathcal{Q}_0, T^\sharp)$ with $\mathcal{Q}_0 \subseteq \mathbb{Q}$ and $T^\sharp \subseteq \mathbb{Q}^\sharp \times \mathbb{L} \times \mathbb{Q}^\sharp$.

A macrotransition system is made of a set of initial microstates and a set of labelled transitions between macrostates. Each macrotransition $(q^\sharp, \lambda, q^{\sharp\prime})$ denotes the set of the transitions $(q, \lambda, q') \in \mathbb{Q} \times \mathbb{L} \times \mathbb{Q}$ for which there exists a set of macrostates $X \subseteq \mathbb{Q}^\sharp$ satisfying both following conditions:

- $\{q\} = \bigcap\{\Gamma_{\mathbb{Q},\mathbb{Q}^\sharp}(x) \mid x \in X \cup \{q^\sharp\}\}$;
- $\{q'\} = \bigcap\{\Gamma_{\mathbb{Q},\mathbb{Q}^\sharp}(x) \mid x \in X \cup \{q^{\sharp\prime}\}\}$.

We denote by $\Gamma'_{\mathbb{Q},\mathbb{Q}^\sharp}(q^\sharp, \lambda, q^{\sharp\prime})$ the set of the transitions that are denoted by the macrotransition $(q^\sharp, \lambda, q^{\sharp\prime})$. Intuitively, each macrotransition models an update of some constraints over the microstates. This update is encoded by the pair of macrostates $(q^\sharp, q^{\sharp\prime})$. In order to fully identify a pair of states, these constraints must be completed by some others, encoded by the set X, that are not modified by the transition. Then, a macrotransition system $(\mathcal{Q}_0, T^\sharp)$ stands for a classical transition system $\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}(\mathcal{Q}_0, T^\sharp)$, that is obtained by gathering all the transitions that are encoded by the macrotransitions (i. e. $\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}(\mathcal{Q}_0, T^\sharp) = (\mathcal{Q}_0, \cup\{\Gamma'_{\mathbb{Q},\mathbb{Q}^\sharp}(t) \mid t \in T^\sharp\})$).

In Kappa, a macrostate is a site-graph that can be embedded in a local view. Each macrostate denotes the set of the local views it can be embedded in. This way, a microtransition is obtained by applying a rule over its frame while the state of the other sites can be completed by using the other local views.

C. Fix-point computation

Now, we mimic the computation of $\mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}$ in macrotransition systems. We define the function $\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp}$ mapping each macrotransition system $(\mathcal{Q}_0, T^\sharp)$ to the macrotransition system $(\mathcal{Q}'_0, T^{\sharp\prime})$ where:

1) $\mathcal{Q}'_0 := \mathcal{Q}_0 \cup \{\beta_n(q_0) \mid n \in \mathcal{A}(q_0),\ type_{q_0}(n) = A\}$;
2) The set $T^{\sharp\prime}$ is obtained in two steps. Firstly, we compute the union between the set $T^\sharp$ and the set of the transitions of the form $(\text{FRAME}_\lambda(v), \lambda, v')$ for every local view $v \in \mathbb{Q}_A$ that is reachable in the transition system $\gamma_{\mathbb{Q}_A,\mathbb{L}_A}(\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}(\mathcal{Q}_0, T^\sharp))$; Secondly we remove every macrotransition t such that there exists a macrotransition t' satisfying $\Gamma(t) \subset \Gamma(t')$ in this union.

The function $\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp}$ is not monotonic in general. Yet, for every macrotransition system $X^\sharp$, both following inclusions:

1) (*soundness*)
$$\mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}(\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}(X^\sharp)) \sqsubseteq \Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}(\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp}(X^\sharp)).$$
2) (*relative completeness*)
$$\gamma_{\mathbb{Q}_A,\mathbb{L}_A}(\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}(\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp}(X^\sharp))) \subseteq \rho_{\mathcal{C}}(\gamma_{\mathbb{Q}_A,\mathbb{L}_A}(\mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}(\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}(X^\sharp))));$$

are satisfied.

The first property formalises the fact that the function $\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp}$ is a sound counterpart to the function $\mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}$. In one iteration of the function $\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp}$, all the macrotransitions that are inserted are enough to simulate all the transitions that would have been inserted in one iteration of the function $\mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}$. The second property establishes the relative completeness of our approach. In one iteration of the function $\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp}$, every new trace that can be simulated by the new set of macrotransitions, can be obtained by changing the interleaving order of commutative events in some trace induced by the local transition system that would have been obtained at the next iteration of the function $\mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}$.

