Interactions between causal structures in graph rewriting systems

Ioana Cristescu Walter Fontana
Department of Systems Biology, Harvard Medical School, Boston, USA
{ioana_cristescu,walter_fontana}@hms.harvard.edu

Jean Krivine
IRIF, Universite Paris 7, Paris, France
jean.krivine@irif.fr

Graph rewrite formalisms are a powerful approach to modeling complex molecular systems. They capture the intrinsic concurrency of molecular interactions, thereby enabling a formal notion of mechanism (a partially ordered set of events) that explains how a system achieves a particular outcome given a set of rewrite rules. It is then useful to verify whether the mechanisms that emerge from a given model comply with empirical observations about their mutual interference. In this work, our objective is to determine whether a specific event in the mechanism for achieving X prevents or promotes the occurrence of a specific event in the mechanism for achieving Y. Such checks might also be used to hypothesize rules that would bring model mechanisms in compliance with observations. We define a rigorous framework for defining the concept of interference (positive or negative) between mechanisms induced by a system of graph-rewrite rules and for establishing whether an asserted influence can be realized given two mechanisms as an input.

1 Introduction

A persistent challenge across molecular biology is to understand how a multitude of diverse and asynchronous interactions between molecular entities give rise to coherent system behavior. One difficulty arises from the combinatorial complexity inherent in chemistry: A reaction (or interaction) between structured entities, such as molecules, consists in the transformation of specific parts in a manner that depends on a few rather than all aspects specifying the reactants. Combinatorial complexity then arises because a given reactant combination can exhibit several distinct reactive patterns and the same pattern can occur across many distinct reactant combinations. This idea generalizes beyond chemistry.

A molecular system can thus be described in terms of rewrite *rules*. In this way, rule-based modeling tackles combinatorial complexity without succumbing to it because it only specifies rules of pattern transformation and not the multitude of possible carriers of these patterns. Many physical systems can be conveniently described as graphs. A rule-based approach then becomes a graph rewriting formalism with a domain-specific execution model that determines the probability with which a rule fires at a given time. The currently most developed approaches are the Kappa language [**?**, **?**] and BNGL [**?**] for molecular biology and Mød [**?**] for organic chemistry.

A rule formalizes the interaction between physical entities at some chosen level of abstraction. Processes occurring below that level are abstracted away, yet not ignored: They inform what a rule should say, but they are not explicitly represented by it. For instance, in organic chemistry, a rule of reaction between molecules expresses a local reconfiguration of bonds among atoms without explicitly representing the underlying mechanism of electron pushing that engenders such reconfiguration. In molecular biology, an interaction between proteins is typically expressed by asserting the conditions for a change of protein state without representing the structural mechanisms enabling that change. In essence, a mechanism below the chosen abstraction level becomes an axiomatic rule at the abstraction level [**?**].

Submitted to:
CREST 2018

Many observations of system behavior are *assertions* rather than rules. For example, an assertion might claim that the activation of protein X inhibits the assembly of molecular machine Z. It is desirable to determine whether and why an assertion holds in terms of the joint action among rules that represent a particular system. This is tantamount to providing a *mechanism* that *explains* a given assertion at the level of abstraction at which rules are defined.

The stochastic application of rules (a simulation) typically generates a long trace of state transitions. A mechanism is a set of transitions that were jointly necessary in producing a specified outcome. Mechanisms so-defined can be extracted from traces [**?**, **?**] and abstracted into partial orders (posets) of events[1].

Here we propose a formal logic to express and verify a particular kind of assertion about a model written in the Kappa language. We focus on assertions in which the occurrence of one event is claimed to interfere with another event. Our approach takes as input two posets of events (i.e. mechanisms), which might be hypothesized or abstracted from a simulation, and provides evidence whether the two posets interfere with one another at the specified events. The key is that each poset builds up a context that is required for its terminal event. These contexts can be reconstructed and checked for mutual consistency. To lay the foundation for this approach requires setting up some formal machinery which occupies the bulk of this paper.

Interaction between graph rewriting posets. The graphs in Kappa consist of nodes, called agents, meant to represent proteins. Agents are equipped with *sites* through which they connect to one another. A site represents a resource and hence can bear at most one edge. Such graphs are called *site-graphs*.

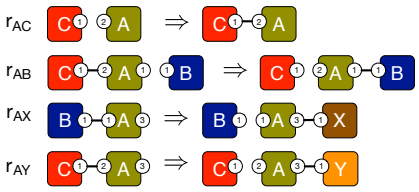


Figure 1: A Kappa model.

An event is the application of a rewrite rule to a usually large graph representing the state of the system. Events are partially ordered by a relation of *precedence*. Intuitively, an event e_1 precedes an event e_2 if e_1 contributes to establishing the context necessary for e_2. Consider, for example, the simple model in Figure 1 with the initial state consisting of nodes $\{\mathtt{A},\mathtt{B},\mathtt{C}\}$, all unbound. Suppose furthermore that the binding of agent X to A (rule r_{AX}) and of agent Y to A (rule r_{AY}) are two significant events e_{AX} and e_{AY}, respectively. We wish to verify the assertion that either event inhibits the other. The assertion is cast in terms of two mechanisms (posets) that could have been extracted from a simulation trace of this model, one mechanism resulting in e_{AX}, the other in e_{AY} (Figure 2A).

A static inspection of rules r_{AX} and r_{AY}, underlying the events that are the subject of our query, shows that both use the same site of A. This might suggest that the two events are in conflict and therefore inhibit each other. This, however, is not a valid conclusion. Given the poset AX of Figure 2A, we can reconstruct the context—specifically the site-graph G_{AX} of Figure 2B—in which rule r_{AX} fires. Note that G_{AX} specifies that site 2 of A must be unbound. Likewise, the firing of rule r_{AY} is contingent upon context G_{AY}, which is built up by poset AY. G_{AY} requires that site 2 of A be bound. These two contexts are in conflict and thus cannot be realized at the same time. This means, in turn, that there is *no* inhibition *at this point between the two mechanisms*: Whether a particular A gets bound to X or to Y is already decided before the mechanisms reach the events whose relationship of inhibition we queried. As a whole, the mechanisms AX and AY must interfere with one another negatively, as A cannot be bound to both X and

[1] In Ref.[**?**], a partially ordered set of events that account for an outcome was dubbed a "story", which is akin to the biological notion of a "pathway".

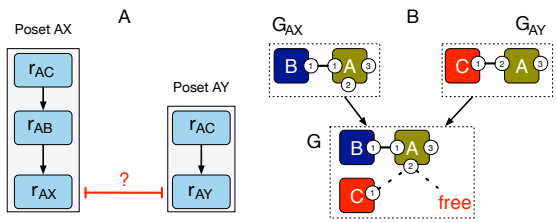


Figure 2: Panel A: Two mechanisms (posets) and a query for conflict between the specified events. Black arrows are precedence; events are labeled by the underlying rules. Panel B: The graph G_{AX} represent the context in which the rule r_{AX} is applied in the Poset AX. There is no scenario in which the posets interact, since the graph G is not a site-graph. The site 3 of agent `A` has to be bound to an agent `C` and be free at the same time, which is not representable in site-graphs.

`Y` at the same time; but the point of conflict is somewhere else. (It is between event r_{AB} and r_{AY}.) To determine the earliest event combination at which two mechanisms conflict with one another can be done by scanning all events of one against all events of the other.

If we change the model by replacing rule r_{AB} with rule r'_{AB} (Figure 3), the context for the application of rule r_{AX} becomes consistent with the context of application of rule r_{AY}. This means both rules can fire. Since the firing of r_{AX} destroys part of the context needed by r_{AY} (and vice versa), the mechanisms inhibit each other at the events queried. In sum, the key in determining whether two events in the scope of distinct mechanisms are mutually exclusive consists in reconstructing from the given mechanisms the context required for both events and determining whether it can be realized. We call this critical context a "scenario".

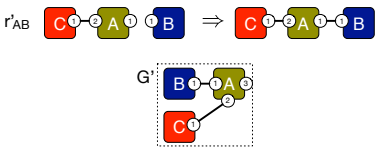


Figure 3: Rule r'_{AB} replaces rule r_{AB}. The graph G' represent the context in which the rule r_{AX} is applied in the Poset AX. In this scenario, which coincides with graph G', there is an inhibition between the two posets AX and AY.

Related work. The notion of *rule influence*, introduced in Refs. [**?**, **?**], is used to detect inhibition and promotion between posets (Definition 18). Our approach to abstracting traces of state transitions into partial orders is similar to Refs. [**?**, **?**], but we use more fine-grained relations on graphs (the enablement and prevention relations of Section 2.4). As a consequence, we do not need Petri nets as an intermediate encoding between state transitions and posets. In any case, our main focus is on reconstructing a trace from a poset, which is obtained from a causal structure extracted from a Kappa simulation. This extraction is the subject of Ref. [**?**] and outside the scope of this paper.

Outline. In Section 2 we introduce site-graphs, the graph rewriting framework of Kappa, and the notion of rule influence. To define partial orders between events in a manner informed by rule influence, we need to take a detour via the transition system induced by the rules (Section 2.4). In order to determine the scenario that establishes enablement or prevention between posets, we need to reconstruct a trace from a poset. To this end, in Section 3, we formalize trace reconstruction as the reverse of poset abstraction from traces. In Section 4 we define a logic for expressing assertions on the posets provided as inputs. We conclude in Section 5.

Length limitations preclude a description of our implementation, which can be found at `https://github.com/Kappa-Dev/PosetLogic`. All constructions on site-graphs presented here can be adapted to simple graphs.

2 Graph rewriting and transition systems

2.1 Site-graphs

Let A be a set of *agents*, ranged over by a, b and $\mathsf{K} = \{A, B, ..\}$ be a set of *agent types*, equipped with a map $\mathsf{site} : \mathsf{K} \to \mathbb{N}_{>0}$. The function $\mathsf{type} : \mathsf{A} \to \mathsf{K}$ assigns a type to each agent.

Definition 1 (Site-graph)**.** A site-graph is a structure $(\mathscr{A}, \mathscr{N}, \mathscr{E})$ where

- $\mathscr{A} \subseteq \mathsf{A}$ is a set of agents;
- $\mathscr{N} \subseteq \mathscr{A} \times \mathbb{N}_{>0} \uplus \{\mathrm{free}\}$ is a set of nodes, with a special node free, and where each non-free node is a pair (a, i) of an agent $\mathrm{a} \in \mathscr{A}$ and site $i < \mathsf{site}(\mathsf{type}(\mathrm{a}))$;
- $\mathscr{E} \subseteq \mathscr{N} \times \mathscr{N}$ is a symmetric set of edges with the constraint that it is *conflict-free*: $\forall (n_1, n_2), (n_1', n_2') \in \mathscr{E}$, $(n_1 = n_1' \wedge n_2 = n_2') \vee (n_1 = n_2' \wedge n_2 = n_1') \vee (\{n_1, n_2\} \cap \{n_1', n_2'\} \subseteq \{\mathrm{free}\})$.

Definition 2 (Morphism on site-graphs)**.** A morphism $f : G \to H$, for G and H two site-graphs, is a pair of functions $f = (v, e)$ with

- $v : \mathscr{A}_G \to \mathscr{A}_H$ a function on agents that preserves types: $\mathsf{type}(v(\mathrm{a})) = \mathsf{type}(\mathrm{a})$ and that can be extended to a function on nodes: $v(\mathrm{a}, i) = (v(\mathrm{a}), i)$ and $v(\mathrm{free}) = \mathrm{free}$, for all $\mathrm{a} \in \mathscr{A}_G$ and for all $i < \mathsf{site}(\mathsf{type}(\mathrm{a}))$;
- and $e : \mathscr{E}_G \to \mathscr{E}_H$ a function on edges such that for any two nodes $n_1, n_2 \in \mathscr{N}_G$, if $(n_1, n_2) \in \mathscr{E}_G$ then $e(n_1, n_2) = (v(n_1), v(n_2))$.

Site-graphs and their morphisms form a category, denoted $\mathscr{G}$. Morphisms in $\mathscr{G}$ preserve the node type and the edge structure of nodes in site-graphs. Isomorphisms are denoted with $\cong$. A *mono* is a morphism with injective functions on nodes and edges. We denote the empty graph with ε and write $\overrightarrow{f} = \langle f_1, f_2 \rangle$ for the span $G_1 \xleftarrow{f_1} G_2 \xrightarrow{f_2} G_3$. The same notation is used to denote the cospan $G_1 \xrightarrow{f_1} G_2 \xleftarrow{f_2} G_3$. For simplicity, we write f for v (or e) in $f = (v, e)$. Finally, we write $\hom(\mathscr{G})$ and $\mathrm{span}(\mathscr{G})$ for the class of morphisms and spans of $\mathscr{G}$, respectively.

2.2 Graph rewriting

A rule-based model consists of graph-rewriting rules that are applied in a stochastic fashion to a typically large graph representing the state of a system. In Kappa the stochastic application of rewrite rules follows basic principles of stochastic chemical kinetics [**?**, **?**]. Each graph-rewrite action constitutes a state transition and a temporal sequence of such transitions is a trace. We also refer to the state of the system as a "mixture".

Definition 3 (Pushout)**.** The *pushout* of a span $\overrightarrow{g}$ is a cospan $\overrightarrow{f}$ such that $f_1 g_1 = f_2 g_2$[2] and such that for any other cospan $\overrightarrow{f'}$ for which $f_1' g_1 = f_2' g_2$, there is a unique morphism $M \to M'$ that makes diagram PO below commute.

