Bio-curation for cellular signalling
the KAMI project

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

LIP, Université de Lyon – CNRS – ENS Lyon – Université Claude Bernard Lyon 1

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`[1] bio-curation tool, based on this approach, is detailed along with our plans for future development.

1 The bio-curation problem

In multi-cellular organisms, tissue development, maintenance and repair are largely 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 which, in consequence, triggers various intra-cellular signalling 'pathways' [9].

Despite their name, these latter do not exist physically, as actual 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.

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.

* Corresponding author: `russell.harmer@ens-lyon.fr`

[1] `K`nowledge `A`ggregator & `M`odel `I`nstantiator

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.

1.1 Modelling pathways

Considerable work has been done, *e.g.* [14,19,18], to determine statistical 'models from data', highly specific to the *context* of a particular cell type. Although able to recapitulate successfully the principal highways known to operate in that context, such models (unsurprisingly) tend to have limited predictive power in other contexts. Indeed, this kind of work never intended, nor claimed, to seek such predictive power; on the contrary, it was exploiting extreme contextuality to provide deeper insight into the workings of particular cells. However, it also 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 initially advocated in [5], adopts a different stance: we step down a level, instead seeking a *de-contextualized* representation of the PPIs that underlie pathways; then provide the means to *re-instantiate* automatically that knowledge in any context in the form of an *executable* model [2]. We then attempt to reconstruct the biologist's notion of pathway either by the extraction of a (suitably post-processed) *causal trace* from a (stochastic) simulation of the model [5,4]; or by direct construction of such a causal trace through static analysis of the model [15].

This factorization of the modelling process allows us to focus attention on *bio-curation*: the construction of the de-contextualized representation of PPIs. The consequences of this knowledge in any particular cell context will be revealed by the automatic generation of an executable model and subsequent analysis. This contrasts with most modelling methodologies that require the modeller first to understand sufficiently the very system they are seeking to model; instead, we aim to enable an *exploratory* form 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, our approach poses certain constraints on what constitutes an appropriate executable model. The principal requirement is that the model provides a notion of *execution trace* based on discrete *events*, *i.e.* occurrences of PPIs, from which *causal traces* can be extracted, cf. Mazurkiewicz traces [17]. This immediately rules out ODE models. More subtly, although Mazurkiewicz's theory applies to reaction-based models—formulated either in terms of Petri nets or multi-set rewriting—the resulting causal traces contain a great deal of *spurious* causality since a single PPI is typically encoded as a family of reactions.

For example, suppose a protein B can independently bind proteins A and C to form a complex ABC via intermediates AB or BC. In the event that an A and B first react to form AB, via the reaction $A, B \to AB$, a spurious causality would be identified to the subsequent $AB, C \to ABC$ event. Indeed, the *independence* of B's bindings to A and C are expressed by the fact that the system also admits $A, BC \to ABC$ and $B, C \to BC$. If these latter reactions were removed from the system, this would imply a sequential assembly of ABC and the above causality would no longer be spurious. This mismatch between the level of representation and the desired notion of causality vastly complicates—and compromises the scalability of—the use of reaction-based models for our purposes.

This mismatch can be alleviated through the use of models based on graph rewriting, an approach known as *rule-based modelling*, exemplified by the BioNetGen[2] [13] and Kappa[3] [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). More generally, Mazurkiewicz traces can be generalized to such graph rewriting settings [1,4,12] although questions still remain as to the most appropriate notion(s) of causal trace in the context of reversible systems[4].

Kappa provides three notions of causal trace: an *uncompressed* trace that may contain many uninformative 'do-undo' event pairs; a *weakly compressed* trace that employs heuristics to eliminate such 'do-undo's; and a *strongly compressed* trace that further quotients by conflating all instances, *i.e.* individual proteins, of each agent, *i.e.* type of protein [4,15]. The latter two notions correspond closely, in many cases, to the intuitive notions of pathway employed by biologists.

[2] `http://bionetgen.org/index.php/Main_Page`
[3] `http://dev.executableknowledge.org`
[4] Ioana Cristescu, private communication

1.2 Representing PPIs

The protein-centric representation of Kappa—as opposed to the complex-centric representation of reaction-based models—fixes, at least to a good first approximation, the mismatch with the desired notion of causality. However, for the purposes of providing a de-contextualized representation of PPIs, it has some serious shortcomings. The principal difficulty comes from the fact that, although one Kappa rule corresponds to one PPI, in practice many PPIs share a single *mechanism*. If we wish to update our knowledge about such a mechanism, this necessitates identifying, and then making 'the same' change to, every Kappa rule corresponding to that mechanism. 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 [6,11].

