Graphs, Rewriting and Pathway Reconstruction for Rule-Based Models

Vincent Danos[3], Jérôme Feret[4], Walter Fontana[5], Russell Harmer[1], Jonathan Hayman[4,2], Jean Krivine[1], Chris Thompson-Walsh[2], and Glynn Winskel[2]

1 **CNRS & Université Paris-Diderot, Paris, France**
2 **Computer Laboratory, University of Cambridge, UK**
3 **LFCS, School of Informatics, University of Edinburgh, UK**
4 **LIENS (INRIA/ÉNS/CNRS), Paris, France**
5 **Harvard Medical School, Boston, Massachusetts**

Abstract

In this paper, we introduce a novel way of constructing concise causal histories (pathways) to represent how specified structures are formed during simulation of systems represented by rule-based models. This is founded on a new, clean, graph-based semantics introduced in the first part of this paper for Kappa, a rule-based modelling language that has emerged as a natural description of protein-protein interactions in molecular biology [1]. The semantics is capable of capturing the whole of Kappa, including subtle side-effects on deletion of structure, and its structured presentation provides the basis for the translation of techniques to other models. In particular, we give a notion of trajectory compression, which restricts a trace culminating in the production of a given structure to the actions necessary for the structure to occur. This is central to the reconstruction of biochemical pathways due to the failure of traditional techniques to provide adequately concise causal histories, and we expect it to be applicable in a range of other modelling situations.

1998 ACM Subject Classification F.1.1 Models of Computation, F.3.1 Specifying and Verifying and Reasoning about Programs, J.3 Biology and genetics, I.6.6 Simulation Output Analysis

Keywords and phrases concurrency, rule-based models, graph rewriting, pathways, causality

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2012.276

1 Introduction

Kappa [8] has emerged as a powerful tool in modelling biochemical systems, supporting sophisticated and efficient simulation [5] and static analysis [6] techniques. It is centered around *rules* that describe how links between sites on entities called agents are modified when local conditions on the link structure (or, more generally, the state of sites) are satisfied. In biological applications, agents typically represent proteins and links correspond to non-covalent associations between domains (sites) of proteins; rules then are intended to capture empirically sufficient conditions for modifications in the binding (or other) state of protein molecules. The use of rule-based approaches has the potential to make a profound impact in these fields [1], making it possible to analyze systems that would be intractable using traditional systems of ordinary differential equations.

Kappa has an intuitive graphical interpretation. In this paper, we formalise the structures involved and characterize the rewriting operation using a single-pushout (SPO) technique [13]. The aim is to develop a natural, stable foundation for Kappa, and the use of morphisms

32nd Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012).
Editors: D. D'Souza, J. Radhakrishnan, and K. Telikepalli; pp. 276–288

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

illuminates choices made in earlier work. The foundation will provide a basis for connections to existing work (such as [3, 9]) on graph rewriting. Critically, the semantics captures side-effects in the application of rules; these are important in the efficient and natural representation of complex systems, so the careful analysis of them underpins the provision of a rewriting semantics for the whole of Kappa. This is a significant progression from the rewriting semantics presented in [7], where, with the aim of developing a theory of dynamic restriction of reductions by type, it was appropriate to study a side-effect free form of Kappa, specifically one without external links and where only whole connected components could be deleted, that permitted a double-pushout (DPO) semantics. As we shall see, DPO rewriting could not have been used without restricting to a side-effect-free fragment of Kappa.

In the second half of the paper, we introduce forms of *trajectory compression*, novel causality analyses that have been implemented in KaSim, the Kappa simulator [14]. When a specified pattern is seen in the state during simulation, for example two of a particular kind of agent being linked together, they allow the automatic generation of a refined causal account of the rule applications that led to the production of the pattern, called by biologists its *pathway*. Traditional techniques fail to provide sufficiently concise histories, leaving in place rule applications that a biologist would not expect to see in a pathway; compression addresses this important problem. The forms of compression introduced progressively increase in their ability to remove actions that are irrelevant to the production of the pattern, for example being able to remove pairs of events that remove and subsequently re-introduce structure upon which the pattern depends. It should be emphasized that, though we present compression for Kappa, we see these as *general* operations. When a user is involved in verifying any kind of model based on rewriting and the model indicates that a specified kind of event of interest can occur, there is the potential to apply these forms of compression to provide the modeller with a concise diagnosis for the event's occurrence.

2 Kappa

Kappa models how structures called *mixtures* are affected by *rules*. A mixture describes the state of *sites* on entities called *agents*. Agents and sites are labelled; each agent has at most one site with each label, and a *signature* governs which sites are allowed on each type of agent. Sites can be *linked* to other sites and can also be tagged to record any other *internal properties* that hold. A natural level of abstraction for viewing biochemical pathways is to view proteins as agents, the signature lists the (functional) sites of each type of protein, and internal properties can be used for example to model the phosphorylation state of each site.

Rules are applied to transform mixtures. Each rule consists of three *site graphs* (patterns): a left-hand side, a domain of definition and a right-hand side. A rule can be applied when its left-hand side is *matched* in the mixture. The domain of definition is a sub-pattern of the left-hand side; anything matched by something in the left-hand outside the domain of definition is deleted by application of the rule. The right-hand side extends the domain of definition with new agents, links and properties to be added to the mixture.

Significantly, to reduce the number of rules that need to be written, the test pattern of a rule does does not need to specify the full state of all agents involved, but only a partial aspect of some agents; this is neatly summed-up as: *"Don't care? Don't write."* If whether the action can occur is not determined by the state of a site, the site does not have to occur in the test pattern. However, the application of a rule induces a reaction, *i.e.* a transition in which agents are fully specified, in the mixture.

Example rules and a sequence of reactions, called a *trajectory*, are presented in Figure 1.

