Incremental Update for Graph Rewriting

Pierre Boutillier[1], Thomas Ehrhard[2], and Jean Krivine[2](✉)

[1] Harvard Medical School, 200 Longwood Avenue, Warren Alpert Building 513, Boston, MA 02115, USA
[2] IRIF and CNRS, IRIF - Université Paris Diderot - Case 7014, 8, place Aurélie Nemours, 75205 Paris Cedex 13, France
Jean.Krivine@irif.fr

Abstract. Graph rewriting formalisms are well-established models for the representation of biological systems such as protein-protein interaction networks. The combinatorial complexity of these models usually prevents any explicit representation of the variables of the system, and one has to rely on stochastic simulations in order to sample the possible trajectories of the underlying Markov chain. The bottleneck of stochastic simulation algorithms is the update of the propensity function that describes the probability that a given rule is to be applied next. In this paper we present an algorithm based on a data structure, called extension basis, that can be used to update the counts of predefined graph observables after a rule of the model has been applied. Extension basis are obtained by static analysis of the graph rewriting rule set. It is derived from the construction of a qualitative domain for graphs and the correctness of the procedure is proven using a purely domain theoretic argument.

1 Introduction

1.1 Combinatorial Models in Systems Biology

As the quest for a cure for cancer is progressing through the era of high throughput experiments, the attention of biologists has turned to the study of a collection of signaling pathways, which are suspected to be involved in the development of tumors.

These pathways can be viewed as channels that propagate, via protein-protein interactions, the information received by the cell at its surface down to the nucleus in order to trigger the appropriate genetic response. This simplified view is challenged by the observation that most of these signaling cascades share components, such as kinases (which tend to propagate the signal) and phosphatases (which have the opposite effect). This implies that signaling cascades not only propagate information, but have also evolved to implement robust probabilistic "protocols" to trigger appropriate responses in the presence of various (possibly conflicting) inputs [1].

As cancer is now believed to be caused by a deregulation of such protocols, usually after some genes coding for the production of signaling components have

H. Yang (Ed.): ESOP 2017, LNCS 10201, pp. 201–228, 2017.
DOI: 10.1007/978-3-662-54434-1_8

mutated, systems biologists are accumulating immense collections of biological facts about proteins involved in cell signaling[1]. The hope of such data accumulation is to be able to identify possible targets for chemotherapy that would be specialized to a specific oncogenic mutation.

Although biological data are being massively produced thanks to high throughput experiments, the production of comprehensive models of cell signaling is lagging. One of the reasons for the unequal race between data production and data integration is the difficulty to make large combinatorial models executable.

1.2 Rule-Based Modeling

Site (or port) graph rewriting techniques, also called *rule-based modeling* [2,3], provide an efficient representation formalism to model protein-protein interactions in the context of cell-signaling. In these approaches, a cell state is abstracted as a graph, the nodes of which correspond to elementary molecular agents (typically proteins). Edges of site graphs connect nodes through named sites (sometimes called ports) that denote a physical contacts between agents. Biological mechanisms of action are interpreted as rewriting rules given as pairs of (site) graphs patterns.

Importantly, rules are applied following a stochastic strategy, also known as SSA or Gillespie's algorithm for rule-based formalisms [4]. KaSim [5] and NFSim [6] are two efficient rule-based simulators that implement this algorithm.

A critical part of the stochastic rewriting procedure is the maintenance, after each rewriting event, of all possible matches that rules may have in the current state of the system, which is a (large) site graph called *mixture*[2]. This number determines the probability that a rule is to be applied next. In general we call *observables* the graph patterns the matches of which need to be updated after each rewriting event. If all rules's left hand sides are mandatory observables, any biologically relevant observation the modeler wishes to track over time has to be declared as an observable as well.

1.3 Rewrite and Update

Beside the initialization phase where all observable matches are identified in the initial graph, observable matches need to be updated after a state change. The update phase can be split into two steps: the *negative update* in which observable matches that no longer hold in the new state are removed, and the *positive update* in which observable matches that have been created by a rule application should be added.

[1] More than 18,000 papers mentioning the protein EGFR, a major signaling protein, either in their title or abstract have been published since 2012. For the year 2015 only there are nearly 5,000 papers for EGFR (source pubmed).

[2] To fix the intuition, let us say that a realistic model of cell signaling would have a few million agents of about a hundred protein types, and several hundreds of rewrite rules, possibly thousands when refinements are automatically generated.

Contrarily to multiset rewriting, in graph rewriting the effect of a rule on a mixture cannot be statically determined. Once a rule has been applied it is necessary to explore the vicinity of the modification to detect potential new or obsolete matches. During this exploration, one may restrict to searching for graph patterns that have a chance to be added (resp. removed) by the modification. In the algorithm presented in Ref. [4], a relation called *activation* (resp. *inhibition*) is computed statically during the initialization phase of a simulation. After a rule r has been applied, the algorithm will look for new instances of observable o only if r activates o. Similarly, a given instance of an observable o' may disappear after the application of r only if r inhibits o'.

There are two essential problems left aside by this simple update method: first, knowing that a new instance of an observable might be created (or deleted) as a consequence of a rewrite does entail one knows in how many *ways* this observable can be found. In particular when dealing with a large amount of possible symmetries, there might be several *equivalent* ways to find a new match. So the first problem to deal with is of combinatorial nature: we wish to statically identify all the *different* manners an observable can be discovered, starting the exploration from a particular sub-graph of this observable (which corresponds to part that may be created or erased by a rewrite). The second issue is to avoid having redundant explorations of the graph after a rewrite: with the classical method sketched above, each observable activated (or inhibited) by the occurrence of a rule need to be searched for. Yet several observables might share sub-graphs. This is particularly true in models that present a lot of refinements of the same rule [7]. In other terms, we wish to design an update method that factors explorations of the graph that can be shared by several observables.

1.4 Outline

This paper presents a novel method for incremental graph rewriting that addresses both issues listed above. We first introduce a *domain of concrete graphs* (Sect. 2), which can be tuned to various types of graphs and, importantly for the application case of this work, to site-graphs.

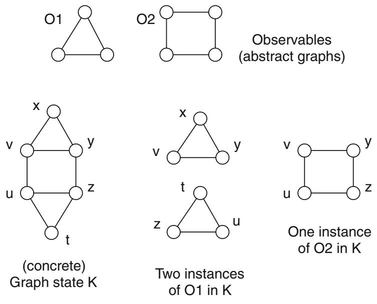
This domain will serve as mathematical foundation in order to specify the *incremental update function* (Sect. 3).

We will then describe *extension bases* (Sect. 4), which can be viewed as a representation of the activation and inhibition relations taking into account sharing and conflict between observables.

These extension bases enable us to implement our incremental update procedure (Sect. 5), and we eventually check that the method is correct using the domain theoretic argument developed in Sect. 3.

2 Concrete Domain

Graph terms can be viewed in two different manners: observables of the system (for instance the left hand sides of the rules) are understood as *abstract* graphs, while the graph representing the state that is to be rewritten is *concrete*. In abstract graphs, nodes identifiers are used up-to consistent renaming, and two isomorphic observables denote in fact *the same* observable, for instance any graph in the shape of a triangle or a rectangle (see right figures).



The state of the rewriting system, however, can be viewed as a concrete graph in the sense that its nodes are used as a reference to track (potentially isomorphic) observables.

Thus, observable *instances* in the state are concrete: a new instance may appear in the state although isomorphic instances existed before.

Since the present work deals with the problem of updating observable instances in a graph state, following a rewriting event, we begin by establishing a simple mathematical framework, which enables us to describe in an extensional fashion, the universe in which concrete graphs live.

2.1 Graphs as Sets

Let $\mathcal{N}$ be a countable set of *nodes* with meta-variables $\{u, v, \dots\}$. *Edges* $\mathcal{E} \subseteq \mathcal{P}_2(\mathcal{N})$ are unordered pairs of nodes with meta-variables $\{e, e' \dots\}$. We say that e and e' are *connected*, whenever $e \cap e' \neq \emptyset$. We use meta-variables $G, H, \dots$ to denotes elements of $\mathcal{P}(\mathcal{E})$.

We consider a *coherence* predicate $\mathsf{Coh} : \mathcal{P}(\mathcal{E}) \to 2$, which is downward closed by inclusion, i.e. $\mathsf{Coh}\ G$ and $H \subseteq G$ implies $\mathsf{Coh}\ H$. A *concrete graph* is a coherent element of $\mathcal{P}(\mathcal{E})$. We use $\mathcal{G} \subseteq \mathcal{P}(\mathcal{E})$ to denote the set of concrete graphs, i.e. $\mathcal{G} =_{\mathsf{def}} \{G \in \mathcal{P}(\mathcal{E}) \mid \mathsf{Coh}\ G\}$ and for all $G \in \mathcal{G}$, we use the notation $|G| =_{\mathsf{def}} \bigcup \{e \in G\}$.

Concrete graphs and set inclusion form the *concrete domain* of coherent graphs. Note that $\mathcal{G}$ is an instance of Girard's qualitative domain [8]. For all $\mathcal{H} \subseteq \mathcal{G}$, we use $\uparrow\mathcal{H}$ and $\downarrow\mathcal{H}$ to denote the upper and lower sets of $\mathcal{H}$ in $\mathcal{G}$. Note that in particular $\downarrow\{G\} = \mathcal{P}(G)$.

For all graphs G, H we say that G is more *abstract* than H (resp. more *concrete*) whenever $G \subseteq H$ (resp. $H \subseteq G$).

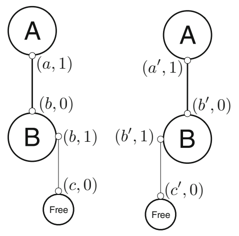
Kappa Graphs. Since efficient Kappa graph rewriting is the main motivation of the present work, we spend some time now to describe Kappa and show how the formalism fits into our general framework.

Kappa graphs are particular kinds of coherent graphs where a node denotes a protein patch, which can either be *free* (i.e. not connected to any other protein) or in *contact* with another protein patch. We encode this by adding a bit of structure to nodes, taking $\mathcal{N} \subseteq \mathcal{A} \times \mathbb{N}$ where $\mathcal{A} =_{\mathsf{def}} \{a, b, \dots\}$ is a countable set of *agents* (protein individuals) that are *sorted* by the map $\kappa : \mathcal{A} \to \mathsf{K}$, where $\mathsf{K} = \{A, B, C, \dots, \mathbf{free}\}$ is a finite set of node *kinds* (the biological name of the protein) with a distinguished element **free**. Therefore in Kappa, a node is of the form $u = (a, i)$ where $i \in \mathbb{N}$ is called the *site* of agent a (a patch of a). A *signature* $\Sigma : \mathsf{K} \to \mathbb{N}$ maps a kind to a (finite) sequence of *sites* identified by natural numbers, with $\Sigma(\mathbf{free}) =_{\mathsf{def}} 1$.

The coherence relation for Kappa is $\mathsf{Coh} =_{\mathsf{def}} \mathsf{Sorted} \wedge \mathsf{ConfFree}$, where:

$$\begin{aligned} \mathsf{Sorted}(G) &=_{\mathsf{def}} \forall e \in G. \, ((a,i) \in e \implies i \leq (\Sigma \circ \kappa) a) \\ \mathsf{ConfFree}(G) &=_{\mathsf{def}} \forall e, e' \in G. \, (e = e' \vee e \cap e' = \emptyset) \end{aligned}$$

We picture on the right an example of a Kappa graph. Nodes (small circles) sharing an agent are depicted attached to the same large circle, named after the kind of the agent. The node $(b, 1)$ that is connected to a **free** node encodes the fact that this protein patch is available for a future interaction. The corresponding graph is obtained as the union of $\{\{(a,1),(b,0)\},\{(b,1),(c,0)\}\}$ and $\{\{(a',1),(b',0)\},\{(b',1),(c',0)\}\}$.