In the category of site-graphs, the pushout does not always exist. For a span $\overrightarrow{g}$ of monos, if the pushout exists, then it asserts a gluing of G_1 and G_2, resulting in M, based on the identifications (gluing instructions) expressed by $\overrightarrow{g}$.

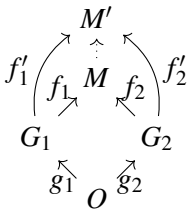
Definition 4 (Rule)**.** A *rule* is a span of monos $\overrightarrow{r} = L \xleftarrow{p} K \xrightarrow{q} R$ such that for some $\mathrm{a} \in \mathscr{A}_K$ and $i < \mathsf{site}(\mathsf{type}(\mathrm{a}))$, if $(\mathrm{a}, i) \in \mathscr{N}_K$ then $\big((q(\mathrm{a}), i), n\big) \in \mathscr{E}_R \iff \big((p(\mathrm{a}), i), n'\big) \in \mathscr{E}_L$, with $n \in \mathscr{N}_R$ and $n' \in \mathscr{N}_L$.

[2] We write $fg(x) = f(g(x))$, with x in the domain of g, for morphisms composition.

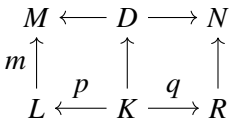
In site-graphs the site of an agent can be specified without specifying if the site is free or bound to another site. Formally, in a site-graph G with an agent $\mathrm{a} \in \mathscr{A}_G$, we can have $(\mathrm{a}, i) \in \mathscr{N}_G$ for which there is no edge $(n_1, n_2) \in \mathscr{E}_G$ such that $n_1 = (\mathrm{a}, i)$ or $n_2 = (\mathrm{a}, i)$. Rules however, need to satisfy a constraint related to sites: if an edge exists for a site in either sides of a rule, then it exists in both sides.

Definition 5 (Double-pushout rewriting [**?**])**.** Let $\vec{r} = L \overset{p}{\leftarrow} K \overset{q}{\rightarrow} R$ be a rule. Let M be a site-graph (typically a system state) and let $m : L \rightarrow M$ be a mono, called *a matching*. The *double pushout rewriting* consists in defining the site-graph D, called the *context* graph, and the site-graph N such that the two squares in diagram DPO are pushouts. We refer to the dpo rewrite of M to N as $M \overset{m,\vec{r}}{\Rightarrow} N$ and denote the state transition associated with the application of rule $\vec{r} = \langle p, q \rangle$ at "location" m of the system state M (i.e. the mixture) as $\mathsf{mix}(M \overset{m,\vec{r}}{\Rightarrow} N) = M \leftarrow D \rightarrow N$.

PO:



DPO:



Given the definition above, a context graph D need not always exist. We use dpo rewriting for the sake of simplicity, but our work extends to other graph rewriting techniques.

2.3 Influence

The postcondition resulting from the application of a rule $\vec{r_1}$ can satisfy or, more generally, contribute (in conjunction with other rules) to satisfying the precondition for the application of another rule $\vec{r_2}$. Alternatively, $\vec{r_1}$ might destroy the precondition of $\vec{r_2}$. In the former case we speak of a positive influence of $\vec{r_1}$ on $\vec{r_2}$ and, in the latter case, of a negative influence. Of course, a rule may also have no influence on a particular other rule.

Influence[3] belongs to the realm of possibility: it is a latent relation between rules that becomes manifest as a relation between events (i.e. actual rule applications) in the specific context of a trace, as we discuss formally in Section 3.

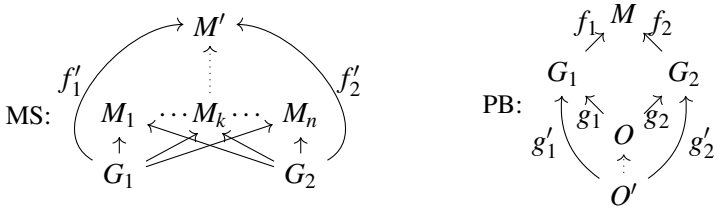
We next define two categorical concepts needed for capturing influence. Multisums are meant to characterize all possible ways of gluing together two graphs G_1 and G_2.

Definition 6 (Multisum in the subcategory of monos [**?**])**.** Let G_1 and G_2 be two graphs. The *multisum* of G_1 and G_2, denoted with $\mathsf{multisum}(G_1, G_2)$, is a family of cospans of monos $\vec{f_i} = \langle f_{1,i}, f_{2,i} \rangle$, with $f_{j,i} : G_j \rightarrow M_i$, $i \leq n$, $j \in \{1, 2\}$, such that for any other cospan of monos $\vec{f'}$, with $f'_j : G_j \rightarrow M'$, there exists an M_k, $k \leq n$ and a unique mono $M_k \rightarrow M'$ that makes diagram MS below commute. Moreover, for any monos $M_k \rightarrow M'$ and $M_i \rightarrow M'$, $i, k \leq n$, for which diagram MS commutes, we have $M_k \cong M_i$.

Unlike other constructions, which are defined in $\mathscr{G}$, multisums are defined in the subcategory of $\mathscr{G}$ whose morphisms are restricted to monos. Multisums always exists in this subcategory.

Definition 7 (Pullback)**.** The *pullback* of the cospan $\vec{f}$ consists of a span $\vec{g}$ such that $f_1 g_1 = f_2 g_2$. In addition, for any other span $\vec{g'}$ such that $f_1 g'_1 = f_2 g'_2$, there is a unique morphism $O' \rightarrow O$ that makes diagram PB commute.

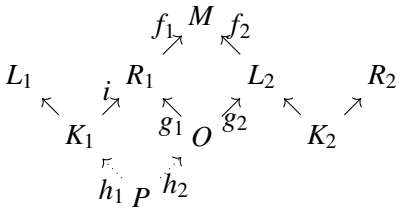
[3]Positive and negative influence were referred to as *activation*, *inhibition* or *overlaps* in Refs.[**?**, Section 3.4],[**?**, Section 4.2.3][**?**].



The pullback always exists in $\mathscr{G}$. Using these notions we can define influence.

Definition 8 (Positive influence [**?**])**.** Given two rules $\vec{r_1} = L_1 \leftarrow K_1 \xrightarrow{i} R_1$ and $\vec{r_2} = L_2 \leftarrow K_2 \rightarrow R_2$, consider an overlap between R_1 and L_2, i.e. a cospan $\vec{f} \in \mathsf{multisum}(R_1, L_2)$, and let $\vec{g}$ be the pullback of $\vec{f}$. Moreover, let $\vec{h}$ be the pullback of $\langle i, g_1 \rangle$. The rule $\vec{r_1}$ has a *positive influence* on rule $\vec{r_2}$, if h_2 is not an iso. In other words, if O is not contained in P and, thus, in K_1. The influence is induced by the overlap $\vec{g}$ corresponding to $\vec{f}$ and is denoted by $\vec{r_1} \xrightarrow{+\vec{g}} \vec{r_2}$.

The diagram on the right depicts the relationships used in Definition 8. The rule $\vec{r_1}$ has a positive influence on $\vec{r_2}$ if it creates a subgraph of L_2. By requiring h_2 not to be an iso, we assert that O is not already present in L_1 and must, therefore, be produced by $\vec{r_1}$. Negative influence $\vec{r_1} \xrightarrow{-\vec{g}} \vec{r_2}$ is defined analogously, but with $\vec{g}$ now the pullback of a cospan $\vec{f} \in \mathsf{multisum}(L_1, L_2)$ between the left hand sides: $\vec{r_1}$ has a negative influence on $\vec{r_2}$, if it destroys a subgraph of L_2.



Example 1. The rule $\vec{r_1}$ has a positive influence on $\vec{r_2}$, because $\vec{r_1}$ produces an agent `B` needed for a subsequent application of $\vec{r_2}$ shown in Figure 4. Similarly $\vec{r_2}$ has a negative influence on $\vec{r_1}$ since it erases an agent `A` needed by $\vec{r_1}$.

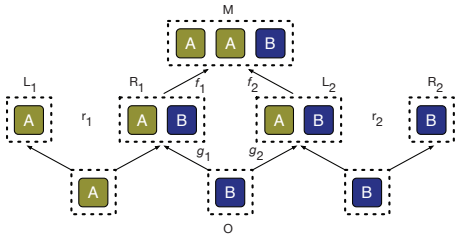


Figure 4: Positive influence between two rules. For simplicity, sites are omitted.

2.4 Transition systems

Following Refs. [**?**, **?**] we introduce the notion of transition system (TS) on state graphs and an independence relation between transitions. We then propose new relations of *enablement* and *prevention* between transitions, based on the notions of rule influence just defined, and connect them to independence.

Definition 9 (TS on graphs [**?**])**.** A transition system $TS = (Q, \mathscr{R}, T)$ on graphs consists of:

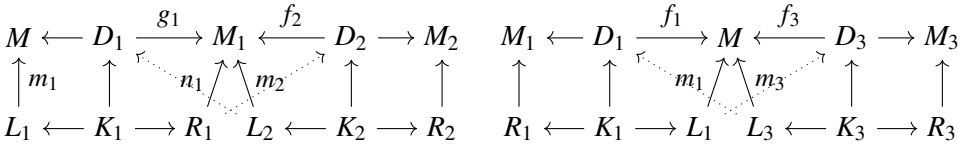
- a set of states $Q \subseteq \mathscr{G}$, where each state is a graph;
- a set of rules $\mathscr{R}$;
- a set of labeled transitions $T \subseteq Q \times \hom(\mathscr{G}) \times \mathscr{R} \times Q$, where each transition t is a dpo rewriting step $M \overset{m,\vec{r}}{\Rightarrow} N$ with $M, N \in Q$, an underlying rule $\vec{r} : L \leftarrow K \rightarrow R \in \mathscr{R}$, and a matching $m : L \rightarrow M \in \hom(\mathscr{G})$.

Transitions can be composed $t_1;t_2$ if the source state of t_2 matches the destination state of t_1. A *trace* θ is a (possibly empty) sequence of composable transitions: $\theta = t_1;t_2;\cdots;t_n$.

Definition 10 (Independence relation on transitions [**?**])**.** Let $t_1 : M \overset{m_1,\vec{r_1}}{\Rightarrow} M_1$, $t_2 : M_1 \overset{m_2,\vec{r_2}}{\Rightarrow} M_2$ and $t_3 : M \overset{m_3,\vec{r_3}}{\Rightarrow} M_3$ be transitions with underlying rules $\vec{r_i} = L_i \leftarrow K_i \rightarrow R_i \in \mathscr{R}$, $i \in \{1,2,3\}$ and corresponding matchings m_i as indicated in the diagrams below.

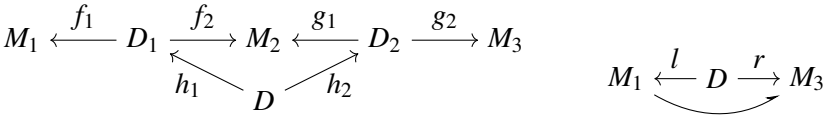
sequential independence t_1 and t_2 are sequentially independent, written $t_1 \diamond_{\text{seq}} t_2$, iff there exist morphisms $i : R_1 \to D_2$ and $j : L_2 \to D_1$ such that $f_2 i = n_1$ and $g_1 j = m_2$.

parallel independence t_1 and t_3 are parallel independent, written $t_1 \diamond_{\text{par}} t_3$, iff there exist morphisms $i : L_1 \to D_3$ and $j : L_3 \to D_1$ such that $f_3 i = m_1$ and $f_1 j = m_3$.



sequential independence parallel independence

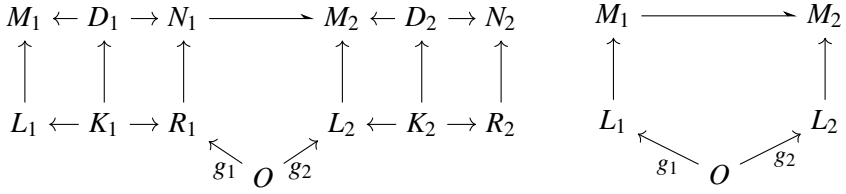
In the following, we use the function mix (of Definition 5) to chain transitions by span composition (see diagrams below). Given two spans $\vec{f} = \langle f_1, f_2 \rangle$ and $\vec{g} = \langle g_1, g_2 \rangle$, we define their composition as $\vec{g}\vec{f} = \langle f_1 h_1, g_2 h_2 \rangle$ where $\vec{h}$ is the pullback of $\langle f_2, g_1 \rangle$. A partial morphism $f : M_1 \rightharpoonup M_3$ is a total morphism from the subgraph $\text{dom}(f)$ of M_1 to M_3, that is $f : M_1 \supseteq \text{dom}(f) \to M_3$. Given a span $M_1 \overset{l}{\leftarrow} D \overset{r}{\rightarrow} M_3$, its corresponding partial morphism, denoted $M_1 \rightharpoonup M_3$, is defined on $l(D)$ as $l^{-1}r$ and undefined otherwise [**?**].