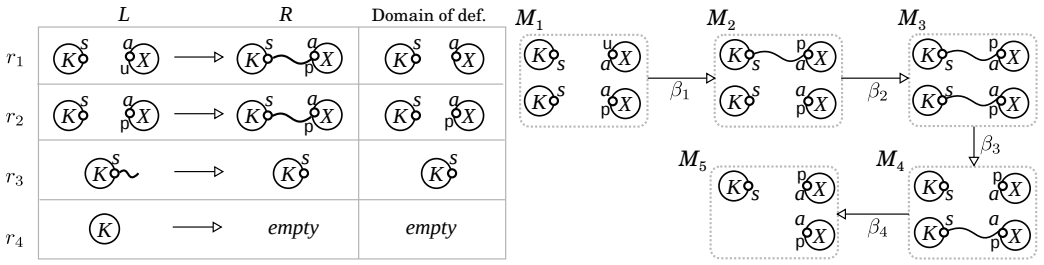
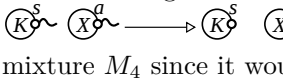


Figure 1 Example set of rules (left) and trajectory (right). Reaction β_1 is obtained by applying r_1 to the top two agents; β_2 by applying r_2 to the bottom two agents; β_3 by applying r_3 to the top-left agent; β_4 by applying r_4 to the lower-left agent.

We adopt the convention of drawing agents as large circles (labelled K and P), sites as small circles (labelled a and s) and using a sans-serif font to indicate the internal properties that hold at sites (u for unphosphorylated and p for phosphorylated).

Since rules do not have to fully specify the state of all agents involved, rule applications can have *side-effects*: changes to the mixture outside the image of the test pattern. For example, the rule r_4 in Figure 1 deletes a K-agent without reference to any of its sites; as seen in the reaction labelled β_4 in the trajectory, its application has the side-effect of removing the link to the lower X-agent. As mentioned in the introduction, Kappa allows such actions to allow the efficient representation of systems, and this will motivate our use of SPO rewriting to capture these highly intricate operations. This is in contrast to rewriting as described in [7], which considered a side-effect free fragment of Kappa, in which, for example, any deleted agent has to have all its sites specified in the rule.

To allow further compact representation, test patterns do not need to specify both ends of links: links can be one-ended, signifying the existence of a link to *some* agent; we call such links *external* links. Rule r_3 which allows reaction β_3 shows such a link being used to specify a link to be deleted. External links in a pattern can additionally specify the type of agent to which the link must connect and, optionally, the identifier of the site. In this paper, for conceptual clarity, we shall take the view that the right-hand side of an applied rule should be matched in the produced mixture. Some care will therefore be needed to manage the interaction of side-effects and external links; we shall do so by requiring the images of external links under matchings to connect to agents outside the image of the matching. Thereby, for example, we rule out application of the rule  (with domain of definition equal to the right-hand side) to the mixture M_4 since it would specify that the link connecting the lower two agents were simultaneously to be destroyed and preserved. The techniques presented in this paper are, however, completely amenable to other design decisions such as choosing to favour deletion in these cases, and the more abstract framework referred-to in the conclusion is of use when studying them.

We briefly remark that, in full Kappa, rules can be equipped with *rate constants*: values that influence how likely it is that an occurrence of a rule's test pattern in the mixture leads to application of the rule. In this paper, however, it will not be necessary to consider any stochastic aspects of the calculus.

3 Σ-graphs and morphisms

Graphs with a given signature, Σ-*graphs*, shall play a central rôle in the semantics for the Kappa that we now proceed to give: they shall include *mixtures* and represent patterns (*site graphs*), and can be used to represent types (*contact graphs*).

▶ **Definition 1.** A *signature* is a 4-tuple $\Sigma = (\Sigma_{\mathsf{ag}}, \Sigma_{\mathsf{st}}, \Sigma_{\mathsf{ag-st}}, \Sigma_{\mathsf{prop}})$, where Σ_{ag} is a set of *agent types*, Σ_{st} is a set of *site identifiers*, $\Sigma_{\mathsf{ag-st}} : \Sigma_{\mathsf{ag}} \to \mathcal{P}_{\mathsf{fin}}(\Sigma_{\mathsf{st}})$ is a *site map*, and Σ_{prop} is a set of *internal property identifiers*.

The set of agent types Σ_{ag} consists of labels to describe the nature of the agents of interest, for example the set of *kinds* proteins to be considered, and we use capital letters $A, B, A', \ldots$ to range over them. The set Σ_{st} represents the set of labels that can appear to identify sites on agents and is ranged over by $i, j, i', \ldots$. The function $\Sigma_{\mathsf{ag-st}}$ specifies the site identifiers that can be present on agents: any site on an agent of type A must be labelled with an identifier in $\Sigma_{\mathsf{ag-st}}(A)$. Finally, the set Σ_{prop} indicates the set of internal properties that sites might possess; for example, $\{\mathsf{u}, \mathsf{p}\}$ to represent 'unphosphorylated' and 'phosphorylated'.

As discussed in the previous section, there are three forms of external link. The first form, written $-$, indicates just that the link connects to some other site. The second form, written A for some $A \in \Sigma_{\mathsf{ag}}$, indicates that the link connects to some site on some agent of type A. The final form, written (A, i) for some $A \in \Sigma_{\mathsf{ag}}$ and $i \in \Sigma_{\mathsf{ag-st}}(A)$, indicates that the link connects to site i on some agent of type A. The set of all possible external link labels is defined to be $\mathsf{Ext} = \{-\} \cup \Sigma_{\mathsf{ag}} \cup \{(A, i) \ : \ A \in \Sigma_{\mathsf{ag}} \ \& \ i \in \Sigma_{\mathsf{ag-st}}(A)\}$.