It follows the following theorem.

Theorem 1: When k increases to infinity, the sequence:
$$[\gamma_{\mathbb{Q}_A,\mathbb{L}_A} \circ \Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}]((\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp})^k(\emptyset,\emptyset)))_{k\in\mathbb{N}}$$
stations ultimately to the value: $\gamma_{\mathbb{Q}_A,\mathbb{L}_A}(\mathit{lfp}\ \mathbb{F}^\sharp_{\mathcal{Q}_0,T,A})$.

Proof 1: We have $\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}(\emptyset,\emptyset) = (\emptyset,\emptyset)$.

Then, by induction over $k \in \mathbb{N}$, we have:

$$(\mathbb{F}^\sharp_{\mathcal{Q}_0,T,A})^k(\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}(\emptyset,\emptyset)) \sqsubseteq \Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}((\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp})^k(\emptyset,\emptyset))$$

and:

$$\gamma_{\mathbb{Q}_A,\mathbb{L}_A}(\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}((\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp})^k(\emptyset,\emptyset))) \subseteq \gamma_{\mathbb{Q}_A,\mathbb{L}_A}(\mathit{lfp}\ \mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}).$$

Ultimately, we get:

$$\gamma_{\mathbb{Q}_A,\mathbb{L}_A}(\mathit{lfp}\ \mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}) \subseteq \gamma_{\mathbb{Q}_A,\mathbb{L}_A}(\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}((\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp})^k(\emptyset,\emptyset)))$$

and:

$$\gamma_{\mathbb{Q}_A,\mathbb{L}_A}(\Gamma_{\mathcal{Q}_0,T,\mathbb{Q}^\sharp}((\mathbb{G}^\sharp_{\mathcal{Q}_0,T^\sharp})^k(\emptyset,\emptyset))) \subseteq \gamma_{\mathbb{Q}_A,\mathbb{L}_A}(\mathit{lfp}\ \mathbb{F}^\sharp_{\mathcal{Q}_0,T,A}).$$

□

Thm. 1 establishes that ultimately, the iterates of the macrotransitions systems induce exactly the set of local traces.

model	version	number of agents	number of rules	analysis time	number of transitions in local transition systems average	number of transitions in local transition systems max	number of dead rules	number of separating rules
TGFβ	V20	35	99	0.46 s / 0.52 s	12 / 8	107 / 37	10	150
TGFβ	V21	36	211	1.06 s / 1.14 s	31 / 18	531 / 106	10	150
TGFβ	2017-03	36	215	1.01 s / 1.13 s	27 / 17	364 / 107	0	21
Wnt	2015-12	17	356	134 s / 11 s	144 / 17	934 / 124	1	833
Wnt	2016-09	17	1419	276 s / 136 s	572 / 156	20992 / 2118	0	30
Wnt	2017-03	29	1486	293 s / 148 s	429 / 121	19969 / 2118	12	61
Ras	V1	288	3916	49 s / 49 s	2 / 2	23 / 23	1610	2016
Ras	V2	112	12896	282 s / 283 s	11 / 11	457 / 457	874	8296
Ras	V3	290	5750	202 s / 205 s	4 / 4	39 / 39	885	1397

Fig. 18. Benchmarks (performed on a MacBook Pro, 3.3 Ghz intel Core i7). We tested the computation of all local transition systems and all macrotransition systems. For each version of each model, we report the number of agents and rules, the computation time for both analysis modes, the average and the max number of transitions per transition system for both analysis modes, the number of dead rules and the number of rules that may induces a non weakly reversible transition. When it applies, the first data refers to the computation of the local transition systems, whereas the second ones refers to the computation of the (local) macrotransition systems. Each model involves agent synthesis and polymerization so their complete transition systems are all infinite.

VII. Benchmarks

The implementation is integrated within an open-source static analyser [2]. In: http://www.di.ens.fr/~feret/local_traces, we provide the source code of some toy models (including those of Sect. II), pointers to the binaries of the analyser, and pointers to its handbook.

We have applied our analysis to three models in development: a model of the extracellular matrix of the protein TGFβ written by Nathalie Théret and Jean Coquet (Dyliss, Project TGFβsysbio, Plan Cancer 2014), a model of the Wnt signalling pathway, written by Héctor Francisco Medina Abarca (Fontana Lab, BigMechanisms DARPA Project), a model of the Ras signalling pathways, assembled by John Bachman and Benjamin Gyori (Sorger lab, BigMechanisms DARPA Project). The first two models have been assembled by hand, by inspection of the literature. The second model is using some scripts to refine the kinetics of rules according to some contextual information about the proteins. The third one has been designed in three steps: automatic natural language processing, automatic assembling into Kappa, and human curation, following the method that is described in [16].