Definition 11 (Causality)**.** Let $t_1 : M_1 \overset{m_1,\vec{r_1}}{\Rightarrow} N_1$ and $t_2 : M_2 \overset{m_2,\vec{r_2}}{\Rightarrow} N_2$ be two transitions bracketing a trace $\theta : t_1;t'_1;t'_2;\cdots;t'_n;t_2$. The rules inducing $t_i, i \in \{1,2\}$, are $\vec{r_i} = L_i \leftarrow K_i \rightarrow R_i$ with matchings $m_i \in \hom(\mathscr{G})$ into M_1 and M_2, respectively.

enablement Let $\vec{g}$ be a span such that $\vec{r_1} \xrightarrow{+\vec{g}} \vec{r_2}$. If the diagram below on the left commutes then t_1 enables t_2, denoted $t_1 <_\theta t_2$. In the diagram below on the left, the partial morphism $N_1 \rightharpoonup M_2$ is obtained from the composition of $\text{mix}(t'_1) \circ \cdots \circ \text{mix}(t'_n)$.

prevention Let $\vec{g}$ be a span such that $\vec{r_2} \xrightarrow{-\vec{g}} \vec{r_1}$. If the diagram below on the right commutes then t_2 prevents t_1, denoted $t_2 \dashv_\theta t_1$. In the diagram below on the right the partial morphism $M_1 \rightharpoonup M_2$ is the composition of $\text{mix}(t_1) \circ \text{mix}(t'_1) \circ \cdots \circ \text{mix}(t'_n)$.



enablement prevention

To make the underlying span explicit, we sometimes write $(t_1, t_2, \overrightarrow{g}) \in <_\theta$ and $(t_1, t_2, \overrightarrow{g}) \in \dashv_\theta$.

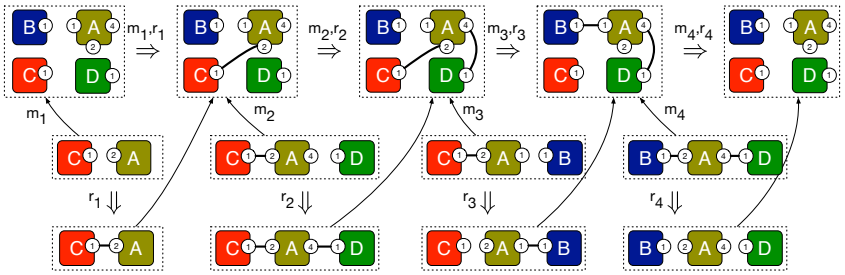


Figure 5: Transition t_2 binds agents `A` and `D`, needed by transition t_4. Transition t_3 needs to happens between the two, as it (i) binds agents `A` and `B` as needed by t_4 and (ii) unbind `A` from `C` which was necessary for t_2.

Example 2. Consider the trace of Figure 5. The first transition enables the second and third transitions. The second transition enables the last transition. However, it is a "delayed" enabler: it partially fulfills the precondition of the last transition, but the third transition has to happen before. Note that such type of causality is not captured by the graph rewriting framework of [**?**, **?**, **?**][4]. Lastly note that the third transition is a preventer for the second one.

3 Posets of graph rewriting events

In this section we abstract a trace into a poset of events, and concretize a poset back into a set of traces. Each transition becomes an event in a poset with the underlying rule as its label. Similarly, in the concretization, each event in a poset corresponds to a transition such that transitions compose into a trace. The abstraction is used to reduce the number of simulation traces to a small set of posets, and the concretization recomputes a "representative" trace from each poset. Concretized traces are used in the next section.

3.1 From traces to posets

A transition t (Definition 5) is a pair of spans—$\mathsf{mix}(t) = M \leftarrow D \rightarrow N$ and the underlying rule $\overrightarrow{r} : L \leftarrow K \rightarrow R$—and a matching $m : L \rightarrow M$. When abstracting a trace into a partial order, we drop the span $\mathsf{mix}(t)$ and m. The enabling and prevention relations between transitions in a trace (Section 2.4) translate into a partial order on events, labeled by the underlying rules.

We proceed in two steps. Enabling and prevention between transitions hinge on positive and negative influence between the underlying rules (see Definition 11). Recall that when transition t enables transition t' within a trace θ there exists a span $\overrightarrow{f}$ for which $(t, t', \overrightarrow{f}) \in <_\theta$. The first abstraction, $\mathscr{A}_1$, forgets the matching and the span $\mathsf{mix}(t)$ of a transition t, but preserves enablement and prevention relations between transitions and the positive and negative influence between the underlying rules.

Definition 12 (Abstraction step 1)**.** Let $\theta = t_1; t_2; \cdots; t_n$ be a trace and $E = \{e_1, e_2, \cdots, e_n\}$[5] be a set of events. Events are labeled using a function $\ell : E \rightarrow \mathscr{R}$ such that $\ell(e_i) = r_i$ if $t_i : M_i \overset{m_i, \overrightarrow{r_i}}{\Rightarrow} N_i$, for $i \leq n$. We then define two relations $\cdot \xrightarrow{+\cdot} \cdot, \cdot \xrightarrow{-\cdot} \cdot \subseteq E \times E \times \mathrm{span}(\mathscr{G})$:

[4]For immediate transitions, enablement and prevention coincide with the sequential dependence and critical pairs, respectively, of Refs. [**?**, **?**, **?**]. See the appendix for more details.

[5]We can define a function $\mathrm{id} : T \rightarrow \mathbb{N}$ from transitions to natural numbers such that $\mathrm{id}(t_i) = i$. The set of events is then $E = \{1, 2, \cdots n\}$.

$$e_i \xrightarrow{+\vec{f}} e_j \iff (t_i, t_j, \vec{f}) \in <_\theta \text{ and } e_i \xrightarrow{-\vec{f}} e_j \iff (t_i, t_j, \vec{f}) \in \dashv_\theta,$$

for $e_i, e_j \in E$, $i, j \leq n$ and $\vec{f} \in \text{span}(\mathscr{G})$. We denote this first abstraction of θ with $\mathscr{A}_1(\theta) = (E, \ell, \xrightarrow{+}, \xrightarrow{-})$.

The notation $e \xrightarrow{+\vec{f}} e'$, for some span $\vec{f}$, overloads the notation $\ell(e) \xrightarrow{+\vec{f}} \ell(e')$. Keep in mind, however, that the first is *defined* on events whereas the second can be *inferred* from the rules on which it holds (see Definition 8).

In the second abstraction step, $\mathscr{A}_2$, we map the relations $\xrightarrow{+}$ and $\xrightarrow{-}$ to corresponding partial orders on events. This step simply forgets the spans responsible for the enablement and prevention relations on transitions.

Definition 13 (Abstraction step 2)**.** Let E be a set of events equipped with a labeling function $\ell : E \to \mathscr{R}$ and two relations $\cdot \xrightarrow{+} \cdot, \cdot \xrightarrow{-} \cdot \subseteq E \times E \times \text{span}(\mathscr{G})$. We translate the relations on events from Definition 12 into two new relations $<, \Vdash \subseteq E \times E$:

$$e_i < e_j \iff e_i \xrightarrow{+\vec{f}} e_j \text{ and } e_i \Vdash e_j \iff e_j \xrightarrow{-\vec{f}} e_i.$$

The associated poset is defined as $\mathscr{A}_2(E, \ell, \xrightarrow{+}, \xrightarrow{-}) = (E, \ell, \leq, \vdash)$, where $\leq$ and $\vdash$ are the transitive and reflexive closure of $<$ and $\Vdash$, respectively. We call the two relations $\leq$ and $\vdash$, *(enabling) precedence* and *non-enabling precedence*, respectively[6].

Lemma 1. Let θ be a trace and let $e, e' \in E$ be two events with $\mathscr{A}_2\mathscr{A}_1(\theta) = (E, \ell, \leq, \vdash)$. If $e < e'$ then there exists a span $\vec{f}$ such that $\ell(e) \xrightarrow{+\vec{f}} \ell(e')$. Similarly, if $e \Vdash e'$, then there exists a span $\vec{f} \in \text{span}(\mathscr{G})$ such that $\ell(e') \xrightarrow{-\vec{f}} \ell(e)$.

A morphism on posets is a function on events that preserves labels and the two precedence relations. An isomorphism between two posets s_1 and s_2 is denoted by $s_1 \cong s_2$. For a set of traces $\Theta = \{\theta_1, \cdots, \theta_n\}$, we write $\mathscr{S} = (s_1, \cdots, s_k)/_\cong$ with $k \leq n$ for the set of posets obtained via $\mathscr{A}_2\mathscr{A}_1$ and quotiented by iso.

Example 3. Consider the trace $\theta = t_1; t_2; t_3; t_4$ of Example 2. The corresponding poset consists of the events $\{e_1, e_2, e_3, e_4\}$ with the relations $<= \{(e_1, e_2); (e_1, e_3); (e_2, e_4); (e_3, e_4)\}$ and $\Vdash = \{(e_2, e_3)\}$. Note that e_2 is a non-enabling precedent of e_3, as in the original trace transition t_3 prevents transition t_2.

3.2 From posets to traces

We next specify the concretization from posets to traces. Again, we proceed in two steps. The first concretization retrieves the intermediate structure $(E, \ell, \xrightarrow{+}, \xrightarrow{-})$ from a poset $(E, \ell, \leq, \vdash)$. This step recovers the influence (positive or negative) between the rules underlying two events that are in a particular precedence or non-enabling precedence relation.

Definition 14 (Concretization step 1)**.** Let $(E, \ell, \leq, \vdash)$ be a poset. We define the relations $\xrightarrow{+}, \xrightarrow{-} \subseteq E \times E \times \text{span}(\mathscr{G})$ as follows:

- $e_i \xrightarrow{+\vec{f}} e_j \iff \ell(e_i) \xrightarrow{+\vec{f}} \ell(e_j)$ and $e_i < e_j$, for some $\vec{f} \in \text{span}(\mathscr{G})$;
- $e_i \xrightarrow{-\vec{f}} e_j \iff \ell(e_i) \xrightarrow{-\vec{f}} \ell(e_j)$ and $e_j \Vdash e_i$, for some $\vec{f} \in \text{span}(\mathscr{G})$

[6] In order to not introduce unnecessary terminology, we abuse the term *poset* to mean the structure $(E, \ell, \leq, \vdash)$ where the set of events E is equipped with *two* partial orders. We could instead define $(E, \ell, (< \cup \Vdash)^\star)$ but in this case we forget the distinction between $<$ and $\Vdash$.

where $<$ and $\Vdash$ are the reduced relation of $\leq$ and $\vdash$, respectively. The concretization of a poset is then $\mathscr{C}_1(E,\ell,\leq,\vdash) = (E,\ell,\overset{+}{\rightarrow},\overset{-}{\rightarrow})$.

Example 4. Consider a poset of events e_1, e_2 and e_3 with labels $\vec{r_1}$, $\vec{r_2}$ and $\vec{r_3}$, respectively, as shown in Figure 6. Furthermore, suppose that events e_1 and e_2 both precede e_3. For the pair $e_1 < e_3$, one can infer the positive influence $\vec{r_1} \xrightarrow{+\vec{f}} \vec{r_3}$. For the pair $e_2 < e_3$, we need to consider two possibilities: either

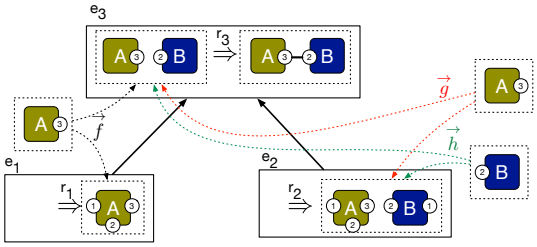


Figure 6: All possible influences between three rules.

$\vec{r_2} \xrightarrow{+\vec{g}} \vec{r_3}$ or $\vec{r_2} \xrightarrow{+\vec{h}} \vec{r_3}$. The relation induced by the span $\vec{g}$ is problematic. Intuitively, the events e_1 and e_2 should produce a distinct set of agents for event e_3. Specifically, both cannot produce the *same* agent `A` that binds to `B` in e_3. The *consistent* span attributes the creation of agent `A` to e_1 and the creation of `B` to e_2 (in addition to a further `A` not used in e_3). In this manner, both e_1 and e_2 are necessary for the occurence of e_3.

As the example indicates, it is not trivial to retrieve the influence between events from the influence between rules. The problem is that influence between events is a *global* property of the poset, whereas influence between rules is *local* to the two rules. Lack of space prevents us from characterizing the correct concretizations of a poset. Informally, a concretization of a poset s is correct if (i) every relation on events in s is due to a shared resource (i.e. an agent or an edge) and if (ii) every resource in s is consistent throughout s.

The second concretization maps events into transitions such that: (i) the transitions compose into a valid trace and (ii) the relations defined on the events hold on the transitions of the trace. We call a *candidate for concretization* any function from events to transitions that satisfies condition (i).

Definition 15. Let E be a set of events with a labeling function $\ell : E \to \mathscr{R}$ and a total order on events $\sqsubset \subseteq E \times E$. A function $\mathsf{concrete} : E \to T$ is called a *candidate for concretization* if $\mathsf{concrete}(e) = M \overset{m,\vec{r}}{\Rightarrow} N$ such that $\ell(e) = \vec{r}$, for some graphs M, N, and a morphism m. Moreover $\mathsf{concrete}(e_1);\cdots;\mathsf{concrete}(e_n)$, with $e_i \sqsubset e_{i+1}$, $i \leq n$, compose into a trace.

