Pattern Graphs and Rule-Based Models: The Semantics of Kappa

Pattern graphs and rule-based models: the semantics of Kappa

Jonathan Hayman[1,3]* and Tobias Heindel[2]**

[1] DIENS (INRIA/ÉNS/CNRS), Paris, France
[2] CEA, LIST, Gif sur Yvette, France
[3] Computer Laboratory, University of Cambridge, UK

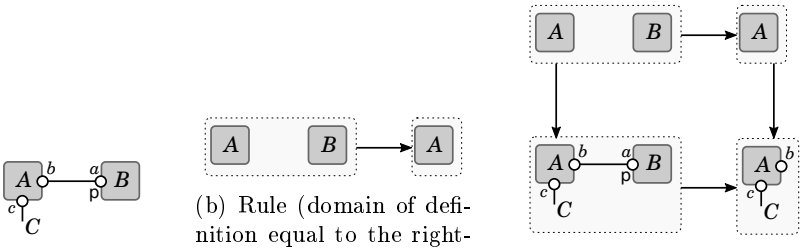
Abstract. Domain-specific rule-based languages to represent the systems of reactions that occur inside cells, such as Kappa and BioNetGen, have attracted significant recent interest. For these models, powerful simulation and static analysis techniques have been developed to understand the behaviour of the systems that they represent, and these techniques can be transferred to other fields. The languages can be understood intuitively as transforming graph-like structures, but due to their expressivity these are difficult to model in 'traditional' graph rewriting frameworks. In this paper, we introduce pattern graphs and closed morphisms as a more abstract graph-like model and show how Kappa can be encoded in them by connecting its single-pushout semantics to that for Kappa. This level of abstraction elucidates the earlier single-pushout result for Kappa, teasing apart the proof and guiding the way to richer languages, for example the introduction of compartments within cells.

1 Introduction

Rule-based models such as Kappa [6] and BioNetGen [2] have attracted significant recent attention as languages for modelling the systems of reactions that occur inside cells. Supported by powerful simulation and static analysis tools, the rule-based approach to modelling in biochemistry offers powerful new techniques for understanding these complex systems [1].

Many of the ideas emerging from rule-based modelling have the potential to be applied much more widely. Towards this goal, in this paper we frame the semantics of Kappa developed in [4] in a more general setting. In [4], an SPO semantics [10] is described by showing that specific pushouts in categories of partial maps between structures specifically defined for Kappa called Σ-graphs correspond to rewriting as performed by Kappa. The richness of the Kappa language means that this construction is highly subtle: it cannot be understood in the more widely studied DPO approach to graph rewriting.

* JH gratefully acknowledges the support of the ANR *AbstractCell* Chair of Excellence and the ERC Advanced Grant *ECSYM*.
** TH is thankful for the financial support of the ANR project PANDA ANR-09-BLAN-0169.



(a) Σ-graph (b) Rule (domain of definition equal to the right-hand side) (c) Rule application

Fig. 1: Example Σ-graph, rule and rule application

In this paper, we study pushouts in categories of simpler, more general structures called pattern graphs. Via an encoding of Σ-graphs into pattern graphs, we determine precisely when pushouts exist and what they are. The original motivation for this work was to tease-apart and generalise the pushout construction in [4]; the more abstract structures certainly provide new insight here by revealing the subtlety of the previous categories, for example in their not admitting all pushouts. But additionally, the study of pattern graphs both exports the fundamentals of Kappa rewriting to a more general setting and provides a uniform target for encodings of other rule-based models. The intention is to use this framework to obtain directly a categorical semantics for BioNetGen and for the enhancement of Kappa with regions.

Overview: In Section 2, we give an overview of Kappa and implicit deletion (called side-effects in [4]) using *closed partial maps*. In Section 3, we introduce *pattern graphs* as an expressive form of graph into which the encoding of Kappa proceeds in Section 4. In Section 5, we isolate the role of *coherent* graphs and determine when they have pushouts. Finally, we show in Section 6 that pushouts in the category of Σ-graphs and pushouts in the category of coherent pattern graphs correspond: the pushout of the encoding is the encoding of the pushout.

2 Kappa and implicit deletion

We shall give a formal account of the semantics of Kappa in Section 4, but essentially we wish to characterise Kappa rewriting as a pushout in a category of special forms of graph called *Σ-graphs*. A rule $\alpha : L \to R$ can be applied to a Σ-graph S if there is a matching of the *pattern* L in S and there is a pushout as follows, generating a rule application $\beta : S \to T$:

$$\begin{array}{ccc} L & \xrightarrow{\alpha} & R \\ {\scriptstyle m}\downarrow & \beta\ulcorner & \downarrow{\scriptstyle m'} \\ S & \longrightarrow & T \end{array}$$

An example Σ-graph is drawn in Figure 1(a). Squares represent entities called *agents* which have circles attached called *sites*. The *signature* Σ describes the

labels that can occur on agents and sites. Links can be drawn between sites, and additionally, for use in patterns which will be used to describe transformation rules, we allow *anonymous* links. The anonymous link drawn at the site c on the A-agent, for example, will represent the existence of a link to some site on a C-agent. Finally, sites can have internal properties attached to them; in this case, the site a on the B-agent has an internal property p, perhaps to represent that the site is *phosphorylated*.

Homomorphisms express how the structure of one Σ-graph embeds into that of another. They are functions on the components of the graph sending agents to agents, sites to sites and links to links, that preserve structure in the sense of preserving the source and target of links and preserving the labels on agents and sites. They allow anonymous links to be sent either to proper links that satisfy any requirements on the target, such as that it is on a C-agent, or to other anonymous links that are at least as specific in their description of the target, for example allowing the link drawn to be sent to another anonymous link that specifies a connection to a site with a particular label on an agent labelled C.

To allow rules to represent the deletion of structure, we consider pushouts of *partial maps*. Partial maps generalise homomorphisms by allowing undefinedness. Formally, we view a partial map $f : L \rightharpoonup R$ to be a span consisting of an inclusion $\mathrm{def}(f) \hookrightarrow L$, where $\mathrm{def}(f)$ is the *domain of definition* of the partial map, and a homomorphism $f_0 : \mathrm{def}(f) \to R$. The interpretation of partial maps as describing transformations is that the rule can be applied if the pattern L is matched; if so, the elements of L that are in the domain of definition indicate what is preserved and the elements of L outside the domain of definition indicate what is to be deleted. Anything in R outside the image of the domain of definition is created by application of the rule.

An example rule is presented in Figure 1(b), showing the deletion of an agent labelled B in the presence of an agent labelled A. Note that the rule does not include any sites, so the state of the sites on agents matched by those in the left-hand side does not determine whether the rule can be applied. This is an instance of the "don't care; don't write" principle in Kappa. Consequently, the rule can be applied to give the application drawn in Figure 1(c). Importantly for capturing the semantics of Kappa, the fact that the B-agent cannot be in the pushout forces the sites on B and the link to A also to be absent from the pushout: the link cannot be in the domain of definition of the rule application since, otherwise, the domain of definition and the produced Σ-graph would not be well-formed. We say that the sites and links are *implicitly deleted* as a side-effect of the deletion of B.