▶ **Definition 2.** A Σ-graph comprises a set $\mathcal{A}$ of *agents*, an *agent type* assignment $\mathsf{type} : \mathcal{A} \to \Sigma_{\mathsf{ag}}$, a set $\mathcal{S}$ of *sites* satisfying $\mathcal{S} \subseteq \{(n, i) \ : \ n \in \mathcal{A} \ \& \ i \in \Sigma_{\mathsf{ag-st}}(\mathsf{type}(n))\}$, a symmetric *link* relation $\mathcal{L} \subseteq (\mathcal{S} \cup \mathsf{Ext})^2 \setminus \mathsf{Ext}^2$, and a *property* set p_k for each $k \in \Sigma_{\mathsf{prop}}$ satisfying

$$p_k \subseteq \{(n, i) \ : \ n \in \mathcal{A} \ \& \ i \in \Sigma_{\mathsf{ag-st}}(\mathsf{type}(n))\}$$

Note that any agent has at most one site with any given identifier: an agent n cannot have two sites identified i. In patterns, we will represent the absence of any link on a site through the site being in $\mathcal{S}$ but not occurring in any element of the link relation. Finally, the set p_k represents the set of sites that have internal property k. It is not in general the case that $p_k \subseteq \mathcal{S}$. The intuition is that we wish the set $\mathcal{S}$ to be the set of sites for which we represent knowledge of *linkage*. In a mixture, this will be all sites, but in a pattern it will be the sites that we require either to be or not to be linked. If we do not care about a site's link state in a pattern, it will not be in $\mathcal{S}$, but we may still wish to modify or test its internal properties, so it may be in the set p_k for some k.

It is useful to introduce a few notational conventions. For a Σ-graph G, we write $\mathcal{A}_G$ for its set of agents, type_G for its typing function, $\mathcal{S}_G$ for its set of link sites, $\mathcal{L}_G$ for its link relation and $p_{k,G}$ for the set of sites satisfying property k. We use m, n to range over agents and x, y to range over elements of $\mathcal{S} \cup \mathsf{Ext}$.

Mixtures, representing the state to which rules are applied, and *site graphs*, representing patterns, are forms of Σ-graph. In particular, site graphs have no links that immediately loop back to the same site and have at most one link from any site. Mixtures additionally specify the link state of all sites and have no external links.

▶ **Definition 3.** A *site graph* is a Σ-graph such that its link relation $\mathcal{L}$ is *irreflexive* and if $((n, i), x) \in \mathcal{L}$ and $((n, i), y) \in \mathcal{L}$ then $x = y$, for all n, i, x and y. A *mixture* is a site graph that additionally satisfies $\mathcal{L} \subseteq \mathcal{S} \times \mathcal{S}$ and $\mathcal{S} = \{(n, i) \ : \ n \in \mathcal{A} \ \& \ i \in \Sigma_{\mathsf{ag-st}}(\mathsf{type}(n))\}$.

It should be noted that Kappa as presented here is slightly different from past presentations in that we allow a *set* of properties to hold at a site in a mixture instead of precisely one. This streamlines the coming account of rewriting, makes no difference to expressivity, and the old definition could be handled with a few straightforward minor changes.

3.1 Homomorphisms and partial morphisms

Homomorphisms between Σ-graphs are structure-preserving functions from the agents of one Σ-graph to the agents of another. They preserve structure in the sense of preserving the presence of sites, preserving properties held on sites and ensuring that if there is a link on a site, the corresponding site on the image of the agent has a link higher in the *link information order*. Given a typing function type, this is the least reflexive, transitive relation $\leq_{\mathsf{type}}$ s.t. for all $A \in \Sigma_{\mathsf{ag}}$ and $i \in \Sigma_{\mathsf{ag-st}}(A)$ and n s.t. $\mathsf{type}_G(n) = A$: $\quad - \leq_{\mathsf{type}} A \leq_{\mathsf{type}} (A, i) \leq_{\mathsf{type}} (n, i)$

▶ **Definition 4.** A *homomorphism* of Σ-graphs $h : G \to H$ is a (total) function on agents $h : \mathcal{A}_G \to \mathcal{A}_H$ satisfying:

- $\mathsf{type}_G(n) = \mathsf{type}_H(h(n))$ for all $n \in \mathcal{A}_G$
- if $(n, i) \in \mathcal{S}_G$ then $(h(n), i) \in \mathcal{S}_H$
- $\{(h(n), i) \ : \ (n, i) \in p_{k,G}\} \subseteq p_{k,H}$ for all $k \in \Sigma_{\mathsf{prop}}$
- if $((n, i), x) \in \mathcal{L}_G$ then there exists y s.t. $((h(n), i), y) \in \mathcal{L}_H$ and $\hat{h}(x) \leq_{\mathsf{type}_H} y$, where $\hat{h}(m, j) = (h(m), j)$ for any $(m, j) \in \mathcal{S}_G$ and $\hat{h}(x) = x$ for any $x \in \mathsf{Ext}$.

It is worth remarking that we could have taken homomorphisms to be functions on agents, sites, links and properties as in [7]. Since in this paper we will primarily focus on site graphs, no difference arises from using this simpler definition.

▶ **Definition 5.** A partial morphism $h : G \rightharpoonup H$ between site graphs G and H is a span $G \hookleftarrow D \xrightarrow{h} H$ where h is a homomorphism and D is a site graph that is a subgraph of G, *i.e.*: $\mathcal{A}_D \subseteq \mathcal{A}_G$ and $\mathcal{S}_D \subseteq \mathcal{S}_G$ and $\mathcal{L}_D \subseteq \mathcal{L}_G$ and $p_{k,D} \subseteq p_{k,G}$ for all $k \in \Sigma_{\mathsf{prop}}$.

We will write $\mathrm{dom}(h)$ for the site graph representing the domain of definition of a partial morphism h, and allow ourselves to write $n \in \mathrm{dom}(h)$ to mean that n is in its set of agents, $(n, i) \in \mathrm{dom}(h)$ to mean that (n, i) is in its set of link sites, and so on. We additionally write $(n, i) \in \mathrm{dom}_{\mathsf{prop},k}(h)$ if (n, i) is in its set representing the property k.

Partial morphisms $f : G \rightharpoonup H$ and $g : H \rightharpoonup K$ compose in the usual way, with the domain of definition of their composition $\mathrm{dom}(g \circ f)$ containing elements (agents/sites/links) of $\mathrm{dom}(f)$ such that their image under f is in $\mathrm{dom}(g)$. This corresponds to taking a pullback of the homomorphism $f_0 : \mathrm{dom}(f) \to H$ against $\mathrm{dom}(g) \hookrightarrow H$ in the category Σ-**Site** of site graphs with homomorphisms between them. We write Σ-$\mathbf{Site}_*$ for the category of site graphs connected by partial morphisms.