Any such function must also satisfy condition (ii), as in the following definition.

Definition 16 (Concretization step 2)**.** Let E be a set of events equipped with a function $\ell : E \to \mathscr{R}$ and two relations $\cdot \overset{+\cdot}{\rightarrow} \cdot, \cdot \overset{-\cdot}{\rightarrow} \cdot \subseteq E \times E \times \mathrm{span}(\mathscr{G})$. Let $\sqsubset$ be a total order on events and let $\mathsf{concrete} : E \to T$ be a function from events to transitions such that the following hold:

- $(t_i, t_j, \vec{f}) \in <_\theta \iff e_i \xrightarrow{+\vec{f}} e_j$ and
- $(t_i, t_j, \vec{f}) \in \dashv_\theta \iff e_i \xrightarrow{-\vec{f}} e_j$

for $e_i, e_j \in E$, $i, j \leq n$. Then the concretized trace is $\mathscr{C}_2(E,\ell,\overset{+}{\rightarrow},\overset{-}{\rightarrow},\mathsf{concrete},\sqsubset) = \mathsf{concrete}(e_1);\cdots;\mathsf{concrete}(e_n)$, for $e_i \sqsubset e_{i+1}$, $i \leq n$.

For $(E,\ell,\leq,\dashv)$ a poset, we write $\mathscr{C}(E,\ell,\leq,\dashv)$ for the set of all possible concretisations, i.e. the set of all traces θ as specified by $\mathscr{C}_1(E,\ell,\leq,\dashv)$ and $\mathscr{C}_2(E,\ell,\overset{+}{\rightarrow},\overset{-}{\rightarrow},\mathsf{concrete},\sqsubset)$. We write $(\theta,\mathsf{concrete})\in\mathscr{C}(E,\ell,\leq,\dashv)$ for the concretization function used in reconstructing a particular θ.

Theorem 1. Let θ be a trace. Then $\theta\in\mathscr{C}\mathscr{A}_2\mathscr{A}_1(\theta)$. Moreover, for any trace $\theta'\in\mathscr{C}\mathscr{A}_2\mathscr{A}_1(\theta)$, $\mathscr{A}_2\mathscr{A}_1(\theta)\cong\mathscr{A}_2\mathscr{A}_1(\theta')$.

4 A logic on posets

In Figure 7 we define a fragment of a first order logic that can be used to express assertions about positive and negative influence between mechanisms, that is, posets. We interpret the logic on the set of posets $\mathscr{S}$, ranged over by s, and on the set of events $\mathscr{E}=\cup_{s_i\in S}E_{s_i}$, where E_{s_i} is the set of events in s_i. To distinguish between the partial orders of different posets in $\mathscr{S}$, we write $s=(E_s,\leq_s,\vdash_s,\ell_s)$. In the following, x stands for variables, t for terms and the superscripts e and s indicate whether the variables and terms range over events or posets, respectively. Formulas are denoted by φ and are built from predicates on variables and terms.

$$
\begin{array}{rll}
x ::= & x^e \mid x^s & \text{(variables on events and posets)}\\
t^s ::= & x^s \mid s & \text{(terms on posets)}\\
t^e ::= & x^e \mid e & \text{(terms on events)}\\
t ::= & t^s \mid t^e & \text{(terms)}\\
\varphi ::= & \exists x.\varphi(x) \mid \forall x.\varphi(x) \mid & \text{(quantifiers)}\\
& \neg\varphi \mid \varphi_1\wedge\varphi_2 \mid & \text{(logical connectors)}\\
& t^e\in t^s \mid \ell(t^e)=\vec{r} \mid t_1^e\leq_{t^s} t_2^e \mid t_1^e\vdash_{t^s} t_2^e & \\
& \mid t_1^e\in t_1^s\overset{-}{\rightsquigarrow} t_2^e\in t_2^s \mid t_1^e\in t_1^s\overset{+}{\rightsquigarrow} t_2^e\in t_2^s & \text{(predicates)}
\end{array}
$$

Figure 7: The grammar of the poset logic.

A *valuation* for φ is a function $v:\mathrm{fv}(\varphi)\to\mathscr{E}\uplus\mathscr{S}$ from the set of free variables of φ to the set of events and posets. The *evaluation* of φ is defined below and requires a valuation function v for the set of free variables of φ; the evaluation is therefore parametric on v. We use two functions, one to evaluate terms $\{\}_v:t\to\mathscr{E}\uplus\mathscr{S}$ and one to evaluate formulas $[\![]\!]_v:\varphi\to\{T,F\}$. A formula φ is satisfiable if there exists v such that $[\![\varphi]\!]_v$ evaluates to true. The interpretation of formulas and terms is shown in Figure 8.

$$
\begin{aligned}
[\![\forall x^s.\varphi]\!]_v &\iff \text{for all } s\in\mathscr{S}, [\![\varphi(s/x)]\!]_v\\
[\![\exists x^s.\varphi]\!]_v &\iff \text{for some } s\in\mathscr{S}, [\![\varphi(s/x)]\!]_v\\
[\![\neg\varphi]\!]_v &= \neg[\![\varphi]\!]_v\\
[\![\varphi_1\wedge\varphi_2]\!]_v &= [\![\varphi_1]\!]_v\wedge[\![\varphi_2]\!]_v\\
[\![t^e\in t^s]\!]_v &\iff \{t^e\}_v\in\{t^s\}_v\\
[\![\ell(t^e)=\vec{r}]\!]_v &\iff \ell(\{t^e\}_v)=\vec{r}\\
[\![t_1^e\leq_{t^s} t_2^e]\!]_v &\iff e_1\leq_s e_2 \text{ where } e_1=\{t_1^e\}_v, e_2=\{t_2^e\}_v, s=\{t^s\}_v\\
[\![t_1^e\vdash_{t^s} t_2^e]\!]_v &\iff e_1\vdash_s e_2 \text{ where } e_1=\{t_1^e\}_v, e_2=\{t_2^e\}_v, s=\{t^s\}_v\\
[\![t_1^e\in t_1^s\overset{+/-}{\rightsquigarrow} t_2^e\in t_2^s]\!]_v &\iff e_1\in s_1\overset{+/-}{\rightsquigarrow} e_2\in s_2 \text{ where } e_1=\{t_1^e\}_v, e_2=\{t_2^e\}_v,\\
&\qquad\qquad s_1=\{t_1^s\}_v, s_2=\{t_2^s\}_v
\end{aligned}
$$

$$
\begin{aligned}
\{x\}_v &= v(x)\\
\{e\}_v &= e\\
\{s\}_v &= s
\end{aligned}
$$

Figure 8: The intepretation of the poset logic.

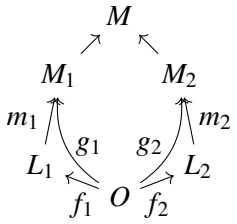
Example 5. We return to the introductory example. The mechanisms of binding an agent A to an agent X or to an agent Y consist in the application of rule $\overrightarrow{r_{\mathrm{AX}}}$ and $\overrightarrow{r_{\mathrm{AY}}}$, respectively. The assertion that the first mechanism prevents (or conflicts with) the second is written as $\exists e_1.(e_1 \in s_1 \wedge \ell(e_1) = \overrightarrow{r_{\mathrm{AX}}}) \wedge \exists e_2.(e_2 \in s_2 \wedge \ell(e_2) = \overrightarrow{r_{\mathrm{AY}}}) \wedge e_1 \in s_1 \overset{-}{\rightsquigarrow} e_2 \in s_2$. The logic allows us to formulate more complex mechanisms. For our example, we can write $\exists e.e \in s \wedge \ell(e) = \overrightarrow{r_{\mathrm{AX}}} \vee \ell(e) = \overrightarrow{r_{\mathrm{AY}}}$ for a mechanism that produces A bound to either X or Y.

The predicates $e_1 \in s_1 \overset{+}{\rightsquigarrow} e_2 \in s_2$ and $e_1 \in s_1 \overset{-}{\rightsquigarrow} e_2 \in s_2$ check for enablement and prevention between two posets. Informally, e_1 and e_2 represent the "meeting point" of the two posets s_1 and s_2. We use these events to reconstruct a graph that represents a context in which s_1 enables or prevents s_2.

The *causal past* of an event is the set of events that preceded it. We denote with $[e]_s$ the causal past of an event $e \in E_s$ and define $[e]_s = (E', \leq', \vdash', \ell')$ with $E' = \{e' : e' \in E, e' \leq e\}$ and $\leq', \vdash', \ell'$ defined like $\leq, \vdash, \ell$ but restricted to E'.

Definition 17 (Occurrence context of an event in a poset)**.** Let s be a poset and let $e \in E_s$ be an event. Furthermore, let $(\theta, \mathsf{concrete}) \in \mathscr{C}([e]_s)$ be a concretization of $[e]_s$. We say that a morphism m is an *occurrence context of e in s* if $\mathsf{concrete}(e) = M \overset{m,\ell(e)}{\Rightarrow} N$, for some graphs M, N.

The occurrence context of e_1 in s_1 and of e_2 in s_2 is specified by matchings $m_1 : L_1 \to M_1$ and $m_2 : L_2 \to M_2$, respectively. The diagram on the right illustrates the prevention of s_2 by s_1. Since the graph M contains both M_1 and M_2, both events e_1 and e_2 can occur in that context. We then say that M is a *scenario* for the prevention of s_2 by s_1, which is induced by a negative influence between the underlying rules, $\ell(e_1) \xrightarrow{-\overrightarrow{f}} \ell(e_2)$. The scenario graph M is formally defined as follows.



Definition 18 (Scenario for prevention)**.** Let m_i be an occurrence context of event e_i in the poset s_i, $i \in \{1,2\}$. Let $\overrightarrow{f}$ be a span such that $\ell(e_1) \xrightarrow{-\overrightarrow{f}} \ell(e_2)$. Define the span $\overrightarrow{g} = \langle g_1, g_2 \rangle$ as $g_i = m_i f_i$, $i \in \{1,2\}$. We say that the graph M obtained by the pushout $\overrightarrow{g}$ is a *scenario* (graph) for the prevention of $e_2 \in s_2$ by $e_1 \in s_1$.

Example 6. Let L_1, L_2 be the left hand sides of rules r_{AX} and r_{AY} from Figure 2. We have a negative influence between the rules r_{AX} and r_{AY} induced by the agent A. The occurence context of e_{AX} in the poset AX is obtained from the concretization of the poset AX and consists of the morphism $L_1 \to G_{\mathrm{AX}}$. Similarly the occurence context of e_{AY} in the poset AY is $L_2 \to G_{\mathrm{AY}}$. There is no scenario for prevention as the graph G (in Figure 2) is not a site-graph.

In a similar manner we interpret the enabling relation between two mechanisms. The predicate $(e_1 \in s_1 \overset{+/-}{\rightsquigarrow} e_2 \in s_2)$ returns true if there exists a scenario M as defined above. The pushout does not always exists and, in consequence, mechanisms do not always interact with one another.

The logic is implemented as a systematic inspection of each poset. The set of posets does not have in itself a structure, and therefore there is no smart strategy for deciding whether a formula holds. The point of the logic is to give a formal language and an interpretation for influence between posets.

Example 7. Let us look at a Kappa model slightly more complicated than the one in the Introduction. We give the rules in the figure below. The two posets build up the graphs G_{AX} and G_{AY}. Then there are two "resources" which can produce an inhibition between the two posets. They produce two scenario graphs for inhibition G_1 and G_2, shown in Figure 9.

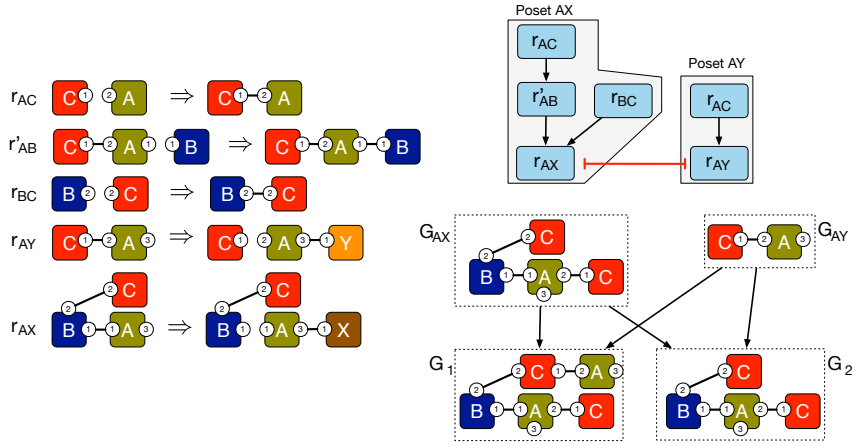


Figure 9: A Kappa model for which there are two scenarios for the prevention between the events labeled r_{AX} and r_{AY}.

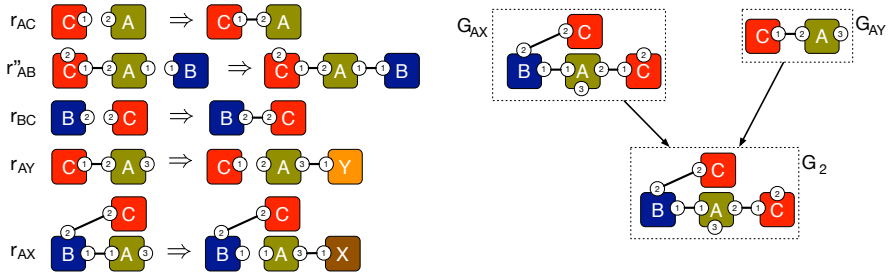


Figure 10: A slightly different Kappa model for which only one of the scenarios is still valid.