Rewriting is abstractly characterised as taking a pushout in the category Σ-graphs with partial maps between them. In the construction of the pushout described in [4], as in the pushout for containment structures in [8], there is a close relationship to the construction in the category of sets and partial functions. For example, in generating the pushout in Figure 1c, we cannot have the B-labelled agent preserved by the rule application due to the morphism α being undefined on the B-agent matching it. On top of this, however, we have implicit

deletion: as remarked, we are additionally forced to remove the link connecting the A-agent and the site a on B since the domain of definition has to be a well-formed Σ-graph. A natural abstraction that captures both the set-like features of the pushout and implicit deletion is to encode the structures as labelled graphs, the nodes of which are the agents, sites and links and internal properties in the Σ-graph. In this way, we simplify the analysis by treating all elements of the Σ-graph uniformly. We represent using links the dependencies of the Σ-graph: sites depend on agents, links on the sites that they connect, and so on. Links are labelled to indicate the role of the dependency, such as a site being the source of a link. The construction is described more fully in Section 4.1.

Before proceeding into detail, we demonstrate in the simplest possible setting how *closed* morphisms allow implicit deletion as described above to be captured.

Definition 1. *Let Λ be a fixed set of labels. A* basic graph *is a tuple $G = (V, E)$ where V is the set of* nodes *(or* vertices*) and $E \subseteq V \times \Lambda \times V$ is the set of* edges.

We adopt the convention of adding subscripts to indicate the components of a structure. For example, we write V_G for the vertices of a graph G.

Definition 2. *A graph* homomorphism $f : G \to H$ *is a function on nodes $f : V_G \to V_H$ such that if $(v, \lambda, v') \in E_G$ then $(f(v), \lambda, f(v')) \in V_H$.*

We write $\mathbb{bG}$ for the category of basic graphs with homomorphisms between them. When considering partial maps, a critical feature will be the following condition called *closure*, which ensures that any partial map will be defined on any node reachable from any defined node. For example, if a partial map is defined on a link of a Σ-graph, which will be encoded as a node, it will be defined on both the sites and agents that the link connects since they shall be reachable from the node representing the link.

Definition 3. *A partial map $f : G \rightharpoonup H$ is a span $G \hookleftarrow \mathrm{def}(f) \xrightarrow{f_0} H$ of homomorphisms such that $\mathrm{def}(f)$ is a subgraph of G, i.e. $V_{\mathrm{def}(f)} \subseteq V_G$ and $E_{\mathrm{def}(f)} \subseteq E_G$, and the following* closure *property holds:*

if $v \in V_{\mathrm{def}(f)}$ and $(v, \lambda, v') \in E_G$ then $v' \in V_{\mathrm{def}(f)}$ and $(v, \lambda, v') \in E_{\mathrm{def}(f)}$.

Write $\mathbb{bG}_*$ for the category of basic graphs with partial maps between them. Partial maps $f : G \rightharpoonup H$ and $g : H \rightharpoonup K$ compose as partial functions, with the domain of definition of $g \circ f$ obtained as the *inverse image* of $\mathrm{def}(g)$ along f_0. It consists of vertices v of G such that $f(v)$ is in the domain of definition of g and edges to satisfy the closure condition, and can be shown to be a pullback of f_0 against the inclusion $\mathrm{def}(g) \hookrightarrow H$ in $\mathbb{bG}$.

Using the category-theoretical account of existence of pushouts of partial maps mentioned in the conclusion or directly, it can be shown that $\mathbb{bG}_*$ has pushouts of all spans of partial maps; we do not present the details here since they shall be subsumed by those in the following section for pattern graphs. In Figure 2, we give an example of a pushout in $\mathbb{bG}_*$. The pushout shows in a simplified way how pushing out against a partial map representing deletion of an agent (the node w_2) requires the implicit deletion of a link (the node ℓ). The key

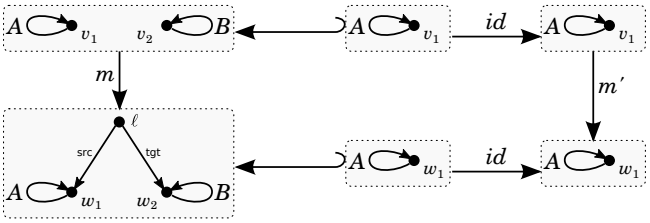


Fig. 2: An example pushout in $\mathbb{bG}_*$. Horizontal spans represent partial maps. Vertical maps m, m' are total and send v_i to w_i for $i \in \{1, 2\}$.

is that, following the argument for pushouts in the category of sets and partial functions, we cannot have the node w_2 in the domain of definition of the lower partial map and hence, by the closure condition, we therefore cannot have ℓ in the domain of definition of the rule application represented by the lower span.

We wish to use the fact that the category $\mathbb{bG}_*$ has pushouts of all spans to consider, via the encoding described above and to be formalised in Section 4.1, pushouts in the category of Σ-graphs. Before we can do so, there are two important details to consider.

The first is that we must ensure that maps between the encoded structures correspond to maps in the category of Σ-graphs: that they send agents to agents, sites to sites and links to links. The standard trick of encoding the types of elements as labelled loops can deal with these aspects, but the presence in Kappa of anonymous links that can be mapped either to proper links or to more specific anonymous links will necessitate a richer structure than basic graphs. There are alternatives, for example recording the type of nodes, but in this paper an elegant treatment is provided by the use of *pattern graphs*.

The second detail is that we will wish the pushout in the category of pattern graphs to correspond to the encoding of some Σ-graph. Certain aspects will follow from the nature of the pushout, but it turns out that a pivotal issue will be *coherence*. The graphs that we form as encodings will be coherent in the sense that any node will have at most one edge with any given label from it. For example, the encoding of a Kappa link will have at most one source and at most one target; Kappa does not have hyper-edges. However, as we shall see, the graphs that we form by taking pushouts in the category of pattern graphs might fail to be coherent, so we desire an operation that forms a coherent pushout from the non-coherent one. This operation can fail: it is not always possible to form a pushout of an arbitrary span of morphisms in the category of coherent pattern graphs with partial morphisms between them, and critically this will inform us that there is no pushout of the given span in the category of Σ-graphs.

3 Pattern graphs

Pattern graphs add to basic graphs the ability to express, via homomorphisms, the existence of an labelled edge from a node to some node satisfying a specifi-

cation. Specifications are just prefix-closed sets of sequences of labels to indicate paths that must exist from the specified node. Assuming a set of link labels Λ, let Λ^* denote the set of all finite sequences of elements of Λ. For sequences p and q, write $p \leq q$ if p is a prefix of q. For a set of sequences $\phi \subseteq \Lambda^*$, we write $\downarrow \phi$ for the set of sequences p such that there exists $q \in \phi$ satisfying $p \leq q$.

Definition 4. *Let $\mathcal{P}_{\leq}(\Lambda^*)$ be the set of all prefix-closed finite sets of sequences of elements of Λ. A tuple (V, E) is a* pattern graph *if V and E are disjoint finite sets and $E \subseteq V \times \Lambda \times (V \cup \mathcal{P}_{\leq}(\Lambda^*))$, where V is disjoint from $\mathcal{P}_{\leq}(\Lambda^*)$.*

The sets V and E represent the sets of vertices and edges of a pattern graph. Edges can either be normal (*i.e.* between vertices) or be specifications, being of the form (v, λ, ϕ): the intention is that specifications are used in patterns when we wish to specify, via homomorphisms, the structure that some other perhaps more refined graph possesses.

Definition 5. *A vertex $v \in V$ in a pattern graph G satisfies $p \in \Lambda^*$, written $v \models_G p$, if either p is empty or $p = \lambda.p_0$ and either:*

- *there exists v' such that $(v, \lambda, v') \in E_G$ and $v' \models_G p_0$, or*
- *there exists $\psi \in \mathcal{P}_{\leq}(\Lambda^*)$ such that $(v, \lambda, \psi) \in E_G$ and $p_0 \in \psi$.*

A vertex $v \in V_G$ satisfies $\phi \in \mathcal{P}_{\leq}(\Lambda^*)$, written $v \models_G \phi$, if $v \models_G p$ for all $p \in \phi$.

