Bio-curation for cellular signalling: the KAMI project

Russ Harmer, Yves-Stan Le Cornec, Sébastien Légaré, Eugenia Oshurko

Bio-curation for cellular signalling

the KAMI project

Russ Harmer, Yves-Stan Le Cornec, Sébastien Légaré and Eugenia Oshurko

Abstract—The general question of what constitutes bio-curation for rule-based modelling of cellular signalling is posed. A general approach to the problem is presented, based on rewriting in hierarchies of graphs, together with a specific instantiation of the methodology that addresses our particular bio-curation problem. The current state of the ongoing development of the `KAMI` bio-curation tool, based on this approach, is outlined along with our plans for future development.

1 The bio-curation problem

In multi-cellular organisms, the development, maintenance and repair of tissues are principally coordinated via decentralized *signalling*: cells send signals—usually small proteins such as hormones, growth factors or cytokines—to be received by other cells through the agency of dedicated receptor proteins embedded in their external membranes. Reception of a signal is typically transduced across the external membrane by a conformational change of the receptor protein that, in consequence, triggers various intra-cellular signalling 'pathways' [1].

Despite their name, these latter do not exist physically, as hard-wired pathways in the cell, but rather as metaphors for the cascaded activation of enzymes that perform post-translational modifications (PTMs)—most commonly phosphorylation and dephosphorylation—in order to control the assembly and disassembly of protein complexes. The metaphorical 'destination' of a pathway is the cell's DNA and the 'journey' ends in the modulation of gene expression as effected by the assembly or disassembly of complexes of transcription factors that bind directly to the DNA. At the cellular level, this typically effects a phenotypic change—advance of the cell cycle, differentiation, movement, &c.—and may also lead to the release of new signals to the environment.

This intrinsic signalling system can be perturbed by modifications to a cell's DNA—mutations or gene ablation, duplication or rearrangement—that 'reroute', 'block' or 'short-cut' its pathways; and by pharmacological interventions intended to counteract such pathological changes. Even in the absence of such extrinsic perturbations, different cells may respond differently to the same signal. In particular, different cell *types*—which express different repertoires of proteins—need not express the same receptors so that the 'starting point' of a pathway may be present in some cases yet absent in others. More generally, the intricate choreography of protein-protein interactions (PPIs)—bindings, unbindings and PTMs—that we conceptualize as pathways clearly depends on the gene expression profile of the cell (including its expression levels): a 'highway' in one cell may be a 'country lane' in another.

• *Y.-S. Le Cornec and S. Légaré were with and the other authors are with the Université de Lyon – CNRS – ENS Lyon – Université Claude Bernard Lyon 1, Laboratoire LIP – UMR5668.*

1.1 Modelling pathways

Considerable work has been done [2], [3], [4] to build cell type-specific statistical 'models from data'. Although able to recapitulate the principal highways known to operate in that cell context, such models have limited predictive power in other contexts. Indeed, this work never intends, nor claims, to seek such predictive power; on the contrary, it exploits contextuality to provide insight into the workings of particular cells. However, it illustrates very clearly the difficulty of trying to model directly in terms of pathways: such models have an inherently holistic nature and, realistically, can only be built by unbiased, statistical learning methods.

Our approach, as advocated in [5], adopts a different stance: we seek a *de-contextualized* representation of the PPIs that underlie pathways; then provide the means to *re-instantiate* that knowledge in any context in the form of an *executable* model [6]. We then attempt to *reconstruct* the biologist's notion of pathway either by extracting a suitably post-processed *causal trace* from a stochastic simulation of the model [5], [7]; or by direct construction of such a causal trace through static analysis of the model [8].

This factorization of the modelling process allows us to focus our attention on *bio-curation*: the construction of the de-contextualized representation. The consequences of this knowledge in any particular context are to be revealed by the automatic generation of an executable model and its subsequent analysis. This contrasts with most modelling methodologies that first require the modeller to understand sufficiently the very system they are seeking to model; instead, we enable an *exploratory* mode of modelling as 'tool for discovery' in order to investigate how a single 'roadmap' of PPIs can be deployed, in varying (normal or pathological) contexts, to exhibit distinct cell type-specific signalling.

However, this places constraints on what constitutes an appropriate executable model. The principal requirement is that it provides a notion of execution trace based on discrete *events*—occurrences of PPIs—from which causal traces can be extracted, cf. Mazurkiewicz traces [9], to be compared to the biologist's imagined pathways. This rules out ODE models—which have no such discrete notion of event—and, although Mazurkiewicz's theory applies to reaction-based models, the resulting causal traces are overly fine-grained, since a single PPI is encoded as a family of reactions, and so tend to correspond badly to the expected pathways.

For example, suppose protein B can independently bind proteins A and C to form a complex ABC via intermediates AB or BC. This means that A's binding to B is insensitive to the presence of C bound to B (and analogously for the binding of C to B). As such, 'A binds B' must be expressed by two reactions: $A, B \rightarrow AB$ and $A, BC \rightarrow ABC$ (and analogously for 'C binds B'). In the event that an A and B first react to form AB, via the first reaction, this would *create an instance* of the $AB, C \rightarrow ABC$ reaction which, if applied, would force Mazurkiewicz's theory to deduce a *spurious* causality: 'C binds B' is supposed to be independent of A but this is confounded by the obligation to represent 'C binds B' by two fine-grained reaction instances—that make explicit the presence or absence of A. This mismatch between the level of representation and the desired notion of causality complicates—and compromises the scalability of—the use of reaction-based models for our purposes.

This problem can be alleviated by an approach known as *rule-based modelling*, based on graph rewriting, exemplified by the BioNetGen[1] [10] and Kappa[2] [5] languages. In this setting, a PPI is represented by a single graph rewriting rule and the above issue of spurious causality no longer arises: the protein B would have two binding *sites*, one for A and one for C, and the rule 'A binds B' would not mention the binding site for C (and vice versa) so Mazurkiewicz's trace theory, which can be generalized to such graph rewriting settings [11], [7], [12], would deduce no causality.

Let us note that the most appropriate notion of causal trace for *reversible* systems remains unclear since the naive generalization of Mazurkiewicz traces does not eliminate loops in causal traces. In the face of this uncertainty, Kappa currently offers three progressively refined notions: *uncompressed* Mazurkiewicz traces that may contain uninformative 'do-undo' event loops; *weakly compressed* traces that employ heuristics to eliminate such 'do-undo' loops; and *strongly compressed* traces that further quotient by conflating all instances, i.e. individual proteins, of each type of protein in the model [7], [8]. Although further clarification is still required, the latter two notions correspond closely, in many cases, to intuitive notions of pathway employed by biologists.

1.2 Representing PPIs

The protein-centric representation of rule-based models fixes, to a good first approximation, the mismatch with the desired notion of causality. However, for the purpose of providing a de-contextualized representation of PPIs, it has some serious shortcomings. The principal difficulty comes from the fact that, although one rule corresponds to one PPI, in practice many PPIs share a single bio-chemical *mechanism*, e.g. bindings or enzymatic modifications may be shared by multiple proteins given sufficient *conservation* of sequence and/or structure. If we wish to update our knowledge about such a mechanism, this necessitates identifying, and then making 'the same' change to, *every* corresponding rule. The significance of this problem became apparent during the first author's development (in 2007–08) of a Kappa model of the erbB signalling network, as partially documented in [5], and led directly to the work on MetaKappa [13], [14].

MetaKappa provides a partial solution to the problem by enabling the definition of mechanisms as *generic* rules—that it then automatically expands into sets of Kappa rules—that can be shared by splice variants, loss-of-function mutants and even homologous genes. However, MetaKappa cannot conveniently treat the important case of gain-of-function mutants and, even more critically, the fact that mechanisms are *defined* in MetaKappa implicitly requires the modeller to *already have in mind* an *intended* set of underlying Kappa rules. In other words, MetaKappa allows us to compress a *known*, contextualized set of Kappa rules—whereas we wish to *discover* those rules by contextualizing mechanisms.

Let us finally state explicitly our *bio-curation problem* for signalling. We are aiming to enable the de-contextualized representation of knowledge about the bio-chemical binding and enzymatic mechanisms that are implicated in cellular signalling: specifically, the known *necessary* conditions for these PPIs to take place. Moreover, we wish to express this knowledge in such a way that a mechanism corresponds to a *single* 'element' of our knowledge representation—from which we can identify all its instances in order to avoid the above-mentioned 'update problem'.

We must also be able to *deploy* this knowledge through *instantiation* to specific cell-type contexts: a mechanism may not apply to a particular splice variant that, for example, lacks a necessary binding site; or a mutated protein may lose, or gain, the ability to participate in a given mechanism [15]. In particular, this requirement implies that the basic unit of our de-contextualized representation is not a *protein* but what we call an *protoform*: the ensemble of products of a gene including all its splice variants and important mutants.

Plan of the paper

This paper motivates and explains the conceptual and mathematical framework of our bio-curation tool `KAMI`[3]: Knowledge Aggregator & Model Instantiator. It extends and supersedes the extended abstract [16] in the light of some small, but significant, changes to the meta-model. It exploits the factorization of the modelling process (discussed above) to focus exclusively on representational issues. In particular, in order to respect the page limit, there is no discussion of dynamics, i.e. how `KAMI` generates executable Kappa models after instantiation; nor do we provide any experimental validation of the tool itself in the form of a significant use case. A follow-up tool paper is being prepared to address both these limitations.

In §2 and the appendix, we provide the necessary mathematical background to define graph rewriting in hierarchies of graphs. In §3, we discuss the requirements on `KAMI`'s knowledge representation (KR) and describe its structure as a hierarchy of graphs, rooted in a domain-specifying meta-model, with domain-specific background knowledge and model-specific content. In §4, we show how to use graph rewriting to update the contents of the KR with particular emphasis on the exploitation of background knowledge to automate this update process. In §5, we use graph rewriting to define the re-contextualization of the contents of the KR into specific PPIs. We conclude with a discussion of future perspectives and related work.