2.2 Effects

In the graph rewriting literature, techniques to decide whether a rule can be applied to a graph come in various flavors [9–11]. In the present work, we do not need to discuss this problem and focus on what happens after a rule has been applied to a graph: we call this the *effect* of a rewrite. The only important point here is that we only consider deterministic effects. For the reader knowledgeable in graph rewriting techniques, they correspond to effects induced by double pushout rewriting [9], where the only way to delete a node is to explicitly delete all edges in which the node appears.

The *effect*, $\eta, \eta', \dots$, of a rewrite can be decomposed as a triple of the form $(G, H^-, H^+) \in \mathcal{G}^3$ where G is the sub-graph that is *tested* by η, and where H^- and H^+ are respectively the *abstraction* and *concretization* steps of η. Intuitively, G are the edges that are required for the rewrite to take place (the match of the left hand side of a rule), H^- and H^+ are the edges that are respectively removed and added during the rewrite step. We do not consider *side-effects*, i.e. those that do not satisfy $H^- \subseteq G$. An effect $\eta = (G, H^-, H^+)$ *occurs* on a graph K if:

– It is *valid*, i.e.: $G \subseteq K$
– It is *visible*, i.e.: $K \cap H^+ = \emptyset$
– It is *defined*, i.e.: $(K \backslash H^-) \cup H^+ \in \mathcal{G}$

For all such effect η and graph K, we define $\eta \cdot K =_{\mathsf{def}} (K \backslash H^-) \cup H^+$. For all effect $\eta = (G, H^-, H^+)$ and for all graph K in which η occurs, we define

$\mathsf{pre}(\eta) =_{\mathsf{def}} G$, the set of edges that are necessarily present in K. Similarly we define $\mathsf{post}(\eta) =_{\mathsf{def}} (G \backslash H^-) \cup H^+$ which is the set of edges that are necessarily present in $\eta \cdot K$. For the remaining of the paper we only discuss defined effects which are both valid and visible[3].

Kappa Effects. To conclude this section and in order to illustrate effects in the context of Kappa, we show below an effect η and its occurrence in the graph K (Fig. 1).

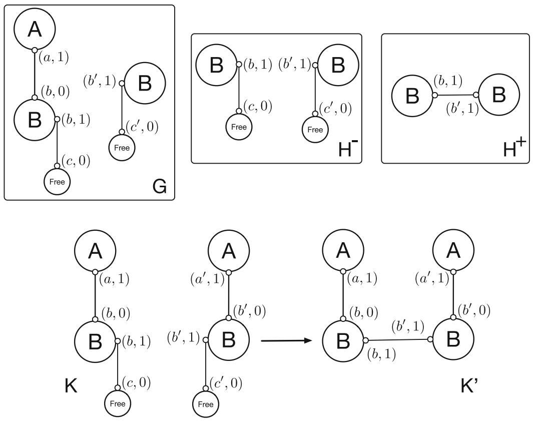


Fig. 1. Illustrating effect in Kappa: $\eta = (G, H^-, H^+)$ occurs in K, with $K' = \eta \cdot K$.

2.3 The Update Problem

Let Φ denote the set of possible effects over graphs in $\mathcal{G}$ and consider a fixed set $\mathcal{O} \subseteq \mathcal{G}$ of *observable* graphs. Let $\mathsf{Obs} : \mathcal{G} \rightarrow \mathcal{P}(\mathcal{O})$ be the *observation map* defined as $\mathsf{Obs}\ G =_{\mathsf{def}} \{O \in \mathcal{O} \mid O \subseteq G\}$.

A *macroscopic effect* $\aleph \in \mathcal{P}(\mathcal{O}) \times \mathcal{P}(\mathcal{O})$ is a pair of observable sets $\aleph = \langle \Omega^-, \Omega^+ \rangle$ satisfying $\Omega^- \cap \Omega^+ = \emptyset$. For all $\mathcal{O}' \subseteq \mathcal{O}$, we define $\aleph \odot \mathcal{O}' =_{\mathsf{def}} (\mathcal{O}' \backslash \Omega^-) \cup \Omega^+$. Intuitively a macroscopic effect describes the set of observables that should be removed (in Ω^-) and added (in Ω^+) after a rewrite has occurred.

We are now in position to state the *incremental update problem*: let K be a graph and η an effect such that $\eta \cdot K$ is defined. We wish to define a function

$$\Delta : \Phi \rightarrow \mathcal{G} \rightarrow \mathcal{P}(\mathcal{O}) \times \mathcal{P}(\mathcal{O})$$

that satisfies the following equation:

$$(\Delta\ \eta\ K) \odot (\mathsf{Obs}\ K) = \mathsf{Obs}\ (\eta \cdot K) \tag{1}$$

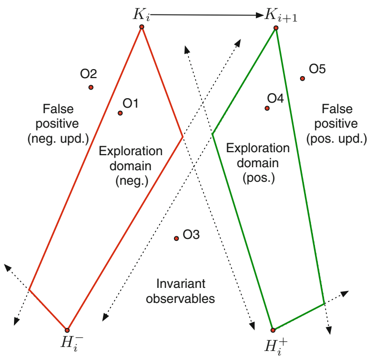
[3] All rewriting techniques satisfy these properties, although only double pushout guarantees the absence of side effect.

Application. Whenever all possible effects $\eta \in \Phi$ satisfy $\mathsf{pre}(\eta) \in \mathcal{O}$, and given the set $\mathcal{O}_K$ of observables that have an occurrence in a graph K, the a priori costly function $\mathsf{Obs}\ (\eta \cdot K)$ can be evaluated by computing $(\Delta\ \eta\ K) \odot (\mathsf{Obs}\ K)$. This is a property that is desirable for any efficient implementation of the stochastic simulation algorithm (SSA) [4], in which $\mathcal{O}_K$ needs to be updated after each rewrite step in order to evaluate the propensity function.

The function Δ will be characterized as a fixpoint of an incremental (one-step) update function on a particular directed sub-domain of $\mathcal{G}$. We turn now to its specification.

3 Exploration Domains

During a sequence of rewrites $K_0, K_1, \ldots, K_n$, the effect $\eta_i = (G_i, H_i^-, H_i^+)$ that occurs during the transition from K_i to K_{i+1} provides the starting point of the update procedure: the observables that should disappear are those that are above H_i^- and below K_i, while the observables that should appear are at those above H_i^+ and below K_{i+1} (see the diagram on the right). Notice that both observables O_4 and O_5 are above H_i^+ but only O_4 is also in $\downarrow\{K_{i+1}\}$. In this case O_5 is not created by the effect and we call it a false positive. In the same example, the instance O_3 is preserved by the effect, as a consequence it cannot be above either H_i^- or H_i^+.


K_i
K_{i+1}
O2
O1
O4
O5
False positive (neg. upd.)
Exploration domain (neg.)
Exploration domain (pos.)
False positive (pos. upd.)
O3
Invariant observables
H_i^-
H_i^+

In the following of this section we assume a fixed graph K and an effect $\eta =_{\mathsf{def}} (G, H_\eta^-, H_\eta^+)$ such that $\eta \cdot K$ is defined. In order to emphasize the symmetry between positive and negative update, we introduce the notation $K^- =_{\mathsf{def}} K$, $K^+ =_{\mathsf{def}} \eta \cdot K$, $\pi_\eta^- =_{\mathsf{def}} \mathsf{pre}(\eta)$ and $\pi_\eta^+ =_{\mathsf{def}} \mathsf{post}(\eta)$ (see Sect. 2.2). In the following, the informal superscript $\_^\varepsilon$ can be replaced globally by either $\_^-$ or $\_^+$ in order to specialize the mathematical description to the negative or positive update.

3.1 Observable Witnesses

Define first the set of *witness* graphs $(W, W', \ldots)$, the presence of which will serve as a proof of negative or positive update of an observable, induced by the occurrence of η:

$$\mathcal{W}_\eta^\varepsilon =_{\mathsf{def}} \{W \in \mathcal{G} \mid \exists O \in \mathcal{O}, O \cap H_\eta^\varepsilon \neq \emptyset \wedge W = O \cup \pi_\eta^\varepsilon\}$$

For all $W \in \mathcal{W}_\eta^\varepsilon$, we say that W is an η-*witness* of $O \in \mathcal{O}$ if $O \cap H_\eta^\varepsilon \neq \emptyset$ and $W = O \cup \pi_\eta^\varepsilon$ and we write $W \rhd_\eta^\varepsilon O$. Notice that W may be the η-witness of several observables.

Proposition 1. *For all $W \in \mathcal{W}^-_\eta$ and $O \in \mathcal{O}$ such that $W \rhd^-_\eta O$:*

$$W \subseteq K^- \iff O \in (\mathsf{Obs}\ K^-) \wedge O \notin (\mathsf{Obs}\ K^+) \tag{2}$$

Similarly, for all $W \in \mathcal{W}^+_\eta$ and $O \in \mathcal{O}$ such that $W \rhd^+_\eta O$:

$$W \subseteq K^+ \iff O \notin (\mathsf{Obs}\ K^-) \wedge O \in (\mathsf{Obs}\ K^+) \tag{3}$$

Proposition 1 guarantees that after η has occurred on K, it is sufficient to extend the graph π^ε_η with edges taken from K^ε in order to reach a witness in $\mathcal{W}^\varepsilon_\eta$. For each $W \subseteq K^\varepsilon$ that are more concrete than π^ε_η, the observable $O \in \mathcal{O}$ that satisfies $W \rhd^\varepsilon_\eta O$ is (positively or negatively) updated.

3.2 Exploration Boundaries

The updatable witnesses after the occurrence of η are the witnesses that are more abstract than K^ε. Since $\downarrow\{K^\varepsilon\}$ forms a complete lattice (it is simply the sub-parts of K^ε), the graph:

$$\widehat{W}^\varepsilon_{\eta,K} =_{\mathsf{def}} \bigcup\{W \in \mathcal{W}^\varepsilon_\eta \mid W \subseteq K^\varepsilon\}$$

is always defined and corresponds to the union of all witnesses that are present in K^ε.

Definition 1 (Optimal update domain). *We call:*

$$\mathcal{X}^\varepsilon_{\eta,K} =_{\mathsf{def}} \downarrow\{\widehat{W}^\varepsilon_{\eta,K}\} \cap \uparrow\{\pi^\varepsilon_\eta\}$$

the optimal (negative or positive) update domain for η and K*.*

Proposition 2. *For all witness $W \in \mathcal{W}^\varepsilon_\eta$, $W \subseteq K^\varepsilon$ if and only if $W \in \mathcal{X}^\varepsilon_{\eta,K}$.*

Proposition 2 indicates that after an effect η has occurred, $\mathcal{X}^\varepsilon_{\eta,K}$ is the smallest domain one needs to explore in order to discover all updatable witnesses. Yet, one cannot hope that a realistic update procedure stays within the boundaries of $\mathcal{X}^\varepsilon_{\eta,K}$ because some witnesses may seem to be updatable given π^ε_η, but are in fact not reachable within K^ε (they are the false positives, discussed in the introduction of this section). The remaining of this section is dedicated to the specification of the directed set that is being explored during the update procedure, and that is defined as an over-approximation of $\mathcal{X}^\varepsilon_{\eta,K}$.