Homomorphisms embed the structure of one pattern graph into that of another: they preserve the presence of normal links between vertices and, if a vertex has a specification, the image of the vertex satisfies the specification. Importantly, they do not record exactly *how* the specification is satisfied.

Definition 6. *A* homomorphism *of pattern graphs $f : G \to H$ is a function on vertices $f : V_G \to V_H$ such that, for all $v, v' \in V_G$, $\lambda \in \Lambda$ and $\phi \in \mathcal{P}_{\leq}(\Lambda^*)$:*

- *if $(v, \lambda, v') \in E_G$ then $(f(v), \lambda, f(v')) \in E_H$, and*
- *if $(v, \lambda, \phi) \in E_G$ then there exists $x \in V_H \cup \mathcal{P}_{\leq}(\Lambda^*)$ such that $(f(v), \lambda, x) \in E_H$ and $x \models_G \phi$ if $x \in V_H$ and $\phi \subseteq x$ if $x \in \mathcal{P}_{\leq}(\Lambda^*)$.*

We write $\mathbb{PG}$ for the category of pattern graphs connected by homomorphisms. For any pattern graph τ, denote by $\mathbb{PG}/\tau$ the slice category above τ. The objects of $\mathbb{PG}/\tau$ are pairs (G, γ) where G is a pattern graph and $\gamma : G \to \tau$ is a homomorphism, and a morphism $h : (G, \gamma) \to (G', \gamma')$ in $\mathbb{PG}/\tau$ is a homomorphism such that $\gamma = \gamma' \circ h$. We can regard τ as representing the structure that the pattern graphs being considered are allowed to possess. Where no ambiguity arises, we shall simply write G for the pair (G, γ) and τ_G for γ.

Partial maps extend homomorphisms by allowing them to be undefined on vertices and edges. Again, we require the closure condition of Section 2.

Definition 7. *Let G and H be objects of $\mathbb{PG}/\tau$. A* partial map *$f : G \rightharpoonup H$ consists of a pattern graph $\mathrm{def}(f) = (V_0, E_0)$ and a homomorphism $f_0 : \mathrm{def}(f) \to H$ in $\mathbb{PG}/\tau$ where:*

- $\mathrm{def}(f)$ *is a pattern graph satisfying* $V_0 \subseteq V_G$ *and* $E_0 \subseteq E_G$*;*
- $\tau_{\mathrm{def}(f)} : \mathrm{def}(f) \to \tau$ *is the restriction of* τ_G *to* $\mathrm{def}(f)$*; and*
- $\mathrm{def}(f)$ *is closed: for all* $(v, \lambda, x) \in E_G$*, if* $v \in V_0$ *then* $(v, \lambda, x) \in E_0$ *(and hence* $x \in V_0$ *if* $x \in V_G$*).*

We write $(\mathbb{PG}/\tau)_*$ for the category of pattern graphs with partial maps between them. As it was for basic graphs, composition is obtained using the inverse image construction and can be shown to be a pullback in $\mathbb{PG}/\tau$.

The category $(\mathbb{PG}/\tau)_*$ can be shown to have pushouts of all spans. However, as we shall see in Section 5, the existence conditions for pushouts of coherent typed pattern graphs will be much more subtle and involve considerable additional work.

Theorem 1. *The category* $(\mathbb{PG}/\tau)_*$ *has pushouts.*

The above result can be proved either by using the category-theoretical conditions for existence of pushouts of partial maps mentioned in the conclusion or directly by showing that the following construction yields a pushout of any span.

Given a span $S \xleftarrow{g} L \xrightarrow{f} R$, we define a cospan $S \xrightarrow{p} T \xleftarrow{q} R$ that forms a pushout in $(\mathbb{PG}/\tau)_*$. Let $V = V_L \cup V_R \cup V_S$. We define $\sim$ to be the least equivalence relation on V such that $v_0 \sim f(v_0)$ for all $v_0 \in V_{\mathrm{def}(f)}$ and $w_0 \sim g(w_0)$ for all $w_0 \in V_{\mathrm{def}(g)}$. For any $v \in V$, we denote by $[v]$ its $\sim$-equivalence class; these equivalence classes will be used to form the vertices of the pushout object T.

Write $[v] \xrightarrow{\lambda} [v']$ iff there exist $v_0 \in [v]$ and $v_0' \in [v']$ such that $(v_0, \lambda, v_0') \in E_L \cup E_R \cup E_S$. The unlabelled transitive closure of this relation, written $[v_1] \to^* [v_n]$, relates $[v_1]$ to $[v_n]$ if there exist $\lambda_1, \ldots, \lambda_{n-1}$ such that $[v_1] \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{n-1}} [v_n]$. Now define $\mathsf{del}_0([v])$ iff there exists $v_0 \in [v] \cap V_L$ such that $v_0 \notin V_{\mathrm{def}(f)} \cap V_{\mathrm{def}(g)}$ and define $\mathsf{del}([v])$ iff there exists $v' \in V$ such that $[v] \to^* [v']$ and $\mathsf{del}_0([v'])$. The vertices and edges of the pushout object are given as:

$$\begin{aligned} V_T &= \{[v] \mid v \in V \ \&\ \neg\mathsf{del}([v])\} \\ E_T &= \{([v], \lambda, [v']) \mid [v], [v'] \in V_T \ \&\ [v] \xrightarrow{\lambda} [v']\} \\ &\quad \cup \{([v], \lambda, \phi) \mid [v] \in V_T \ \&\ \exists v_0 \in [v].(v_0, \lambda, \phi) \in E_L \cup E_R \cup E_S\} \end{aligned}$$

The domain of definition of the pushout morphism p is the closed subgraph of S containing all vertices $v \in V_S$ such that $\neg\mathsf{del}([v])$. Where defined, p sends vertices of S to their $\sim$-equivalence classes. The partial map q is defined similarly. The type map $\tau_T : T \to \tau$ sends an equivalence class $[v]$ to $\tau_L(v)$ if $v \in V_L$, to $\tau_R(v)$ if $v \in V_R$ and $\tau_S(v)$ if $v \in V_S$; well-definedness of this follows from the maps f and g being type-preserving.

4 Kappa and Σ-graphs

We begin this section by briefly describing the semantics given to Kappa in [4], where a fuller explanation and examples can be found. The semantics of Kappa is

given over graphs with a given signature Σ, specifying the labels that can occur on agents Σ_{ag}, sites Σ_{st} and internal properties Σ_{prop}, and, for any agent label A, the set of sites $\Sigma_{\mathsf{ag-st}}(A)$ that are permitted to occur on an agent labelled A.