4 Rewriting

Matchings will be used to express how patterns (site graphs) are found in mixtures. There are three key aspects. Firstly, distinct agents in patterns must match distinct agents in the mixture. Secondly, sites with no link in the pattern should have no link in the mixture, allowing patterns to express the absence of links on sites. Finally, as discussed in Section 2, the image of an external link in the pattern cannot connect to any agent in the image of the pattern. We write $e : G \rightarrowtail H$ to signify that e is a matching.

▶ **Definition 6.** A *matching* of the site graph G in H is an injective homomorphism $e : G \to H$ such that, for all $(n, i) \in \mathcal{S}_G$:

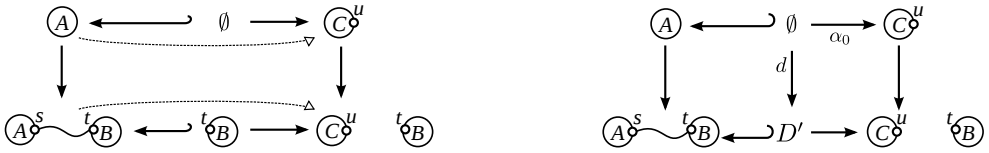


Figure 2 An SPO application (left) that cannot be a DPO application (right)

- if there exists y s.t. $((e(n),i),y) \in \mathcal{L}_H$ then there exists x such that $((n,i),x) \in \mathcal{L}_G$, and
- if there exist m and j s.t. $((e(n),i),(e(m),j)) \in \mathcal{L}_H$ then $((n,i),(m,j)) \in \mathcal{L}_G$.

Every rule is associated with a single *action map*, a special kind of partial morphism, to describe its effect. Via matchings, action maps specify what is deleted from the mixture when they are applied and what is added: anything matched by the left-hand side but not in the domain of definition is deleted by application of the rule, and anything added to the domain of definition in the progression to the right hand side is created.

Action maps have three important aspects. Firstly, the right leg of the span is injective, so rules cannot 'merge' agents together. Secondly, we wish it always to be the case that if the left-hand side of a rule matches part of the mixture, following application of the rule to that part, it matches the right-hand side of the rule. Finally, we wish to ensure that action maps preserve the property of being a mixture. Property (2) below ensures that sites are not added to or deleted from existing agents: we might otherwise introduce a site onto a preserved agent that failed to match due to the presence/absence of links, or fail to generate a mixture by deleting a site from a preserved agent. Property (3) ensures that any preserved external link is not promoted up the link information order; this is again to ensure that the right-hand side of the pattern will be consistent with the obtained mixture. Property (4) ensures that any link that is created is not an external link. Property (5) ensures that if an agent is created then so are all the sites consistent with the signature.

▶ **Definition 7.** An *action map* $\alpha : L \rightharpoonup R$ is a partial morphism of site graphs s.t.

1. α is partial injective, i.e. for all $m, n \in \mathrm{dom}(\alpha)$, if $\alpha(n) = \alpha(m)$ then $n = m$
2. for any $n \in \mathrm{dom}(\alpha)$, $\quad (n,i) \in \mathcal{S}_L$ iff $(\alpha(n),i) \in \mathcal{S}_R$ iff $(n,i) \in \mathrm{dom}(\alpha)$
3. for any $x \in \mathsf{Ext}$ and $((m,i),x) \in \mathrm{dom}(\alpha)$, $\quad ((\alpha(m),i),x) \in \mathcal{L}_R$
4. if $((m,i),x) \in \mathcal{L}_R$ and $\nexists n, y$ s.t. $\alpha(n) = m$ and $((n,i),y) \in \mathrm{dom}(\alpha)$ then there exists $(p,j) \in \mathcal{S}_R$ s.t. $x = (p,j)$.
5. if $m \in \mathcal{A}_R$ and $m \notin \mathrm{image}(\alpha)$ then $(m,i) \in \mathcal{S}_R$ for all $i \in \Sigma_{\mathsf{ag-st}}(\mathsf{type}(m))$.

The effect of applying a rule to a mixture is characterized abstractly as the following pushout in Σ-**Site**$_*$; this result will be used extensively in the following section on compression and causality. The result is dependent on the definitions of matchings and action maps since, in general, Σ-**Site**$_*$ does not have pushouts.

▶ **Theorem 8.** *Given an action $\alpha : L \rightharpoonup R$ and a matching $e : L \rightarrowtail M$, there is a pushout in the category Σ-**Site**$_*$ as follows. Furthermore, N is a mixture, u is a matching and β is an action map.*

$$\begin{array}{ccc} L & \xrightarrow{\alpha} & R \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle u} \\ M & \xrightarrow{\beta} & N \end{array}$$

We now give a simple example showing the necessity, due to side-effects, of SPO rather than DPO rewriting. The SPO characterization gives the application to the left of Figure 2

where the deletion of an A-agent has the side-effect of deleting a link. DPO rewriting allows a rule with an action map $\alpha : L \rightarrowtriangle R$ with domain D and right leg $\alpha_0 : D \to R$ to be applied to M to yield M' if there is a morphism $d : D \to D'$ such that M is the pushout of $D \hookrightarrow L$ against d and M' is the pushout of $\alpha_0 : D \to R$ against d. Hence, for DPO rewriting to produce the same derivation, there would have to exist D' that would allow both the squares in the diagram to the right of Figure 2 to be pushouts, which is clearly impossible.