MetaKappa provided a partial solution to this problem by enabling the definition of mechanisms as *generic* rules—that were automatically expanded into sets of underlying Kappa rules—shared by splice variants, loss-of-function mutants and even related genes. However, it was unable to treat the important case of gain-of-function mutants and, critically, the fact that mechanisms had to be *defined* in MetaKappa implicitly required the modeller to have already in mind an intended set of underlying Kappa rules. In other words, a choice of generic rules expressed only one possible way of compressing a *known*, contextualized set of Kappa rules.

Let us now state explicitly our *bio-curation problem* for signalling. We are seeking to enable the de-contextualized representation of knowledge about PPIs: specifically, the known *necessary* conditions under which a PPI may take place. Furthermore, we need to be able to express this knowledge in such a way that a single *mechanism* corresponds to a single 'element' of our knowledge representation in order to avoid the 'update problem' above. In particular, a mechanism that is potentially shared by a family of splice variants and/or mutants of a given gene should correspond to a single element.

We also need to provide the means to *deploy* this knowledge in context via the automatic determination of which mechanisms give rise to which specific PPIs: a mechanism may not apply to a particular splice variant that lacks, for example, the necessary binding site; or a mutated protein may lose, or gain, the ability to participate in a given mechanism. Finally, this contextualized knowledge should then be automatically transformed into an executable model for detailed analysis.

1.3 Plan of the paper

In §2, we present briefly our `ReGraph`[5] Python library which provides the underlying graph rewriting machinery necessary for our bio-curation tool `KAMI`[6] and discuss its use to support a de-contextualized representation of PPIs. In §3, we discuss the front-end—which performs semi-automatic update of this knowledge—and back-end of `KAMI`—which automatically instantiates this knowledge into an executable Kappa model. We conclude with a discussion of perspectives for future development of `KAMI` in §4.

Acknowledgements. 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 specially Walter Fontana for many discussions over the years related to this work. Thanks also to Pierre Boutillier, John Bachman and Ben Gyori; and to Adrien Basso-Blandin and Ismaïl Lahkim Bennani who worked on prototypes of `KAMI` and `ReGraph` respectively.

2 `KAMI`'s knowledge representation

2.1 The `ReGraph` library

In previous work [2], the first author presented a theoretical framework for graph-based knowledge representation specifically tailored to the needs of representing PPIs for the purposes of rule-based modelling. In this setting, one first defines a so-called *meta-model*, a particular graph intended to define the kinds of entities that can exist: genes, features of genes (regions, key residues, modifiable states) and actions (binding, unbinding and state modification). The meta-model is then used to *type* a second graph called the 'pre-model', but which we rename as *action graph* in this paper, which defines the specific genes, features and actions that occur in a model. By typing, we mean the existence of a homomorphism from the action graph to the meta-model [7,12]. Finally, the action graph types a collection of *nuggets* that represent the PPIs in the model. A *model* thus comprises an action graph typing a collection of nuggets.

[5] `https://github.com/Kappa-Dev/ReGraph`
[6] `https://github.com/Kappa-Dev/KAMI`

This framework supports *sesqui-push-out* graph rewriting [3,12] so it can express adding, deleting, cloning and merging of nodes and edges. An *update* of knowledge about a PPI can thus be expressed as an appropriate step of graph rewriting. An important technical point in this approach is that PPIs—themselves graph rewriting rules—are reified as *graphs*. This enables updates of PPIs to be written as ordinary graph rewriting rules even though, conceptually, they should be thought of as second-order rules that rewrite rules, cf. [16]. This is a particularity of our meta-model and clearly the generic framework could also be used in completely different domains—with or without the need to 'reduce' second-order to first-order rewriting. However, the rather *ad hoc* nature of the graphs used—simple graphs with two kinds of directed edges where nodes can have attributes—imposes unnecessary limitations on applicability of the framework.

We address this by adopting a more general theoretical framework based on simple directed graphs where nodes *and* edges have attributes that can be assigned sets of values. This still provides all the structure necessary to support sesqui-push-out rewriting but provides greater flexibility; in particular, different kinds of edges can be expressed by the use of edge attributes.