1. `http://bionetgen.org/index.php/Main_Page`
2. `https://kappalanguage.org`
3. `https://github.com/Kappa-Dev/KAMI`

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under grant numbers W911NF-14-1-0367 and W911NF-15-1-0544. The views, opinions, and/or findings contained in this report 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.

The first author thanks Pierre Boutillier and especially Walter Fontana for many discussions over the years related to this work. Thanks also to Adrien Basso-Blandin and Ismaïl Lahkim Bennani who worked on early prototypes.

2 Graph rewriting

This section briefly explains the mathematical formalism underlying KAMI's knowledge representation. Specifically, it defines the class of graphs that we use and how rewriting rules are defined and applied.

2.1 Graphs and homomorphisms

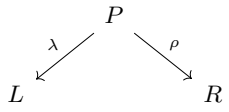
We consider a *graph* to be a simple directed graph with attributes on nodes and edges an attribute consists of a *key* to which a *set* of values may be associated. The widely-used Python library networkX[4] provides this class of graphs and our implementation of graph hierarchies and rewriting—in the Python library ReGraph[5]—is built on top of networkX.

A *homomorphism* $h : G \to G'$ is a total function mapping the nodes of G to the nodes of G' in such a way that (i) if there is an edge from node n_1 to node n_2 of G then there is an edge from $h(n_1)$ to $h(n_2)$ in G'; and (ii) if n_1 has an associated key k with set of values V then $h(n_1)$ also has the key k with set of values $V' \supseteq V$. In words, h preserves all edges and attributes of G.

2.2 Sesqui-pushout graph rewriting

Graph rewriting makes use of three basic operations that correspond to generalized notions of intersection, union and set difference on nodes, edges and attributes. These are called respectively *pullback*, *pushout* and *pullback complement* and are defined, abstractly and in our concrete setting, in the appendix. *Sesqui-pushout* (SqPO) rewriting [17] can be defined in any setting where these three operations are well-defined—with the requirement for pullback complements to exist only when the second homomorphism is injective.

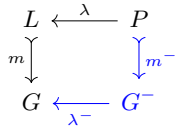
A *rewriting rule* is a *span*, i.e. a pair of homomorphisms from a common source object P



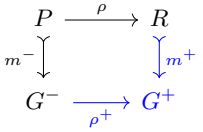
where the left-hand side L defines the *pattern* that the rule must *match*. The articulation between L and R is expressed in P, the *preserved* region of the rule, that specifies which nodes, edges and attributes of L correspond to those of R and which nodes, edges and attributes of L (resp. R) will disappear (resp. appear).

4. https://networkx.github.io
5. https://github.com/Kappa-Dev/ReGraph

The left leg of the span $\lambda : P \to L$ specifies the cloning of nodes and the deletion of nodes, edges and attributes. We give specific details for the graphs used by KAMI in the appendix. An instance—also often called a *matching*—of this in a graph G is an injective homomorphism $e : L \rightarrowtail G$. This *restrictive* phase of a rewriting step from G to G^- is defined by taking the pullback complement.



The right leg of the span $\rho : P \to R$ defines the adding of nodes, edges and attributes and the and merging of nodes. This *expansive* phase of rewriting from G^- to G^+ is defined by taking a push-out.



A rewriting rule can be viewed as a *relation* between L and R: a node n_L of L is related to a node n_R of R iff some node of P is mapped by λ to n_L and by ρ to n_R; we call this node of P the *witness* of the relation between n_L and n_R.

3 KAMI's knowledge representation

Statements about mechanistic aspects of PPIs abound in the signalling literature. They occur with varying granularity and detail ranging from vague, coarse-grained statements such as '*Grb2* binds *Shc*' or the slightly more detailed '*Grb2* binds phosphorylated *Shc*' to detailed, fine-grained statements such as 'the *SH2* domain of *Grb2* binds *Shc* phosphorylated on Y317'. As such, our knowledge representation must be capable of accommodating formal counterparts to this entire spectrum of more or less detailed statements.

In order to fulfill this rôle, we need to represent certain *anatomic* aspects of proteins such as regions, residues, binding sites and (physical or phenomenological) states (that represent the presence or absence of PTMs or consequent phenomenological conditions such as 'activity'). We must also be able to represent the three general classes of PPIs implicated in signalling—*bindings*, enzymatic *modifications* and *unbindings*—as well as tests for the *presence* or *absence* of states and bonds which enable us to express state-dependent and (positive or negative) allosteric control of PPIs.

3.1 The meta-model of KAMI

These various *kinds of* concepts can be naturally represented as the nodes of a particular graph, which we call the *meta-model* of KAMI, and relationships between these nodes as edges between these nodes.

The meta-model is shown in figure 1); its nodes and edges also have attributes that we now discuss. Let us first note that the meta-model enforces certain domain-specific but model-independent constraints on *all* KAMI models. Technically, this means that all graphs that are valid in KAMI must be homomorphic to the meta-model. In particular, *all* attributes and *all* their possible values must be present in the meta-model.

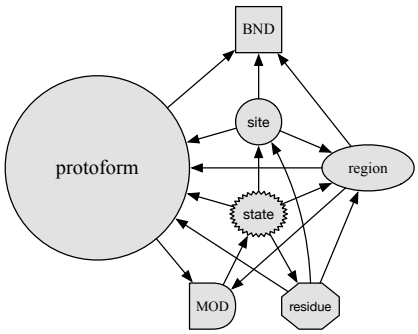


Fig. 1. The (nodes and edges of the) meta-model of KAMI

These attributes are of two distinct types: those serving as pure meta-data; and those that may be tested as one of the necessary conditions of an interaction. Meta-data attributes occur on the protoform, region and site nodes for (string-valued) database references such as UniProt accessions, InterPro IDs and so on; and on the BND and MOD nodes for (positive real-valued) rate constants.

The principal examples of necessary conditions tested by interactions are the presence or absence (i) of PTMs such as phosphorylation; (ii) of bonds; or (iii) of certain amino acids in specific sequence locations. These correspond to (i) states that serve as proxies for the activity (or otherwise) of certain enzymes; (ii) protoform positive or negative *allosteric* control; and (iii) whether or not a wild-type or mutant protein is known to lose or gain an interaction.

The state of a PTM is represented by the state node that has two attributes: value:$\{0,1\}$ and test:$\{+,-\}$. These attributes of the meta-model constraint all values of KAMI states to be subsets of $\{0,1\}$; and all tests of a state to be for the presence $(+)$ or absence $(-)$ of the stated value(s). Phenomenological states, such as the *activity* of an enzyme, can also be represented in this way.

The MOD node represents actions that change the value of states; it has a single (non-meta-data) attribute called value:$\{0,1\}$ that specifies the value to be written to the state to which it points. A MOD node may also have an incoming arrow specifying the agent, e.g. an enzyme, responsible for the modification.

The BND node has two dual aspects: it can represent both an *action* and a *test*—albeit not at the same time. This is handled by an attribute called type:$\{do, be\}$. As for states, there is a second attribute test:$\{+,-\}$. The combination of type:do and test:+ corresponds to a *binding action*; and type:do and test:− to an *unbinding action*. Similarly, type:be and test:+ corresponds to a *bound test*; and type:be and test:− to an *unbound test*.

The third class of necessary conditions requires us to test the amino acid identity of a residue. The residue node has two attributes to enable this: aa, taking values from the set of one-letter codes for amino acids; and test:$\{+,-\}$, as for state and BND nodes. There is one more (integer-valued) attribute loc that represents the sequence location of the residue. However, this is attached not to the residue node itself but to its outgoing arrow to the protoform node. Similarly, the site and region nodes have integer-valued attributes start and end for their sequence intervals.

Let us note that there is no significant technical difference between attributes and states: a state is simply a special kind of attribute whose value we wish to be able to modify *within* a model. Currently, this means either a physical state, representing a PTM, or a phenomenological state such as enzymatic activity. If we wish to be able to modify the value of the aa attribute of a residue *within* a model, i.e. an actual *act* of mutation, we would need to reify aa as a state whose value would then be controllable by a MOD.

3.2 Models in KAMI

The meta-model defines the *kinds* of things that can exist in KAMI. A *model* in KAMI is an *instance* of the meta-model, i.e. the *actual* things that exist in the model. This comprises two distinct components: an *action graph* which defines the 'anatomy' of the model—the proteins, their regions and residues, &c. and their interactions—and a collection of *nuggets* that define its 'physiology'—the detailed descriptions of the necessary conditions for interactions.

The *action graph* is a graph that is homomorphic to the meta-model, i.e. each of its nodes maps to a node of the meta-model—which defines what *kind* of node it is—in such a way that all its edges and attributes are preserved, i.e. that they are permitted by the meta-model.

We give an example of an action graph in figure 2. The mapping of its nodes to those of the meta-model are given implicitly by the shapes: big circles are proteins, ovals are regions, small circles are sites, &c. We label (some) nodes for the sake of readability and ease of reference; technically, these are really string-valued attributes.

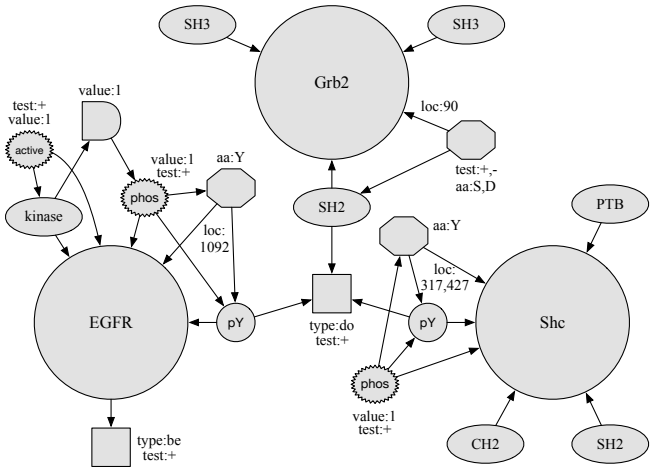


Fig. 2. An example action graph in KAMI

This action graph defines three protoforms and aspects of their anatomies, two BNDs (one of which is a binding action, the other a bound test) and a MOD setting value 1.