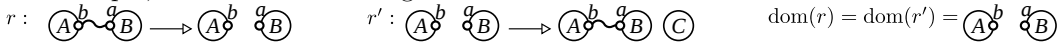
5 Trajectory compression

The pushout theorem described in the previous provides a foundation for understanding how KaSim, the Kappa simulator, simulates biochemical systems. Given an initial mixture and a set of rules described by action maps, KaSim rewrites mixtures to obtain a *trajectory*, a finite sequence of mixtures connected by action maps. Each step in the sequence is justified by a pushout in which each action map α_i is associated with a rule r_i:

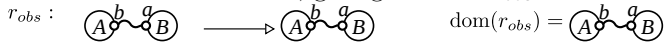
$$M_1 \xrightarrow{\beta_1} M_2 \cdots\cdots M_n \xrightarrow{\beta_n} M_{n+1} \quad \text{where} \quad \begin{array}{ccc} L_i & \xrightarrow{\alpha_i} & R_i \\ {\scriptstyle e_i}\downarrow & & \downarrow{\scriptstyle u_i} \\ M_i & \xrightarrow{\beta_i} & M_{i+1} \end{array}$$

One of the key practical uses of a trajectory is to form a causal account called a *pathway* of how specified patterns (site graphs) called *observables* come to exist in the mixture. The occurrence of an observable can be regarded simply as the application of a rule that tests if a given pattern is matched, making no change to the mixture. A natural first approach is therefore based on consideration of the independence of consecutive actions, called sequential independence in graph rewriting, the intention being to draw upon the well-developed theory of independence models for concurrency. Approximately, two consecutive rule applications are independent if they overlap only on their preserved parts. Critically, any consecutive pair of independent rule applications can be permuted, applying the same rules to the same agents, to yield another valid trajectory to the same ultimate state. Using independence, we may derive from the trajectory a partial order of causal dependence: a rule application causally depends on the rule applications that it always occurs after, under any sequence of permutations of consecutive independent rule applications. This construction forms part of the adjunction between event structures and Mazurkiewicz trace languages in [15] . The pathway of a given observable in a trajectory is the restriction of this partial order to the rule applications upon which the rule application representing the observable causally depends. In general, since the pathway is constructed by removing unnecessary rule applications from the trajectory, we call it the *Mazurkiewicz compression* of the original trace.

In practice, however, this classical approach produces pathways that contain rule applications that a biologist would normally ignore, leaving in place sequences of rule applications that have no combined effect from the perspective of the ultimate production of the observable. For example, consider the following two rules:



Application of r to a mixture exactly the same as its left-hand side followed by application of r' first destroys the link and then re-creates it, in the process creating a C agent. Let the observable be that sites a and b are linked, giving the rule r_{obs}:



There is a trajectory consisting of r, then r' and then finally r_{obs}. Considering consecutive rule applications, the observable cannot occur immediately prior to the application of r', and r' cannot be applied until r has applied, so no Mazurkiewicz compression of this trajectory

with respect to the observable can take place. Mazurkiewicz compression therefore cannot compress to show that the observable is matched by the same part of the graph in the initial mixture. In general, it leaves in place sequences of events that needlessly cycle in removing and adding structure tested by the observable.

The weakness of Mazurkiewicz compression is addressed by *weak compression*, a novel technique that draws its power from the ability simply to test if rules can be applied to the same agents at *arbitrary* prior points in the given trajectory whilst ensuring that the observable is eventually matched in the same way. This is in contrast to the consideration of independence of rules only with their *immediately* prior rule application. Though in this paper we only discuss its application to Kappa, there is potentially great scope for applying it in other rewriting settings such as traditional graph rewriting and multiset rewriting.

Finally, we introduce *strong* compression, which lifts the requirement above that the rule generating the action being compressed is applied to same agents. Though this seemingly loses a significant amount of fidelity in trajectories, in many situations the particular identity of the agents to which rules are applied is not of significance. For example, a kinase may be required to be linked to a protein for a particular sequence of phosphorylation rules to be applied (this is called *processive phosphorylation*); we often wish to disregard the possibility of the link being broken mid-sequence and then some other instance of the same kind of kinase re-establishing the link for the process to continue. It is also worth noting that strongly-compressed traces of a given observable can be extracted *statically*, *i.e.* without the need to run any simulation.

Though in this paper we focus on their foundations, both weak and strong compression have been implemented using constraint satisfaction techniques, weak in KaSim and both in the earlier *simplx* simulator, and they have been found to be of real value when analysing realistic models. The syntax to invoke them and further examples are available at [14].

5.1 Weak compression

Define the equivalence $\sim_{\mathsf{ag}}$ to relate partial morphisms $f, g : G \to H$ iff $\mathcal{A}_{\mathrm{dom}(f)} = \mathcal{A}_{\mathrm{dom}(g)}$ and $\forall n \in \mathcal{A}_{\mathrm{dom}(f)} : f(n) = g(n)$. Partial morphisms are $\sim_{\mathsf{ag}}$-related iff they are between the same site graphs and are equally defined on agents, though not necessarily on sites, properties or links.

Suppose that we have an action map $\beta : M \twoheadrightarrow M_0$ and there is a rule r_0 with action map $\alpha_0 : L_0 \twoheadrightarrow R_0$ and a matching $e_0 : L_0 \rightarrowtail M_0$. The application of r_0 to M_0 can be *weakly compressed* to occur prior to the action β, which may represent the *composition* of action maps induced by rule applications, if there is a matching $e'_0 : L_0 \rightarrowtail M$ such that $\beta \circ e'_0 \sim_{\mathsf{ag}} e_0$. This implies that r_0 can be applied to the same agents (tracking their identity through β) in M as it was in M_0 through the original matching.