Definition 8. *A* signature *is a 4-tuple* $\Sigma = (\Sigma_{\mathsf{ag}}, \Sigma_{\mathsf{st}}, \Sigma_{\mathsf{ag-st}}, \Sigma_{\mathsf{prop}})$*, where* Σ_{ag} *is a finite set of* agent types*,* Σ_{st} *is a finite set of* site identifiers*,* $\Sigma_{\mathsf{ag-st}} : \Sigma_{\mathsf{ag}} \to \mathcal{P}_{\mathsf{fin}}(\Sigma_{\mathsf{st}})$ *is a* site map*, and* Σ_{prop} *is a finite set of* internal property identifiers*.*

As described in Section 2, Σ-graphs consist of sites on agents; sites can have internal properties indicated and be linked to each other. Sites can also have anonymous links attached to them, typically used when the Σ-graph represents a pattern, so the anonymous link represents that a link is required to exist at the image of the site under a homomorphism. There are three types of anonymous link; the first is represented by a dash '$-$' and indicates that the link connects to any site on any agent, the second by A for $A \in \Sigma_{\mathsf{ag}}$ which indicates that the link connects to some site on an agent of type A, and the final kind of anonymous link is (A, i) which indicates that the link connects to site i on some agent labelled A. These form the set $\mathsf{Anon} = \{-\} \cup \Sigma_{\mathsf{ag}} \cup \{(A, i) \mid A \in \Sigma_{\mathsf{ag}} \;\&\; i \in \Sigma_{\mathsf{ag-st}}(A)\}$.

Definition 9. *A* Σ*-graph comprises a finite set* $\mathcal{A}$ *of* agents*, an* agent type *assignment* $\mathsf{type} : \mathcal{A} \to \Sigma_{\mathsf{ag}}$*, a set* $\mathcal{S}$ *of* link sites *satisfying* $\mathcal{S} \subseteq \{(n, i) \;:\; n \in \mathcal{A} \;\&\; i \in \Sigma_{\mathsf{ag-st}}(\mathsf{type}(n))\}$*, a symmetric* link *relation* $\mathcal{L} \subseteq (\mathcal{S} \cup \mathsf{Anon})^2 \setminus \mathsf{Anon}^2$*, and a* property *set* $\mathcal{P} \subseteq \{(n, i, k) \mid n \in \mathcal{A} \;\&\; i \in \Sigma_{\mathsf{ag-st}}(\mathsf{type}(n)) \;\&\; k \in \Sigma_{\mathsf{prop}}\}$*.*

We shall conventionally assume that the sets described above are pairwise-disjoint. A normal link is a pair of sites $((n, i), (m, j))$ and an anonymous link is of the form $((n, i), x)$ where (n, i) is a site and $x \in \mathsf{Anon}$. We use x to range over both sites and Anon. Note that $(n, i, k) \in \mathcal{P}$ does not imply $(n, i) \in \mathcal{S}$: as explained in [4], this is to allow Σ-graphs to represent patterns where we represent a property holding at some site but do not specify anything about its linkage.

Homomorphisms between Σ-graphs are structure-preserving functions from the agents, sites, links and internal properties of one Σ-graph to those of another. They preserve structure by preserving the presence of sites on agents, preserving properties held on sites, preserving the source and target of links and ensuring that the source and target of the image of any link is at least as high in the *link information order* as those of the original link. Given a typing function type, this order is the least reflexive, transitive relation $\leq_{\mathsf{type}}$ s.t. for all $A \in \Sigma_{\mathsf{ag}}$ and $i \in \Sigma_{\mathsf{ag-st}}(A)$ and n s.t. $\mathsf{type}_G(n) = A$: $\quad - \leq_{\mathsf{type}} A \leq_{\mathsf{type}} (A, i) \leq_{\mathsf{type}} (n, i)$.

Definition 10. *A* homomorphism *of* Σ*-graphs* $h : G \to H$ *consists of a function on agents* $h_{\mathsf{ag}} : \mathcal{A}_G \to \mathcal{A}_H$*, a function on sites* $h_{\mathsf{st}} : \mathcal{S}_G \to \mathcal{S}_H$*, a function on links* $h_{\mathsf{ln}} : \mathcal{L}_G \to \mathcal{L}_H$ *and a function on internal properties* $h_{\mathsf{prop}} : \mathcal{P}_G \to \mathcal{P}_H$*, satisfying:*

- $\mathsf{type}_G(n) = \mathsf{type}_H(h_{\mathsf{ag}}(n))$ *for all* $n \in \mathcal{A}_G$
- $h_{\mathsf{st}}(n, i) = (h_{\mathsf{ag}}(n), i)$ *and* $h_{\mathsf{prop}}(n, i, k) = (h_{\mathsf{ag}}(n), i, k)$
- $h_{\mathsf{ln}}((n, i), x) = (h_{\mathsf{st}}(n, i), y)$ *for some* y *such that* $\hat{h}(x) \leq_{\mathsf{type}_H} y$*, where we take* $\hat{h}(m, j) = h_{\mathsf{st}}(m, j)$ *for any* $(m, j) \in \mathcal{S}_G$ *and* $\hat{h}(x) = x$ *for any* $x \in \mathsf{Anon}$*.*

We write $\Sigma\mathbb{G}$ for the category of Σ-graphs with homomorphisms between them. Note that in [4], attention was restricted to graphs with only one link to or from any site, allowing a less general form of homomorphism to be used.

Definition 11. *A partial map $f : G \rightharpoonup H$ between Σ-graphs G and H is a span $G \hookleftarrow \mathrm{def}(f) \xrightarrow{f_0} H$ where f_0 is a homomorphism and $\mathrm{def}(f)$ is a Σ-graph that is a subgraph of G, i.e.: $\mathcal{A}_{\mathrm{def}(f)} \subseteq \mathcal{A}_G$ and $\mathcal{S}_{\mathrm{def}(f)} \subseteq \mathcal{S}_G$ and $\mathcal{L}_{\mathrm{def}(f)} \subseteq \mathcal{L}_G$ and $\mathcal{P}_{\mathrm{def}(f)} \subseteq \mathcal{P}_G$.*

Partial maps between Σ-graphs form a category denoted $\Sigma\mathbb{G}_*$, where partial maps $f : G \rightharpoonup H$ and $g : H \rightharpoonup K$ compose in the usual way, with the domain of definition of their composition $\mathrm{def}(g \circ f)$ containing elements of $\mathrm{def}(f)$ such that their image under f is in $\mathrm{def}(g)$. This corresponds to taking a pullback of the homomorphism $f_0 : \mathrm{def}(f) \to H$ against $\mathrm{def}(g) \hookrightarrow H$ in $\Sigma\mathbb{G}$.

4.1 Encoding Σ-graphs as pattern graphs

We now show how Σ-graphs can be interpreted as pattern graphs. As stated before, the idea behind the encoding of a Σ-graph G is to build a pattern graph $[\![G]\!]$ with vertices that are the agents, sites, links and properties of G. Labelled edges in $[\![G]\!]$ indicate the dependencies between elements of the graph, so for example that deletion of an agent causes the deletion of any edge connecting to that agent. There will be edges from links to their source and target, labelled src or tgt respectively, and edges labelled ag from sites to agents and internal properties to agents (not to sites since, as mentioned, we use the set of sites specifically to represent link state). There will also be an edge labelled symm between every link and its symmetric counterpart to ensure that morphisms preserve the symmetry of the link relation. Specifications are used in the representation of anonymous links.

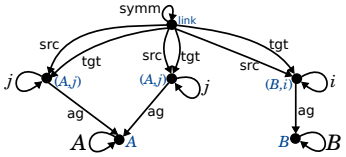
The starting point is to define a pattern graph $\hat{\Sigma}$ to represent the structure of encodings, so that the encoding of a Σ-graph will be an object of $(\mathbb{PG}/\hat{\Sigma})_*$.