The second model contains a protein that works as the one our second study, except that it contains five independent pairs of phosphorylation sites. Our analysis succeeds in deciphering the behaviour of this protein. In the benchmarks, we use macrotransition systems to extract the number of dead rules, i. e. the rules that will never be triggered, and the number of separating rules, i. e. the rule that may take a protein from a configuration a to another configuration b whereas there is no sequence of transitions that may take the protein from the configuration b back to the configuration a (such a transition is usually called non weakly reversible). Separating rules are computed by using Tarjan's strongly connected components algorithm [19] on local transition systems or macrotransition systems. Dead rules usually indicate that some part of the model is missing. Separating rules are also important. They usually come from some phosphorylation rules without corresponding dephosphorylation rules, or from some binding rules without corresponding unbinding rules. In such a case, the model must be completed accordingly, before investigating its dynamical behaviour. In Kappa, knowing whether a given pattern is reachable is not decidable [17]. By reduction to this problem, knowing whether a rule is dead and whether it is separating, are both undecidable.

Benchmarks for different versions of the models are given in Fig. 18.

VIII. Conclusion

Kappa [9] allows for the description of highly combinatorial systems of interactions between proteins. As a result, it is not always obvious to check the consistency of the models that are written in this language, or to get an overview of how these models work. So as to cope with this issue, we have proposed an abstract interpretation framework to over-approximate automatically the behaviour of each agent of a model, independently, by the means of a *local* transition system. Yet, we have noticed that these local transition systems may remain too combinatorial in the case of proteins with too many interaction sites.

Thus we have introduced a more abstract description of the local transition systems, inspired by simplicial systems. This latter description involves transitions between macrostates, as a symbolic representation of a set of transitions between microstates. This representation abstracts efficiently the interleaving order of commutative transitions in local transition systems. Interestingly, transitions between macrostates can be computed directly, without having to consider explicitly the underlying local transition systems.

We have used the fact that granularity of description of biochemical reactions in Kappa rule-based languages, as Kappa, provides a syntactic way to detect beforehand which event commutes and which macrostates should be considered.

Acknowledgments: We would like to thank the modellers, Nathalie Théret, Jean Coquet, Héctor Francisco Medina Abarca, Peter Sorger, John Bachman, and Benjamin Gyori. Their models provide the motivation for these works. We would like to deeply thank Pierre Boutillier, Vincent Danos,

Walter Fontana, Russ Harmer, Jean Krivine, and Jonathan Laurent, for their helpful feedbacks and nice suggestions for the tools.

REFERENCES

[1] Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival. A static analyzer for large safety-critical software. In Ron Cytron and Rajiv Gupta, editors, *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation 2003, San Diego, California, USA, June 9-11, 2003*, pages 196–207, 2003.

[2] Pierre Boutillier, Jérôme Feret, Jean Krivine, and Kim Quyên Lý. Kasim development homepage. http://kappalanguage.org.

[3] Thomas Chatain, Stefan Haar, Loïg Jezequel, Loïc Paulevé, and Stefan Schwoon. Characterization of reachable attractors using petri net unfoldings. In Pedro Mendes, Joseph O. Dada, and Kieran Smallbone, editors, *Computational Methods in Systems Biology - 12th International Conference, CMSB 2014, Manchester, UK, November 17-19, 2014, Proceedings*, volume 8859 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2014.

[4] Patrick Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoretical Computer Science*, 277(1–2):47–103, 2002.

[5] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Robert M. Graham, Michael A. Harrison, and Ravi Sethi, editors, *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*, pages 238–252. ACM, 1977.

[6] Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In Alfred V. Aho, Stephen N. Zilles, and Barry K. Rosen, editors, *Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages, San Antonio, Texas, USA, January 1979*, pages 269–282. ACM Press, 1979.

[7] Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, Jonathan Hayman, Jean Krivine, Christopher D. Thompson-Walsh, and Glynn Winskel. Graphs, rewriting and pathway reconstruction for rule-based models. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, volume 18 of *LIPIcs*, pages 276–288. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[8] Vincent Danos, Jérôme Feret, Walter Fontana, and Jean Krivine. Abstract interpretation of cellular signalling networks. In Francesco Logozzo, Doron A. Peled, and Lenore D. Zuck, editors, *Verification, Model Checking, and Abstract Interpretation, 9th International Conference, VMCAI 2008, San Francisco, USA, January 7-9, 2008, Proceedings*, volume 4905 of *Lecture Notes in Computer Science*, pages 83–97. Springer, 2008.