We first define the *η-domain* which is coarsening of the optimal update domain:

Definition 2 (η-Domain). *For all $\mathcal{H} \subseteq \mathcal{G}$, we define the* $\cup$-closure of $\mathcal{H}$*, written $\uparrow^\cup\mathcal{H}$, as:*

$$\uparrow^\cup\mathcal{H} =_{\mathsf{def}} \max\{G \in \mathcal{G} \mid G = \bigcup_i H_i \in \mathcal{H}\}$$

where for all $\mathcal{H} \subseteq \mathcal{G}$, $\max\mathcal{H}$ *is the set of maximal graphs in $\mathcal{H}$. We use this construction to define the* η-domain*:*

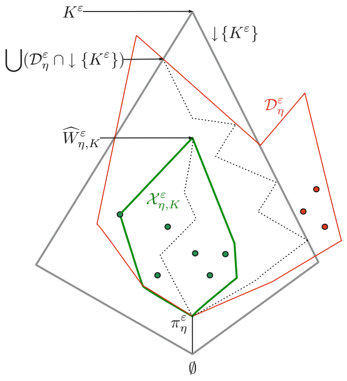
$$\mathcal{D}^\varepsilon(\eta) =_{\mathsf{def}} \downarrow(\uparrow^\cup\mathcal{W}^\varepsilon_\eta) \cap \uparrow\{\pi^\varepsilon_\eta\}$$

Notice that $\uparrow^{\cup}\mathcal{H} = \bigcup\mathcal{H}$ when $\mathcal{H}$ has a supremum.

Contrary to the optimal update domain $\mathcal{X}^{\varepsilon}_{\eta,K}$ (Definition 1), the η-domain $\mathcal{D}^{\varepsilon}(\eta)$ is independent of K^{ε}. By itself it is not a correct over-approximation of the optimal update domain, since it is not in general a directed set. However we get a fine grained approximation of the optimal update domain when one restricts (on the fly) explorations of $\mathcal{D}^{\varepsilon}(\eta)$ to graphs that are also below K^{ε} (see Fig. 2 for illustration):

Proposition 3 (Over-approximation). *For all effect η and graph K such that η occurs on K, the following directed sets are ordered by inclusion:*

$$\mathcal{X}^{\varepsilon}_{\eta,K} \subseteq (\mathcal{D}^{\varepsilon}(\eta) \cap \downarrow\{K^{\varepsilon}\}) \subseteq \downarrow\{K^{\varepsilon}\} \tag{4}$$



K^{ε}
$\downarrow\{K^{\varepsilon}\}$
$\bigcup(\mathcal{D}^{\varepsilon}_{\eta} \cap \downarrow\{K^{\varepsilon}\})$
$\mathcal{D}^{\varepsilon}_{\eta}$
$\widehat{W}^{\varepsilon}_{\eta,K}$
$\mathcal{X}^{\varepsilon}_{\eta,K}$
π^{ε}_{η}
$\emptyset$

Fig. 2. An exploration of $\mathcal{X}^{\varepsilon}_{\eta,K}$ (leftmost dotted line) and an exploration of $\mathcal{D}^{\varepsilon}(\eta) \cap \downarrow\{K^{\varepsilon}\}$ (rightmost dotted line). Circles denote witnesses. In the first exploration all edges that are added along the exploration belong to a witness that is also within K^{ε}. The exploration stops exactly at the supremum of all reachable witnesses, i.e. the sup of the optimal update domain (Definition 1). The rightmost exploration correspond to a path where edges are added only if the resulting graph belong to the η-domain $\mathcal{D}^{\varepsilon}(\eta)$ (Definition 2) and is present in $\downarrow\{K^{\varepsilon}\}$. The difference between the endpoints of the rightmost and leftmost explorations corresponds to the edges that have been inspected with no corresponding updatable witness.

3.3 Specifying the Incremental Update Function

We have now everything in place to specify the incremental update function Δ of Sect. 3. In order to do so, we require that a call to $(\Delta\ \eta\ K)$ be the fixpoint of a *one-step exploration function* that we specify now. Consider a function $\mathsf{inc}^{\varepsilon}_{\eta,K}$:

$$\mathsf{inc}^{\varepsilon}_{\eta,K} : \mathcal{D}^{\varepsilon}(\eta) \times \mathcal{P}(\mathcal{W}^{\varepsilon}_{\eta}) \to \mathcal{D}^{\varepsilon}(\eta) \times \mathcal{P}(\mathcal{W}^{\varepsilon}_{\eta})$$

such that whenever

$$\mathsf{inc}^{\varepsilon}_{\eta,K}\langle X, \mathcal{R}\rangle = \langle X', \mathcal{R}'\rangle$$

the following properties hold:

$$\begin{array}{ll} X' = X \cup G & \text{if there exists } G \subseteq K^{\varepsilon} \text{ satisfying } G \cap X = \emptyset \text{ and } G \in \mathcal{D}^{\varepsilon}(\eta) \\ X' = X & \text{otherwise} \end{array} \tag{5}$$

$$\mathcal{R}' = \{W \in \mathcal{W}^{\varepsilon}_{\eta} \mid W \subseteq X\} \tag{6}$$

Intuitively, the first argument of the function is a graph X (for *explored*) corresponding the current endpoint of the exploration of $\mathcal{D}^{\varepsilon}(\eta) \cap \downarrow\{K^{\varepsilon}\}$. The second argument $\mathcal{R}$ (for *reached*) correspond to the set of observable witnesses that have been discovered so far. Condition (5) ensures that the explored sub-graph X of K^{ε} grows uniformly, and inside the boundaries of $\mathcal{D}^{\varepsilon}(\eta)$ until it reaches its supremum. In the meantime, Condition 6 is making sure that all witnesses that are below X have been collected.

Lemma 1. *Any implementation of* $\mathsf{inc}^{\varepsilon}_{\eta,K}$ *satisfying the above specification admits a least fixpoint of the form:*

$$\langle \top^{\varepsilon}_{\eta,K}, \downarrow\{K^{\varepsilon}\} \cap \mathcal{W}^{\varepsilon}_{\eta}\rangle$$

where $\top^{\varepsilon}_{\eta,K} =_{\mathsf{def}} \bigcup(\mathcal{D}^{\varepsilon}(\eta) \cap \downarrow\{K^{\varepsilon}\})$.

Lemma 1 ensures that the iteration of the one-step incremental update function terminates and returns a pair, the second argument of which is precisely the set of updatable witnesses, i.e. those that are the same time above π^{ε}_{η} and a sub-graph of K^{ε}.

Definition 3 (Incremental update function). *For all effect η and all graph K such that $\eta \cdot K$, and all correct implementation of* $\mathsf{inc}^{\varepsilon}_{\eta,K}$, *let:*

$$(\mathsf{inc}^{-}_{\eta,K})^{\omega}\langle \mathsf{pre}(\eta), \emptyset\rangle = \langle \_, \mathcal{R}^{-}\rangle$$
$$(\mathsf{inc}^{+}_{\eta,K})^{\omega}\langle \mathsf{post}(\eta), \emptyset\rangle = \langle \_, \mathcal{R}^{+}\rangle$$

let also $\Omega^{\varepsilon} : \mathcal{P}(\mathcal{W}^{\varepsilon}_{\eta}) \to \mathcal{P}(\mathcal{O})$ *be:*

$$\Omega^{\varepsilon}\ \mathcal{R} =_{\mathsf{def}} \{O \in \mathcal{O} \mid \exists W \in \mathcal{R}. W \rhd^{\varepsilon}_{\eta} O\}$$

we define the incremental update function *as:*

$$\Delta\ \eta\ K =_{\mathsf{def}} \langle (\Omega^{-}\ \mathcal{R}^{-}), (\Omega^{+}\ \mathcal{R}^{+})\rangle$$

Theorem 1. *For all effect η and all graph K such that $\eta \cdot K$,*

$$(\Delta\ \eta\ K) \odot (\mathsf{Obs}\ K) = (\mathsf{Obs}\ \eta \cdot K)$$

Theorem 1 concludes this section by stating that, provided our one-step incremental update function satisfies its specification, its fixpoint correspond to the macroscopic effect $\langle \Omega^{-}, \Omega^{+}\rangle$ that we are looking for.

4 Abstraction

Although $\mathcal{D}^{\varepsilon}(\eta)$ is an invariant domain, it cannot be used as a data structure *per se* (it is an infinite object). For the update algorithm we use a data structure that can be viewed as a quotient of $\mathcal{D}^{\varepsilon}(\eta)$ in which isomorphic graphs are identified. This quotienting of the concrete domain is naturally described using a categorical terminology.

4.1 Graph: a category of graphs

A graph *homomorphism*, $f : G \to H$, is an injective function on nodes $f : |G| \to |H|$ that preserves edges, i.e.:

$$\{u, v\} \in G \implies \{f(u), f(v)\} \in H$$

We call Graph the category that has graphs as objects and we use $\mathsf{Hom}(\mathcal{G})$ to denote the set of its arrows. We use $\phi, \psi, \ldots$ for graph *isomorphisms.*

A Category for Kappa Graphs. In order to tune Graph to Kappa we require that morphisms should be injective on agents and preserve sorting, i.e. $\kappa \circ f = \kappa$ (see Fig. 5 for an example).

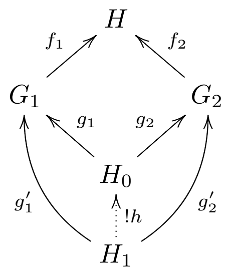
Property 1 (Pullbacks). For all co-span:

$$\boldsymbol{f} : \langle f_1 : G_1 \to H, f_2 : G_2 \to H\rangle$$

there is a unique span:

$$\mathsf{pb}(\boldsymbol{f}) : \langle g_1 : H_0 \to G_1, g_2 : H_0 \to G_2\rangle$$

satisfying $f_1 g_1 = f_2 g_2$ and such that for any alternative span $\boldsymbol{g'}$ there is a unique morphism h that makes the right diagram commute.


$$\begin{tikzcd}
 & H & \\
G_1 \arrow[ur, "f_1"] & & G_2 \arrow[ul, "f_2"'] \\
 & H_0 \arrow[ul, "g_1"'] \arrow[ur, "g_2"] & \\
 & H_1 \arrow[uul, bend left, "g_1'"] \arrow[uur, bend right, "g_2'"'] \arrow[u, dotted, "!h"'] &
\end{tikzcd}$$

We follow now the relative pushout terminology introduced in the context of bigraphical reactive systems [12]. A span:

$$\boldsymbol{f} : \langle f_1 : H \to G_1, f_2 : H \to G_2\rangle$$

admits a *bound*:

$$\boldsymbol{g} : \langle g_1 : G_1 \to H', g_2 : G_2 \to H'\rangle$$

whenever $g_1 f_1 = g_2 f_2$. Given a span $\boldsymbol{f}$ and a bound $\boldsymbol{g}$, we say that $\boldsymbol{f}$ has a bound:

$$\boldsymbol{h} : \langle h_1 : G_1 \to H'', h_2 : G_2 \to H''\rangle$$

relative to $\boldsymbol{g}$, if there exists a morphism $h : H'' \to H'$ such that $hh_1 = g_1$ and $hh_2 = g_2$. We call the triple $(h, \boldsymbol{h})$ a *$\boldsymbol{g}$-relative bound* of $\boldsymbol{f}$.

Property 2 (Relative pushout). A span $\boldsymbol{f}$ that admits bound $\boldsymbol{g}$ also admits a *best* $\boldsymbol{g}$-relative bound $(h, \boldsymbol{h})$ such that for all alternative $\boldsymbol{g}$-relative bound $(k, \boldsymbol{k})$, there exists a unique morphism j such that $jh_1 = k_1$ and $jh_2 = k_2$. This best $\boldsymbol{g}$-relative bound $(h, \boldsymbol{h})$ is called a $\boldsymbol{g}$-*relative pushout* for $\boldsymbol{f}$ (see right diagram).

$$\begin{tikzcd}
 & H' & & \\
 & & K \arrow[ul, "k"'] & \\
G_1 \arrow[uur, bend left, "g_1"] \arrow[urr, "k_1"] \arrow[r, "h_1"'] & H'' \arrow[uu, "h"] \arrow[ur, dotted, "!j"'] & & G_2 \arrow[uull, bend right, "g_2"'] \arrow[ul, "k_2"'] \arrow[ll, "h_2"] \\
 & H \arrow[ul, "f_1"] \arrow[urr, "f_2"'] & &
\end{tikzcd}$$

Note in particular that if $(h, \boldsymbol{h})$ is a $\boldsymbol{g}$-relative pushout for $\boldsymbol{f}$, then $\boldsymbol{h}$ is a bound for $\boldsymbol{f}$. So in the above diagram we have that $(\mathsf{id}_{H''}, \boldsymbol{h})$ is an $\boldsymbol{h}$-relative pushout for $\boldsymbol{f}$ and we simply say that $\boldsymbol{h}$ is an *idem pushout* for $\boldsymbol{f}$, written $\mathsf{IPO}_{\boldsymbol{f}}(\boldsymbol{g})$.