A *nugget* is a graph homomorphic to a given action graph; and a *model* consists of an action graph together with a collection of nuggets. For example, in figure 3, we give a nugget (with respect to the action graph in figure 2) that expresses 'the site *pY* of *EGFR* with residue *Y1092* whose state *phos* has value 1 binds to the region *SH2* of *Grb2*'.

In effect, an action graph can be seen as *a schema* for *a* model and the meta-model as *the schema* of *all* KAMI models.

As things stand, the state *phos* has no *semantics*: its name suggests 'phosphorylation'—but we must wait for the next subsection to make this kind of domain-specific background knowledge precise.

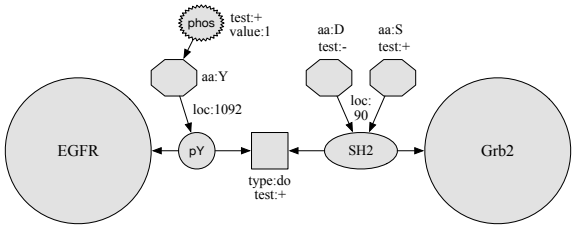


Fig. 3. An example nugget in KAMI

This nugget makes additional tests concerning the amino acid at location 90 of *Grb2* (within the *SH2* domain). In the action graph, there are two possible values—`aa:S` or `aa:D`—that this can take. Of course, in principle, there are twenty possible values; all we mean here is that, to date, we have only considered these two values (one of which is the wild-type and the other a known mutation). The nugget states a positive requirement for `aa:S`, i.e. serine, although there could have been a set of values—meaning that there is a positive requirement for *any one* of those values. However, there is also a negative requirement for `aa:D`, i.e. aspartic acid, meaning that we *know* that the presence of aspartic acid at location 90 *prevents* binding.

Had the nugget not specified this negative constraint, its meaning would have been subtly but crucially different: we *do not know* whether or not the presence of aspartic acid at location 90 affects the binding. The given nugget therefore includes the additional knowledge that 'the *S90D* mutation abrogates binding' by explicitly stating the negative constraint via the `test:-` attribute. In general, the use of the `test:+` and `test:-` attributes allows us to make the critical distinction between *not knowing* and *knowing not*.

As another example, consider the nugget in figure 4 that states 'the region *kinase* of an *EGFR* whose state *active* has value 1 modifies to value 1 the state *phos* of the residue *Y1092* of a second *EGFR* bound to the first'. This nugget includes an explicit test of the presence of a bond between the two *EGFR*s.

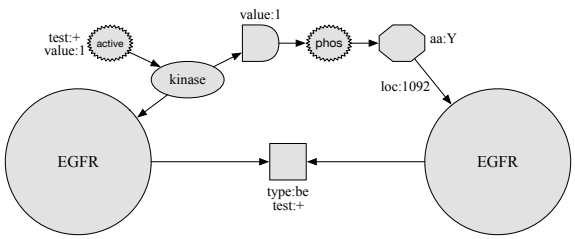


Fig. 4. Another example nugget in KAMI

Note that anatomic aspects of genes may have multiple outgoing edges in the action graph, e.g. the state *active* in figure 2. The purpose of having these multiple edges is to be able to express nuggets with varying levels of detail: one may refer to 'the *active* state of *EGFR*' while another refers to 'the *active* state of the *kinase* region of *EGFR*'.

3.3 Background knowledge in KAMI

The English transliterations of the nuggets in figures 3 and 4 are clumsy and do not correspond to the way that biologists speak about PPIs. However, because nuggets and action graphs are formal, purely syntactic entities, we can only express domain-specific properties, such as phosphorylation, by inventing, and rigorously abiding by, an arbitrary syntactic convention, e.g. any state labelled *phos* should be thought of as *meaning* phosphorylation—and treated accordingly.

For example, if we take the nugget in figure 4 but change the value to be written to 0, the nugget remains perfectly well-formed syntactically but no longer respects the *intended meaning*: a kinase is, by definition, an enzyme that phosphorylates—so it cannot write the value 0 to a state intended to represent phosphorylation.

It would clearly be unwieldy to attempt to maintain the appropriate convention for each and every instance of an intended domain-specific property. A more robust approach would be to state explicitly, in one place somewhere in KAMI, all the domain-specific background knowledge that we wish to represent and the constraints that this implies.

We represent this background knowledge with the so-called *semantic action graph* (SAG), which is homomorphic to the meta-model, and a collection of *semantic nuggets* that are all homomorphic to the SAG—in effect, a semantic counterpart to a model. This is a built-in component of KAMI that can nonetheless be modified and extended over time.

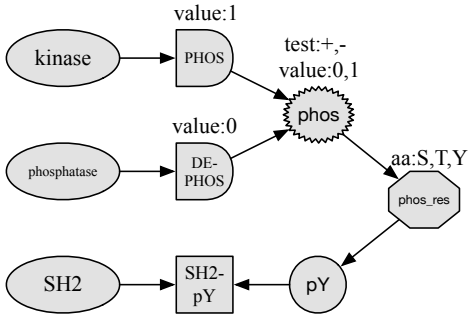


Fig. 5. The (current) semantic action graph of KAMI

The SAG is shown in figure 5. It defines *kinds of* semantic things: *kinase*, *phosphatase* and *SH2* domains, *pY*-binding motifs (sites) and *phos* states. It also defines associated *kinds of* semantic actions—*PHOS*, *DEPHOS* and *SH2–pY* binding—and states some of their associated constraints, e.g. *PHOS* actions act on *phos* states and may only set their value to 1.

KAMI allows us to provide a relation between an action graph and the SAG to specify the semantic attribution, if any, of its regions, residues, sites, states and actions; see the appendix for full details.

A semantic nugget provides a *template* for the generic necessary conditions of a PPI such as *SH2–pY* binding or phosphorylation (figure 6). Note how the residue in the former restricts the permitted values of its `aa` attribute; this expresses its dependence on *tyrosine* phosphorylation.

The relation between the action graph of a model and the SAG lifts to relations between each of its nuggets and the semantic nuggets. This will allow us to express the idea that a nugget can be an instance of a generic semantic PPI.

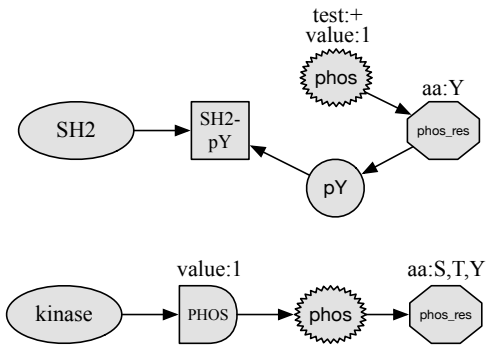


Fig. 6. The semantic nuggets for *SH2*–phospho-tyrosine binding and phosphorylation

In section 4, we explain how we can (sometimes) automatically recognize that a nugget is an instance of a semantic nugget, how missing details can be auto-completed in such cases and how this can be used to guarantee that the resulting nugget respects the intended meaning. In effect, this provides—and provides the means to enforce automatically—a collection of syntactic conventions to capture domain-specific semantic properties such as those of phosphorylation or SH2 domain-binding discussed above.

4 KNOWLEDGE AGGREGATION

In this section, we discuss how knowledge can be imported into KAMI and the various ways in which such inputs may update the state of the KR.

As we have seen, the KR has been designed to accommodate knowledge at varying levels of granularity. However, the representation remains somewhat stylized in that only one 'type' of modification exists formally which, in practice, subsumes a number of distinct but related cases: classical modification of one entity bound to another, self-modification of a single entity and the case where the modifying entity is currently unknown.

Clearly, these can all be captured by our single type of modification; however, the information required and the details of the processes of construction of the corresponding nuggets differ. For this reason, we choose to represent inputs to KAMI with an intermediate language—whose terms are called KAMI *interactions*—that distinguishes these kinds of cases in order to ease the transition from the way in which biological knowledge is generally expressed to the more abstract form required by our KR.

A raw input to KAMI, provided in the intermediate language, gives rise to an update of the KR by building a graph rewriting rule that respects the meta-model and, as such, has an intrinsic *semantic* character: an update expresses more than just a 'diff'; it is stated in terms of a *meaningful* change in an expert's knowledge about something in the KR. An update may apply only to the action graph—typically when we provide some new anatomic information about a gene—or, more commonly, adds a new nugget or modifies an existing one. In the latter case, downward propagation maintains consistency with the action graph if the update requires the introduction of new nodes, edges or attributes to the action graph.

The history of updates thus provides an *audit trail* that recapitulates, in properly semantic, domain-specific terms, the *modelling process* itself. In particular, it maintains a record of how knowledge was aggregated from various sources—principally scientific papers but also potentially from databases—thus providing some transparency and clarity—as well as support for model *maintenance* and future *update*—in the face of the fragmentary, dispersed nature of the primary bio-medical literature. Indeed, let us emphasize that this is the *only* reason why KAMI has been built on top of a rather sophisticated graph rewriting technology.

The raw input, as initially provided in the intermediate language and subsequently re-expressed as a nugget, may be additionally auto-completed in the case that we can exploit background knowledge to fill in missing details. At the time of writing, KAMI performs two distinct types of auto-completion. Firstly, when a gene is mentioned in an input for the very first time, we invoke a specialized module—the *gene anatomizer*—to construct a representation of anatomic aspects of that gene, e.g. its known regions and modifiable residues, from standard databases such as UniProt and InterPro. Secondly, if an input is judged to be an instance of a generic semantic nugget, even though it does not include all the details expressed by the latter, we can construct a rewriting rule that automatically adds those missing details. The identification of an input as being an instance of a semantic nugget may also allow us to decide whether that input refers to a completely new nugget or whether some version of it already exists in the KR.