▶ **Lemma 9.** *Let $\beta : M \twoheadrightarrow M_0$ and $\alpha_0 : L_0 \twoheadrightarrow R_0$ and there be a matching $e_0 : L_0 \rightarrowtail M_0$. Let M_1 be a pushout of α_0 against e_0 with pushout morphisms $\beta_0 : M_0 \twoheadrightarrow M_1$ and $u_0 : R_0 \rightarrowtail M_1$. If there is a matching $e'_0 : L_0 \rightarrowtail M$ such that $\beta \circ e'_0 \sim_{\mathsf{ag}} e_0$ then, letting M'_0 be the pushout of α_0 against e'_0 with pushout morphisms $\beta'_0 : M \twoheadrightarrow M'_0$ and $u'_0 : R_0 \rightarrowtail M'_0$, there is a morphism $\gamma_0 : M'_0 \to M_1$, unique up to $\sim_{\mathsf{ag}}$, such that $\gamma_0 \circ \beta'_0 \sim_{\mathsf{ag}} \beta_0 \circ \beta$ and $\gamma_0 \circ u'_0 \sim_{\mathsf{ag}} u_0$. Furthermore, γ_0 is an action map.*

The situation can be summarized as in the left of Figure 3, drawing the pushouts M'_0 and M_1 described in Theorem 8. Note that the map γ_0 might not be derivable as a trajectory of rules in the given Kappa system. It describes the *residual* of the compression: an action

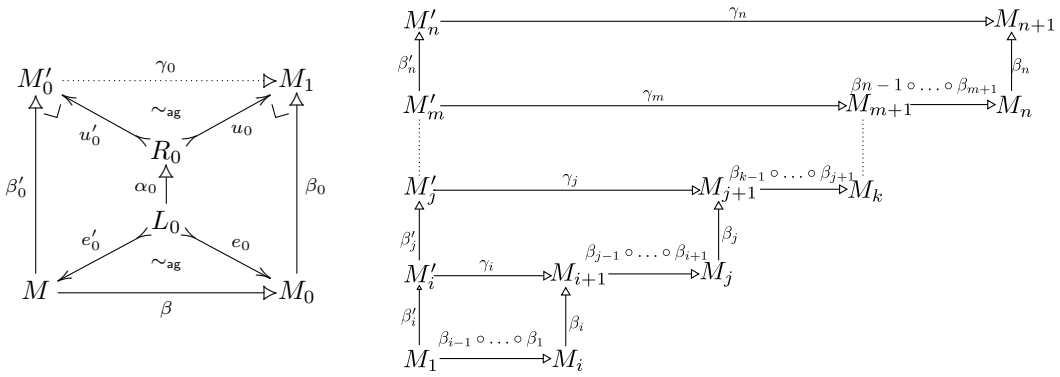


Figure 3 Steps of compression. Left: A single step of compression. Right: Multiple steps of compression, each justified by a single step of compression as drawn on the left.

summarizing the effect of the intermediate actions on the mixture formed by moving the application of the action map $\alpha_0 : L_0 \rightarrowtail R_0$ forward, excluding the modified part.

If $\beta \circ e_0' = e_0$ as opposed to $\beta \circ e_0' \sim_{\mathsf{ag}} e_0$, we call the compression *simple*. All permutations of independent individual actions will be simple, though the discussion following the example at the end of this section will explain why weak compression is more often of interest. In the case of simple compressions, $\sim_{\mathsf{ag}}$ in Lemma 9 can be replaced by equality, and the partial morphism γ_0 exists as a consequence of the left-hand pushout.

Suppose that we are given a trajectory $M_1 \xrightarrow{\beta_1} \cdots \xrightarrow{\beta_{n-1}} M_n \xrightarrow{\beta_n} M_{n+1}$ derived as shown at the beginning of Section 5 and that we wish to produce a causal account for the last action, $\beta_n : M_n \to M_{n+1}$. Intuitively, the goal is to remove as many rule applications as possible from this original trajectory, leaving in place β_n, to yield a valid trajectory from M_1 where each of the rule applications that are selected to remain involve matching the same agents as in the original trajectory. The standard notions of independence on the compressed trajectory can then be applied to obtain a pathway.

Formally, a set of indices $\{i, j, k, \ldots, m\} \subseteq \{1, \ldots, n-1\}$, taking $i < j < k < \ldots < m$, represents a weak compression if there is a sequence of compression steps as drawn in the right of Figure 3. The first step is generated by a matching $e_i' : L_i \to M_1$ commuting (on agents) with the original matching through the composition of the prior rule applications, *i.e.* $e_i \sim_{\mathsf{ag}} \beta_{i-1} \circ \ldots \circ \beta_1 \circ e_i'$. This generates a rule application $\beta_i' : M_1 \to M_i'$ and a residual $\gamma_i : M_i' \to M_{i+1}$. Each later step requires there to exist a matching that commutes (on agents) through the composition of the prior rule applications and the previous residual. For example, for j, there must exist a matching $e_j' : L_j \to M_i'$ such that $\beta_{j-1} \circ \ldots \circ \beta_{i+1} \circ \gamma_i \circ e_j' \sim_{\mathsf{ag}} e_j$; this generates a rule application $\beta_j' : M_i' \to M_j'$ and a residual $\gamma_j : M_j' \to M_{j+1}$. Finally, we require a single step of compression to establish that there is an application of the same rule as that generating β_n in M_m', with the matching being unchanged on agents.

If $\{i, j, k, \ldots, m\}$ is a minimal (with respect to set inclusion) representation of a weak compression, the trajectory $M_1 \overset{\beta_i'}{\rightarrowtail} M_i' \cdots \overset{\beta_m'}{\rightarrowtail} M_m' \overset{\beta_n'}{\rightarrowtail} M_n'$ is *maximally weak compressed* with respect to the rule application $M_n \overset{\beta_n}{\rightarrowtail} M_{n+1}$.

▶ **Example 10.** Consider the following site graph G_0:

This represents an observable matched when there are two X-type agents with no link at their a sites and, on at least one, a is phosphorylated.

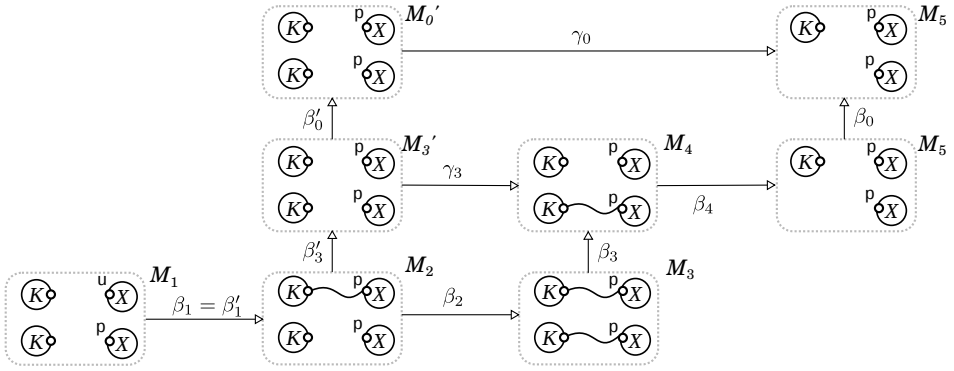


Figure 4 An example of weak compression. Sites on X are a and on K are s.