Define the *multi-pushout* of a span $\boldsymbol{f}$ as the set of its idem pushouts, i.e.:

$$\mathsf{Mpo}(\boldsymbol{f}) =_{\mathsf{def}} \{\boldsymbol{g} \mid \mathsf{IPO}_{\boldsymbol{f}}(\boldsymbol{g})\}$$

The following proposition states that any bound for a span $\boldsymbol{f}$ factors uniquely (up-to iso) through one member of $\mathsf{Mpo}(\boldsymbol{f})$. In other words, elements of the multi-pushout of $\boldsymbol{f}$ are either isomorphic, or conflicting.

Proposition 4. *Let $\boldsymbol{g}$ be a bound for $\boldsymbol{f}$. For all $\boldsymbol{h}, \boldsymbol{h}' \in \mathsf{Mpo}(\boldsymbol{f})$ if there exists k, k' such that $k\boldsymbol{hf} = k'\boldsymbol{h}'\boldsymbol{f} = \boldsymbol{gf}$ then there exists a unique iso ϕ such that $\phi\boldsymbol{hf} = \boldsymbol{h}'\boldsymbol{f}$.*

Proof. (sketch). The proof is a straightforward application of the relative pushout properties. Since both $\boldsymbol{h}$ and $\boldsymbol{h}'$ are $\boldsymbol{g}$-relative pushouts, there is a unique morphism j and a unique morphism j' such that $j\boldsymbol{hf} = \boldsymbol{h}'\boldsymbol{f}$ and $\boldsymbol{hf} = j'\boldsymbol{h}'\boldsymbol{f}$. Since j and j' are injective they are also isos. $\square$

We will need one final construction which defines the *gluings* of two graphs. It is obtained by first using the pullback construction, and then building a multi-pushout:

Definition 4 (gluing). *Let:*

$$\mathsf{Inter}(G, H) =_{\mathsf{def}} \{\boldsymbol{f} : \langle f_1 : I \to G, f_2 : I \to H\rangle \mid I \neq \emptyset \wedge \exists \boldsymbol{g} \;\; s.t\; \boldsymbol{f} = \mathsf{pb}(\boldsymbol{g})\}$$

We define:

$$\mathsf{Gluings}(G, H) =_{\mathsf{def}} \bigcup_{\boldsymbol{f} \in \mathsf{Inter}(G,H)} \mathsf{Mpo}(\boldsymbol{f})$$

We conclude this section by illustrating Fig. 3, the concept of multi-pushout in the context of Kappa, previously described in Ref. [13].

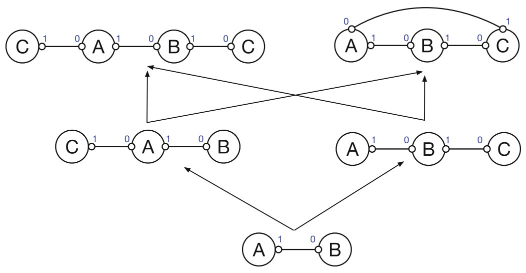


Fig. 3. The multi-pushout of the lower span contains 2 possible bounds (up-to iso). All closed diagrams are commuting.

4.2 Extension and Matching

We wish now to define a way to capture a notion of "abstract exploration". Such exploration is defined by means of *extension steps* (see Definition 7 of this section) along a statically computed "chart", called an *extension basis*. We illustrate Fig. 4 the main ideas of this extension basis. In order to *build* an extension basis, we need a first notion of morphism equivalence called *extension equivalence* that equates morphisms that denote the same "way" of discovering a graph starting from a smaller one. In order to *use* the extension basis during the update procedure we need a second notion of equivalence called *matching equivalence* that equates morphisms that denote the same instance (or match) of a graph into the concrete state. With extension and matching morphisms, we define a procedure, called extension step, that produces an exploration of the concrete domain, which is at the core of the update procedure (see Fig. 4 for an example).

Definition 5 (Extension equivalence). *Two morphisms $f : G \to H$ and $g : G \to H'$ are equivalent with respect to* extension*, written $f \sim_{\mathsf{ext}} g$, if there exists an iso $\phi : H \to H'$ that makes the right diagram commutes. Whenever $f \sim_{\mathsf{ext}} g$ we say that f and g denote the same* extension*.*

$$\begin{tikzcd} G \arrow[r, "f"] \arrow[dr, "g"'] & H \arrow[d, "\phi"] \\ & H' \end{tikzcd}$$

Extension classes come with the dual notion of *matching classes* that enables one to count different instances of a graph into another one.

Definition 6 (Matching equivalence). *Two morphisms $f : G \to H$ and $g : G' \to H$ are equivalent with respect to* matching*, written $f \sim_{\mathsf{mat}} g$, if there exists an iso $\phi : G \to G'$ that makes the right diagram commutes. Whenever $f \sim_{\mathsf{mat}} g$ we say that f and g denote the same* match*.*

$$\begin{tikzcd} G \arrow[r, "f"] \arrow[d, "\phi"'] & H \\ G' \arrow[ur, "g"'] & \end{tikzcd}$$

Another way to describe matching equivalence between f and g is that their codomain coincide:

Property 3. Two morphisms $f : G \to H$ and $g : G' \to H$ are matching equivalent if and only if $f(G) = g(G')$.

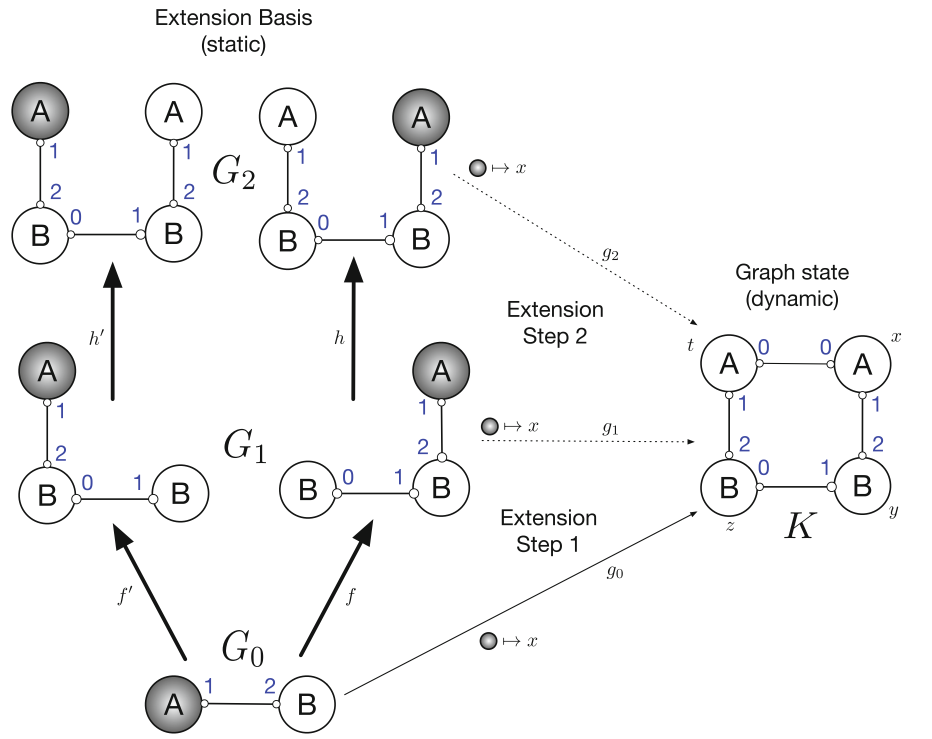


Fig. 4. An extension basis (thick arrows) describing how to "discover" G_1 and G_2 starting from G_0. The colored A node helps tracking the identity of A through the basis: note here that the basis has two distinct ways of discovering G_1 from G_0, each of which has its own extension into G_2. Given an initial match g_0 into K, one may extend g_0 into g_1 through f, and then g_1 into g_2 through the extension h. Note that f' fails to extend g_0 in K.

For all $f \in \mathsf{Hom}(\mathcal{G})$, for all $g, h \in \mathsf{Hom}(\mathcal{G})$, we say that g is *extended* by f into h, whenever $g = hf$. We say that the extension is *trivial* when f is an iso.

Importantly, two maps $g : G \to K$ and $g' : G \to K$ denoting the *same* match can be respectively extended by a map $f : G \to H$ into *distinct* matches of H into K (Fig. 6, left diagram). Similarly, two *distinct* matches $g : G \to K$ and $g' : G \to K$ might be extended by $f : G \to H$ into maps that denote the *same* match (Fig. 6, right diagram).

Definition 7 (Extension step). *Let $\Gamma \subseteq \mathsf{Hom}(G, K)$ and $\Gamma' \subseteq \mathsf{Hom}(H, K)$ for some $G, H, K \in \mathcal{G}$. For all $\mathcal{F} \subseteq \mathsf{Hom}(\mathcal{G})$, the pair (Γ, Γ') defines an $\mathcal{F}$-*extension step *if $\Gamma' = Ext_{\mathcal{F},K}(\Gamma)$ with:*

$$Ext_{\mathcal{F},K}(\Gamma) =_{\mathsf{def}} \{h : H \to K \mid \exists (f : G \to H) \in \mathcal{F}, \exists g \in \Gamma : g = hf\}$$

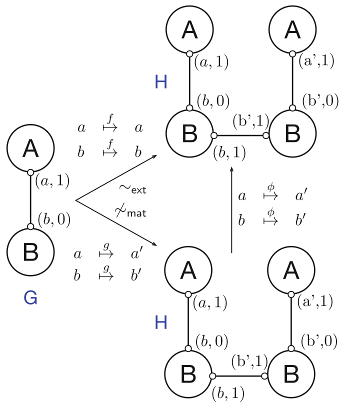

A
B
(a,1)
(b,0)
G
H
(b',1)
(b,1)
(a',1)
(b',0)
$a \overset{f}{\mapsto} a$
$b \overset{f}{\mapsto} b$
$a \overset{g}{\mapsto} a'$
$b \overset{g}{\mapsto} b'$
$a \overset{\phi}{\mapsto} a'$
$b \overset{\phi}{\mapsto} b'$
$\sim_{\mathsf{ext}}$
$\not\sim_{\mathsf{mat}}$

Fig. 5. Two morphisms f and g that belong to the same extension class (since $f = \phi g$) but define two distinct matchings of G in H (there is no iso ψ such that $f = g\psi$).

For all set $\mathcal{S}$ and $\approx \subseteq \mathcal{S} \times \mathcal{S}$ an equivalence relation over elements of $\mathcal{S}$, we define:

$$[\mathcal{S}]_{\approx} =_{\mathsf{def}} \{\mathcal{S}' \subseteq \mathcal{S} \mid \forall s \in \mathcal{S}, \exists ! s' \in \mathcal{S}' : s \approx s'\}$$

Definition 8 (Extension basis). *Consider a set of morphisms $\mathcal{F} \subseteq \mathsf{Hom}(\mathcal{G})$. We say that $\mathcal{F}$ is an* extension basis *if it satisfies $[\mathcal{F}]_{\mathsf{ext}} = \{\mathcal{F}\}$.*

In Definition 7, $Ext_{\mathcal{F},K}(\Gamma)$ extends an arbitrary set of maps Γ into *all* possible extensions of $g \in \Gamma$ by a map in $\mathcal{F}$. This raises two issues: first, we wish to build extension steps between sets of *matches* into K and not all equivalent ways of denoting the same match. So one may wonder what one obtains if, instead of computing $Ext_{\mathcal{F},K}(\Gamma)$, one were to compute $Ext_{\mathcal{F},K}(\Gamma')$ for some $\Gamma' \in [\Gamma]_{\mathsf{mat}}$. Second, the set $\mathcal{F}$ might be arbitrarily large, and we wish to compute the same extension step with a smaller set of maps.