4.1 KAMI interactions

KAMI interactions are defined as standard Python user-defined data structures in the `kami.entities` and `kami.interactions` modules.

KAMI entities are Python classes that reflect the structure of the meta-model in that they define `Protoforms` (potentially containing regions, sites and residues), `Regions` (potentially containing sites and residues), `Sites` (potentially containing residues) and `Residues`; states and attributes may also be specified for each of these kinds of entity. These classes all inherit the class of *physical entities*.

For example, 'phosphorylated *Shc*' and 'an *SH2* domain' (in the sequence interval 58–152 of some protoform) could be written as the following entities:

```
pShc = Protoform(
    uniprotid="P29353",
    states=[State("phos",True)])
sh2 = Region(
    interproid="IPR000980",
    start=58, end=152)
```

At the time of writing, a nugget must be 'rooted' in protoforms: we cannot express that a disembodied region modifies some state; the protoform to which the region belongs must additionally be specified. In consequence, we provide a second class, of `Actors`, that is inherited by `Protoforms` (again), `RegionActors` (with respect to a given protoform) and `SiteActors` (with respect to a given protoform and, optionally, region). These classes do nothing more than provide a fully-grounded context for an entity, e.g. the protoform to which a region belongs.

For example, if we wish to use the *SH2* domain of *Grb2* as one of the two participants in a binding nugget, we must encapsulate the above definition inside a `RegionActor` that additionally specifies the *Grb2* protoform:

```
Grb2_SH2 = RegionActor(
    gene=Protoform(
        uniprotid="P62993",
        states=[State("phos",True)])
    region=sh2)
```

A binding interaction between 'phosphorylated *Shc*' and 'the *SH2* domain of *Grb2*' can now be written simply as

```
Binding(
    left=pShc,
    right=Grb2_SH2)
```

However, strictly speaking, and just as in section 3.3, we have not yet specified the semantics of the state `phos` of `Shc` and so cannot yet speak of 'phosphorylated *Shc*' but only of '*Shc* in state `phos:True`'.

More generally, we need to specify how and when semantic information becomes available to `KAMI`. First of all, in some cases, a `KAMI` interaction already includes certain kinds of semantic attribution, e.g. a UniProt accession number uniquely identifies the protoform in question while an InterPro ID identifies the *type* of a domain, not an individual instance, e.g. a region tagged `IPR000980` is an *SH2* domain. In other cases, we may be able to exploit existing semantic attribution as expressed in the relation between the action graph of our model and the semantic action graph. To do this, we must first construct a nugget from the given `KAMI` interaction and then insert it into the model by providing a homomorphism to the action graph; only then can we determine whether or not an entity or state maps to a node of the action graph that already has a semantic attribution.

Let us note that `KAMI` interactions implicitly introduce some constraints on the shape of the nuggets that we can add to the system. For example, at the time of writing, there is no means of writing a bond displacement—where one binding partner is initially bound to a third party and the other binding partner actively disrupts that pre-existing bond—or a coupled modification and unbinding. Such nuggets would have two action nodes: in the first case, one that unbinds the pre-existing bond; and another that creates the new bond; in the second case, one for the modification and another for the unbinding.

As such, the language of `KAMI` interactions provides only a partial coverage of the entire 'representation space' of `KAMI`. We have made this choice for the pragmatic reason that most real inputs to `KAMI` only need a single action and we anticipate that a small number of *ad hoc* additional `KAMI` interactions will suffice, in practice, to cover the exceptions to this general rule.

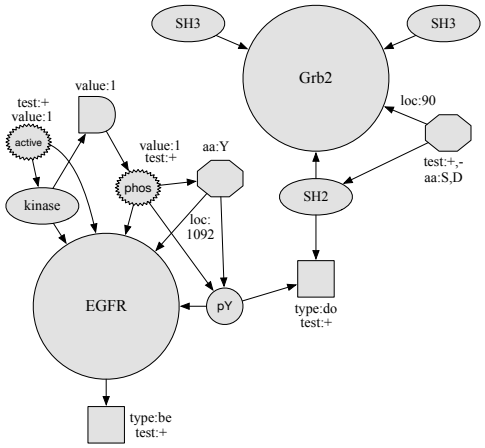
However, in principle, we can write nuggets that express arbitrarily complex combinations of actions in `KAMI`—up to the constraint that the actions must not be in conflict with each other. In other words, entire *sequences* (or even DAGs) of actions could be grouped together into a single mechanism, e.g. a piece of a signalling pathway. We return to this point later in the discussion about future directions for `KAMI` in section 6.

4.2 Nugget construction and aggregation

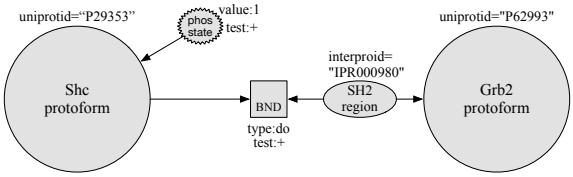
We now detail the process of automatic aggregation of PPIs into a `KAMI` model. The Aggregation Engine of `KAMI` takes as input a `KAMI` interaction and performs a sequence of operations that result in an appropriate update of the model.

First, we generate a proto-nugget graph that is typed by, i.e. homomorphic to, the meta-model; secondly, a *rôle relation* is constructed between this proto-nugget and the appropriate *template* graph; and, finally, a relation between the proto-nugget and the current action graph is constructed.

Let us illustrate these steps on our above `KAMI` interaction with respect to the following action graph:



We begin by parsing the interaction into the following proto-nugget graph; the typing into the meta-model is determined directly from the `KAMI` interaction.



At the current time, there are two templates in `KAMI`: one for `MOD` actions and the other for `BND` actions (figure 7). A *relation* between a template and a nugget can be thought of as an assignment of the rôles given by the nodes of the template to those of the nugget.

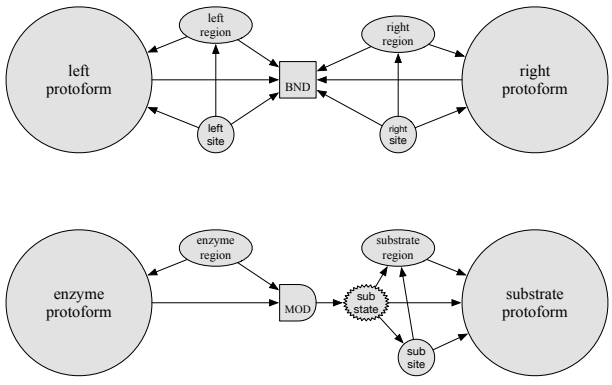
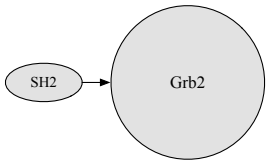


Fig. 7. `BND` and `MOD` nugget templates

The *rôle* relation between our proto-nugget and the BND template identifies *Shc* as the left and *Grb2* as the right protoform; and *SH2* as the right region. (Clearly, the assignment of left and right is completely arbitrary.)

The *grounding* meta-data in the proto-nugget allows us to read off that the protoform with UniProt accession number P62993, i.e. *Grb2*, already exists in the action graph whereas the other with accession number P529353, i.e. *Shc*, does not. We can also see that the region with InterPro ID IPR000980, i.e. the *SH2* domain of *Grb2*, already exists.

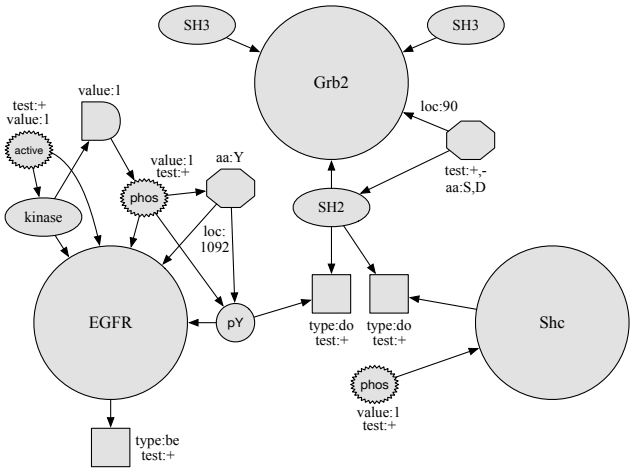
This allows us to define a relation between our proto-nugget and the current action graph: its witness R_T is



corresponding precisely to the fact that the *Grb2* protoform and its *SH2* region are the only entities that already exist in the action graph T.

The proto-nugget R can now be *aggregated* into the model essentially by performing the push-out of the relation $T \leftarrow R_T \rightarrow R$ to construct the minimal extension of T that is required to type R. This is a straightforward instance of the downward propagation of rewriting as described in the appendix.

In our running example, we obtain:



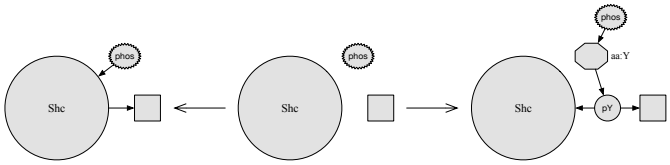
Note that this adds a new BND action to the action graph. This is inevitable since no standard grounding, analogous to UniProt or InterPro IDs, has yet been defined for PPIs[6].

However, if we can further identify the newly-added nugget as being an instance of a semantic nugget, we may be able to *sharpen* this update by taking into account constraints coming from background knowledge. Indeed, in this case, the bio-physical constraint that *SH2* domains can bind at most one phospho-tyrosine motif at a time implies that KAMI should merge the new BND action with the existing one—provided it can satisfy itself that the new nugget really is an instance of the *SH2–pY* semantic nugget.