Returning to the Kappa rules and trajectory in Figure 1, this matches the mixture M_5. Let the matching $e_0 : G_0 \rightarrowtail M_5$ take the left X in G_0 to the top X in M_5 and the right X in G_0 to the bottom X in M_5. Weak compression can be applied as shown in Figure 4 to compress the trajectory $M_1 \xrightarrow{\beta_1} M_2 \xrightarrow{\beta_2} M_3 \xrightarrow{\beta_3} M_4 \xrightarrow{\beta_4} M_5 \xrightarrow{\beta_0} M_5$ where β_0 is the action testing the observable G_0. The residual map γ_3 records the creation of a link between the bottom two agents. The domain of definition of γ_0 is equal to M_4 with the lower K-agent removed, but otherwise γ_0 acts as the identity on agents. The result is a maximally compressed trajectory with the sequence of rule applications β'_1, β'_3 and β'_0, showing that the pattern can be matched in the same way as it was for the original trajectory simply following application of the rule r_1 to phosphorylate the site a on the top X, and r_3 to remove the created link. We have 'compressed' the trajectory by removing the actions that unnecessarily affected the bottom X-agent. Mazurkiewicz compression could not have been applied to this trajectory since the action that tests the observable is not independent of the rule application that removes the link from the lower X-agent, which itself is not independent of the rule application that creates this link.

In fact, the example above shows a simple compression. Whilst, as we have seen, simple compression allows a rule application to be pushed back along a trajectory that temporarily adds and then removes links or properties that prevent application of the rule, it does not allow the event to be pushed back if links or properties are temporarily *removed*. This asymmetry is addressed by requiring commutation up to $\sim_{\mathsf{ag}}$ in *weak* compression.

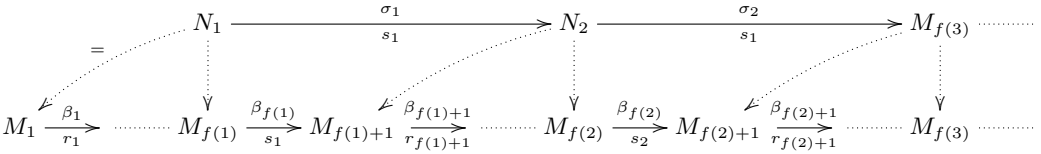
We conclude this section by noting that, in contrast with Mazurkiewicz compression, there are trajectories that have more than one maximal weak compression. For example, recalling the rules in Figure 1, from an initial mixture where an X-agent is linked to a K-agent, r_3, r_2 and r_4 could be applied in sequence followed by an observable testing that the site a on X is unlinked. Both the application of r_3 followed by the observable and the application of r_4 followed by the observable are maximal weak compressions.

5.2 Strong compression

Weak compression tests whether rules can be applied to act on the same agents at earlier points in the given trajectory. Strong compression relaxes this requirement: it asks whether a given rule in a trajectory can be applied through *any* matching; the rule does not have to be applied to the same agents. Though seemingly a very strong notion, as mentioned at the beginning of Section 5, it is appropriate in situations where we wish to ignore the identity of agents, for example in pathways for processive phosphorylation.

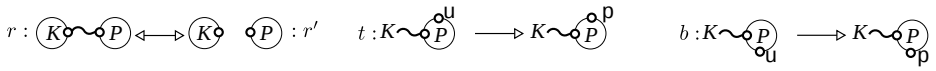
We begin by saying that a sequence of rules $s_1 \dots s_m$ is *realisable* from mixture N_1 if there is a trajectory $N_1 \overset{\sigma_1}{\to} N_2 \overset{\sigma_2}{\to} \dots \overset{\sigma_m}{\to} N_{m+1}$ and the rule applied to generate σ_i is s_i.

Consider a trajectory $M_1 \overset{\beta_1}{\to} M_2 \overset{\beta_2}{\to} \dots \overset{\beta_n}{\to} M_{n+1}$ where the rule applied to generate β_i is r_i. A *strong compression* with respect to the rule application $\beta_n : M_n \to M_{n+1}$ is a sequence $s_1 \dots s_m$ that is realisable from M_1 for which there is a monotone injection $f : \{i \ : \ 0 < i \leq m\} \to \{j \ : \ 0 < j \leq n\}$ such that $s_i = r_{f(i)}$ and $f(m) = n$. The compression is *maximal* if no rule can be removed from the sequence to yield a shorter strong compression. Labelling the action maps with the rules that generate them, we obtain:

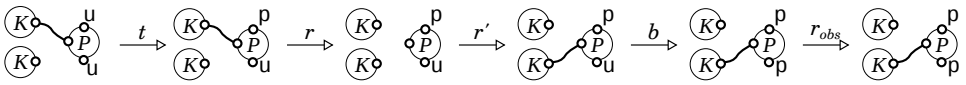


Any simple or weak compression is a strong compression, in which case the earlier constructions yield action maps for the dotted arrows. In the case of simple compression, the dotted areas commute up to equality, whereas they commute up to $\sim_{\mathsf{ag}}$ for weak compression

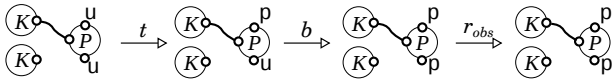
▶ **Example 11.** The following example shows how strong compression can be applied to disregard the identity of kinase agents in processive phosphorylation. In the following three rules, for clarity we have omitted site identifiers. The first is a reversible rule, so it can be applied in either direction, representing a kinase binding to/unbinding from a protein. The second and third rules represent the phosphorylation of upper and lower sites on the protein.