The Extension theorem below provides an answer to these two issues: computing $Ext_{\mathcal{F},K}(\Gamma)$ is essentially equivalent to computing $Ext_{\mathcal{F}',K}(\Gamma')$ if one picks $\mathcal{F}' \in [\mathcal{F}]_{\mathsf{ext}}$ and $\Gamma' \in [\Gamma]_{\mathsf{mat}}$.

Theorem 2 (Extension). *Let $\mathcal{F} \subseteq \mathsf{Hom}(G, H)$, and $\Gamma \subseteq \mathsf{Hom}(G, K)$. For all $\Gamma' \in [\Gamma]_{\mathsf{mat}}$, for all extension basis $\mathcal{F}' \in [\mathcal{F}]_{\mathsf{ext}}$, we have:*

$$Ext_{\mathcal{F}',K}(\Gamma') = \emptyset \iff Ext_{\mathcal{F},K}(\Gamma) = \emptyset \tag{7}$$

$$[Ext_{\mathcal{F}',K(\Gamma')}]_{\mathsf{mat}} \subseteq [Ext_{\mathcal{F},K(\Gamma)}]_{\mathsf{mat}} \tag{8}$$

Importantly, replacing $\mathcal{F}$ by an arbitrary extension basis and Γ by an arbitrary member of $[\Gamma]_{\mathsf{mat}}$ is not a neutral operation with respect to extension step. However the resulting set of maps is *indistinguishable* from $Ext_{\mathcal{F},K}(\Gamma)$ if one equates matching equivalent maps.

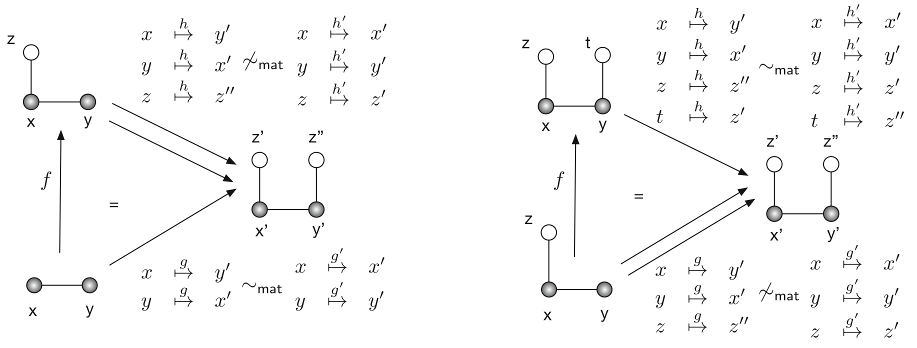


Fig. 6. Left diagram (loosing symmetry): two equivalent matches g and g' can be extended by f into two distinct matches. Right diagram (gaining symmetry): two distinct matches g and g' can be extended by f into the same match.

Notice also that although $[\Gamma']_{\mathsf{mat}} = \{\Gamma'\}$ (Γ' is already stripped of any redundant map), in general $[Ext_{\mathcal{F}',K}(\Gamma')]_{\mathsf{mat}} \neq \{Ext_{\mathcal{F}',K}(\Gamma')\}$ because one extension step, even using purely non equivalent extensions, may produce matching-equivalent maps (see Fig. 6, right example). However we can prove that it is not possible to extend the same map g into two matching-equivalent h and h' unless one has used extension-equivalent maps to do so:

$$\begin{tikzcd}
 & H \arrow[dr, "h"] \arrow[d, dotted, "\phi"] & \\
G \arrow[ur, "f"] \arrow[dr, "f'"'] & H' \arrow[r, phantom, "="] & K \\
 & &
\end{tikzcd}$$

$$\begin{tikzcd}
 & H \arrow[dr, "h"] \arrow[dd, dotted, "\phi"] \arrow[ddr, phantom, "="] & \\
G \arrow[ur, "f"] \arrow[dr, "f'"'] & & K \\
 & H' \arrow[ur, "h'"'] &
\end{tikzcd}$$

Proposition 5. *Consider the commuting diagram on the right, if there exists an iso $\phi : H \to H'$ such that $h = h'\phi$, then $\phi f = f'$. As a consequence $h \sim_{\mathsf{mat}} h' \implies f \sim_{\mathsf{ext}} f'$.*

Proof. By hypothesis we have $hf = h'f'$. Suppose we have $h = h'\phi$ for some iso ϕ. So we have $h'\phi f = h'f'$ by substituting h in the hypothesis. Since h' is injective, we deduce $\phi f = f'$ and $f \sim_{\mathsf{ext}} f'$ follows by Definition 5. □

In combination with the example of Fig. 6 (right diagram), this proposition essentially guarantees that, when using an extension basis, the only way to produce matching-equivalent extensions is to start from two maps that were not matching equivalent. This remark will become handy when we describe our update algorithm in Sect. 5.

4.3 Proof of the Extension Theorem

We begin by a lemma that shows one cannot lose any match into K after the extension step if one disregards extension-equivalent maps in $\mathcal{F}$:

$$\begin{tikzcd}
 & H \arrow[dr, "h"] \arrow[dd, dotted, "\phi"] & \\
G \arrow[ur, "f"] \arrow[dr, "f'"'] \arrow[r, phantom, "="] & \, & K \\
 & H' \arrow[ur, "h'"'] &
\end{tikzcd}$$

Lemma 2. *Consider the commuting diagram on the right, if there exists an iso $\phi : H \to H'$ such that $f' = \phi f$, then there exists $h'' \sim_{\mathsf{mat}} h'$ such that $g = h''f$.*

Eventually we need a Lemma that shows it is also not possible to lose a match into K after the extension step, if one disregards matching-equivalent maps in Γ.

Lemma 3. *Let $g : G \to K$, $f : G \to H$ and $h : H \to K$ an f-extension of g. The following proposition holds:*

$$\forall g' \sim_{\mathsf{mat}} g, \exists h' \sim_{\mathsf{mat}} h \text{ s.t either } \begin{cases} g' = h'f \text{ or} \\ \exists f' \not\sim_{\mathsf{ext}} f \text{ s.t } g' = h'f' \end{cases}$$

We are now in position to prove the Extension Theorem.

Proof (Theorem 2). We first prove Eq. (7).

- Suppose $Ext_{\mathcal{F}',K}(\Gamma') = \emptyset$, by definition this implies:

$$\{h \mid \exists f \in \mathcal{F}' : hf \in \Gamma'\} = \emptyset$$

  This can be either true because $\Gamma' = \emptyset$ (point 1) or because no f in $\mathcal{F}'$ satisfies $hf \in \Gamma'$ for some h (point 2).
  1. Since $\Gamma' = \emptyset$ and $\Gamma' \in [\Gamma]_{\mathsf{mat}}$, we have that $\Gamma = \emptyset$. In turn, this entails that $Ext_{\mathcal{F},K}(\Gamma) = \emptyset$.
  2. Looking for a contradiction, suppose that some $f \in \mathcal{F}$ is such that $hf \in \Gamma$ for some h. Since we supposed that no $f \in \mathcal{F}'$ satisfies $hf \in \Gamma'$ necessarily $f \notin \mathcal{F}'$. But since $\mathcal{F}' \in [\mathcal{F}]_{\mathsf{ext}}$ there exists $f' \in \mathcal{F}'$ such that $f' \sim_{\mathsf{ext}} f$. According to Lemma 2, there exists $h' \sim_{\mathsf{mat}} h$ such that $h'f' \in \Gamma'$ which entails a contradiction. Therefore no f in $\mathcal{F}$ satisfies $hf \in \Gamma$ for any h and $\Gamma = \emptyset$.
- Suppose that $Ext_{\mathcal{F},K}(\Gamma) = \emptyset$. Since $\Gamma' \subseteq \Gamma$ it follows immediately that $Ext_{\mathcal{F}',K}(\Gamma') \subseteq Ext_{\mathcal{F}',K}(\Gamma')$ and hence $Ext_{\mathcal{F}',K}(\Gamma') = \emptyset$. □

We now prove Eq. (8). Note that it is equivalent to proving:

$$\forall h \in Ext_{\mathcal{F},K}(\Gamma), h \notin Ext_{\mathcal{F}',K}(\Gamma') \implies \exists h' \sim_{\mathsf{mat}} h : h' \in Ext_{\mathcal{F}',K}(\Gamma') \tag{9}$$

So let us suppose there is some h such that $h \in Ext_{\mathcal{F},K}(\Gamma)$ and $h \notin Ext_{\mathcal{F}',K}(\Gamma')$. Recall that $h \in Ext_{\mathcal{F},K}(\Gamma)$ implies that $hf \in \Gamma$ for some $f \in \mathcal{F}$. In addition, $h \notin Ext_{\mathcal{F}',K}(\Gamma')$ implies that for all $f' \in \mathcal{F}'$, $hf' \notin \Gamma'$. Now there are several cases to consider:

- $f \in \mathcal{F}'$ and $hf \in \Gamma'$. This would imply that $h \in Ext_{\mathcal{F}',K}(\Gamma')$ which would contradict our hypothesis.
- $f \in \mathcal{F}'$ and $hf \notin \Gamma'$. Since $\Gamma' \in [\Gamma]_{\mathsf{mat}}$ we know there exists $g \in \Gamma'$ such that $g \sim_{\mathsf{mat}} hf$. We apply Lemma 3 to deduce that there exists $h' \sim_{\mathsf{mat}} h$ and f' such that $h'f' = g$. Still according to Lemma 3, either $f' = f$ (point 1) or $f' \not\sim_{\mathsf{ext}} f$ (point 2).
  1. Since $f \in \mathcal{F}'$ we have that $h' \in Ext_{\mathcal{F}',K}(\Gamma')$.

2. Since $\mathcal{F}' \in [\mathcal{F}]_{\mathsf{ext}}$, $f' \not\sim_{\mathsf{ext}} f$ implies there is $f'' \sim_{\mathsf{ext}} f'$ such that $f'' \in \mathcal{F}'$. We apply Lemma 2 to deduce that there exists $h'' \sim_{\mathsf{mat}} h'$ such that $f''h'' \in \Gamma'$. By transitivity of $\sim_{\mathsf{mat}}$ we have $h'' \sim_{\mathsf{mat}} h$ and we do have $h'' \in Ext_{\mathcal{F}',K}(\Gamma')$.

– $f \notin \mathcal{F}'$ and $hf \in \Gamma'$. Since $\mathcal{F}' \in [\mathcal{F}]_{\mathsf{ext}}$ we know there exists $f' \in \mathcal{F}'$ that satisfies $f' \sim_{\mathsf{ext}} f$. We apply Lemma 2 to deduce that there is $h' \sim_{\mathsf{mat}} h$ such that $h'f' \in \Gamma'$. It entails that $h' \in Ext_{\mathcal{F}',K}(\Gamma')$.

– $f \notin \mathcal{F}'$ and $hf \notin \Gamma'$ and we proceed by combining the arguments of the two previous points. □

5 The Update Algorithm

In this section we show how to utilize extension bases and extension steps to implement the incremental update function specified in Sect. 3. We describe Fig. 7 the interplay of extension steps and exploration of the concrete domain.