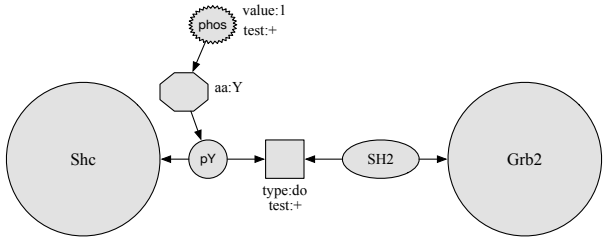
6. Indeed, a notable longer-term side effect of KAMI is precisely to provide such a grounding!

KAMI takes this decision by determining whether or not the new nugget 'sufficiently matches' the semantic nugget. Specifically, in our current implementation, if the left (resp. right) protoform is phosphorylated—either directly, on a site or on a tyrosine residue, i.e. having the attribute aa:Y—and the right (resp. left) protoform binds through an *SH2* domain then this is taken to correspond to an instance of the *SH2–pY* semantic nugget[7]. In the event of a match, the system uses the semantic nugget to construct an *auto-completion* rewriting rule[8] to be applied to the nugget.

In our example, the auto-completion rule is:



We can now apply this rule to our nugget to obtain its auto-completion:



This nugget update propagates downwards to the action graph to reflect the fact that new entities have been created.

The update is completed by a final phase that consists in the construction of rewriting rule that is applied only to the action graph. This rule performs some pedestrian 'book-keeping' updates that guarantee the transitive closure of anatomic structure, e.g. a state that belongs to a residue that belongs to a site also belongs directly to that site.

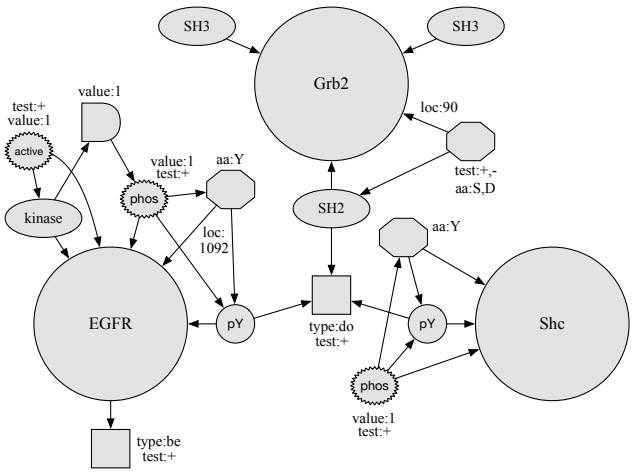
However, it may also be required to restore certain invariants, demanded by background knowledge, that have been broken by the downward propagation of auto-completion. For example, in our running example, we must merge the two *SH2–pY* binding actions incident to the *SH2* domain of *Grb2* in order to satisfy the constraint that an *SH2* domain can sustain a single phospho-tyrosine binding at any given time. Similar constraints apply to the cases of phosphorylation and dephosphorylation since kinase and phosphatase domains have a single mode of enzymatic action—and so a single edge to a PHOS or DEPHOS node[9].

7. Clearly, these conditions could be made more or less stringent. They could also be made 'programmable' by allowing the express of *mandatory* nodes, edges and attributes in the semantic nugget to specify the notion of sufficient matching.

8. At the time of writing, this rule is constructed by cases; however, by adding a protoform node to the semantic action graph, we could construct this rule automatically by computing an appropriate pullback. Such an approach is likely to prove more scalable as we add further semantic nuggets of this general domain–motif binding type.

9. At present, this part of the rule is constructed explicitly by cases. However, simple constraints of the form 'at most one incident edge', as in SH2 domains or enzymatic modifications, could be expressed directly in the semantic action graph from which the necessary rule could be computed.

The final action graph, where the two SH2−pY nodes incident to the $SH2$ domain of $Grb2$ have at last been merged, is:



This action graph formalizes the fact that the two nuggets that have an action that maps to this node—for $Grb2$'s binding to $EGFR$ and Shc respectively—are two instances of the *same* mechanism. This enables us to solve the update problem discussed in the introduction while allowing us to maintain the flexibility of being able to add some further *ad hoc* necessary conditions to one nugget or the other without necessarily having to propagate this to the other.

4.3 The Anatomizer

The Anatomizer is a module of KAMI that can be used during the process of nugget construction and aggregation—but also independently. It makes use of a number of databases—most notably UniProt and InterPro—to retrieve anatomic information and further meta-data about protoforms. At the time of writing, it uses InterPro to produce a list of the known *regions* of the protoform, accompanied by a name, sequence interval and InterPro ID. It also uses UniProt to collect meta-data such as the HGNC symbol, synonyms and references to other databases.

Once it has collated this information, the Anatomizer generates an expansive rewriting rule $P \to R$, i.e. where the left leg of the rule is the identity on P and is therefore trivial, called an *anatomization* rule which performs the corresponding update of the action graph. If the protoform being anatomized does not yet exist in the action graph, P is simply the empty graph and R is built by adding the *protoform* and all its *regions*; if it and/or some of its *regions* already exist, they constitute the graph P and R extends P by adding all the newly discovered *regions*.

If we anatomize *Shc* in our running example, we obtain—modulo the sequence location of the tyrosine residue—the action graph of figure 2 where Shc has gained three extra regions, one of which is an $SH2$ domain.

The fact that the Anatomizer extracts InterPro IDs means that it is implicitly collecting information about semantic attribution. This information is used to update the relation between the (updated) action graph and the semantic action graph, e.g. Shc's $SH2$ domain is designated as such. As discussed above, this may be of use to subsequent nugget construction and aggregation processes.

5 Model Instantiation

The basic unit of KAMI's KR is that of a (small) neighbourhood in sequence space around a protoform. A nugget specifies the known necessary conditions on its participating protoforms; this includes required regions and sites as well as amino acid constraints that capture the abrogation or creation of interactions by mutations.

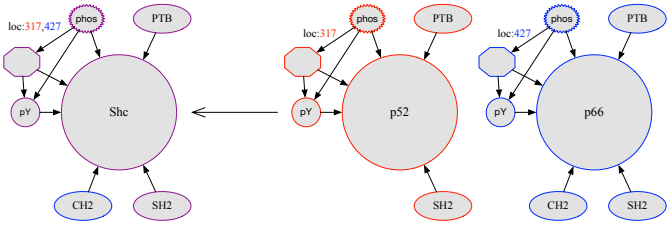
As discussed in the introduction, this *de-contextualized* representation expresses potential interactions that may or may not be possible for actual proteins, i.e. the particular inhabitants of such a neighbourhood in sequence space. For each protein of interest, an interaction can only occur if its necessary conditions can be *realized* for that protein, e.g. a splice variant may lack a certain region or a mutant may violate an amino acid constraint.

In order to re-contextualize nuggets for a given collection of proteins, we need an additional ingredient: for each desired protein, a *definition* that specifies a *single* sequence in the appropriate sequence neighbourhood that determines which anatomic aspects of the protein are actually present and which amino acid is present at each residue.

5.1 Protein definitions

The *Shc* gene gives rise to several proteins including the so-called *p52* and *p66* proteins (named after their molecular weights in kDa). The shorter *p52* lacks the first 110 residues of *p66*; as such, residue location 427 in *p66* becomes 317 in *p52*. Moreover, the *CH2* region of *Shc* occurs within those first 110 residues and does not occur in the *p52* protein.

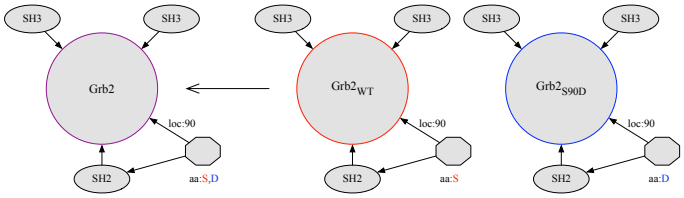
We express the *definition* of these proteins by the rule



which clones the *Shc* protoform, its various regions, residues and states—with the exception of *CH2* that only occurs in *p66*—and *dispatches* sequence locations and their amino acid identities appropriately to the proteins on the right-hand side of the rule.

Let us note that, in each case, the rewriting rule defining proteins is purely restrictive, i.e. of the form $L \leftarrow P$ with a trivial right leg, precisely because the de-contextualized protoform contains all possible regions, sites, residues, states and attributes of all its possible proteins; indeed, it can be thought of as their superposition. As such, we only ever need to clone and delete in order to define specific proteins. Indeed, each definition specifies a combination of deletions that precisely specifies the passage from a superposition to an actual protein.

The *Grb2* gene does not have any known splice variants but it does have the possibility of the *S90D* mutation; as such, we must resolve this sequence nondeterminism in the definition of its proteins (even if we only wish to instantiate one of the two possible proteins).



Moreover, the left-hand side of a rewriting rule that defines gene products must be *injectively typed* by, i.e. a subgraph of, the action graph: typing by the action graph is fundamental simply because that is where `protoforms` are defined; we further require injectivity because we wish to be able to *apply* these rules to rewrite the action graph in order to determine which nuggets a given gene product can actually use, i.e. to effect re-contextualization.

5.2 Re-contextualization

Given a model and a collection of protein definitions, we can apply them to the action graph and, through the upward propagation mechanism of `ReGraph` described in the appendix, propagate their effects to the nuggets of the model.

For example, the propagation of the two above protein definitions to the '*Shc* binds *Grb2*' nugget is shown in figure 8. This *instantiated* nugget can be read[10] as saying that *p52* phosphorylated on Y317 or *p66* phosphorylated on Y427 binds the *SH2* domain of wild-type *Grb2* or the *SH2* domain of S90D *Grb2*'.