Taking the observable to be a P-type agent with upper and lower sites phosphorylated, we have the following trajectory (labelling reactions only with the rules that give rise to them).



Strong compression can be applied to this to obtain a trajectory consisting of rules t and b without the intervening r' and r events, where now only one of the two kinases acts.



6 Related work and conclusion

In this paper, we have seen how a semantics based on SPO graph rewriting [13] can be given to Kappa through the careful specification of various kinds of morphism. We then studied forms of trajectory compression, novel techniques that used the SPO characterization of rewriting and underlie the automatic reconstruction of biochemical pathways. This was motivated by traditional techniques based on the permutation of consecutive independent rule applications providing causal accounts with too much redundant information.

The necessity of an SPO instead of DPO semantics, due to Kappa allowing side-effectful actions, was shown in Section 4. It is worth remarking on the related work in [7] which developed a theory of dynamic restriction by type, where reduction is restricted to graphs over a given contact graph. There, a side-effect-free fragment of Kappa was considered, ruling out side effects by not having external links and only allowing rules to delete fully-specified connected components, and was shown to permit a DPO semantics.

The SPO result and characterization of compression provide an important concrete foundation for a number of areas of further research. Firstly, though there has not been space to present it here, the work in [7] on dynamic restriction translates to the SPO semantics. The analogue of the adjunction representing change of contact graph presented there, with the presence of external links introduced in this paper, can furthermore be used as a foundation for understanding *views*, special kinds of graph that are important in techniques based on static analysis for computing an approximation of the set of reachable connected components (complexes) [6]. We also intend to study how this adjunction can be used to describe techniques for providing a flexible degree of context-sensitivity in the abstraction of information flow that is used for reducing quantitative semantics [4]. Additionally, demonstrating the generality of the approach to rewriting, in [10] it is shown how it supports the addition of *regions* to site graphs to represent membranes. More abstractly, it would be interesting to determine whether sesqui-pushout rewriting [3] could fruitfully be applied to model Kappa, and to connect to the wider graph-rewriting literature, for example by comparing to [9]. Another potentially interesting connection that could be made is to the framework recently introduced in [11], which shows how negative application conditions in DPO rewriting can be specified using 'open maps'. Negative application conditions should allow us to understand abstractly constraints of the form seen in matchings, where we require the image of any unlinked site under a matching to itself be unlinked. Finally on rewriting, the work here suggests a more general framework in which to characterize pushouts of partial maps, where we represent how the preservation of elements such as links depends on the preservation of other elements such as their corresponding agents. This can be used to tease-apart the concrete proofs presented here and reveals the robustness of the pushout result when the 'externality' constraint on matchings is relaxed.

On compression, though we chose to formulate the technique for Kappa, we see such techniques for automatically providing more concise causal accounts of rule applications as being applicable much more widely in the verification of concurrent and distributed systems. Towards this, in future work we shall give an abstract framework to describe compression in any category of structures with distinguished forms of action and matching morphisms alongside a full description of its algorithmic techniques. This shall be used to show how it translates equally-well to other models such as Petri nets, traditional graph rewriting and other models for biochemical pathways such as BioNetGen[2], which differs from Kappa in allowing rules to have non-local tests (though, in terms of complexity, there is an inherent cost in dealing with these). The more abstract framework for compression may also provide a useful setting for a comparison with [12], which can be seen as defining when a single rule application can be permuted with a number of prior rule applications in the style of Mazurkiewicz compression. This would provide a form of compression between Mazurkiewicz and weak compression, in which the residual represents the composition of a number of rule applications. It is worth noting that this will be less powerful than weak compression since it will leave in place rule applications in asymmetric conflict with production of the observable. Finally, we are currently studying a framework that allows the description of other more sophisticated forms of compression lying between weak and strong, for example compression allowing the identity only of particular types of agent to be disregarded.

Acknowledgments: JH and GW gratefully acknowledge the support of the ERC through the Advanced Grant *ECSYM*, JF and JH the support of the ANR *AbstractCell* Chair of Excellence and JK the ANR Avenir *ICEBERG*.

References

1 J. A. Bachman and P. Sorger. New approaches to modeling complex biochemistry. *Nature Methods*, 8(2):130–131, 2011.

2 M. L. Blinov, J. Yang, J. R. Faeder, and W. S. Hlavacek. Graph theory for rule-based modeling of biochemical networks. In *Proc. CSB*, number 4230 in LNCS, 2006.

3 A. Corradini, T. Heindel, F. Hermann, and B. König. Sesqui-pushout rewriting. In *Proc. ICGT*, 2006.

4 V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Abstracting the differential semantics of rule-based models: exact and automated model reduction. In *Proc. LICS*, 2010.

5 V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable simulation of cellular signaling networks. In *Proc. APLAS*, 2007.

6 V. Danos, J. Feret, W. Fontana, and J. Krivine. Abstract interpretation of cellular signalling networks. In *Proc. VMCAI*, 2008.

7 V. Danos, R. Harmer, and G. Winskel. Constraining rule-based dynamics with types. *MSCS*, 2012.

8 V. Danos and C. Laneve. Formal molecular biology. *TCS*, 325, 2004.

9 H. Ehrig, J. Padberg, U. Prange, and A. Habel. Adhesive high-level replacement systems: A new categorical framework for graph transformation. In *Proc. ICGT '04*, volume 74 of *Fundamenta Informaticae*, 2006.

10 J. Hayman, C. Thompson-Walsh, and G. Winskel. Simple containment structures in rule-based modelling of biochemical systems. In *Proc. SASB*, 2011.

11 R. Heckel. DPO transformation with open maps. In *Proc. ICGT*, number 7562 in LNCS, 2012.

12 F. Hermann. Permutation equivalence of DPO derivations with negative application conditions based on subobject transformation systems. *ECEASST*, 16, 2008.

13 M. Löwe. Algebraic approach to single-pushout graph transformation. *TCS*, 109, 1993.

14 `kappalanguage.org`. The kappa simulator.

15 G. Winskel and M. Nielsen. Models for concurrency. In *Handbook of Logic and the Foundations of Computer Science*. OUP, 1995.