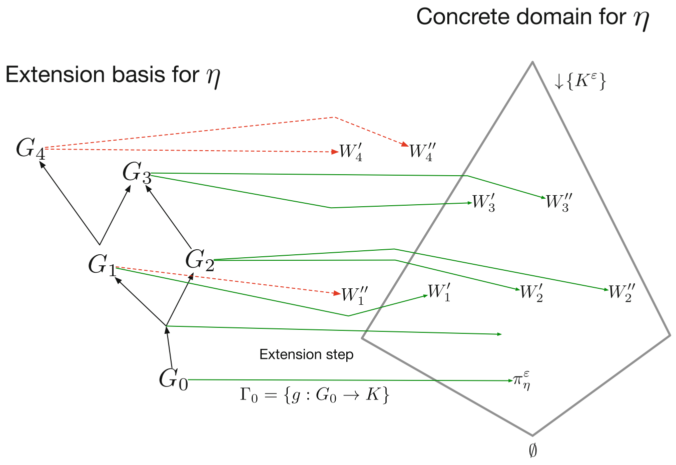


Fig. 7. Extension steps and domain exploration. The occurrence of η on K provides a map $g : G_0 \to K$ and the concrete identity of G_0 in K. The algorithm looks for all possible extension steps above G_0 in the statically computed basis. The extensions that succeed are represented with plain line arrows. Those that fail are represented with dotted line arrows. For instance no extension step is able to provide a match for G_4 in K.

5.1 Abstract Effects

Graph rewriting systems are given as a set of rewriting rules of the form

$$\boldsymbol{r} : L \rightharpoonup R$$

where $\boldsymbol{r}$ is a partial map between $L \in \mathcal{G}$ and $R \in \mathcal{G}$. Formally such a partial map is given as a span $\boldsymbol{r} = \langle \mathsf{lhs} : D \to L, \mathsf{rhs} : D \to R \rangle$ where $D \in \mathcal{G}$ is the domain of definition of $\boldsymbol{r}$ and lhs (resp. rhs) stands for the left hand side map of $\boldsymbol{r}$ (resp. right hand side). For all such span $\boldsymbol{r}$, we define:

$$H^+_{\boldsymbol{r}} =_{\mathsf{def}} R \setminus \mathsf{rhs}(D) \qquad H^-_{\boldsymbol{r}} =_{\mathsf{def}} L \setminus \mathsf{lhs}(D)$$

Definition 9 (abstract effect). *Let $\boldsymbol{r}$ be a rule. The* abstract effect *of $\boldsymbol{r}$, written $\eta^\sharp_{\boldsymbol{r}}$, is the pair of maps:*

$$\eta^\sharp_{\boldsymbol{r}} =_{\mathsf{def}} \langle f^-_{\boldsymbol{r}} : H^-_{\boldsymbol{r}} \to L, f^+_{\boldsymbol{r}} : H^+_{\boldsymbol{r}} \to R \rangle$$

where $f^\varepsilon_{\boldsymbol{r}}$ is the identity on its domain.

We give Fig. 8 an example of the derivation of an abstract effect from a Kappa rule.

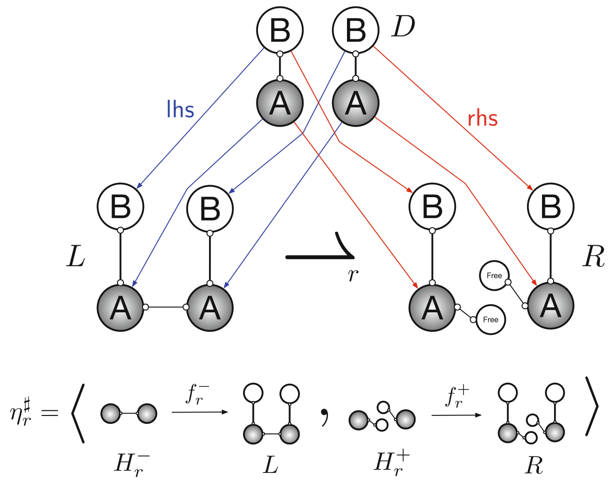


Fig. 8. Deriving an abstract effect from a Kappa rule. The Kappa rule is given as a partial map $\boldsymbol{r} : L \rightharpoonup R$ (upper part). The corresponding abstract effect is a pair of maps $(f^-_{\boldsymbol{r}}, f^+_{\boldsymbol{r}})$ describing respectively the edges that are removed and added by the rule.

Definition 10 (K-occurrence). *Consider an abstract effect*

$$\eta^\sharp_{\boldsymbol{r}} = \langle f^-_{\boldsymbol{r}} : H^-_{\boldsymbol{r}} \to L, f^+_{\boldsymbol{r}} : H^+_{\boldsymbol{r}} \to R \rangle$$

*For all concrete state K, a K-*occurrence *$m_{K,\boldsymbol{r}}$ of $\eta^\sharp_{\boldsymbol{r}}$ is a pair of maps:*

$$m_{K,\boldsymbol{r}} =_{\mathsf{def}} (g^- : L \to K, g^+ : R \to K)$$

and we write $m_{K,\boldsymbol{r}}(\eta^\sharp_{\boldsymbol{r}}) = \eta$ whenever:

$$\eta = (g^-(L), g^- f^-_{\boldsymbol{r}}(H^-_{\boldsymbol{r}}), g^+ f^+_{\boldsymbol{r}}(H^+_{\boldsymbol{r}}))$$

5.2 Extension Basis Synthesis

Recall from Sect. 2.3 that we consider a set $\mathcal{O} \subseteq \mathcal{G}$ of *observable graphs*. Since the observables (including any match of the left hand sides of a rule) are intentionally given as a finite set of abstract graphs (see Sect. 2), we may assume that every elements of $[\mathcal{O}]_{\mathsf{iso}}$ is finite, where $\sim_{\mathsf{iso}}$ is the graph isomorphism equivalence relation. Let us thus consider an arbitrary $\hat{\mathcal{O}} \subseteq \mathcal{G}$ such that $\hat{\mathcal{O}} \in [\mathcal{O}]_{\mathsf{iso}}$.

For all rule $\boldsymbol{r}$, we define now the procedure to build a *negative and positive* $\boldsymbol{r}$*-extension basis*, respectively $\mathfrak{B}_{\boldsymbol{r}}^{+} \subseteq \mathsf{Hom}(\mathcal{G})$ and $\mathfrak{B}_{\boldsymbol{r}}^{-} \subseteq \mathsf{Hom}(\mathcal{G})$.

Similarly to Sect. 3 we adopt the following naming convention: for all $r : L \rightharpoonup R$ we write $\pi_{\boldsymbol{r}}^{+} =_{\mathsf{def}} R$ and $\pi_{\boldsymbol{r}}^{-} =_{\mathsf{def}} L$ and the superscript $\_^{\varepsilon}$ should be globally replaced by either $\_^{+}$ or $\_^{-}$ to specialize a definition to the positive or negative update.

Procedure 1: *"Backbone" extension basis synthesis* (see Sect. 4.1 for the categorical constructions used in the procedure).

Input: a rule $\boldsymbol{r}$ and the set of (abstract) observables $\hat{\mathcal{O}}$.

```
1. Compute the abstract effect $\eta_{\boldsymbol{r}}^{\sharp} = \langle f_{\boldsymbol{r}}^{-} : H_{\boldsymbol{r}}^{-} \to \pi_{\boldsymbol{r}}^{-}, f_{\boldsymbol{r}}^{+} : H_{\boldsymbol{r}}^{+} \to \pi_{\boldsymbol{r}}^{+} \rangle$
2. For all $O \in \hat{\mathcal{O}}$, build $\Lambda_{O}^{\varepsilon} \in [\mathsf{Gluings}(H_{\boldsymbol{r}}^{\varepsilon}, O)]_{\mathsf{ext}}$
3. For all $\langle f : O \to G, h : H_{\boldsymbol{r}}^{\varepsilon} \to G \rangle \in \Lambda_{O}^{\varepsilon}$, build $\Lambda_{O,h}^{\varepsilon} \in [\mathsf{Mpo}(h, f_{\boldsymbol{r}}^{\varepsilon})]_{\mathsf{ext}}$
4. $\mathfrak{B}_{\boldsymbol{r}}^{\varepsilon} =_{\mathsf{def}} \{f \mid \exists O, \exists h, \exists g \text{ s.t } \langle g, f \rangle \in \Lambda_{O,h}^{\varepsilon}\}$
5. return $\mathfrak{B}_{\boldsymbol{r}}^{\varepsilon}$
```

An important point with respect to combinatorial explosion is the following:

Proposition 6. *At step 3 and for all h, $\Lambda_{O,h}^{\varepsilon}$ contains at most one element.*

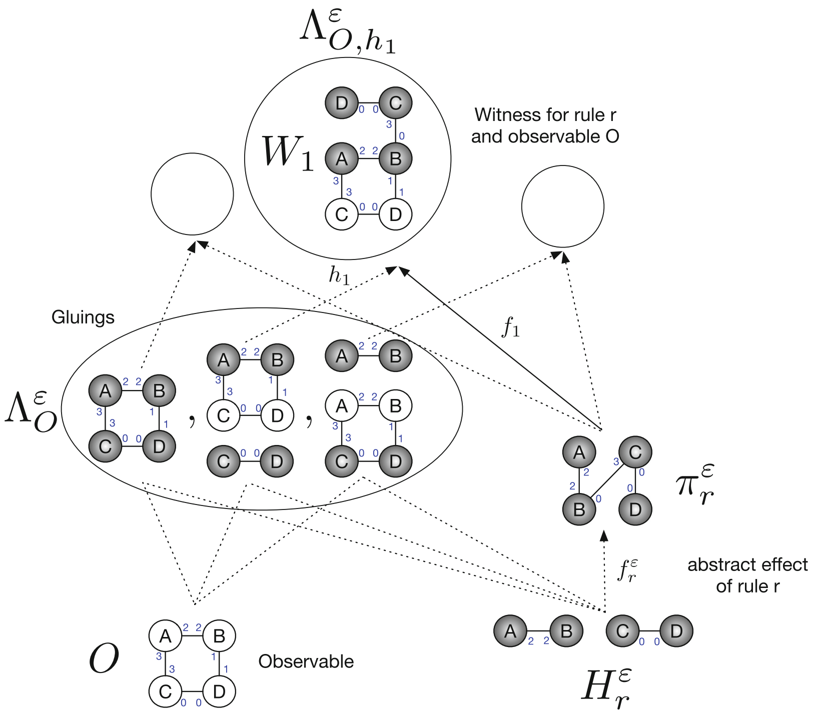
In a nutshell, at step 2 one computes all possible gluings of $H_{\boldsymbol{r}}^{\varepsilon}$ with some observable O. At step 3 we build abstract witnesses by means of multi-pushout construction. Finally step 4 assembles into $\mathfrak{B}_{\boldsymbol{r}}^{\varepsilon}$ all extensions $f : \pi_{\boldsymbol{r}}^{\varepsilon} \to W$ that are the left component of an idem-pushout built in the previous step. We provide and example, Fig. 9, of the construction of the "backbone" extension basis in the context of Kappa.

We call this extension basis a "backbone" because it only contains direct extensions from $\pi_{\boldsymbol{r}}^{\varepsilon}$ to some witness. We will see shortly how to enrich this backbone basis into a new basis that takes into account sharing between witnesses.

In the meantime, we may readily state a lemma that guarantees that extensions steps along $\mathfrak{B}_{\boldsymbol{r}}^{\varepsilon}$ produce concrete witnesses. Consider a basis $\mathfrak{B}_{\boldsymbol{r}}^{\varepsilon}$ build from a rule $\boldsymbol{r}$ and a set of abstract observables following Procedure 1. Recall that any K-occurrence of $\eta_{\boldsymbol{r}}^{\sharp}$ is a pair of maps (g_0^{-}, g_0^{+}) that identify the edges that are respectively removed and added in K. Whenever the $\mathfrak{B}_{\boldsymbol{r}}^{\varepsilon}$-extension of $\{g_0^{\varepsilon}\}$ (see Definition 7) builds a non empty set Γ of witness matches into K, then those matches indeed provide η-witnesses (see Sect. 3.1) that are also below K^{ε}.