The well-known Python library `networkX`[7] provides exactly this class of graphs; as such, we chose to build our `ReGraph` library for (sesqui-push-out) graph rewriting on top of `networkX`. The `ReGraph` library also provides support for typing *hierarchies*: collections of graphs connected by (i) typing homomorphisms that form a forest or, more generally, a DAG (provided all typing paths between two graphs coincide); and (ii) binary *relations* in the form of spans of typing homomorphisms.

The notion of typing immediately extends to rewriting rules and, given a rule and a graph G typed by T, the result of rewriting G remains typed by T [12]. Conversely, if we rewrite T, we can restore typing by *propagating* the rewrite to G: if a node/edge is deleted or cloned in T, we delete or clone all nodes/edges typed by it in G [12]. This allows us to update an entire hierarchy upon rewriting of one of its constituent graphs: we propagate the rewrite to all other graphs typed—directly or transitively—by the rewritten graph and restore all typing homomorphisms. This is exploited by the back-end of `KAMI` for knowledge instantiation; see §3.2.

The notion of typing can be refined by placing *constraints* on the in- or out-degree of certain nodes: a constraint in T must be satisfied by all graphs G typed—directly or transitively—by T. This is used to express domain-specific semantic constraints in the front-end of `KAMI`; see §3.1.

[7] `https://networkx.github.io`

2.2 The meta-model

The heart of `KAMI` is an instance of `ReGraph` with a particular hierarchy, rooted in a *meta-model*, that includes—in addition to the action graph and nugget graphs—background knowledge in the form of (i) domain-specific PPI templates, *e.g.* 'phosphorylation', used to perform semantic checks or auto-completion; and (ii) definitions of gene products, *e.g.* splice variants and mutants, used to instantiate knowledge into specific contexts.

The meta-model, shown in Fig. 1, remains more or less unchanged from that originally proposed in [2]. The principal difference lies in two new nodes, defining *tests* of binding status, that were previously encoded in a rather opaque fashion; these allow nuggets to express conditions that are tested, but not modified, by the graph rewriting rules they reify. The 'source' and 'target' nodes, which played a purely formal rôle in [2], have been replaced by a single kind of *site* which should be thought of as representing a template of a physical binding site that can occur in multiple genes. As before, there are two kinds of arrows—distinguished by attributes: dotted arrows represent a *belongs to* relation, *i.e.* hierarchical structuring of actors; while solid arrows relate actions and actors.

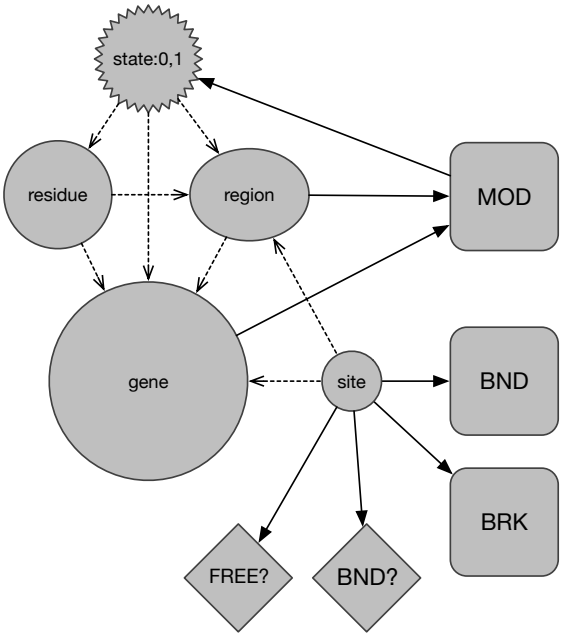


Fig. 1. The meta-model of `KAMI`

The meta-model also defines some standard meta-data as attributes:

- for *genes*, a string-valued attribute for the `UniProt`[8] accession number;
- for *residues*, an attribute `aa` with values in the set of twenty one-letter codes for amino acids;
- for the dotted arrow *from residues to genes*, a positive integer-valued attribute `pos` for its position in the sequence;
- for all actions, a positive real-valued attribute `rc` for its rate constant;
- for *MOD* actions, a $\{0, 1\}$-valued attribute `val` specifying the value written by the modification.

Note that a *state* is simply an attribute whose value can be modified by actions from *within* the system; as such, in order to be able to express such a MOD action, it must be reified explicitly as a node.