Let us change rule r'_{AB} into r''_{AB} and keep everything else the same. With the new rule the graph build up by Poset AX requires the site 2 of agent *C* to be free. In this case only one scenario for inhibition can still occur, shown in Figure 10.

5 Conclusions

Given a categorical notion of graph rewrite system, we defined positive and negative influence between rules. This allowed us to define sequential and parallel independence between state transitions and the relations of enablement and prevention. These were then lifted to the poset abstraction of a trace of state transitions, where they became enabling and non-enabling precedence relations *within* a poset. The formulation of a logic on posets then allows us to formulate questions about enablement and prevention relations *between* posets. We ended by specifying how the concretization of posets back into a trace provides a scenario graph that establishes the truth (or falsity) of a statement about poset interaction. These notions, together with their implementation, are meant to assist a modeler in checking the consistency between observations and the mechanisms that are implied by a rule-based model.

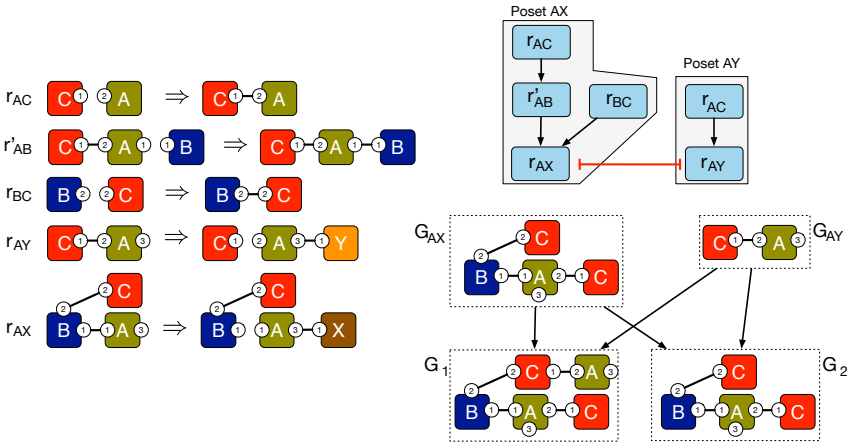
Acknowledgements. We gratefully acknowledge illuminating discussions with Russ Harmer, Jerome Feret, and Jonathan Laurent. Special thanks to Pierre Boutillier for his help in developing and integrating the model checker resulting from this contribution into the Kappa software framework.

A Appendix

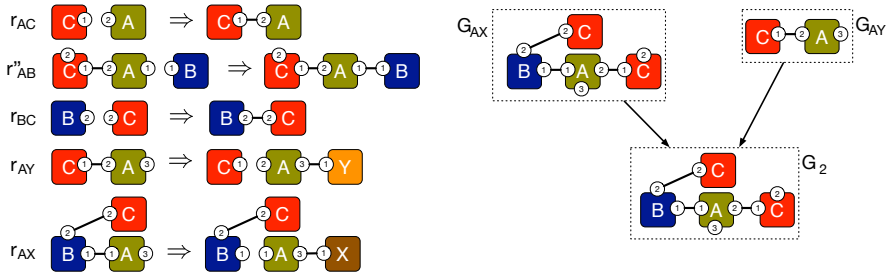
In this appendix we present the proofs missing in our presentation as well as providing more examples and remarks.

A.1 Another Example

Let us look at a Kappa model slightly more complicated than the one in the Introduction. We give the rules in the figure below. The two posets build up the graphs G_{AX} and G_{AY}. Then there are two "resources" which can produce an inhibition between the two posets. They produce two scenario graphs for inhibition G_1 and G_2, shown below.



Let us change rule r'_{AB} into r''_{AB} and keep everything else the same. With the new rule the graph build up by Poset AX requires the site 2 of agent C to be free. In this case only one scenario for inhibition can still occur, shown below.



A.2 Proofs of Section 2.1

Lemma 2. Site-graphs and their morphisms form a category.

Proof. The category of site-graphs has as objects the site-graphs of 1 and as arrows the morphisms of 2. Let us show morphisms compose. Given three site-graphs G_1, G_2, G_3 and two morphisms $f : G_1 \to G_2$, $g : G_2 \to G_3$ let $h = gf$ be the (pairwise) composition of the two. It follows

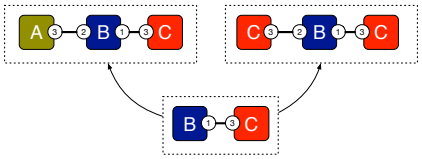
- for an agent $\mathrm{a} \in \mathscr{A}_{G_1}$, $\mathsf{type}(h(\mathrm{a})) = g(f(\mathsf{type}(\mathrm{a}))$ and we have that $\mathsf{type}(h(\mathrm{a})) = h(\mathsf{type}(\mathrm{a}))$;
- for a node $(\mathrm{a}, i) \in \mathscr{N}_{G_1}$, $h(\mathrm{a}, i) = g(f(\mathrm{a})), i))$ and therefore $h(\mathrm{a}, i) = (g(f(\mathrm{a})), i) = (h(\mathrm{a}), i)$; h trivially preserves the free node;

- for an edge $(n_1,n_2) \in \mathscr{E}_{G_1}$, $h(n_1,n_2) = (g(f(n_1),g(f(n_2))$, hence h preserves edges.

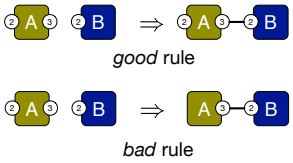
If f and g are injective, then so is h. The axioms of associativity and identity law easily hold. □

A.3 Examples of Section 2.3

Example 8 (Of no pushout in site-graphs)**.** In the figure below, the pushout of the span below is not a site-graph: there is a conflict on site 2 of agent B.



Example 9 (Rules)**.** The *good rule* at the right binds agents `A` and `B` to one another. Note that we do not need to specify all sites of the agents `A` and `B`. The sites that are specified are preconditions for the application of the rule, that is the rule only applies on agents `A` with the sites 2 and 3 free and on agents `B` with site 2 free.



Note that is site 2 of agent A is mentioned on the left hand side, it also needs to appear in the right hand side. This is expressed in condition of the Definition 4. So, for example the *bad* rule on the right does not satisfy this condition.

A.4 Proofs and examples of Section 2.2

In Ref. [**?**] there are two conditions that need to hold for the dpo rewriting of Definition 5. One of the two condition is the dangling condition defined below. The second one (called the *identification condition*) always holds in our setting because we only consider monos in the dpo rewriting.

Property 1 (Dangling conditions)**.** *Let* $\vec{r} = L \xleftarrow{p} K \xrightarrow{q} R$ *be a rule and let* $L \xrightarrow{m} M$ *be a matching in a graph* M*. Define the* gluing points *and* dangling points *as subsets of the nodes in* $\mathscr{N}_L$*, as follows:*

$$GP = p(\mathscr{N}_K)$$
$$DP = \{n \in \mathscr{N}_L : \exists l \in \mathscr{E}_M \setminus m(\mathscr{E}_L)\ s.t.\ l = (m(n), \_)\ or\ l = (\_, m(n))\}.$$

$$\begin{array}{ccccc} M & \xleftarrow{f} & D & \xrightarrow{g} & N \\ {\scriptstyle m}\uparrow & & \uparrow & & \uparrow \\ L & \xleftarrow{p} & K & \xrightarrow{q} & R \end{array}$$

Then we say that the dangling condition holds if $DP \subseteq GP$*.*

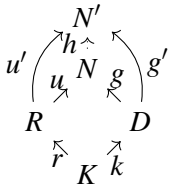
In this work, we only consider rule applications for which the dangling conditions hold. In the following lemma we show that there exists a unique D such that $L \to M \leftarrow D$ is the pushout of the span $L \leftarrow K \to D$. Note that the condition on rules (of Definition 4) is needed for the following result.

Lemma 3. Let $L \xleftarrow{l} K \xrightarrow{r} R$ be a rule and let M be a site-graph and let $m : L \to M$ be a matching. The DPO rewriting can be applied whenever the dangling conditions hold.

Proof.
We first define a pushout in a "constructive" manner. Then we construct the graph D and show that M is the pushout of the span $\langle k,l \rangle$. Lastly, we construct N and show that it is the pushout of the span $\langle k,r \rangle$.

$$\begin{array}{ccccc} M & \xleftarrow{f} & D & \xrightarrow{g} & N \\ {\scriptstyle m}\uparrow & & \uparrow{\scriptstyle k} & & \uparrow{\scriptstyle u} \\ L & \xleftarrow{l} & K & \xrightarrow{r} & R \end{array}$$

1. Let $R \xleftarrow{r} K \xrightarrow{k} D$ be a span. We define $\equiv \subseteq \mathscr{N}_R \times \mathscr{N}_D$ to be the smallest equivalence relation with $(k(n), r(n)) \in \equiv$, for a graph N to be the following graph:
   - $\mathscr{N}_N = (\mathscr{N}_D \cup \mathscr{N}_R)|_{\equiv}$
   - $\mathscr{A}_N = \{\mathrm{a} : \mathrm{a} \in n, n \in \mathscr{N}_N\}$
   - $\mathscr{E}_N = \{(n_1, n_2) : n_1, n_2 \in \mathscr{N}_N$
     if $n_1, n_2 \in \mathrm{image}(r)$ then $[r^-(n_1), r^-(n_2)$
     if $n_1, n_2 \in \mathrm{image}(k)$ then $[k^-(n_1), k^-(n_2)$



The pushout in the category of site-graphs does not always exists. When the pushout does not exists, then the graph N is not a site-graph, i.e. there is a conflict in $\mathscr{E}_N$. We show that if N is a site-graph then it is the pushout. For that, we have to show that the graph N has the universal property: for any site-graph N' and two morphisms $g' : D \to N'$ and $u' : R \to N'$ such that the diagram above commutes, there exists a unique morphism $h : N \to N'$ such that the diagram above commutes. Let the morphism $h : N \to N'$ be as follows:

$$
\begin{aligned}
h(\mathrm{a}) = & u'(u^-(\mathrm{a})) = g'(g^-(\mathrm{a})) && \text{if } \mathrm{a} \in \mathrm{image}(u) \cap \mathrm{image}(g) \\
& u'(u^-(\mathrm{a})) && \text{if } \mathrm{a} \in \mathrm{image}(u) \\
& g'(g^-(\mathrm{a})) && \text{if } \mathrm{a} \in \mathrm{image}(g) \\
h(n_1, n_2) = & u'(u^-(n_1, n_2)) = \\
& g'(g^-(n_1, n_2)) && \text{if } (n_1, n_2) \in \mathrm{image}(u) \cap \mathrm{image}(g) \\
& u'(u^-(n_1, n_2)) && \text{if } (n_1, n_2) \in \mathrm{image}(u) \\
& g'(g^-(n_1, n_2)) && \text{if } (n_1, n_2) \in \mathrm{image}(g)
\end{aligned}
$$

The morphism preserves agent types which follows from the composition of morphisms and from the fact the diagram commutes for the morphisms u' and g'. For a node $(\mathrm{a}, i) \in \mathscr{N}_N$, suppose that $(\mathrm{a}, i) \in \mathrm{image}(u) \cap \mathrm{image}(g)$. It follows that $(u'(u^-(\mathrm{a})), i) = (g'(g^-(\mathrm{a})), i) \in \mathscr{N}'_N$. Therefore h also preserves nodes.

The morphism h is also unique. It follows from the fact that any agent or edge in N is also either in graph R or in graph D. Therefore for the diagram to commute, there is only one possible mapping for any agent or edge from R (or D) into N'.

2. The graph D is defined as a subgraph of M as follows:
   - $\mathscr{N}_D = \mathscr{N}_M \setminus m(\mathscr{N}_L) \cup m(l(\mathscr{N}_K))$
   - $\mathscr{E}_D = \mathscr{E}_M \setminus m(\mathscr{E}_L) \cup m(l(\mathscr{E}_K))$
   - $\mathscr{A}_D = \{\mathrm{a} : \mathrm{a} \in n, n \in \mathscr{N}_D\}$

   It is a site-graph, as it is a subgraph of M. The morphism $f : D \to M$ is defined as the inclusion morphism. The morphism $k : K \to D$ is defined as $k = lm$ restricted to the subgraph D of M.

Let us now show that the cospan $\langle f, m\rangle$ is the pushout of the span $\langle l, k\rangle$. For that, we show that M can be obtained as in the first item of the proof from the graphs D and L.

By manipulating the definition of the set $\mathscr{N}_D$ we obtain

$$\mathscr{N}_D \setminus m(l(\mathscr{N}_K)) \cup m(\mathscr{N}_L) = \mathscr{N}_M$$

which is equivalent to first merging sets $\mathscr{N}_D$ and $m(\mathscr{N}_L)$ and then define an equivalence class on nodes such that $(m(l(n)), k(n)) \in \equiv$, for all $n \in \mathscr{N}_K$. Therefore $\mathscr{N}_M = (\mathscr{N}_D \cup m(\mathscr{N}_L))|_{\equiv}$. We can make a similar argument to show that the edges of M can be obtained as in the construction above, from the edges of graphs L and D.