Definition 12. *With respect to signature Σ, the pattern graph $\hat{\Sigma}$ is*

$$
\begin{aligned}
V_{\hat{\Sigma}} = \quad & \{\mathsf{link}\} \quad \cup \quad \Sigma_{\mathsf{ag}} \quad \cup \quad \{(A,i) \mid A \in \Sigma_{\mathsf{ag}} \;\&\; i \in \Sigma_{\mathsf{ag-st}}(A)\} \\
& \cup \{(A,i,p) \mid A \in \Sigma_{\mathsf{ag}} \;\&\; i \in \Sigma_{\mathsf{ag-st}}(A) \;\&\; p \in \Sigma_{\mathsf{prop}}\} \\
E_{\hat{\Sigma}} = \quad & \{(\mathsf{link}, \mathsf{symm}, \mathsf{link})\} \\
& \cup \{(\mathsf{link}, \mathsf{src}, (A,i)), \quad (\mathsf{link}, \mathsf{tgt}, (A,i)) \;\mid\; A \in \Sigma_{\mathsf{ag}} \;\&\; i \in \Sigma_{\mathsf{ag-st}}(A)\} \\
& \cup \{((A,i), \mathsf{ag}, A) \mid A \in \Sigma_{\mathsf{ag}} \;\&\; i \in \Sigma_{\mathsf{ag-st}}(A)\} \\
& \cup \{((A,i,p), \mathsf{ag}, A) \mid A \in \Sigma_{\mathsf{ag}} \;\&\; i \in \Sigma_{\mathsf{ag-st}}(A) \;\&\; p \in \Sigma_{\mathsf{prop}}\} \\
& \cup \{((A,i,p), (i,p), (A,i,p)) \mid A \in \Sigma_{\mathsf{ag}} \;\&\; i \in \Sigma_{\mathsf{ag-st}}(A) \;\&\; p \in \Sigma_{\mathsf{prop}}\} \\
& \cup \{(A,A,A) \mid A \in \Sigma_{\mathsf{ag}}\} \cup \{(A,i), i, (A,i) \mid A \in \Sigma_{\mathsf{ag}} \;\&\; i \in \Sigma_{\mathsf{ag-st}}(A)\}
\end{aligned}
$$

Example 1. Given Σ as follows, the graph $\hat{\Sigma}$ is:

$$\Sigma_{\mathsf{ag}} = \{A, B\}$$
$$\Sigma_{\mathsf{st}} = \{i, j\}$$
$$\Sigma_{\mathsf{prop}} = \emptyset$$
$$\Sigma_{\mathsf{ag-st}} = \{A \mapsto \{i, j\}, B \mapsto \{i\}\}$$



The loops on the vertices for agents and sites will be used in patterns for anonymous links. We now define a functor $[\![\cdot]\!] : \Sigma\mathbb{G}_* \to (\mathbb{PG}/\hat{\Sigma})_*$ that embeds the category of Σ-graphs into the category of pattern graphs over $\hat{\Sigma}$.

Definition 13. *For a Σ-graph G, the pattern graph $[\![G]\!]$ is:*

$$\begin{aligned}
V_{[\![G]\!]} = \quad & \mathcal{A}_G \cup \mathcal{S}_G \cup \mathcal{L}_G \cup \mathcal{P}_G \\
E_{[\![G]\!]} = \quad & \{(n, \mathsf{type}_G(n), n) \mid n \in \mathcal{A}\} \cup \{((n,i), i, (n,i)) \mid (n,i) \in \mathcal{S}\} \\
& \cup \{((n,i,p), (i,p), (n,i,p)) \mid (n,i,p) \in \mathcal{P}_G\} \\
& \cup \{((n,i), \mathsf{ag}, n) \mid (n,i) \in \mathcal{S}_G\} \cup \{((n,i,p), \mathsf{ag}, n) \mid (n,i,p) \in \mathcal{P}_G\} \\
& \cup \{(((n,i),x), \mathsf{src}, (n,i)) \mid ((n,i),x) \in \mathcal{L}_G\} \\
& \cup \{((x,(n,i)), \mathsf{tgt}, (n,i)) \mid (x,(n,i)) \in \mathcal{L}_G\} \\
& \cup \{((x,(n,i)), \mathsf{src}, anon(x) \mid x \in \mathsf{Anon} \ \&\ (x,(n,i)) \in \mathcal{L}_G\} \\
& \cup \{(((n,i),x), \mathsf{tgt}, anon(x) \mid x \in \mathsf{Anon} \ \&\ ((n,i),x) \in \mathcal{L}_G\}
\end{aligned}$$

where anon gives a specification for anonymous links:

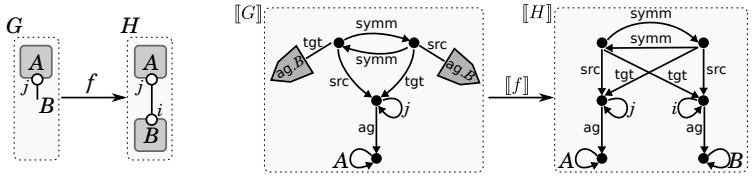
$$anon(-) = \emptyset \qquad anon(A) = \downarrow \{\mathsf{ag}.A\} \qquad anon(A,i) = \downarrow \{i.\mathsf{ag}.A\}$$

The function $\tau_{[\![G]\!]} : [\![G]\!] \to \hat{\Sigma}$ sends links to link*, agents n to $\mathsf{type}_G(n)$, sites (n,i) to $(\mathsf{type}_G(n), i)$ and properties (n,i,p) to $(\mathsf{type}_G(n), i, p)$.*

The encoding $[\![g]\!] : [\![G]\!] \rightharpoonup [\![H]\!]$ of a partial map $g : G \rightharpoonup H$ in $\Sigma\mathbb{G}_*$ has domain of definition $[\![\mathrm{def}(g)]\!]$ and sends a vertex $v \in V_{[\![\mathrm{def}(g)]\!]}$ to $g_{\mathsf{ag}}(v)$ if $v \in \mathcal{A}_G$, or $g_{\mathsf{st}}(v)$ if $v \in \mathcal{S}_G$, or $g_{\mathsf{link}}(v)$ if $v \in \mathcal{L}_G$, or $g_{\mathsf{prop}}(v)$ if $v \in \mathcal{P}_G$.

It is straightforward to check that the encoding defines a functor. The key is that the partial map $[\![f]\!]$ satisfies the closure condition due to the domain of definition of f being a well-formed Σ-graph.

Example 2. The encoding of the unique homomorphism from G to H as drawn is the unique homomorphism from $[\![G]\!]$ to $[\![H]\!]$. Specifications are drawn as kites.

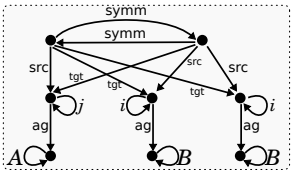


Lemma 1. *The encoding $[\![\cdot]\!]$ is an embedding: it is a full and faithful functor and is injective on objects.*

As such, the image of $[\![\cdot]\!]$ is a full subcategory of $(\mathbb{PG}/\hat{\Sigma})_*$ isomorphic to $\Sigma\mathbb{G}_*$.

5 Coherence

The pushout in $(\mathbb{PG}/\hat{\Sigma})_*$ of the encoding of a span of morphisms in $\Sigma\mathbb{G}_*$ can fail to be an encoding of a Σ-graph. For example, in $(\mathbb{PG}/\hat{\Sigma})_*$ the pushout against itself of the morphism $[\![f]\!] : [\![G]\!] \to [\![H]\!]$ drawn in Example 2 is:



The encodings of Σ-graphs have links with at most one source and at most one target: we say that they are *coherent*.