2.3 Action graphs

An instance of `KAMI`'s hierarchy contains two action graphs: one that is built up during the development of a model; and a second that frames the built-in domain-specific background knowledge. In ontological terms, where the meta-model defines *general* concepts—genes, actions, &c.—the action graphs define which entities *actually* exist: the specific genes, actions, &c. under consideration; and the entities—binding domains, PTM states, &c.—for which the system has background knowledge.

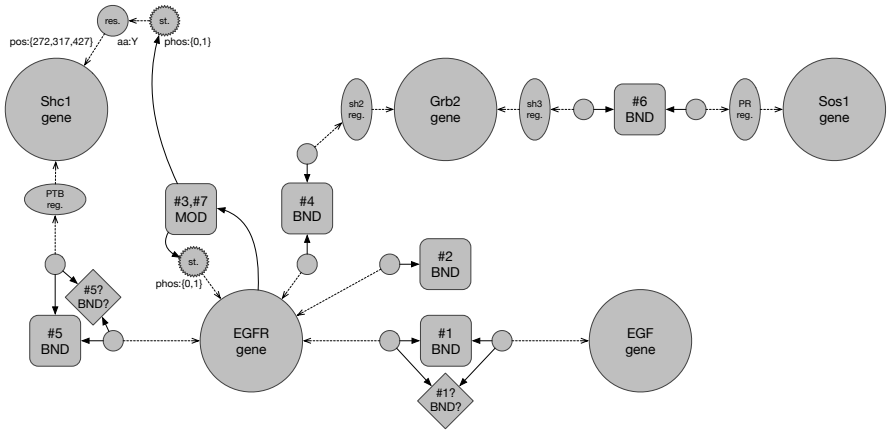


Fig. 2. An example of action graph

[8] `http://www.uniprot.org/uniprot/`

Fig. 2 shows a typical (small) example of the first kind of action graph. It defines five actual *genes*, in the sense that those five nodes are typed by the *gene* node of the meta-model, each of which defines a type—*Shc1*, *Grb2*, *EGFR*, *EGF* and *Sos1*—that can be used by nugget graphs. The other nodes also have this dual typing aspect which occurs in any graph which is neither a sink nor a source node of its hierarchy.

The current *semantic* action graph of `KAMI` is shown in Fig. 3. It defines three types of regions—*kinase* domains, *phosphatase* domains and *SH2* domains—and other associated entities that will be referenced by semantic nuggets. These four domains participate in three kinds of actions—*phosphorylation*, *dephosphorylation* and *SH2–phospho-tyrosine motif binding*.

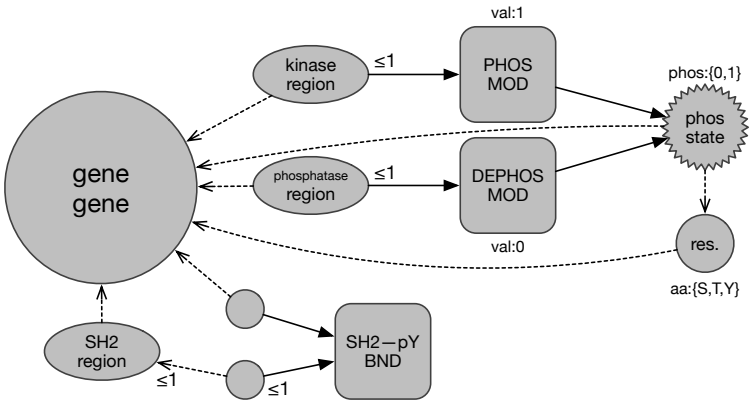


Fig. 3. The semantic action graph

The *constraints* state that (i) kinase (resp. phosphatase) domains have at most one associated phosphorylation (resp. dephosphorylation) action; and (ii) SH2 domains have at most one binding site for, and mechanism of binding to, phospho-tyrosine motifs. These statements correspond to real physical constraints but, more importantly for our purposes, also allow `KAMI` to identify whether or not an incoming input corresponds to a pre-existing action; see §3.1 for a detailed discussion.

This semantic action graph is clearly very incomplete as it stands; our approach has been to develop the ideas—and the code—in a small number of illustrative cases that should generalize broadly with little or no complication. We return to this in §4 on future work.