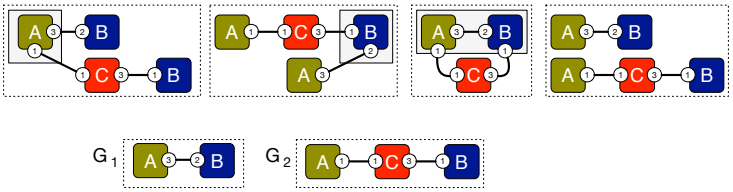
3. Let N be the graph constructed in the first item of this proof. Suffices then to show that N is a site-graph, i.e. we have to show that $\mathscr{E}_N$ is conflict free. Suppose by contradiction that there exists $\big((\mathrm{a}, i), (\mathrm{b}, j)\big) \in \mathscr{E}_D$ and $\big((\mathrm{a}, i), (\mathrm{c}, j)\big) \in \mathscr{E}_R$ such that $(\mathrm{a}, i) \in \mathscr{N}_K$. We have then that both $\big((\mathrm{a}, i), (\mathrm{b}, j)\big)$ and $\big((\mathrm{a}, i), (\mathrm{c}, j)\big)$ are edges in N, which are conflicting.

   From the Definition 4 of a rule, there exists $n \in \mathscr{N}_L$ such that $\big((\mathrm{a}, i), n\big) \in \mathscr{E}_L$. As $m : L \to M$ is a mono, $\big((m(\mathrm{a}), i), m(n)\big) \in \mathscr{E}_M$. From the construction of D above it follows that there is no $n' \in \mathscr{N}_D$ such that $((\mathrm{a}, i), n') \in \mathscr{E}_D$. Contradiction.

□

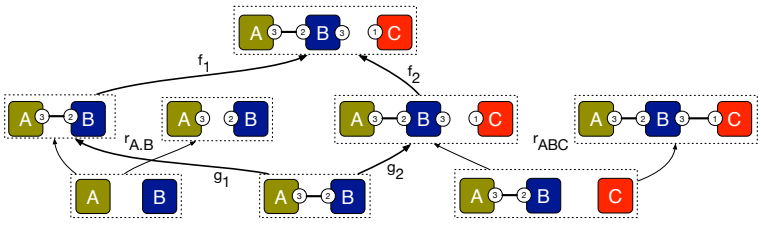
A.5 Proofs and examples of Section 2.3

Example 10 (Multisum)**.** For two site-graphs G_1 and G_2, the site-graphs obtained from the multisum are the top four site-graphs in the diagram below.



In each of the graph of the multisum, the overlapping of G_1 and G_2 is highlighted by a square, i.e. the two graphs overlap on agents A, B, on either A or B or do not overlap.

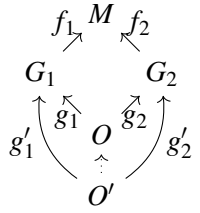
Example 11 (Negative influence between rules)**.** In the figure below, we have two rules $r_{\mathrm{A.B}}, r_{\mathrm{ABC}}$ written as spans. The rule $r_{\mathrm{A.B}}$ has a negative influence on rule r_{ABC} as it unbinds agent A from agent B. Formally, this is expressed by the span $\langle g_1, g_2\rangle$ obtained by pullback from $\langle f_1, f_2\rangle \in \mathsf{multisum}(L_1, L_2)$.



Lemma 4. The pullback always exists in $\mathscr{G}$.

Proof.

Let G_1, G_2, M be two site-graphs and $\overrightarrow{f}$ be a cospan as shown on the right. We define a site-graph O as a subgraph of M (and is therefore a site-graph), which projects into G_1 and G_2. The span $\overrightarrow{g}$ is given by a restriction on the inverse of $\overrightarrow{f}$. Lastly we show the universal property.

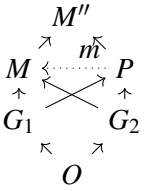


Let $\mathscr{A}_O = f_1(\mathscr{A}_1) \cap f_2(\mathscr{A}_2) \subseteq \mathscr{A}_M$ be a set of agents. We have that the maps $g_1 = f_1^-(\mathscr{A}_O)$ and $g_2 = f_2^-(\mathscr{A}_O)$ are well defined functions on the agents $\mathscr{A}_O$. Let $\mathscr{N}_O = \{(a,i) : (a,i) \in \mathscr{N}_M, a \in \mathscr{A}_O\} \cup \{\text{free}\} \subseteq \mathscr{N}_M$ be a set of nodes. Finally, let $\mathscr{E}_O = \{(n_1,n_2) : (n_1,n_2) \in \mathscr{E}_M, n_1,n_2 \in \mathscr{N}_O\}$ be a set of edges which is by construction symmetric and conflict free. Then $O = (\mathscr{A}_O, \mathscr{N}_O, \mathscr{E}_O)$ is a site-graph, included in M. The maps g_1, g_2 are defined on nodes and edges as expected and are morphisms in $\mathscr{G}$ by construction.

Let $\mathscr{A}'$ be the set of agents of a site-graph O' and let $\overrightarrow{g'}$ be a span such that the diagram above commutes. Then define $h : O' \to O$ a map such that $h(a') = a \iff f_1(g_1'(a')) = a$, for $a' \in \mathscr{A}'$. The map extends to nodes and edges of O'. Moreover, h is the unique morphism that commutes, which follows by contradiction. □

Lemma 5. Let $G_1 \leftarrow O \rightarrow G_2$ be the pullback of the cospan $G_1 \rightarrow M \leftarrow G_2 \in \mathsf{multisum}(G_1, G_2)$. Then, $G_1 \rightarrow M \leftarrow G_2$ is the pushout of $G_1 \leftarrow O \rightarrow G_2$.

Proof. First, we have to show that M' is a site-graph, where the span $G_1 \leftarrow O \rightarrow G_2$ is the pullback of a cospan $G_1 \rightarrow M \leftarrow G_2$ and $G_1 \rightarrow M' \leftarrow G_2$ is the pushout of $G_1 \leftarrow O \rightarrow G_2$. If M' is obtained from the pushout, then there exists a unique mono $M' \rightarrow M$. Therefore, from the Definition 2 of morphism, if there is a conflict in M' then there is a conflict in M as well.
Secondly, let us show the hypothesis. Let us suppose by contradiction, that the cospan $G_1 \rightarrow M \leftarrow G_2$ is not the pushout of the span $G_1 \leftarrow O \rightarrow G_2$. We denote the pushout as the cospan $G_1 \rightarrow P \leftarrow G_2$. From the definition of the pushout we have that there exists a unique morphism $m : P \rightarrow M$ for which the diagram on the right commutes.



Let us distinguish between two cases:

- $G_1 \rightarrow P \leftarrow G_2 \notin \mathsf{multisum}(G_1, G_2)$. If the pushout is not in the multisum, then there exists $M' \in \mathsf{multisum}(G_1, G_2)$ such that there exists a morphism $m' : M' \rightarrow P$. But then there exists a morphism $mm' : M' \rightarrow M$. However as both M, M' are in the multisum, then $M \cong M'$ and, hence $M \cong P$.
- $G_1 \rightarrow P \leftarrow G_2 \in \mathsf{multisum}(G_1, G_2)$. Then let us denote with M'' the pushout-object of the span $M \leftarrow O \rightarrow P$. It follows that both M and P embed into M''. From the definition of the multisum, it follows that $M \cong P$.

□

Let us note that the Lemma above is used in the definition of influence between rules in Ref.[**?**].

A.6 Proofs of Section 3

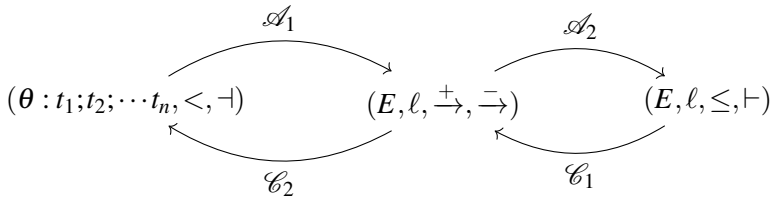
Lemma 6. Let θ be a trace and let $e, e' \in E$ be two events with $\mathscr{A}_1(\theta) = (E, \ell, \xrightarrow{+}, \xrightarrow{-})$. If $e \xrightarrow{+\overrightarrow{f}} e'$ then $\ell(e) \xrightarrow{+\overrightarrow{f}} \ell(e')$, for some $\overrightarrow{f} \in \text{span}(\mathscr{G})$ (and similarly for negative influence).

Proof. Let θ be a trace such that $\mathscr{A}_1(\theta) = (E, \ell, \xrightarrow{+}, \xrightarrow{-})$. For the two events e_1, e_2 in E and a cospan f such that $e \xrightarrow{+\vec{f}} e'$, there exists two transitions $t_1, t_2 \in \theta$ such that $t_1 : M_1 \overset{m_1, r_1}{\Rightarrow} N_1$ and $t_2 : M_2 \overset{m_2, r_2}{\Rightarrow} N_2$, for some graphs M_i, N_i, morphisms m_i and the underlying rules $\vec{r_i} = L_i \leftarrow K_i \rightarrow R_i$, $i \in \{1,2\}$. Moreover, from the Definition 12 it follows that $(t_1, t_2, \vec{f}) \in <_\theta$ and $\ell(e_i) = r_i$, $i \in \{1,2\}$. From the Definition 11 it follows that $r_1 \xrightarrow{+\vec{f}} r_2$. □

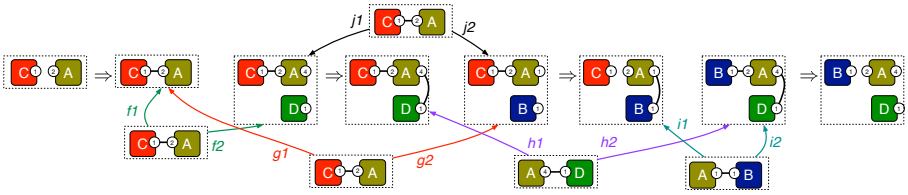
Proof of Lemma 1. Let θ be a trace, let $\mathscr{A}_1(\theta) = (E, \ell, \xrightarrow{+}, \xrightarrow{-})$ be the intermediate structure and $\mathscr{A}_2(E, \ell, \xrightarrow{+}, \xrightarrow{-}) = (E, \ell, \leq, \vdash)$ be a poset. Let $e, e' \in E$ be two events such that $e < e'$. Then from the Definition 13, it follows $e \xrightarrow{+\vec{f}} e'$, for some span $\vec{f}$. From Lemma 6, then we have that $\ell(e) \xrightarrow{+\vec{f}} \ell(e')$.

The proof is similar for the non causal precedence. □

Schematically, the abstractions and concretisations of Definitions 12,13,14,16 are represented as follows:



Example 12 (The two-steps abstraction: Example 3 revisited)**.** Let us consider the trace $\theta = t_1; t_2; t_3; t_4$ of example 2. The first abstraction constructs the structure $(E, \ell, \xrightarrow{+}, \xrightarrow{-})$. Let $\vec{f}, \vec{g}, \vec{h}, \vec{i}, \vec{j} \in \text{span}(\mathscr{G})$ be the spans shown in the figure below:



Then the relations $\xrightarrow{+}, \xrightarrow{-}$ are defined as follows:

$$e_1 \xrightarrow{+\vec{f}} e_2 \quad e_1 \xrightarrow{+\vec{g}} e_3 \quad e_2 \xrightarrow{+\vec{h}} e_4 \quad e_3 \xrightarrow{+\vec{i}} e_4 \quad e_3 \xrightarrow{-\vec{j}} e_2.$$

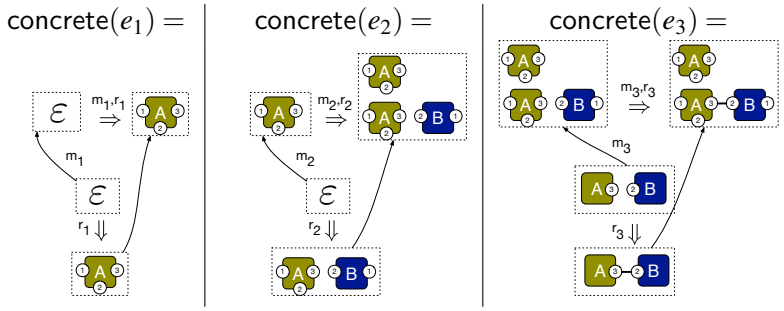
The second abstraction removes the spans in the relations $\xrightarrow{+}, \xrightarrow{-}$. At the end we obtain the structure $(E, \ell, <, \Vdash)$ with $< = \{(e_1, e_2); (e_1, e_3); (e_2, e_4); (e_3, e_4)\}$ and $\Vdash = \{(e_2; e_3)\}$.

Example 13 (The two-steps concretisation: Example 4 revisited)**.** Consider a poset of events e_1, e_2 and e_3 with labels $\vec{r_1}$, $\vec{r_2}$ and $\vec{r_3}$, as in Example 4. The first concretisation step, as we have argued, retrieves the structure $(E, \ell, \xrightarrow{+}, \xrightarrow{-})$ with the relation $\xrightarrow{+}$ defined as follows:

$$e_1 \xrightarrow{+\vec{f}} e_2 \quad e_2 \xrightarrow{+\vec{h}} e_3$$

and $\xrightarrow{-} = \emptyset$.