Definition 14. *A pattern graph G is* coherent *if $(v, \lambda, x_1) \in E_G$ and $(v, \lambda, x_2) \in E_G$ implies $x_1 = x_2$ for all $v \in V_G$, all $\lambda \in \Lambda$ and all $x_1, x_2 \in V_G \cup \mathcal{P}_{\leq}(\Lambda^*)$.*

Lemma 2. *For any Σ-graph G, the encoding $[\![G]\!]$ is coherent.*

We now characterise pushouts in the category $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$, the full subcategory of $(\mathbb{PG}/\tau)_*$ with coherent pattern graphs over τ as objects. By restricting ourselves to coherent graphs, remarkably we lose the property that all spans of partial (and even total) maps have pushouts. It is not hard, for example, to verify that there is no pushout in $(\mathbb{PG}/\hat{\Sigma})^{\mathfrak{c}}_*$ of the morphism $[\![f]\!] : [\![G]\!] \to [\![H]\!]$ in Example 2 against itself. The significance of this is that exactly the same phenomenon occurs in the category of Σ-graphs, which fails to have pushouts of all spans for the same reason. For example, analogously there is no pushout of the morphism of Σ-graphs $f : G \to H$ drawn in Example 2 against itself in $\Sigma\mathbb{G}_*$ (noting that the form of homomorphism in this paper includes a component to give the target of links, unlike in [4] where the less general form of homomorphism could be used).

We shall see in Section 6 that pushouts of encodings in $(\mathbb{PG}/\hat{\Sigma})^{\mathfrak{c}}_*$ will correspond to those in $\Sigma\mathbb{G}_*$. We now study when pushouts in $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$ exist. We do so by characterising the largest full subcategory $\mathcal{C}$ of $(\mathbb{PG}/\tau)_*$ for which $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$ is a reflective subcategory of $\mathcal{C}$. When the pushout in $(\mathbb{PG}/\tau)_*$ lies in the category $\mathcal{C}$, the pushout in coherent graphs will be obtained by applying the left adjoint of the reflection. Otherwise, if the pushout in $(\mathbb{PG}/\tau)_*$ is outside $\mathcal{C}$, there is no pushout of the span in $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$.

The process for determining if a pattern graph G in $(\mathbb{PG}/\tau)_*$ lies in $\mathcal{C}$ is somewhat intricate. We begin by removing from G vertices on which any partial map to any coherent graph in $(\mathbb{PG}/\tau)_*$ must be undefined due to the type constraint τ; we go through this in more detail below. We then successively merge *joinable* edges, continuing until there is no joinable pair of edges. If the result is a coherent graph, then G lies in $\mathcal{C}$ and the left adjoint applied to G is the constructed graph; otherwise, G is not in $\mathcal{C}$. It is convenient to begin the formalisation of this with the merging operation.

Definition 15. *Given a pattern graph G, a distinct pair of edges $e_1 = (v_1, \lambda_1, x_1)$ and $e_2 = (v_2, \lambda_2, x_2) \in E_G$ is* engaged *if $v_1 = v_2$ and $\lambda_1 = \lambda_2$.*

Definition 16. *Let $e_1 = (v, \lambda, x_1)$ and $e_2 = (v, \lambda, x_2)$ be an engaged pair of edges in a graph G. The graph obtained by merging them, denoted $G[e_1 \bowtie e_2]$, and homomorphism $(e_1 \bowtie e_2) : G \to G[e_1 \bowtie e_2]$ is defined as follows:*

- *if x_1 and x_2 are vertices,*

$$G[e_1 \bowtie e_2] = (V_G \setminus \{x_2\}, E_G[x_2 \mapsto x_1]) \qquad (e_1 \bowtie e_2)(w) = \begin{cases} w & \text{if } w \neq x_2 \\ x_1 & \text{if } w = x_2 \end{cases}$$

  where $E_G[x_2 \mapsto x_1] = \{((e_1 \bowtie e_2)w, \lambda', (e_1 \bowtie e_2)w') \mid (w, \lambda', w') \in E_G\}$
- *if x_1 is a specification and x_2 is a vertex,*

$$G[e_1 \bowtie e_2] = (V, E \setminus \{e_1\} \cup \{(x_2, \lambda_1, \{p_1\}) \mid \lambda_1.p_1 \in x_1\}) \qquad (e_1 \bowtie e_2)(w) = w$$

- *if x_2 is a specification and x_1 is a vertex,*

$$G[e_1 \bowtie e_2] = (V, E \setminus \{e_2\} \cup \{(x_1, \lambda_2, \{p_2\}) \mid \lambda_2.p_2 \in x_2\}) \qquad (e_1 \bowtie e_2)(w) = w$$

- *if x_1 and x_2 are specifications,*

$$G[e_1 \bowtie e_2] = (V_G, E_G \setminus \{e_1, e_2\} \cup \{(v, \lambda, x_1 \cup x_2)\}) \qquad (e_1 \bowtie e_2)(w) = w$$

The process of merging engaged pairs of edges is locally confluent up to isomorphism.

Lemma 3. *Let G be a graph containing engaged pairs of edges e_1, e_2 and f_1, f_2. Either $G[e_1 \bowtie e_2] \cong G[f_1 \bowtie f_2]$ or there exist engaged pairs of edges e_1', e_2' and f_1', f_2' such that $G[e_1 \bowtie e_2][f_1' \bowtie f_2'] \cong G[f_1 \bowtie f_2][e_1' \bowtie e_2']$.*

Repeatedly joining engaged nodes, we obtain (up to isomorphism) a coherent graph denoted $\mathsf{collapse}(G)$ and a homomorphism $\mathsf{collapse}_G : G \to \mathsf{collapse}(G)$.

For a vertex v of G, let $\lceil v \rceil_G$ denote the pattern graph obtained by restricting G to vertices and edges reachable from v.

Definition 17. *Let $e_1 = (v_1, \lambda_1, x_1)$ and $e_2 = (v_2, \lambda_2, x_2)$ be an engaged pair of edges. They are* joinable *if for all $i, j \in \{1, 2\}$ such that $i \neq j$, if x_i is a vertex then either $x_i \to^* v$ or x_j is a specification and $\mathsf{collapse}_{\lceil x_i \rceil_G}(x_i) \models x_j$ in $\mathsf{collapse}(\lceil x_i \rceil_G)$, where $\to^*$ denotes reachability in G.*

We now return to the initial stage, deleting vertices on grounds of the 'type' τ. As an example, let $G = \tau$ be the non-coherent graph with $V_G = \{v, w_1, w_2\}$ and $E_G = \{(v, a, w_1), (v, a, w_2), (w_1, b, w_1), (w_2, c, w_2)\}$. Let C be any coherent graph with a homomorphism $\tau_C : C \to \tau$ and $f : G \to C$ be a morphism in $(\mathbb{PG}/\tau)_*$. We cannot have $v \in \mathrm{def}(f)$ since, if it were, by closure and coherence there would have to exist a vertex $f(w_1) = f(w_2)$ with outgoing edges labelled b and c, contradicting the assumption of a homomomorphism from C to τ. The graph remaining after removal of vertices that cannot be in the domain of definition of any partial map to any coherent graph is defined as follows. Let $\tau_{\lceil v \rceil_G} : \lceil v \rceil_G \to \tau$ be the restriction of the homomorphism $\tau_G : G \to \tau$ to $\lceil v \rceil_G$.