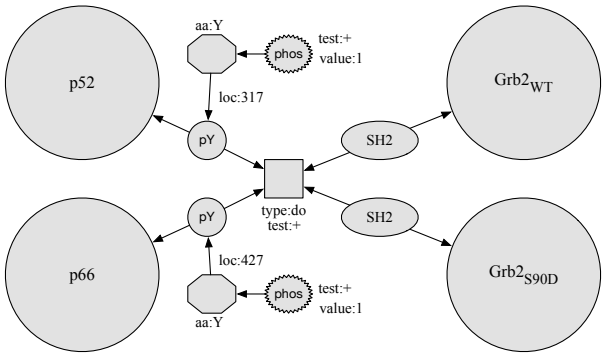


Fig. 8. The instantiation of '*Shc* binds *Grb2*'

However, if we consider propagation to the '*EGFR* binds *Grb2*' nugget, we obtain the instantiated nugget of figure 9: the `aa` attribute has the empty set of values for the residue performing the *negative* test (in wild-type *Grb2*) and for the residue performing the *positive* test (in S90D *Grb2)*. This means that wild-type *Grb2* fulfills the necessary conditions for binding but that S90D *Grb2* does not; indeed, the S90D mutant fulfills a necessary condition for *not*-binding! Let us note that, even if the original nugget did not have the negative constraint, the S90D mutant would still fail to fulfill the necessary condition for binding; it would just not fulfill any necessary condition for *not*-binding.

10. Although the `BND` node no longer has arity 2, there is no ambiguity because the *relation* between the nugget and the binding template is propagated to the instantiated nugget. In particular, if *Shc* was related to the *left* node of the template then so are both *p52* and *p66*.

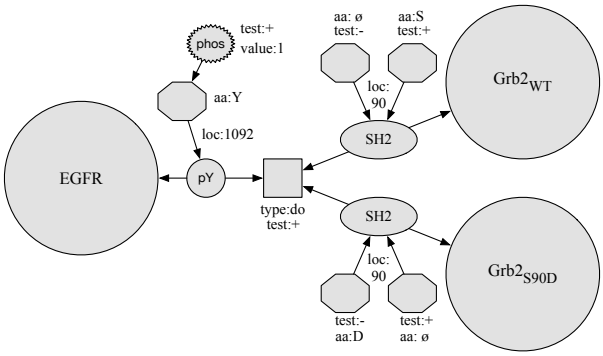


Fig. 9. The instantiation of '*EGFR* binds *Grb2*'

From the point of view of a rule-based model, these two situations are completely indistinguishable: rules can *only* express positive constraints; the effect of a negative constraint, as in this example, manifests by the *non-existence* of a rule, not by the existence of a rule that turns out never to be applicable. As such, the absence of a rule may be due to a lack of knowledge—not knowing—or because we know better—knowing not.

If we build a rule-based model directly, we have no way to distinguish these two cases and, in any large-scale curation effort, it would have to be explicitly noted as meta-data; otherwise, it would be unclear whether a rule had been forgotten or deliberately left out. `KAMI` allows the human curator to express these distinctions directly, in the very definitions of nuggets, and relegates the task of generating (and not-generating) a potentially huge number of rules to the machine. This simultaneously sidesteps the cognitive bottleneck and the scalability failure of the rule-based approach for *building* models.

6 Conclusions and Future Perspectives

We have presented an overview of the aims and current functionality of our bio-curation tool `KAMI` with particular focus on the importance of capturing mechanisms, not just individual PPIs, together with a curation procedure which exploits domain-specific background knowledge and intrinsically provides an audit trail, by constructing and storing explicit rewriting rules, documenting the curation process. The tool is based on solid theoretical foundations, as discussed specifically for `KAMI` in the appendix and also, to some extent and in greater generality, in [6], [12].

Rule-based modelling has been around for over a decade now. The stochastic simulation technology has greatly matured over this time, so that very large models can be run, and the causal analysis technology specific to Kappa has also significantly developed—although more work remains to be done. `KAMI` is intended as a further tool to allow the full exploitation of this technology by scaling up the size of models that can be realistically built. In this way, we seek to enable new kinds of questions that models can be used to address: instead of seeking to build a model of 'the EGF pathway' (or whatever), we seek to avoid this omnipresent bias by collating a large collection of PPIs and investigating, via causal analysis, the pathways that emerge during simulation.

In this way, we hope that KAMI can become an authentic 'tool for discovery' that provides (semi-)automated support for the *book-keeping* aspects of curation, allowing the expert user to focus on hypothesis testing and investigating the consequences of curated knowledge in various contexts. Let us note, in particular, the potential strength of our approach—exploiting a de-contextualized representation of PPIs—to address the critical issue of reproducibility and robustness of biological knowledge [18], [19]: if a *single* body of de-contextualized knowledge can be instantiated to a number of different cell type contexts, providing pertinent explanations for their diverse behaviours, our confidence in that body of knowledge will be much greater than if we had achieved the same results with *independently*-produced models for each cell type which may derive from different, or even apparently conflicting, bodies of knowledge.

The development of KAMI continues in earnest. We are planning to extend the current Anatomizer to collect information about *homologous* proteins in order to build *orthology* relations between the action graphs representing different species and *paralogy* relations within the action graph of each species. This would provide the Aggregation Engine with a further powerful source of background knowledge about *conserved regions* of proteins, which typically share mechanisms, that would enable the systematic *transfer* of knowledge, through such homology relations, so that a single *observed* PPI could be used to generate a collection of *inferred* PPIs—that would be tagged as such but which could be 'upgraded' to observed PPIs subsequently.

On a more technical front, the current implementation of semantics-based reasoning in the Aggregation Engine uses hard-wired 'by cases' code in a number of places (see footnotes 7–9 above). A more flexible approach would be to enable the expression of domain-specific constraints—such as the 'at most one adjacent edge' properties discussed in this paper—in the semantic action graph and/or semantic nuggets; and for these to determine all required rewriting rules for the semantic auto-completion of a model. We have not yet done this as it does not directly concern KAMI so much as the underlying ReGraph library; however, it will be a necessary step for the scalability of semantics-based reasoning as we add further generic PPIs and extend the Aggregation Engine accordingly.

In the longer term, we intend to broaden KAMI's purely mechanistically-oriented representation to incorporate certain *phenomenological* aspects. These come in essentially two kinds: phenomenological *states*, such as 'activation' of an enzymatic domain or 'local counters' that would enable us to express conditions such as 'any 3 of these 5 residues must be phosphorylated'; and *actions at a distance* that express the overall effect of an entire cascade of mechanistic actions. The class of actions at a distance would serve a double purpose: firstly, as *assertions* that capture experimentally-observed dependencies, as expressed in the BEL[11] language for example, that we would like to reconstitute with an underlying mechanistic 'implementation'; and secondly, as *templates* to support the gradual refinement of phenomenological knowledge—of which there is a great deal in the bio-medical literature—into mechanistic implementations.

This interplay between [what it called] Assembly and Explanation were central to DARPA's Big Mechanism programme [20] within which the development of KAMI was initiated. Indeed, KAMI should be seen as an attempt to build a semantically-rigorous Assembly machine that feeds the (already-existing) Explanation machine provided by rule-based modelling. In order to complete this schema to a bona fide Big Mechanism would require at least two further aspects. Firstly, integration with natural language processing (NLP), to solve the human Reading bottleneck intrinsic to the vast literature on cellular signalling; and, secondly, the means to revisit Assembly in the light of Explanation, to refine phenomenological knowledge into mechanisms.

Related work

Our work bears a superficial similarity to the INDRA project developed in the Sorger Lab at Harvard Medical School [21]. However, the level of representation employed by INDRA corresponds to that of rule-based modelling: their *agents* are specific gene products, so mutants must be treated as distinct agents. As such, INDRA *statements* have none of the disjunctive flavour of our nuggets that comes from the notion of a *neighbourhood* in sequence space—and therefore fail to solve the 'update problem'.

Indeed, INDRA sets out to solve a different problem: its aim is not the de-contextualization of knowledge but the automation of model construction—notably through the use of NLP to extract knowledge from the literature. Unlike KAMI, which places priority on building a transparent and semantically rigorous curation procedure, INDRA invests in a battery of techniques—some based on background knowledge, others on heuristics—to infer conflicts and other relationships between large numbers of machine-Read INDRA statements. The outcome of this *assembly* procedure is an executable model, either ODEs or rule-based, whose provenance and built-in assumptions necessarily remain rather opaque since no *meaningful* audit trail can be provided.

Our work also relates to the BioPAX[12] project [22] which aims to represent biological pathways at multiple levels of abstraction. As such, BioPAX has a more general ambition than KAMI as it also treats metabolic pathways and interactions with DNA. Moreover, BioPAX allows the user to express pathways directly—along the lines of the discussion above concerning the perspective of higher-level mechanisms and assertions in KAMI. On the other hand, BioPAX employs the same level of representation as rule-based modelling and so, like INDRA, offers no treatment of the effects of mutations nor of the update problem.

Let us conclude by mentioning briefly the first author's MetaKappa [13], [14] which provided the means to express an *existing* Kappa model compactly through the use of a hierarchy of agents and generic rules acting upon them. Its principal defect was in forcing the modeller to *define* this hierarchy which inevitably led to cases where the generic rules failed to express the desired Kappa rules. To put it another way, MetaKappa forced the modeller to *encode*, rather than *represent*, and therefore reinforced the (old) idea of modelling as a means of codifying understanding rather than as a tool for discovery.

11. http://openbel.org

12. http://www.biopax.org

REFERENCES

[1] J. Gerhart, "1998 Warkany lecture: Signaling Pathways in Development," *Teratology*, vol. 60, no. 4, pp. 226–239, 1999.

[2] K. A. Janes *et al.*, "A systems model of signaling identifies a molecular basis set for cytokine-induced apoptosis," *Science*, vol. 310, no. 5754, pp. 1646–1653, 2005.

[3] S. Nelander *et al.*, "Models from experiments: combinatorial drug perturbations of cancer cells," *Molecular Systems Biology*, vol. 4, no. 1, p. 216, 2008.

[4] E. J. Molinelli *et al.*, "Perturbation biology: inferring signaling networks in cellular systems," *PLoS Computational Biology*, vol. 9, no. 12, p. e1003290, 2013.

[5] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine, "Rule-based modelling of cellular signalling," in *Concurrency Theory: 18th International Conference*. Springer, 2007, pp. 17–41.