The second concretisation step needs a function concrete. For our example the following works:



Providing an implementation for the function f is outside the scope of this paper and we leave it as future work. For the case of Kappa, an implementation is available at `https://github.com/Kappa-Dev/PosetLogic`.

Example 14 (More coarse-grained abstractions)**.** Recall that for the trace θ of example 2 our abstraction constructs the poset s of the example 3. Note that there are other abstractions possible from a causal trace. For example, one can also forget the prevention relations between events and retrieve "regular" posets, i.e. sets of events equipped with only one partial order, the precedence relation.

In Example 2 if we forget prevention on trace θ, we obtain the same set of events $\{e_1,e_2,e_3,e_4\}$) but equipped with only the precedence relation $<= \{(e_1,e_2);(e_1,e_3);(e_2,e_4);(e_3,e_4)\}$. From this poset we can concretise a trace in which event e_3 occurs before event e_2, which is not possible in our concretisation.

Another possible abstraction consists of remembering the direct precedence instead of the indirect one. In this case we do not detect that event e_2 is a precedent for event e_4.

None of these abstractions are "wrong" in any sense, but do not keep as much information about the original trace as the abstraction we defined. Our choice is motivated by the application to Kappa, where the indirect enablement and prevention are deemed important.

Proof of theorem 1. Suffices to show that for any trace θ there exists a concretisation function for the poset $\mathscr{A}_2\mathscr{A}_1(\theta)$ such that the constraints in Definitions 16,14 hold. It follows from the Definitions 12,13 of abstractions.

Consider now a trace $\theta' \in \mathscr{C}\mathscr{A}_2\mathscr{A}_1(\theta)$. Both traces θ and θ' meet the constraints in Definitions 16,14. It follows that $\mathscr{A}_2\mathscr{A}_1(\theta) \cong \mathscr{A}_2\mathscr{A}_1(\theta')$ using the Definitions 12,13. □

A.7 Independence, enablement and prevention in graph rewriting

In this section we revisit the graph rewriting theory from Ref. [**?**] adapted to site-graphs.

The next Lemma asserts that when $t_1 \diamond_{\text{par}} t_2$, we can construct a t_1' and t_2' to sequentialize either way, $t_1;t_2'$ or $t_2;t_1'$, with the same net effect. Likewise, when $t_1 \diamond_{\text{seq}} t_2$, we can find a t_1' and t_2' to swap the order from $t_1;t_2$ into $t_2';t_1'$ with, again, the same net effect. The Lemma is a variant of a result from Ref. [**?**] adapted to site-graphs.

Lemma 7 (Permutation of independent transitions)**.** Consider two transitions $t_1 : M \overset{m_1,r_1}{\Rightarrow} M_1$ and $t_2 : M \overset{m_2,r_2}{\Rightarrow} M_2$. If $t_1 \diamond_{\text{par}} t_2$ then there exists an $M' \in Q$ and two transitions $t_2' : M_1 \overset{m_2',r_2}{\Rightarrow} M'$ and $t_1' : M_2 \overset{m_1',r_1}{\Rightarrow} M'$ for some matchings $m_2', m_1' \in \hom(\mathscr{G})$. Moreover, $t_1 \diamond_{\text{seq}} t_2'$ (and $t_2 \diamond_{\text{seq}} t_1'$).

Consider two transitions $t_1 : M \overset{m_1,r_1}{\Rightarrow} M_1$ and $t_2 : M_1 \overset{m_2,r_2}{\Rightarrow} M_2$. If $t_1 \diamond_{\text{seq}} t_2$ then there exists an $M' \in Q$ and two transitions $t_2' : M \overset{m_2',r_2}{\Rightarrow} M'$ and $t_1' : M' \overset{m_1',r_1}{\Rightarrow} M_2$ for some matchings $m_2', m_1' \in \hom(\mathscr{G})$. Moreover, $t_1 \diamond_{\text{par}} t_2'$.

Proof. From the Definition 10 of parallel independence we have that if $t_1 \diamond_{\text{par}} t_2$ then there exists two transitions $t_2' : M_1 \overset{m_2', r_2}{\Rightarrow} M_1'$ and $t_1' : M_2 \overset{m_1', r_1}{\Rightarrow} M_2'$ for some matchings $m_2', m_1' \in \hom(\mathscr{G})$. We have to show that $M_1' \cong M_2'$. For this we show that the parts of M modified by r_1 and the parts modified by r_2 are disjoint and therefore the order in which they occur does not matter.

We fix some notations. We write $n_1 : R_1 \to M_1$, $n_2 : R_2 \to M_2$, $n_1' : R_1 \to M_2'$ and $n_2' : R_1 \to M_1'$. Let us also denote $\mathscr{A}_1' \subseteq \mathscr{A}_M$ the set of agents deleted by the application of r_1 and $\mathscr{A}_1'' \subseteq \mathscr{A}_{M_1}$ the set of agents created by r_1. We use similar notations for $\mathscr{A}_2' \subseteq \mathscr{A}_M$, $\mathscr{A}_2'' \subseteq \mathscr{A}_{M_2}$.

Then we have that

$$\mathscr{A}_{M_1} = \mathscr{A}_M \setminus \mathscr{A}_1' \cup \mathscr{A}_1''$$

and similarly for M_2:

$$\mathscr{A}_{M_2} = \mathscr{A}_M \setminus \mathscr{A}_2' \cup \mathscr{A}_2''$$

The sets $\mathscr{A}_1'$ and $\mathscr{A}_2'$ are disjoint, as the transitions are independent. It implies that $m_2'(m_2^-(\mathscr{A}_2')) \in \mathscr{A}_{M_1}$ and similarly for $n_2'(n_2^-(\mathscr{A}_2'')) \in \mathscr{A}_{M_1'}$ the agents created by r_2. Therefore

$$\begin{aligned}\mathscr{A}_{M_1'} &= \mathscr{A}_{M_1} \setminus m_2'(m_2^-(\mathscr{A}_2')) \cup n_2'(n_2^-(\mathscr{A}_2'')) \\ &= \mathscr{A}_M \setminus \big(\mathscr{A}_1' \cup m_2'(m_2^-(\mathscr{A}_2'))\big) \cup \big(\mathscr{A}_1'' \cup n_2'(n_2^-(\mathscr{A}_2''))\big).\end{aligned}$$

and similarly,

$$\begin{aligned}\mathscr{A}_{M_2'} &= \mathscr{A}_{M_2} \setminus m_1'(m_1^-(\mathscr{A}_1')) \cup n_1'(n_1^-(\mathscr{A}_2'')) \\ &= \mathscr{A}_M \setminus \big(\mathscr{A}_2' \cup m_1'(m_1^-(\mathscr{A}_1'))\big) \cup \big(\mathscr{A}_2'' \cup n_1'(n_1^-(\mathscr{A}_1''))\big).\end{aligned}$$

From the Definition 10 of parallel independence we have that there exists the bijections

$$\begin{aligned}&f_1 : \mathscr{A}_1' \to m_1'(m_1^-(\mathscr{A}_1')); \; f_2 : \mathscr{A}_2' \to m_2'(m_2^-(\mathscr{A}_2')); \\ &g_1 : \mathscr{A}_1'' \to n_1'(n_1^-(\mathscr{A}_1'')); \; g_2 : \mathscr{A}_2'' \to n_2'(n_2^-(\mathscr{A}_2'')).\end{aligned}$$

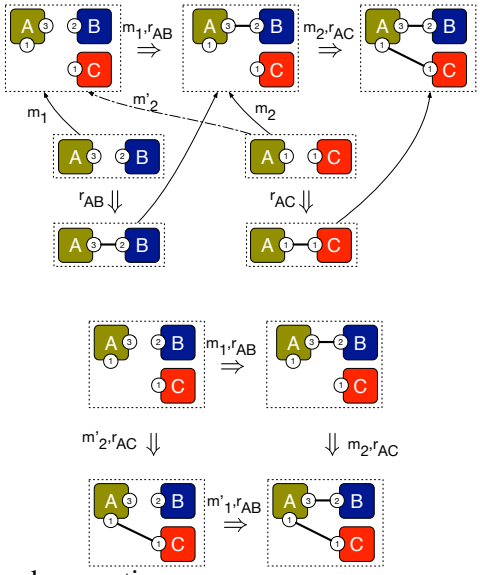
Therefore the we can define a bijection $h : \mathscr{A}_{M_1'} \to \mathscr{A}_{M_2'}$ between the agents of M_1' and the agents of M_2' as follows

$$h = \mathsf{id} \setminus \big(f_1 \cup f_2\big) \cup \big(g_1 \cup g_2\big),$$

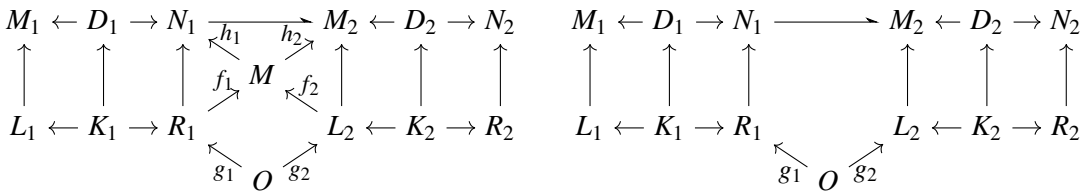
where id is the identity function on the agents of M. As it is obtained from operations on morphisms, the bijection h preserves agent types and nodes. Similarly we proceed to construct the bijection between the egdes of M_1' and M_2'. We obtain a isomorphism between the site-graph M_1' and M_2'.

In a similar fashion we can prove the second part of this lemma. □

Example 15 (Parallel and sequential independence)**.** In the figure on the right we have two sequential transitions that we denote with $t_1 : M_1 \overset{m_1,r_{AB}}{\Rightarrow} M_2$ and $t_2 : M_2 \overset{m_2,r_{AC}}{\Rightarrow} M_3$. The two transitions are sequential independent: there exists m'_2 a morphism from the left hand side of r_{AC} to M_1 such that $m'_2 \mathsf{mix}(t_1) = m_2$. Thanks to Lemma 7 we can also rewrite transition t_2 into a transition $t'_2 : M_1 \overset{m'_2,r_{AC}}{\Rightarrow} M'_2$ as shown in the figure below. Transitions t_1 and t'_2 become then parallel independent. Lastly, note that in the figure, we can also rewrite t_1 into $t'_1 : M'_2 \overset{m'_1,r_{AB}}{\Rightarrow} M_3$, with t'_2 and t'_1 sequential independent.



Let us now revisit the Definition 11 of enablement and prevention.



direct enablement

indirect enablement

Definition 19 (Direct and indirect enablement)**.** Let $t_1 : M_1 \overset{m_1,r_1}{\Rightarrow} N_1$ and $t_2 : M_2 \overset{m_2,r_2}{\Rightarrow} N_2$ be two transitions bracketing a trace $\theta : t_1;t'_1;t'_2;\cdots t'_n;t_2$. The rules inducing $t_i, i \in \{1,2\}$, are $\vec{r_i} = L_i \leftarrow K_i \rightarrow R_i$ with matchings $m_i \in \hom(\mathscr{G})$ into M_1 and M_2, respectively. The span $N_1 \leftarrow D \rightarrow M_2$ is the composition of $\mathsf{mix}(t'_1) \circ \cdots \circ \mathsf{mix}(t'_n)$.

direct enablement Let $\vec{g}$ be a span such that $\vec{r_1} \xrightarrow{+\vec{g}} \vec{r_2}$ and such that $\vec{g}$ is the pullback of $\vec{f} \in \mathsf{multisum}(R_1, L_2)$. If there exists a span $\vec{h}$ such that the diagram below on the left commutes then t_1 directly enables t_2, denoted $t_1 \prec t_2$.

indirect enablement Let $\vec{g}$ be a span such that $\vec{r_1} \xrightarrow{+\vec{g}} \vec{r_2}$. If the diagram below on the right commutes then t_1 indirectly enables t_2, denoted $t_1 < t_2$.

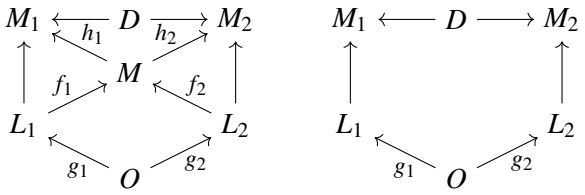
The notion of direct enablement is useful in the characterization of non-independent transitions (see Lemmas 8, 10). However, as we suggested in the example 2, indirect enablement captures more dependencies between transitions.

Example 16 (Example 2 revisited)**.** Transition t_1 is a direct enabler of t_2 and t_3. Transition t_2 is an indirect enabler for t_4.