Definition 18. *For a pattern graph G and homomorphism $\tau_G : G \to \tau$, define the graph $\mathsf{td}(G, \tau_G) = (V_0, E_0)$ to have vertices $v \in V_G$ such that there exists a homomorphism $\tau' : \mathsf{collapse}(\lceil v \rceil_G) \to \tau$ such that $\tau_{\lceil v \rceil_G} = \tau' \circ \mathsf{collapse}_{\lceil v \rceil_G}$ and $E_0 = \{(v, \lambda, x) \in E_G \mid v \in V_0\}$.*

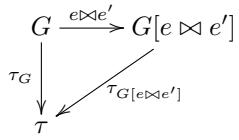
Write $v \notin \mathsf{td}(G, \tau_G)$ if $v \in V_G \setminus V_{\mathsf{td}(G,\tau_G)}$; it is the predicate that determines if the vertex v is deleted on grounds of type.

Note that for each v, the morphism τ', if it exists, must be the unique such morphism since $\mathsf{collapse}_{\lceil v \rceil_G}$ is an epimorphism (it is surjective on vertices).

Lemma 4. *Let G be a pattern graph and $\tau_G : G \to \tau$ and $\mathsf{td}(G, \tau_G) = (V_0, E_0)$. Let $\tau_{\mathsf{td}(G,\tau_G)} : \mathsf{td}(G, \tau_G) \to \tau$ be the restriction of τ_G to V_0. The span $\mathsf{td}_G = (G \hookleftarrow \mathsf{td}(G, \tau_G) \xrightarrow{\mathsf{id}_{\mathsf{td}(G,\tau_G)}} \mathsf{td}(G, \tau_G))$ is a partial map in $(\mathbb{PG}/\tau)_*$. Furthermore, for any coherent pattern graph C and morphism $f : G \to C$ in $(\mathbb{PG}/\tau)_*$, there is a unique morphism $f^\sharp : \mathsf{td}(G, \tau_G) \to C$ such that $f = f^\sharp \circ \mathsf{td}_G$.*

The process for determining if a pattern graph lies in the category $\mathcal{C}$ begins by forming the graph $\mathsf{td}(G, \tau_G)$. We then repeatedly merge *joinable* pairs of vertices. If the resulting graph is coherent, the graph is in $\mathcal{C}$. Otherwise, if we obtain a graph with no joinable pair of edges but some engaged pair of edges, the graph cannot be in the category $\mathcal{C}$. Importantly, as we merge vertices we never re-introduce grounds for removal of other vertices due to type incompatibility.

Lemma 5. *Let G be in $(\mathbb{PG}/\tau)_*$ and there be no $v \in V_G$ such that $v \notin \mathsf{td}(G, \tau_G)$. For any joinable pair of edges e, e' in E_G, there is a unique homomorphism $\tau_{G[e \bowtie e']}$ such that the following diagram commutes:*



Furthermore, there is no $v \in V_{G[e \bowtie e']}$ such that $v \notin \mathsf{td}(G[e \bowtie e'], \tau_{G[e \bowtie e']})$.

The following lemma represents one step of proving that the constructed graph lies in the subcategory in reflection with coherent graphs.

Lemma 6. *Let G in $(\mathbb{PG}/\tau)_*$ contain no vertex v such that $v \notin \mathsf{td}(G, \tau_G)$. Let e, e' be any joinable pair of edges in G. For any C in $(\mathbb{PG}/\tau)_*$ such that C is coherent and morphism $f : G \to C$ in $(\mathbb{PG}/\tau)_*$, there is a unique morphism $f' : G[e \bowtie e'] \to C$ such that $f = f' \circ (e \bowtie e')$.*

Conversely, the following lemma is used to show that if the process of merging joinable nodes from $\mathsf{td}(G, \tau_G)$ fails, leaving a non-coherent graph with no joinable pair of edges, the graph G lies outside the category $\mathcal{C}$.

Lemma 7. *Let G in $(\mathbb{PG}/\tau)_*$ contain no vertex v such that $v \notin \mathsf{td}(G, \tau_G)$ and no pair of joinable edges. If G is not coherent, there is no P in $(\mathbb{PG}/\tau)_*$ and morphism $\phi : G \to P$ in $(\mathbb{PG}/\tau)_*$ such that P is coherent and any morphism $f : G \to C$ in $(\mathbb{PG}/\tau)_*$ to a coherent graph C factors uniquely through ϕ.*

For a sequence of pairs of edges $\mathcal{E}$ and pattern graph G, let $G\lfloor e \bowtie e' \rfloor_{\mathcal{E}}$ denote G with all pairs up to but not including (e, e') in $\mathcal{E}$ merged as in Definition 16 in sequence. Let $G[\mathcal{E}]$ denote G with all pairs in $\mathcal{E}$ merged in sequence.

Theorem 2. *The largest full subcategory $\mathcal{C}$ of $(\mathbb{PG}/\tau)_*$ for which there is a reflection*

$$(\mathbb{PG}/\tau)^{\mathfrak{c}}_* \underset{F}{\overset{\top}{\rightleftarrows}} \mathcal{C}$$

consists of pattern graphs G for which there exists a sequence of pairs of edges $\mathcal{E}$ such that $\mathsf{collapse}(G_0) = G_0[\mathcal{E}]$, where $G_0 = \mathsf{td}(G, \tau_G)$, and e and e' are joinable in $G_0\lfloor e \bowtie e' \rfloor_{\mathcal{E}}$ for all $(e, e') \in \mathcal{E}$. The functor F sends G to $\mathsf{collapse}(G_0)$.

It follows categorically that this is sufficient to show the key required characterisation of pushouts:

Theorem 3. *Let $R \xleftarrow{f} L \xrightarrow{g} S$ be a span in $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$ and let the cospan $R \xrightarrow{g'} T \xleftarrow{f'} S$ be its pushout in $(\mathbb{PG}/\tau)_*$.*

- *If T is not in $\mathcal{C}$ then the span $R \xleftarrow{f} L \xrightarrow{g} S$ has no pushout in $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$.*
- *If T is in $\mathcal{C}$ then the cospan $R \xrightarrow{\mathsf{collapse}_{T_0} \circ \mathsf{td}_T \circ g'} \mathsf{collapse}(T_0) \xleftarrow{\mathsf{collapse}_{T_0} \circ \mathsf{td}_T \circ f'} S$ is a pushout of the span $R \xleftarrow{f} L \xrightarrow{g} S$ in $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$, where $T_0 = \mathsf{td}(T, \tau_T)$.*

6 Pushouts of Σ-graphs

In the previous section, we saw a necessary and sufficient condition for the existence of pushouts in the category $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$. We now tie the result back to the category of Σ-graphs. The aim is that this should involve a minimal amount of effort specific to Σ-graphs since similar analyses will be required when considering other models, for example Σ-graphs equipped with regions. In fact, the only requirement that has to be proved specifically for Kappa is Lemma 10, which establishes that if there is a regular epi from the encoding of a Σ-graph to a coherent pattern graph C then C is isomorphic to the encoding of some Σ-graph.