[6] A. Basso-Blandin, W. Fontana, and R. Harmer, "A knowledge representation meta-model for rule-based modelling of signalling networks," *EPTCS*, vol. 204, pp. 47–59, 2016.

[7] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Hayman, J. Krivine, C. Thompson-Walsh, and G. Winskel, "Graphs, rewriting and pathway reconstruction for rule-based models," in *Foundations of Software Technology and Theoretical Computer Science*, 2012.

[8] J. Laurent, "Causal analysis of rule-based models of signaling pathways," Master's thesis, ENS Paris, France, 2015.

[9] A. Mazurkiewicz, "Introduction to trace theory," *The Book of Traces*, pp. 3–41, 1995.

[10] L. A. Harris *et al.*, "BioNetGen 2.2: advances in rule-based modeling," *Bioinformatics*, vol. 32, no. 21, pp. 3366–3368, 2016.

[11] P. Baldan, "Modelling concurrent computations: from contextual petri nets to graph grammars," Ph.D. dissertation, Department of Computer Science, University of Pisa, 2000.

[12] R. Harmer, "Rule-based meta-modelling for bio-curation," Habilitation à Diriger des Recherches, ENS Lyon, France, 2017.

[13] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine, "Rule-based modelling and model perturbation," in *Transactions on Computational Systems Biology XI*, 2009, pp. 116–137.

[14] R. Harmer, "Rule-based modelling and tunable resolution," *EPTCS*, vol. 9, pp. 65–72, 2009.

[15] X. Yang *et al.*, "Widespread expansion of protein interaction capabilities by alternative splicing," *Cell*, vol. 164, no. 4, pp. 805–817, 2016.

[16] R. Harmer, Y.-S. Le Cornec, S. Légaré, and I. Oshurko, "Bio-curation for cellular signalling: the KAMI project," in *Computational Methods in Systems Biology*, 2017, pp. 3–19.

[17] A. Corradini, T. Heindel, F. Hermann, and B. König, "Sesqui-pushout rewriting," in *International Conference on Graph Transformation*. Springer, 2006, pp. 30–45.

[18] F. Prinz, T. Schlange, and K. Asadullah, "Believe it or not: how much can we rely on published data on potential drug targets?" *Nature reviews Drug discovery*, vol. 10, no. 9, p. 712, 2011.

[19] C. G. Begley and L. M. Ellis, "Raise standards for preclinical cancer research," *Nature*, vol. 483, no. 7391, p. 531, 2012.

[20] P. R. Cohen, "Darpa's Big Mechanism program," *Physical biology*, vol. 12, no. 4, p. 045008, 2015.

[21] B. M. Gyori *et al.*, "From word models to executable models of signaling networks using automated assembly," *Molecular Systems Biology*, vol. 13, no. 11, p. 954, 2017.

[22] E. Demir *et al.*, "The BioPAX community standard for pathway data sharing," *Nature biotechnology*, vol. 28, no. 9, pp. 935–942, 2010.

[23] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe, "Algebraic approaches to graph transformation, part I: Basic concepts and double pushout approach." in *Handbook of Graph Grammars*, 1997, pp. 163–246.

[24] H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, and A. Corradini, "Algebraic approaches to graph transformation, part II: Single pushout approach and comparison with double pushout approach," in *Handbook of Graph Grammars*, 1997, pp. 247–312.

[25] R. Dyckhoff and W. Tholen, "Exponentiable morphisms, partial products and pullback complements," *Journal of Pure and Applied Algebra*, vol. 49, no. 1-2, pp. 103–116, 1987.

Russ Harmer obtained his PhD degree in Computer Science from Imperial College, London in 1999 and the *Habilitation à Diriger des Recherches* (HDR) from ENS Lyon in 2017. His research interests include formal semantics and graph-based knowledge representation and their application to the curation of the biological knowledge relevant to building executable rule-based models.

Yves-Stan Le Cornec obtained his PhD degree in Computer Science from the Université Paris-Saclay in 2016. His research interests concern formal methods for the analysis and verification of concurrent systems, biological networks and safety-critical software.

Sébastien Légaré obtained his PhD degree in biochemistry from Laval University, Quebec in 2014. His research interests include systems biology, structural biology and stochastic simulations applied to the modelling and prediction of biological systems behaviour.

Eugenia Oshurko obtained her Masters degree in Computer Science, with specialization in modelling of complex systems, from ENS Lyon in 2017 and is currently working towards her PhD degree at the same institution. Her research interests include graph rewriting, graph-based databases and knowledge representation and their application to the curation of biological knowledge relevant to building executable rule-based models.

APPENDIX
OPERATIONS ON GRAPHS

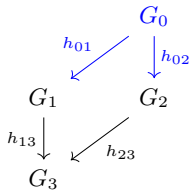
Graph rewriting makes use of three basic operations on graphs corresponding to generalized notions of intersection, union and set difference on nodes and edges [23], [24], [17].

Each of these operations takes, as its starting data, some graphs and homomorphisms and constructs new graphs and homomorphisms that satisfy a so-called *universal* property that guarantees the canonicity of the construction—just as the intersection of two sets is the *largest* set contained in both; or their union is the *smallest* containing both.

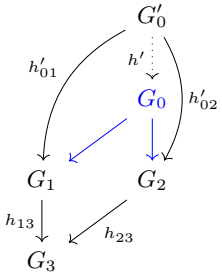
Let us first introduce some terminology. A pair of arrows with a common source is called a *span* (just like a rewriting rule) while a pair of arrows with a common target, often called the *sink* object, is called a *co-span*.

Pullback

The first operation is called *pullback*. If we are given, as starting data, a co-span $h_{13} : G_1 \to G_3$ and $h_{23} : G_2 \to G_3$, we construct its pullback by defining a span, i.e. a graph G_0 and homomorphisms $h_{01} : G_0 \to G_1$ and $h_{02} : G_0 \to G_2$, such that $h_{13} \circ h_{01} = h_{23} \circ h_{02}$, i.e.



such that, for any other choice of span h'_{01} and h'_{02} to G_1 and G_2 respectively such that $h_{13} \circ h'_{01} = h_{23} \circ h'_{02}$, there is a unique homomorphism from $h' : G'_0 \to G_0$ such that $h'_{01} = h_{01} \circ h'$ and $h'_{02} = h_{02} \circ h'$, i.e.



This definition characterizes G_0 mathematically (up to unique isomorphism) but does not provide us with a direct definition of its nodes and edges. We can easily do this: the nodes of G_0 are all pairs of nodes of G_1 and G_2 mapped to the same node in G_3 by h_{13} and h_{23}, i.e. $\{(n_1, n_2) | h_{13}(n_1) = h_{23}(n_2)\}$; we have an edge from (n_1, n_2) to (n'_1, n'_2) in G_0 if there is an edge from n_1 to n'_1 in G_1 *and* from n_2 to n'_2 in G_2. The above *unique factorization* condition means that G_0 is *as large as possible*, cf. set intersection.

We give an example in figure 10. To compute which nodes should occur in the pullback object, we look at each node of the sink object and ask whether or not it occurs in the image of *both* homomorphisms; this is only the case for the black square and the white circle (twice) so the pullback object contains one black square and two white circles. We do not show the nodes of the pullback object explicitly as pairs; this is implicit in the way we have calculated them.

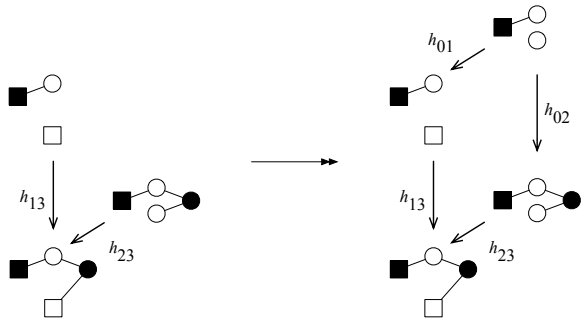
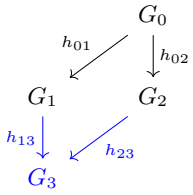


Fig. 10. An example of a pullback

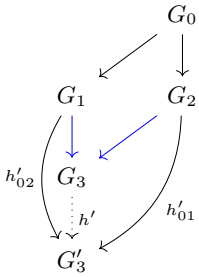
The fact that G_0, h_{01} and h_{02} always exists is a concrete property of the class of graphs under consideration. In the special case that the two given homomorphisms h_{13} and h_{23} are injective, it can be shown that h_{01} and h_{02} are also necessarily injective and, in this case, the nodes of G_0 are the intersection of those of G_1 and G_2 (and likewise for the edges).

Pushout

The second operation is dual to the first and is called *pushout*. Given a span $h_{01} : G_0 \to G_1$ and $h_{02} : G_0 \to G_2$, its pushout consists of a co-span, i.e. a graph G_3 and homomorphisms $h_{13} : G_1 \to G_3$ and $h_{23} : G_2 \to G_3$, such that $h_{13} \circ h_{01} = h_{23} \circ h_{02}$, i.e.



such that, for any other choice of co-span h'_{13} and h'_{23} from G_1 and G_2 respectively such that $h'_{13} \circ h_{01} = h'_{23} \circ h_{02}$, there is a unique homomorphism from $h' : G_3 \to G'_3$ such that $h'_{13} = h' \circ h_{13}$ and $h'_{23} = h' \circ h_{23}$, i.e.



We give a direct definition by taking the disjoint union of G_1 and G_2 and quotienting by the reflexive and transitive closure of the gluing relation: a node n_1 in G_1 is related to n_2 in G_2 iff, for some n_0 in G_0, $h_{01}(n_0) = n_1$ and $h_{02}(n_0) = n_2$. This time, the unique factorization condition means that G_3 is *as small as possible*, cf. set union.