Definition 20 (Direct and indirect prevention)**.** Consider two transitions $t_1 : M_1 \overset{m_1,r_1}{\Rightarrow} N_1$ and $t_2 : M_2 \overset{m_2,r_2}{\Rightarrow} N_2$ in a trace $\theta : t_1;t'_1 : t'_2;\cdots t'_n;t_2$. The rules inducing $t_i, i \in \{1,2\}$, are $\vec{r_i} = L_i \leftarrow K_i \rightarrow R_i$ with matchings $m_i \in \hom(\mathscr{G})$ into M_1 and M_2, respectively. The span $M_1 \leftarrow D \rightarrow M_2$ is the composition of $\mathsf{mix}(t_1) \circ \mathsf{mix}(t'_1) \circ \cdots \circ \mathsf{mix}(t'_n)$.

direct prevention Let $\vec{g}$ be a span such that $\vec{r_2} \xrightarrow{-\vec{g}} \vec{r_1}$ and such that $\vec{g}$ is the pullback of $\vec{f} \in \mathsf{multisum}(L_1, L_2)$. If there exists a span $\vec{h}$ such that the diagram below on the left commutes then t_2 directly prevents t_1, denoted $t_2 \Rightarrow t_1$.

indirect prevention Let $\vec{g}$ be a span such that $\vec{r_2} \xrightarrow{-\vec{g}} \vec{r_1}$. If the diagram below on the right commutes then t_2 indirectly prevents t_1, denoted $t_2 \dashv t_1$.



direct prevention indirect prevention

Note that $t_1 \prec t_2 \implies t_1 < t_2$ and $t_1 \Rightarrow t_2 \implies t_1 \dashv t_2$, but not the reverse.

We next relate enablement and prevention to non-independence. First we show that when restricting our definitions to the case of consecutive transitions, enablement and prevention are *included* in the negation of sequential and parallel independence.

Lemma 8. Consider two sequential transitions $t_1 : M \overset{m_1,r_1}{\Rightarrow} M_1$ and $t_2 : M_1 \overset{m_2,r_2}{\Rightarrow} M_2$ and let g_1 be the morphism $D_1 \to M_1$, where D_1 is the context graph of t_1. Transition t_1 direct enables transition t_2 iff there exists no morphism $j : L_2 \to D_1$ such that $g_1 j = m_2$.

Lemma 9. Consider two parallel transitions $t_1 : M \overset{m_1,r_1}{\Rightarrow} M_1$ and $t_2 : M \overset{m_2,r_2}{\Rightarrow} M_2$ and let f_1 be the morphism $D_1 \to M$, where D_1 is the context graph of t_1. Transition t_1 is directly prevents transition t_2 iff there is no morphism $j : L_2 \to D_1$ such that $f_1 j = m_2$.

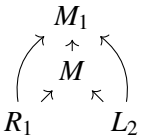
Lastly, the following two Lemmas assert that two non-independent transitions are either in an enabling relation or they can be rewritten as preventing transitions.

Lemma 10. Consider two sequential transitions $t_1 : M \overset{m_1,r_1}{\Rightarrow} M_1$ and $t_2 : M_1 \overset{m_2,r_2}{\Rightarrow} M_2$. Let f_1 and g_1 be the morphisms $D_1 \to M$ and $D_1 \to M_1$, respectively, where D_1 is the context graph of t_1. If the sequential transitions t_1 and t_2 are not independent, then either (i) $t_1 < t_2$ or (ii) there exists a morphism $j : L_2 \to M$ such that $g_1 j = m_2$; and there exists a graph M_2' and a transition $t_2' : M \overset{f_1 j,r_2}{\Rightarrow} M_2'$ with $t_2' \Rightarrow t_1$.

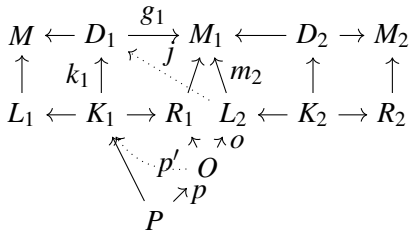
Lemma 11. If two parallel transitions t_1 and t_2 are not parallel independent then either $t_1 \Rightarrow t_2$ or $t_2 \Rightarrow t_1$ (or both).

A.7.1 Proofs of Appendix subsection A.7

Proof of Lemma 8.
Let us first make an observation. For the two rules r_1 and r_2, where we have the morphisms $R_1 \to M_1$ and $L_2 \to M_1$, there exists a unique $R_1 \to M \leftarrow L_2 \in \mathsf{multisum}(R_1, L_2)$ that make the diagram on the right commutes (from Definition 6). The pullback of $R_1 \to M_1 \leftarrow L_2$ is the same as the pullback of $R_1 \to M \leftarrow L_2$, which follows from Lemma 5.



We depict the two transitions in the figure below. Let $R_1 \leftarrow O \to L_2$ be the pullback of the cospan $R_1 \to M_1 \leftarrow L_2$ and let $K_1 \leftarrow P \to O$ be the pullback of $K_1 \to R_1 \leftarrow O$. We proceed by showing that in the diagram below



the mono $p : P \to O$ is an iso iff there exists a morphism $j : L_2 \to D_1$ such that $g_1 j = m_2$. Let us denote $o : O \to L_2$ the morphism between O and L_2 and $o' : O \to R_1$ be the morphism between O and R_1.

- Suppose that $p : P \to O$ is an iso. We denote $p' : O \to K_1$ the composition of p and $P \to K_1$. We define $j : L_2 \to D_1$ as follows

$$\begin{aligned} j(\mathrm{a}) =& k_1(p'(o^-(\mathrm{a}))) && \text{for a} \in \mathrm{image}(o)(\mathscr{A}_O) \\ & g_1^-(m_2(\mathrm{a})) && \text{for a} \in \mathscr{A}_{L_2} \setminus \mathrm{image}(o)(\mathscr{A}_O) \\ j((\mathrm{a},i)) =& (j(\mathrm{a}),i) && \text{for a node } (\mathrm{a},i) \in \mathscr{N}_{L_2} \\ j([n_1,n_2]) =& [j(n_1),j(n_2)] && \text{for an edge } [n_1,n_2] \in \mathscr{E}_{L_2}. \end{aligned}$$

  In what follows we show that j is (i) well defined on agents; (ii) preserves nodes and edges; (iii) is a mono.

  Let $o(O) = L_2' \subseteq L_2$ be the image of $o : O \to L_2$. For any agent a in L_2', we have that $j(\mathrm{a}) = k_1(p'(o^-(\mathrm{a})))$. The function $o^- : L_2' \to O$ is not necessarily a morphism, as it might not preserve nodes and the edges between nodes. It is however a function well defined on agents, which follows from o being a mono. Moreover, for any node or edge in L_2', o^- preserves them into O.

  Consider now x an agent, a node or an edge in L_2 such that $x \notin L_2'$. We have that $R_1 \leftarrow O \to L_2$ is the pullback of $R_1 \to M_1 \leftarrow L_2$ and therefore, if $x \in L_2$, $x \notin \mathrm{image}(o)(O)$ then $m_2(x) \in M_1$, which we denote x^M. We also have that there exists no $x^R \in R_1$, $x^R \in \mathrm{image}(o')(O)$, that maps into x^M.

  We have that $D_1 \to M_1 \leftarrow R_1$ is the pushout of $D_1 \leftarrow K_1 \to R_1$. For $x^M \in M_1$ and not in R_1 we have that there exists $x^D \in D_1$ such that $g_1(x^D) = x^M$. Therefore we can define $j(x) = g_1^-(m_2(x))$, for all $x \in L_2 \setminus \mathrm{image}(o)$. This proves (i) and (ii).

  The function j satisfies (iii) as any agent, node or edge in graph L_2 is either in L_2' or in $L_2 \setminus L_2'$.

- Let us now show that if there exists $j : L_2 \to D_1$ such that $g_1 j = m_2$ then the morphism $p : P \to O$ is an iso.

  For any x (agent, node or edge) in O we have that there exists $x^L \in L_2$ such that $o(x^L) = x$ and similarly for $x^R \in R_1$, $o'(x) = x^R$. As O is the pullback, both x^L and x^R map into the same $x^M \in M_1$. Also, $m_2(x^L) = g_1(j(x^L))$ and therefore there exists $x^D \in D_1$ such that $j(x^L) = x^D$ and $g_1(x^D) = x^M$.

  The span $D_1 \to M_1 \leftarrow R_1$ is the pushout of $D_1 \leftarrow K_1 \to R_1$ and there exists $x^D \in D_1$, $x^R \in R_1$ that map into x^M. It implies that there exists $x^k \in K_1$ such that x^k maps into x^D and x^R.

  The span $K_1 \leftarrow P \to O$ is the pullback of $K_1 \leftarrow R_1 \to O$ and there exists $x^k \in K_1$, $x^O \in O_1$ that map into x^R. Therefore there exists $x^P \in P$ such that $p(x^P) = x$ for all nodes or edges $x \in O$. It implies that p is surjective.

  As p is both surjective and injective (by definition) we conclude that p is indeed an iso.

□

Proof of Lemma 9. This proof is similar to the one of Lemma 8. □

Proof of Lemma 10. If $M \overset{m_1,r_1}{\Rightarrow} M_1$ and $M_1 \overset{m_2,r_2}{\Rightarrow} M_2$ are not sequentially independent, from Definition 10, it follows two possible cases:

- there is no morphism $j : L_2 \to D_1$ such that $g_1 j = m_2$. From Lemma 8, in this case $t_1 < t_2$. We have proved then case (i) from the hypothesis.
- there is no morphism $i : R_1 \to D_2$ such that $f_2 i = n_1$, but there exists the morphism $j : L_2 \to D_1$ such that $g_1 j = m_2$. It implies, from Lemma 7 that there exists $M' \in Q$ and $t_2' : M \overset{m_2',r_2}{\Rightarrow} M'$, for some morphism m_2' such that t_1 with is not parallel independent of t_2'.

  From the Definition 10 of parallel independence if t_1 and t_2' are not parallel independent it follows that there is no morphism $i' : L_1 \to D_2$ such that $f_2 i' = n_1$. Then by Lemma 9, we conclude that $t_2' \dashv t_1$.

□

Proof of Lemma 11. It follows from Definition 10 and Lemma 9. □

A.8 On the implementation

In the implementation we use two simpliying assumptions, that holds for stories generated by KaSim:

- Let θ be a trace. For all M_i mixtures of θ, for all agents $a \in M_i$, there exists a transition of θ $M_j \overset{m,r}{\Rightarrow} M_{j+1}$, with $\overrightarrow{r} = L \leftarrow K \to R$, such that $a \notin \mathscr{A}_L$ and $a \in \mathscr{A}_R$.
- Let $\overrightarrow{r} = L \leftarrow K \to R$ be a rule. For any $a \in \mathscr{A}_R$, if $a \notin q(\mathscr{N}_K)$ then $\forall i < \mathsf{site}(\mathsf{type}(a)), (a,i) \in \mathscr{N}_R$ and $\exists n \in \mathscr{N}_R$ such that $((a,i),n) \in \mathscr{E}_R$.

The first condition, informally, asks that any agent used at some point in a trace has an "introductory" rule, i.e. a rule that creates the agent. The second condition requires that the agents created by a rule are fully specified, i.e. all sites are involved in an edge.

These two assumptions simplify the implementation of the concretisation function, but do not have an impact on the theoretical development presented in the main text.

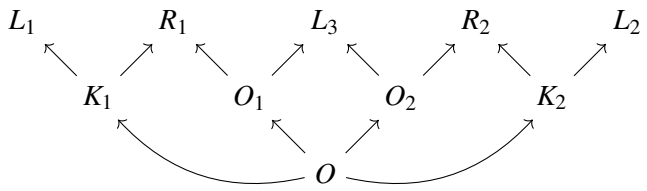
A correct concretisation, as mentioned in the main text, correct concretisations are cumbersome to define. Here we give two conditions that have to hold, the rest being similar.

Definition 21 (Valid poset)**.** A poset $s = (E, \overset{+}{\longrightarrow}, \overset{-}{\longrightarrow}, \ell)$ is *valid* if it is directed w.r.t. the transitive and reflexive closure of $\overset{+}{\longrightarrow}$ and if the following constraints are meet:

no influence is empty Let $e_1, e_2 \in s$ be two events. If $e_1 \overset{+\overrightarrow{f}}{\longrightarrow} e_2$ or $e_1 \overset{-\overrightarrow{f}}{\longrightarrow} e_2$ then $\overrightarrow{f}$ is not empty;

constraints on the influence between rules for positive meets Let e_1, e_2, e_3 be three events in s such that there exists $\overrightarrow{f} = R_1 \leftarrow O_1 \to L_3$ and $\overrightarrow{g} = R_2 \leftarrow O_2 \to L_3$ two spans with $e_1 \overset{+\overrightarrow{f}}{\longrightarrow} e_3$ and $e_2 \overset{+\overrightarrow{g}}{\longrightarrow} e_3$. Let $O_1 \leftarrow O \to O_2$ be the pullback of the span $O_1 \to L_3 \leftarrow O_2$. Then one of the following holds:

- either there exists the morphisms $O \to K_1$ and $O \to K_2$ that commute in the diagram below



- or there exists the morphism $O \to K_1$ but no morphism $O \to K_2$ that commutes. Then there exists $\overrightarrow{f'} : R_2 \leftarrow O' \to L_1$, with $O \subseteq O'$ for which $e_2 \xrightarrow{+\overrightarrow{f'}} e_1$.

...

We call a *resource* any non empty site-graph that appears in one but not both sides of a rule, as for instance an agent or an edge. A correct concretisation is consistent w.r.t. all resources used in the concretised trace.

In the definition above a resource is modeled as a span. The first condition says that if there is a relation between two events in a poset, then necessarily they have a non-empty shared resource. The second condition checks that whenever an event e_3 shares the same resource with two other events e_1 and e_2, then the two events also share the resource.