Lemma 8. *Let $\mathcal{C}$ be a full subcategory of $\mathcal{D}$ and suppose that every morphism in $\mathcal{D}$ is equal to a regular epi followed by a mono, $\mathcal{C}$ has finite coproducts preserved by the inclusion and any regular epi $e : C \to D$ in $\mathcal{D}$ from some C in $\mathcal{C}$ implies that $D \cong C'$ for some C' in $\mathcal{C}$. For a span of morphisms in $\mathcal{C}$, any pushout in $\mathcal{D}$ is also a pushout in $\mathcal{C}$ and any pushout in $\mathcal{C}$ is also a pushout in $\mathcal{D}$.*

Following the remark after Lemma 1, we regard the category $\Sigma\mathbb{G}_*$ as a full subcategory of $(\mathbb{PG}/\hat{\Sigma})^{\mathfrak{c}}_*$. Firstly, note that $\Sigma\mathbb{G}_*$ has coproducts obtained by taking the disjoint union of Σ-graphs and that these are preserved by the inclusion. For any τ, the regular epis of $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$ are characterised as follows:

Lemma 9. *A morphism $f : G \to H$ in $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$ is a regular epi if, and only if:*

- *for all $w \in V_H$ there exists $v \in V_G$ such that $f(v) = w$,*
- *if $(w, \lambda, w') \in E_H$ then there exist $v, v' \in V_G$ such that $(v, \lambda, v') \in E_G$ and $f(v) = w$ (and hence $f(v') = w'$), and*
- *if $(w, \lambda, \phi) \in E_H$ for $\phi \in \mathcal{P}_{\leq}(\Lambda^*)$ then $\mathcal{S} = \{\psi \mid (v, \lambda, \psi) \in E_G \ \&\ f(v) = w\}$ is non-empty and $\phi = \bigcup \mathcal{S}$.*

Monos in $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$ are total, injective functions on vertices. It is easy to see that any morphism in $(\mathbb{PG}/\tau)^{\mathfrak{c}}_*$ factors as a regular epi followed by a mono. All that remains before we can apply Lemma 8 to obtain the required result about pushouts in $\Sigma\mathbb{G}_*$ is the following straightforward result:

Lemma 10. *For any Σ-graph S and regular epi $e : [\![S]\!] \to G$ in $(\mathbb{PG}/\hat{\Sigma})^{\mathfrak{c}}_*$, there exists a Σ-graph T such that $G \cong [\![T]\!]$.*

We conclude by applying Lemmas 8 and 10 to characterise pushouts in $\Sigma\mathbb{G}_*$.

Theorem 4. *If there is no pushout in $\Sigma\mathbb{G}_*$ of a span $S \xleftarrow{g} L \xrightarrow{f} R$ then there is no pushout in $(\mathbb{PG}/\hat{\Sigma})^{\mathfrak{c}}_*$ of $[\![S]\!] \xleftarrow{[\![g]\!]} [\![L]\!] \xrightarrow{[\![f]\!]} [\![R]\!]$. If there is a pushout*

$$\begin{array}{ccc} L & \xrightarrow{f} & R \\ {\scriptstyle g}\downarrow & \ulcorner & \downarrow{\scriptstyle g'} \\ S & \xrightarrow[f']{} & T \end{array} \quad \text{in } \Sigma\mathbb{G}_* \text{ then there is a pushout} \quad \begin{array}{ccc} [\![L]\!] & \xrightarrow{[\![f]\!]} & [\![R]\!] \\ {\scriptstyle [\![g]\!]}\downarrow & \ulcorner & \downarrow{\scriptstyle [\![g']\!]} \\ [\![S]\!] & \xrightarrow[[\![f']\!]]{} & [\![T]\!] \end{array} \quad \text{in } (\mathbb{PG}/\hat{\Sigma})^{\mathfrak{c}}_*.$$

In summary, we have the following chain of functors:

$$\Sigma\mathbb{G}_* \xrightarrow{[\![\cdot]\!]} (\mathbb{PG}/\hat{\Sigma})^{\mathfrak{c}}_* \underset{\text{collapse}}{\overset{\top}{\rightleftarrows}} \mathcal{C} \hookrightarrow (\mathbb{PG}/\hat{\Sigma})_*$$

To determine the pushout of a span in $\Sigma\mathbb{G}_*$, we take the pushout in $(\mathbb{PG}/\hat{\Sigma})_*$ of its encoding. If this is outside $\mathcal{C}$ as characterised in Theorem 2, there is no pushout of the span in $\Sigma\mathbb{G}_*$. Otherwise, the pushout is the cospan in $\Sigma\mathbb{G}_*$ that is isomorphic under the encoding to the collapse of the pushout taken in $(\mathbb{PG}/\hat{\Sigma})_*$.

7 Conclusion

This paper has begun the work of placing Kappa in a more general graph rewriting setting, abstracting away features of Σ-graphs that are tailored to the efficient representation of biochemical signalling pathways to arrive at rewriting based on pattern graphs. The central features are the capture of implicit deletion through the use of closed partial maps and the characterisation of pushouts for categories of pattern graphs.

Alongside the work presented in this paper, we have studied categorical conditions for the existence of pushouts in categories of partial maps, that for example can show that $(\mathbb{PG}/\tau)_*$ has pushouts. In this paper, we have focused on the encoding of Σ-graphs, showing how pushouts in $\Sigma\mathbb{G}_*$ can be obtained in

$(\mathbb{PG}/\hat{\Sigma})_*$. At the core of this was the intricate consideration of pushouts in the subcategory of coherent pattern graphs.

Though there has not been space to present it here, we expect that this work applies without complications to a wide range of biochemical models. For example, categories for Kappa with regions [8] can be encoded in pattern graphs, resulting in a simpler characterisation of their pushouts. We also intend to give the BioNetGen Language [2] its first categorical interpretation using the framework developed. Another area for further research is to translate the work on dynamic restriction by type in Kappa presented in [5] to the current setting.

On rewriting, there are areas where further generalisation would be of interest. A more expressive logical formalism for patterns could be adopted, connected to the work on application conditions for rules in graph transformation [7]. It would also be interesting to consider the role of negative application conditions, perhaps specified as open maps [9], in this setting; in [4], these were used to constrain matchings. Finally, and more speculatively, by studying bi-pushouts in bi-categories of spans, a new perspective on span-based rewriting approaches (see, e.g. [11, 3]) that allows duplication of entities might be obtained.

More abstractly, the restriction to finite structures in this paper can be lifted fairly straightforwardly; we intend to present the details in a journal version of this paper. However, for modelling biochemical pathways, the restriction to finite structures is no limitation.

References

1. Bachman, J.A., Sorger, P.: New approaches to modeling complex biochemistry. Nature Methods **8**(2) (2011) 130–131
2. Blinov, M.L., Yang, J., Faeder, J.R., Hlavacek, W.S.: Graph theory for rule-based modeling of biochemical networks. In: Proc. CSB. Number 4230 in LNCS (2006)
3. Corradini, A., Heindel, T., Hermann, F., König, B.: Sesqui-pushout rewriting. In: Proc. ICGT. Number 4178 in LNCS (2006)
4. Danos, V., Feret, J., Fontana, W., Harmer, R., Hayman, J., Krivine, J., Thompson-Walsh, C., Winskel, G.: Graphs, rewriting and pathway reconstruction for rule-based models. In: Proc. FSTTCS 2012, LIPICs (2012)
5. Danos, V., Harmer, R., Winskel, G.: Constraining rule-based dynamics with types. MSCS (2012)
6. Danos, V., Laneve, C.: Formal molecular biology. TCS **325** (2004)
7. Habel, A., Pennemann, K.H.: Nested constraints and application conditions for high-level structures. In: Formal Methods in Software and Systems Modeling. Number 3393 in LNCS. Springer (2005)
8. Hayman, J., Thompson-Walsh, C., Winskel, G.: Simple containment structures in rule-based modelling of biochemical systems. In: Proc. SASB. (2011)
9. Heckel, R.: DPO transformation with open maps. In: Proc. ICGT. Number 7562 in LNCS (2012)
10. Löwe, M.: Algebraic approach to single-pushout graph transformation. TCS **109** (1993)
11. Löwe, M.: Refined graph rewriting in span-categories. In: Proc. ICGT. Number 7562 in LNCS (2012)