Lemma 4 (Soundness). *Let $\boldsymbol{r}$ be a rule with an abstract effect $\eta_{\boldsymbol{r}}^{\sharp}$. Let also $m_{K,r} =_{\mathsf{def}} (g_0^{-}, g_0^{+})$ be a K-occurrence of $\boldsymbol{r}$ with $\eta = m_{K,r}(\eta_{\boldsymbol{r}}^{\sharp})$ a concrete effect. For all $(h : W \to K) \in Ext_{\mathfrak{B}_{\boldsymbol{r}}^{\varepsilon}, K^{\varepsilon}}(\{g_0^{\varepsilon}\})$, there exists $f_K : O \to K$ such that:*

$$h(W) \rhd_{\eta}^{\varepsilon} f_K(O)$$



$\Lambda^{\varepsilon}_{O,h_1}$
W_1
Witness for rule r and observable O
h_1
f_1
Gluings
$\Lambda^{\varepsilon}_{O}$
π^{ε}_{r}
f^{ε}_{r}
abstract effect of rule r
O
Observable
H^{ε}_{r}

Fig. 9. Construction of the "backbone" extension basis $\mathfrak{B}^{\varepsilon}_{r} = \{f_1\}$. Grey coloring of nodes helps tracking nodes of H^{ε}_{r} through the morphisms (all closed diagrams are commuting). In dotted line, the maps that are used for the construction of the basis but that are not morphisms of $\mathfrak{B}^{\varepsilon}_{r}$. Consistently with Proposition 6, the multi-pushouts in the upper part of the diagram have at most one element ($\Lambda^{\varepsilon}_{O,h_1} = \{W_1\}$, the other gluings are incompatible with π^{ε}_{r}).

A simple saturation procedure enables one to add sharing between graphs of the "backbone" extension basis we have constructed so far:

Procedure 2: *Add sharing to an extension basis.*

Input: an extension basis $\mathfrak{B}$.

1. if there exists $f, f' \in \mathfrak{B}$ and $g, h, h' \notin \mathfrak{B}$ such that $f = hg$ and $f' = h'g$ then $\mathfrak{B} = \mathfrak{B} \cup \{g, h, h'\}$ and go to 1.
2. else return $\mathfrak{B}$.

We write $G <^{1}_{\mathfrak{B}} H$ if there exists $f \in \mathfrak{B}$ such that $f : G \to H$. The relation $\leq_{\mathfrak{B}}$ is the transitive and reflexive closure of $<^{1}_{\mathfrak{B}}$ and denotes a partial order.

5.3 Implementing the Incremental Update Function

This section is dedicated to the implementation of the incremental update function, according to the specification that was given Sect. 3.3. The algorithm relies on a pre-computation of the $\boldsymbol{r}$-extension bases (with sharing) of all rule $\boldsymbol{r}$ contained in the rule set:

Procedure 3: *Compute the extension bases.*

Input: a finite rule set $\mathcal{R}$.

1. For all $r \in \mathcal{R}$, build $\mathfrak{B}^{\varepsilon}_{\boldsymbol{r}}$ following Procedure 1.
2. For all $\mathfrak{B}^{\varepsilon}_{\boldsymbol{r}}$, add sharing following Procedure 2.
3. return $\bigcup_{\boldsymbol{r}}\{\mathfrak{B}^{\varepsilon}_{\boldsymbol{r}}\}$.

Each time a rule $\boldsymbol{r}$ is applied to a graph K, one obtains the corresponding K-occurrence $m_{K,\boldsymbol{r}} = (g_0^+ : \pi_{\boldsymbol{r}}^+ \to K, g_0^+ : \pi_{\boldsymbol{r}}^+ \to K)$. We use g_0^{ε} as an input to the update procedure, which is a breadth-first traversal of the extension basis $\mathfrak{B}^{\varepsilon}_{\boldsymbol{r}}$:

Procedure 4: *Incremental update.*

Input: a basis $\mathfrak{B}^{\varepsilon}_{\boldsymbol{r}}$, a *$K$-occurrence $g_0^{\varepsilon} : \pi^{\varepsilon}_{\boldsymbol{r}} \to K$* of *$\eta^{\sharp}_{\boldsymbol{r}}$, and a predicate $w : \mathcal{G} \to 2$ such that $w(G)$ holds if G is an abstract witness of $\mathfrak{B}^{\varepsilon}_{\boldsymbol{r}}$.* For all set of morphisms $\mathcal{F} \subseteq \mathfrak{B}^{\varepsilon}_{\boldsymbol{r}}$ we define:

$$\min(\mathcal{F}) =_{\mathsf{def}} \{f \in \mathcal{F} \mid f : G \to H \wedge \forall (g : G' \to H') \in \mathcal{F} : H' \not\leq_{\mathfrak{B}} H\}$$

```
 1. Initialize $\gamma : \mathcal{G} \to \mathcal{P}(\mathsf{Hom}(\mathcal{G}))$ as $\gamma(G) := \emptyset$ for all $G \neq \pi^{\varepsilon}_{\boldsymbol{r}}$ and $\gamma(\pi^{\varepsilon}_{\boldsymbol{r}}) := \{g_0\}$
 2. $\mathcal{F} := \emptyset$ (for the extensions yet to explore), $x := \pi^{\varepsilon}_{\boldsymbol{r}}$ (for the current point in
    the basis) and $\mathcal{W} := \emptyset$ (for the reached witnesses).
 3. if $w(x)$ then $\mathcal{W} := \mathcal{W} \cup \{x\}$
 4. for all $(f : x \to G) \in \mathfrak{B}^{\varepsilon}_{\boldsymbol{r}}$ do $\mathcal{F} := \mathcal{F} \cup \{f\}$
 5. if $\mathcal{F} \neq \emptyset$ then
 6.       choose $(f : G \to H) \in \min(\mathcal{F})$
 7.       $\gamma(H) := [\gamma(H) \cup \mathit{Ext}_{\{f\},K^{\varepsilon}}(\gamma(G))]_{\mathsf{mat}}$
 8.       $\mathcal{F} := \mathcal{F} \backslash \{f\}$ and $x := H$
 9.       go to step 3.
10. else return $\mathcal{W}^{\varepsilon}_{\eta,K}$ where:
```

$$\mathcal{W}^{\varepsilon}_{\eta,K} := \bigcup \{W \mid \exists G \in \mathcal{W}, \exists g \in \gamma(G) : g(G) = W\}.$$

The procedure builds the function γ that maps the graphs of the basis to the matches they have in K. Initially only $\pi^{\varepsilon}_{\boldsymbol{r}}$ has a match given by g_0^{ε} and γ is updated at step 7 each time an extension step is performed. We give Fig. 10 an example of the construction of the map γ for a specific extension basis.

We conclude this section by proving that the above procedure complies with the specification of the incremental update function given Sect. 3.3.

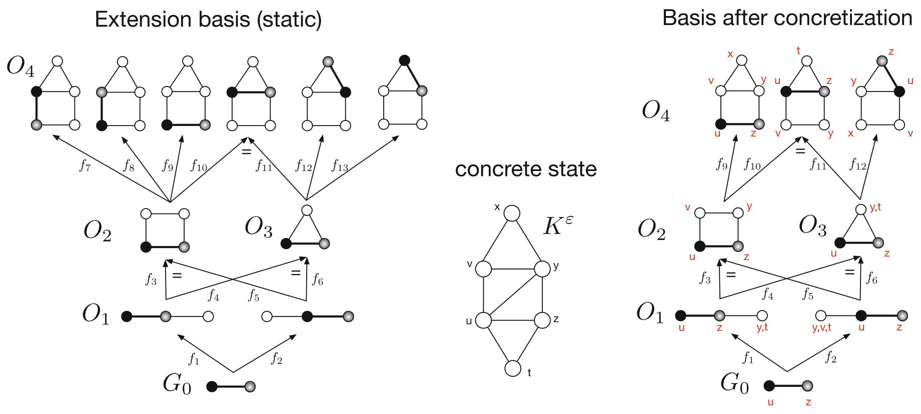

Extension basis (static)
Basis after concretization
concrete state
O_4
O_2
O_3
O_1
G_0
K^ε
f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 f_9 f_{10} f_{11} f_{12} f_{13}
x y v u z t
y,t
y,v,t

Fig. 10. An extension basis (left) for the observables O_1 (line), O_2 (square), O_3 (triangle) and O_4 (house) and $\pi_r^\varepsilon = H^\varepsilon = G_0$ (creation or deletion of a single edge). A concrete graph state K^ε (middle) and a representation of the final γ map (right). The concretization is performed using the initial match $G_0 \mapsto \{u, z\}$ (the edge $\{u, z\}$ of K^ε has been created or deleted by the effect occurrence). Colored nodes help tracking the identity of the nodes of G_0 (all closed diagrams are commuting). Overall 5 new instances of O_1 were found, 1 instance of O_2, 2 instances of O_3 and 3 instances of O_4.

5.4 Correctness Proof

We essentially need to follow the guidelines of Sect. 3.3. Let:

$$X(\gamma) =_{\mathsf{def}} \bigcup \{H \mid \exists (g : G \to K) \in \gamma(G) : g(G) = H\}$$

and

$$\mathcal{R}(\gamma, \mathcal{W}) =_{\mathsf{def}} \bigcup \{H \mid \exists W \in \mathcal{W}, \exists g \in \gamma(W) : g(W) = H\}$$

We write $\mathsf{inc}^\varepsilon_{\eta,K}(X_0, \mathcal{R}_0) = (X_1, \mathcal{R}_1)$ if at step 3 we have $X(\gamma) = X_0$ and $\mathcal{R}(\gamma, \mathcal{W}) = \mathcal{R}_0$ and the next values of $X(\gamma)$ and $\mathcal{R}(\gamma, \mathcal{W})$ are respectively X_1 and $\mathcal{R}_1$.

We begin by proving that $\mathsf{inc}^\varepsilon_{\eta,K}(X_0, \mathcal{R}_0) = (X_1, \mathcal{R}_1)$ satisfies the requirements for X_i. After one loop of the procedure we have two cases:

- if Step 5 was satisfied, then at step 7, $X_1 = X_0 \cup X$ where:

$$X = \bigcup \{G' \subseteq K^\varepsilon \mid \exists g \in \gamma(H) : G' = g(H)\}$$

  by construction $X \subseteq K^\varepsilon$ and $X \cap X_0 \neq \emptyset$ since a new extension step has been performed. Furthermore $X \in \mathcal{D}^\varepsilon(\eta)$ by construction of the basis: it is either itself an abstract witness or it is below some other witnesses. Therefore we satisfy Eq. (5) (Sect. 3.3).
- if Step 10 was satisfied then $X_0 = X_1$ since γ is not modified.

We need to prove now that $\mathsf{inc}^{\varepsilon}_{\eta,K}(X_0, \mathcal{R}_0) = (X_1, \mathcal{R}_1)$ satisfies the requirements for $\mathcal{R}_i$. The only step where $\mathcal{W}$ is modified is at step 3. Using Lemma 4 we know that for all $G \in \mathcal{W}$, we have:

$$g : W \to K \in \gamma(G) \implies g(W) \rhd^{\varepsilon}_{\eta} O$$

for some concrete observable $O \subseteq K^{\varepsilon}$. Thus we have either $\mathcal{R}_0 = \mathcal{R}_1$ or:

$$\mathcal{R}_1 = \mathcal{R}_0 \cup \{g(W) \mid g(W) \rhd^{\varepsilon}_{\eta} O\}$$

and therefore the Eq. (6) is also satisfied. □

6 Conclusion

We have investigated in this paper the problem of efficiently updating observable counts in a graph after a rewrite step has occurred. We believe our approach has several merits.