[9] Vincent Danos and Cosimo Laneve. Formal molecular biology. *TCS*, 325(1), 2004.

[10] Vijay D'Silva, Daniel Kroening, and Marcelo Sousa. Independence abstractions and models of concurrency. In Ahmed Bouajjani and David Monniaux, editors, *Verification, Model Checking, and Abstract Interpretation - 18th International Conference, VMCAI 2017, Paris, France, January 15-17, 2017, Proceedings*, volume 10145 of *Lecture Notes in Computer Science*, pages 151–168. Springer, 2017.

[11] Manuel Fähndrich and Francesco Logozzo. Static contract checking with abstract interpretation. In Bernhard Beckert and Claude Marché, editors, *Formal Verification of Object-Oriented Software - International Conference, FoVeOOS 2010, Paris, France, June 28-30, 2010, Revised Selected Papers*, volume 6528 of *LNCS*, pages 10–30. Springer, 2010.

[12] Lisbeth Fajstrup, Eric Goubault, and Martin Raußen. Detecting deadlocks in concurrent systems. In Davide Sangiorgi and Robert de Simone, editors, *CONCUR '98: Concurrency Theory, 9th International Conference, Nice, France, September 8-11, 1998, Proceedings*, volume 1466 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 1998.

[13] Jérôme Feret. Reachability analysis of biological signalling pathways by abstract interpretation. In *Computational Methods in Sciences and Engineering, 18th International Conference of Computational Methods in Sciences and Engineering ICCMSE 2007, Corfu, Greece, Sept 25-30, 2007, Proceedings*, volume 963.(2) of *American Institute of Physics Conference Proceedings*. American Institute of Physics, 2007.

[14] Jérôme Feret and Kim Quyên Lý. Local traces: An over-approximation of the behaviour of the proteins in rule-based models. In Ezio Bartocci, Pietro Liò, and Nicola Paoletti, editors, *Computational Methods in Systems Biology - 14th International Conference, CMSB 2016, Cambridge, UK, September 21-23, 2016, Proceedings*, volume 9859 of *Lecture Notes in Computer Science*, pages 116–131. Springer, 2016.

[15] Patrice Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems - An Approach to the State-Explosion Problem*, volume 1032 of *LNCS*. Springer, 1996.

[16] Benjamin M Gyori, John A Bachman, Kartik Subramanian, Jeremy L Muhlich, Lucian Galescu, and Peter K Sorger. From word models to executable models of signaling networks using automated assembly. *bioRxiv*, 2017.

[17] Peter Kreyßig. Chemical organisation theory beyond classical models: Discrete dynamics and rule-based models.

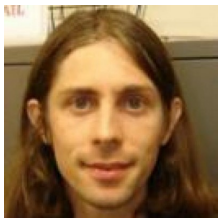
[18] Jonathan Laurent. Causal analysis of rule-based models of signaling pathways, 2015. Internship report.

[19] Robert Tarjan. Depth first search and linear graph algorithms. *SIAM Journal On Computing*, 1(2), 1972.

[20] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2), 1955.

[21] Glynn Winskel. Event structures. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1986.

[22] Jai You. Darpa sets out to automate research. *Science*, 347, 2015.



Jérôme Feret entered École normale supérieure in 1997, where he received his training in theoretical computer sciences, with a focus on semantics of programming languages and abstract interpretation. In 2005, he defended his Ph.D about abstract interpretation of mobile systems, under the supervision of Prof. Patrick Cousot. Jérôme Feret has been trained in applications of computer sciences in Systems Biology by Vincent Danos since 2006, and by Walter Fontana in 2007. He joined INRIA as a junior researcher in 2008.
His research interest is to extend the scope of abstract interpretations by developing new applications. He is one of the co-author of the ASTRÉE static analyser, that is used to prove the absence of run time errors in critical embedded C code. He is also developing static analysis, causality analysis, and model reduction tools within the Kappa modelling platform.

Kim Quyên Lý defended her Ph.D thesis about automated verification of termination certificates in 2014, under the supervision of Prof. Jean-François Monin and Frédéric Blanqui. She joined INRIA as a research engineer in 2014, funded by the DARPA grant number W911INF-14-10367. She is developing static analysis and model reduction tools for Kappa models