For example, in figure 11, the pushout object contains a single black square and, due to quotienting, a single white square—and otherwise adds in everything from G_1 and G_2 that is not in the image of h_{01} and h_{02} respectively. Note also the slight difference in G_3 compared with the pullback example: there is no edge incident to the white square.

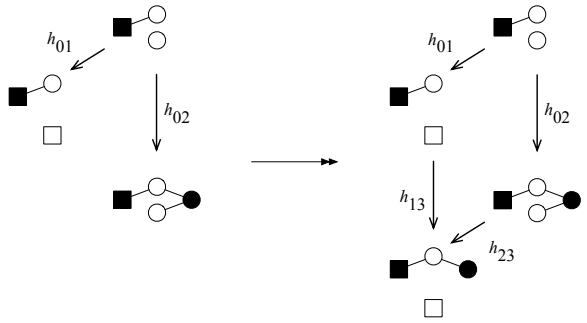


Fig. 11. An example of a pushout

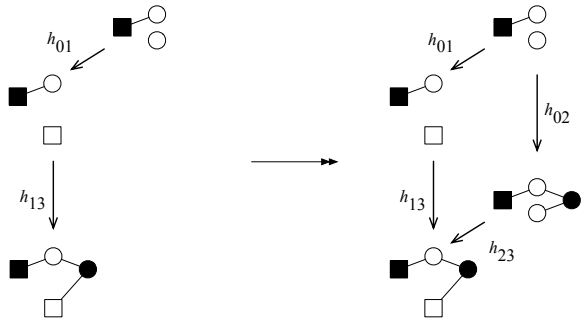
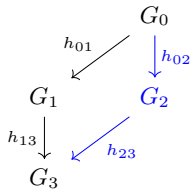


Fig. 12. An example of a pullback complement

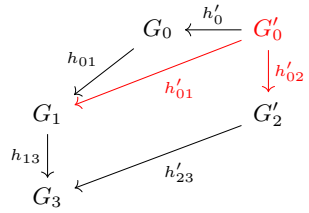
As for the pullback, the fact that G_3, h_{13}, h_{23} always exists is a concrete property of our class of graphs. In the special case where h_{01} and h_{02} are injective, it can be shown that h_{13} and h_{23} are also injective and, in this case, the nodes of G_3 are the union of those of G_1 and G_2.

Pullback complement

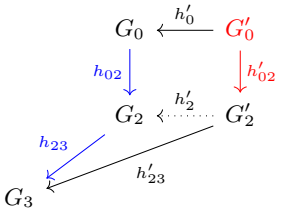
The third operation is called *pullback complement*. Given two homomorphisms $h_{01} : G_0 \to G_1$ and $h_{13} : G_1 \to G_3$, their pullback complement is a graph G_2 and homomorphisms $h_{02} : G_0 \to G_2$ and $h_{23} : G_2 \to G_3$ such that G_0, h_{01}, h_{02} is a *pullback* of h_{13} and h_{23}, i.e.



and such that any other pullback that factors through G_0, i.e. where $h'_{01} = h_{01} \circ h'_0$



also necessarily factors *uniquely* through G_2



in such a way that $h'_{23} = h_{23} \circ h'_2$ and $h'_2 \circ h'_{02} = h_{02} \circ h'_0$.

This abstract characterization may seem obscure—it has its origins in algebraic topology [25]—but, concretely, it corresponds to defining G_2 directly by starting with G_3 and modifying it by replacing the image of h_{13} in G_3 by G_0. The (rather complex) unique factorization condition means that G_2 is *as large as possible*, cf. set difference.

For example, in figure 12, the white square in G_3 is in the image of h_{13} but not of h_{01} so it disappears in G_2—otherwise, by the requirement that the square is a pullback, it would have to occur in G_0 (but it does not). Moreover, the white circle in G_3 is duplicated in G_2 because it is duplicated in G_0—but only one of the copies preserves the edge to the black square (as in G_0 whereas both acquire an edge to the black circle.

In our concrete setting, pullback complements only exist (necessarily) when h_{13} is injective. Note also that, for any given $h_{01} : G_0 \to G_1$ and $h_{13} : G_1 \rightarrowtail G_3$, there are many ways to define a graph G_2 and homomorphisms $h_{02} : G_0 \rightarrowtail G_2$ and $h_{23} : G_2 \to G_3$ such that G_0, h_{01}, h_{02} is a pullback, e.g. taking G_2 to be G_0. However, only the largest possible G_2 satisfies the factorization condition expressed in the abstract characterization—and this is the one captured by our direct definition.

Rewriting hierarchies of graphs

Let us now formalize the hierarchy of `KAMI` and discuss the process of rewriting in such a hierarchy.

We say that a graph G is *typed* by a graph T iff there is a homomorphism $h : G \to T$.

The full hierarchy of `KAMI` is shown in figure 13. There can be many nuggets $\overrightarrow{N}$ and semantic nuggets $\overrightarrow{SN}$, typed respectively by an action graph AG and the semantic action graph SAG that are themselves both typed by the meta-model MM. The dotted lines represent relations so that a nugget can be related, by a rewriting rule, to a semantic nugget and the action graph likewise to the semantic action graph. This enables us to capture the idea that a concrete mechanism, represented by a nugget, uses the generic mechanism specified by a semantic nugget. The templates $\overrightarrow{T}$ are used internally for nugget aggregation.

The typing hierarchy of `KAMI` is a tree where the meta-model is its root and the nuggets and semantic nuggets are its leaves. If we wish to rewrite one of the graphs G in this hierarchy, it is possible that this will 'break' the hierarchy. Specifically, if the rule is restrictive, we may no longer know to type all graphs previously typed by G—because we have deleted, or cloned, some of the nodes, edges and attributes of G—but we *do* still know how to type the rewritten G^-. Conversely, if the rule is expansive, we still know how to type all graphs previously typed by G but—because we have added, or merged, some nodes, edges and attributes to G—we do *not* know how to type the rewritten G^+.

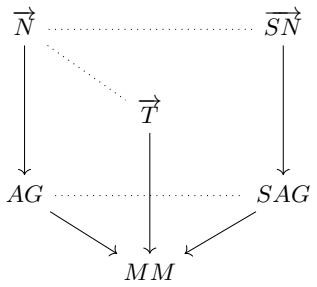
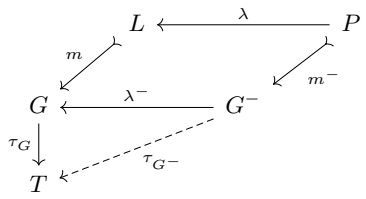


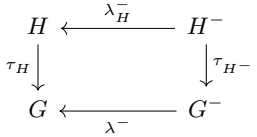
Fig. 13. The graph hierarchy of KAMI

However, it is possible to maintain all typings in the hierarchy, after rewriting G, by *propagating* restrictive effects 'upwards', i.e. in the opposite direction of the typing arrows, and expansive effects 'downwards'—provided that the rule satisfies some mild conditions. Let us consider the two cases separately for ease of exposition.

Given a *restrictive* rule $\lambda : P \to L$ and typings $\tau_P : P \to T$ and $\tau_L : L \to T$, we say that λ respects typing iff $\tau_L \circ \lambda = \tau_P$. If we additionally have a matching $m : L \rightarrowtail G$ and a typing $\tau_G : G \to T$, the rewritten G^- is indeed still typed by T by setting $\tau_{G^-} := \tau_G \circ \lambda^-$:

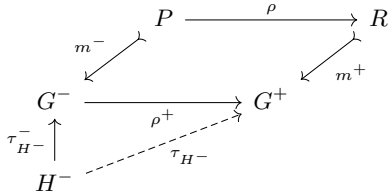


However, if we have a further typing $\tau_H : H \to G$, to maintain consistency 'above' G^- we must propagate the changes in G to H by taking the pullback:



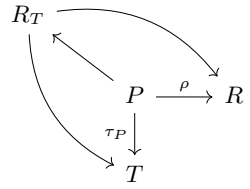
Concretely, if the rule deletes (resp. clones) a node n in G, all nodes of H typed by n are deleted (resp. cloned) in H^-. This propagates in exactly the same way to relations: deleting n in G deletes all pairs of a relation with G^- in which it occurs; and cloning n clones all those pairs. We call this operation *upward propagation*.

If we continue with an *expansive* rule $\rho : P \to R$ where we have a typing $\tau_R : R \to T$ such that $\tau_P = \tau_R \circ \rho$, G^+ is still typed by T (by the universal property of the pushout that defines G^+). Intuitively, the fact that R is typed by T means that nothing is being added to G^- that T does not already know about. Moreover, H^- is still typed by the rewritten G^+ by setting $\tau_{H^-} := \rho^+ \circ \tau^-_{H^-}$.

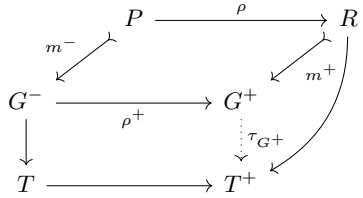


However, if R is *not* typed by T, we will have things in G^+ that we have no means of typing in T.

In this case, we require an extra piece of information that specifies those nodes, edges and attributes of R that *do* already exist in T. This is given by a relation between R and T, $R \leftarrow R_T \to T$, and a homomorphism $P \to R_T$ making the two triangles commute:



We can now take the pushout of the relation $R \leftarrow R_T \to T$ to construct $T \to T^+ \leftarrow R$ from which we obtain a typing $\tau_{G^+ \to T^+}$ of the fully rewritten G^+ (by the universal property of its defining pushout):



In concrete terms, T^+ adds any extra nodes, edges of attributes mentioned in R but absent in T and also merges nodes, edges and attributes of T when required—typically if two nodes of different type in P are merged in R. We call this operation *downward propagation*.

In practice, upward propagation only occurs in KAMI when we rewrite the current action graph with a restrictive rule—and this only generally occurs during the process of model instantiation, as discussed in section 5. On the other hand, downward propagation occurs every time we update a nugget with details that do not yet exist in the action graph. This is discussed extensively in section 4.