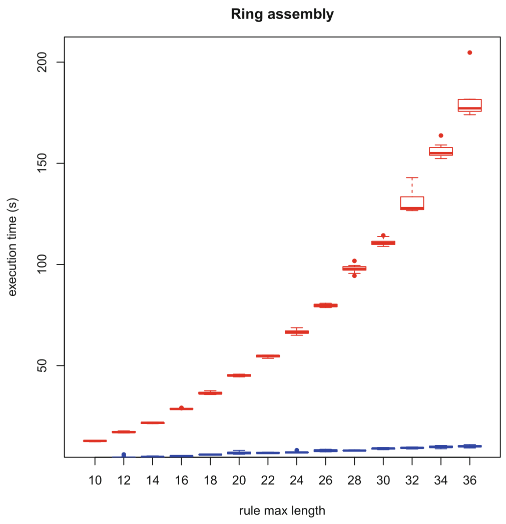


Fig. 11. A comparison between the average time of KaSim 3 runs (in red) vs. KaSim 4 runs (in blue) on successive variants of the "ring assembly" model. KaSim 4. Scales linearly with the maximal size of the largest observable (the left hand side of the largest rule), while KaSim 3. Scales with the total number of rules in each model. (Color figure online)

The first one is of methodological nature: to our knowledge it is the first attempt to describe a problem that is usually treated in a purely algorithmic fashion [14], using domain theoretic arguments for proofs and categorical constructions for the implementation. In particular algorithmic approaches tend to consider quite concrete graphs, represented by their adjacency matrices, while graph rewriting literature uses morphisms to track node identity. We have seen here that it is possible to conciliate both worlds through the interplay between extension maps and matchings.

The second merit is of qualitative nature. The incremental update procedure that is described in this paper has been implemented in the version 4.x of KaSim, a stochastic simulator for the rewriting of Kappa models. We show Fig. 11 a comparison between runs of KaSim 3 vs. 4. On variations of the "ring assembly model" that is designed to highlight the benefits of sub-graph sharing: the number of rules each variants of the model has, grows exponentially with the size of their largest left hand side: each variant of the model is characterized by the length of a ring-like graph it is trying to form. The first variant is forming all rings up-to length 10, while the last variant is forming all rings up-to length 36. Forming all possible rings up-to length n requires $!n$ rules, and the largest left hand side of these rules has length n.

As usual there are multiple continuations of this work one may envision. Just to mention a promising one, it would be interesting to see what happens if instead of incrementally maintaining $\mathcal{O}_K$ (the observable that are present in state K) one were to maintain $\downarrow\mathcal{O}_K$. In theory one could benefit from having partial observables already explored in order to minimize what remains to be discovered after an effect has occurred. From an implementation point of view this may lead to potentially memory intensive data structures but to very minimalist update phases.

Acknowledgments. This work was supported by the ANR grant ICEBERG (ANR-10-BINF-0006) and is partially sponsored by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under grant number W911NF-14-1-0367. The views, opinions, and/or findings contained in this paper are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

Appendix

Proofs Omitted in Section 3

Proof (Proposition 1). We first prove Eq. (2), $\Rightarrow$. By unfolding the def. of $W \rhd_\eta^- O$ and by $W \subseteq K$ we have $O \cup \mathsf{pre}(\eta) \subseteq K$. As a consequence, $O \subseteq K$ and by def. of Obs (Sect. 3) we have $O \in (\mathsf{Obs}\ K)$. Still by definition of $W \rhd_\eta^- O$ we have $O \cap H_\eta^- \neq \emptyset$ (i). By def. of $\eta \cdot K$ (Sect. 2.2) we have $O \backslash H_\eta^- \cup H_\eta^+ \subseteq \eta \cdot K$ (ii). In addition $H_\eta^- \cap H_\eta^+ = \emptyset$. By (i) and (ii) we have $O \nsubseteq \eta \cdot K$ and consequently $O \notin (\mathsf{Obs}\ \eta \cdot K)$.

We now prove Eq. (2), $\Leftarrow$. By def. $W = O \cup \mathsf{pre}(\eta)$ (i), and by hyp. we have $O \in (\mathsf{Obs}\ K)$ implies $O \subseteq K$ (ii). Moreover since $\eta \cdot K$ is defined we have $\mathsf{pre}(\eta) \subseteq K$ (iii). From (i)–(iii) we get $O \cup \mathsf{pre}(\eta) \subseteq K$. In order to conclude that $W = O \cup \mathsf{pre}(\eta) \in \mathcal{W}_\eta^-$ we need to additionally show that $O \cap H_\eta^- \neq \emptyset$.

Since $O \notin (\mathsf{Obs}\ \eta \cdot K)$ and $O \in (\mathsf{Obs}\ K)$ we have $O \not\subseteq (O \backslash H_\eta^- \cup H_\eta^+)$ (iv). Since $H_\eta^- \cap H_\eta^+ = \emptyset$ by def. of η, the only possibility to satisfy (iv) is $O \cap H_\eta^- \neq \emptyset$. $\square$

Proofs Omitted in Section 4

Proof (Lemma 2). It suffices to take $h'' = h'\phi$ which, by Definition 6 implies that $h'' \sim_{\mathsf{mat}} h'$. By hypothesis we have $g = h'f'$ and $f' = \phi f$. From these equalities we get $g = h'\phi f$. By substituting $h'\phi$ by h'' we obtain $g = h''f$. $\square$

Proofs Omitted in Section 5

Proof (Lemma 3). By hypothesis we start from the following commuting diagram:

Since $g' \sim_{\mathsf{mat}} g$, by Definition 6, we have $g = g'\phi$ for some iso ϕ. We have two cases:

$$\begin{tikzcd}
H \arrow[drr, "h"] & & \\
 & = & K \\
G \arrow[uu, "f"] \arrow[urr, "g"'] & &
\end{tikzcd}$$

- either f is ϕ-preserving and there exists an iso ψ such that $f\phi = \psi f$. Then by construction $h\psi^{-1}f = g'$ and we can conclude by noticing that $h' =_{\mathsf{def}} h\psi^{-1} \sim_{\mathsf{mat}} h$ (by Definition 6).
- or f is not ϕ-preserving and there is no iso ψ such that $f\phi = \psi f$. It entails that $f\phi \not\sim_{\mathsf{ext}} f$ (by Definition 5) and (by symmetry) $f' =_{\mathsf{def}} f\phi^{-1} \not\sim_{\mathsf{ext}} f$. Now we can conclude, since by construction $hf' = g'$. $\square$

Proof (Proposition 6). We have the following diagram:

$$\begin{tikzcd}
 & G & \pi_r^\varepsilon \\
O \arrow[ur, "h'"'] & = & H_r^\varepsilon \arrow[ul, "h"] \arrow[u, "f_r^\varepsilon"'] \\
 & I \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

where $\boldsymbol{h} = \langle h', h \rangle$ is a gluing of O and H_r^ε. Now the procedure attempts to build the multi-pushout of the span $\boldsymbol{f} = \langle h, f_r^\varepsilon \rangle$. Suppose it has at least two elements, we have the following diagram:

$$\begin{tikzcd}
 & U & U' \\
 & G \arrow[u, "i"] \arrow[ur, bend left, "j"] \arrow[r, phantom, "="] & \pi_r^\varepsilon \arrow[u, "l"'] \arrow[ul, bend right, "k"'] \\
O \arrow[ur, "h'"'] & = & H_r^\varepsilon \arrow[ul, "h"] \arrow[u, "f_r^\varepsilon"'] \\
 & I \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

where $\langle ih', kf_r^\varepsilon \rangle$ and $\langle jh', lf_r^\varepsilon \rangle$ are bounds for $\boldsymbol{h} = \langle h, h' \rangle$. By construction $\boldsymbol{h}$ is a relative pushout, therefore, by Proposition 4, there exists an iso $\phi : U \to U'$ that equates the bounds $\langle i, k \rangle$ and $\langle j, l \rangle$. This would entail $i \sim_{\mathsf{ext}} j$ which contradicts the hypothesis. □

Proof (Lemma 4). Recall from Sect. 3.1 that a (concrete) η-witness W for a (concrete) observable O must satisfy:

$$O \cap H_\eta^\varepsilon \neq \emptyset \wedge W = O \cup \pi_\eta^\varepsilon \tag{10}$$

By construction of the basis, and using the hypothesis of the lemma we have the following diagram:

$$\begin{tikzcd}
 & & & W \arrow[ddr, "h"] & \\
 & G \arrow[urr, "f_2"] & = & & = \\
O \arrow[ur, "f_1"] & & & \pi_r^\varepsilon \arrow[uu, "f"] \arrow[r, "g_0^\varepsilon"'] & K \\
 & = & & H_r^\varepsilon \arrow[u, "f_r^\varepsilon"'] \arrow[uull] & \\
 & & I \neq \emptyset \arrow[uull, "h_1"] \arrow[ur, "h_2"'] & &
\end{tikzcd}$$

We take $f_K =_{\mathsf{def}} hf_2f_1$, $g^\varepsilon = g_0^\varepsilon f_r^\varepsilon$, and we have:

$$f_K(O) \cap g^\varepsilon(H_r^\varepsilon) = g^\varepsilon h_2(I) = f_K h_1(I) \neq \emptyset$$

and

$$h(W) = f_K(O) \cup g_0^\varepsilon(\pi_r^\varepsilon) = f_K(O) \cup \pi_\eta$$

which verifies Eq. (10). □

References

1. Rowland, M.A., Deeds, E.J.: Crosstalk and the evolution of specificity in two-component signaling. PNAS **111**(25), 9325 (2014)
2. Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Rule-based modelling of cellular signalling. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 17–41. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74407-8_3
3. Faeder, J.R., Blinov, M.L., Hlavacek, W.S.: Rule-based modeling of biochemical systems with bionetgen. Methods Mol. Biol. **500**, 113–167 (2009)
4. Danos, V., Feret, J., Fontana, W., Krivine, J.: Scalable simulation of cellular signaling networks. In: Shao, Z. (ed.) APLAS 2007. LNCS, vol. 4807, pp. 139–157. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76637-7_10
5. Boutillier, P., Feret, J., Krivine, J. (2008). https://github.com/kappa-dev/kasim

6. Sneddon, M.W., Faeder, J.R., Emonet, T.: Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. Nat. Methods **8**, 177–183 (2011)
7. Danos, V., Heckel, R., Sobocinski, P.: Transformation and refinement of rigid structures. In: Giese, H., König, B. (eds.) ICGT 2014. LNCS, vol. 8571, pp. 146–160. Springer, Heidelberg (2014). doi:10.1007/978-3-319-09108-2_10
8. Girard, J.-Y.: The system F of variable types fifteen years after. Theor. Comput. Sci. **45**, 159–192 (1986)
9. Ehrig, H., Pfender, M., Schneider, H.J.: Graph grammars: an algebraic approach. In: Proceedings of IEEE Conference on Automata and Switching Theory, pp. 167–180 (1973)
10. Raoult, J.C.: On graph rewriting. TCS **32**, 1–24 (1984)
11. Corradini, A., Heindel, T., Hermann, F., König, B.: Sesqui-pushout rewriting. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G. (eds.) Proceedings of ICGT 2006, pp. 30–45 (2006)
12. Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press, Cambridge (2009)
13. Feret, J., Danos, V., Fontana, W., Harmer, R., Krivine, J.: Internal coarse-graining of molecular systems. PNAS **106**, 6453–6458 (2009)
14. Varró, G., Varró, D.: Graph transformation with incremental updates. ENTCS **109**, 71–83 (2004). Proceedings of the Workshop on Graph Transformation and Visual Modelling Techniques (GT-VMT 2004)