Fig. 4 shows the hierarchy introduced so far. The dotted line between the action graph (AG) and the semantic action graph (SAG) represents a *relation* between the two graphs which, internally, corresponds to a span from the graph • to AG and SAG: the typing from • to AG picks out those nodes of AG that have been assigned a semantic attribution in SAG; and the typing from • to SAG specifies that assignment. Note that, in order to be a valid hierarchy, the two paths from • to the meta-model (MM) must commute.

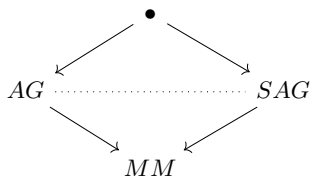


Fig. 4. The (partial) hierarchy of `KAMI`

In our example, the node *#3* of the AG is assigned to the *PHOS* node of the SAG and the (unique) state of *EGFR* is assigned to the *phos* state of the SAG; *EGFR* is also assigned to the *gene* node of the SAG. This means that node *#3 is* a phosphorylation and any domain-specific constraints—expressed in the SAG—of phosphorylation therefore apply. Additionally, node *#4* of the AG is assigned to the *SH2–pY* node of the SAG and the region *sh2* of *Grb2* is assigned to the *SH2* node of the SAG; *Grb2* is also assigned to the *gene* node of the SAG. This means that node *sh2 is* an SH2 domain and node *#4 is* an SH2 domain–phospho-tyrosine binding.

2.4 Nuggets

An instance of `KAMI`'s hierarchy may contain many nuggets, representing specific (families of) PPIs, typed by the action graph. It also contains a built-in—but modifiable—collection of *semantic* nuggets, typed by the semantic action graph, that provide *templates* for certain generic PPIs such as domain-domain or domain-motif bindings. These enable us to perform *semantic checks* that can reject *non-sense* nuggets.

Fig. 5 shows an example of a nugget typed by the action graph of Fig. 2. Note how the nugget specifies all and only the (known) context—in this case, the test that a state of *EGFR* called *phos* has value 1 and that *Grb2* has a region *sh2*—necessary for this PPI to occur.

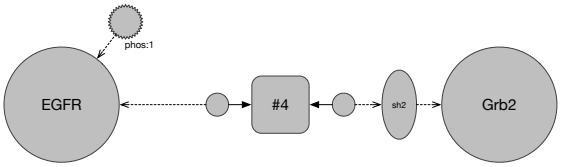


Fig. 5. An example of nugget

A nugget N *matches* a semantic nugget SN iff there is a span of injective homomorphisms $N \hookleftarrow \bullet \rightarrowtail SN$. A matching is *complete* iff the right leg $\bullet \rightarrowtail SN$ of the span is an isomorphism, *i.e.* there is an injective homomorphism $SN \rightarrowtail N$. For example, the nugget in Fig. 5 matches the semantic nugget in Fig. 6—which defines a template for SH2 domain–phospho-tyrosine binding—via the evident complete matching.

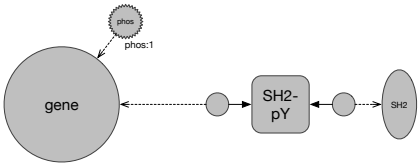


Fig. 6. An example of semantic nugget

A given semantic action may have several associated semantic nuggets, *e.g.* Fig. 7 shows a more *refined* semantic nugget for SH2 domain–phospho-tyrosine binding. These two semantic nuggets are related by a span which also serves as a rewriting rule that can be applied to a nugget—provided (i) there is a complete matching to the LHS semantic nugget; and (ii) we supply a typing of the RHS into the action graph. This allows us to *upgrade* nuggets systematically once we have all extra needed details.

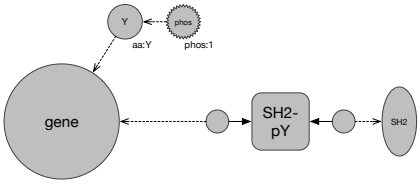


Fig. 7. Another example of semantic nugget

2.5 Protein definitions

We represent *gene products*, *i.e.* proteins, as rewriting rules typed by the meta-model whose LHSs are injectively typed by the action graph, cf. complete matchings. A LHS comprises one gene and all features belonging to it; the RHS can have multiple gene products, each of which must resolve all disjunctive aspects of those features: a residue that has several admissible values of its `aa` or `pos` attributes—due to mutations or different sequence numbering due to splice variants or truncations—must here be assigned *exactly one* for each. Moreover, each feature may be removed, *e.g.* a region of a gene may not occur in some splice variants.

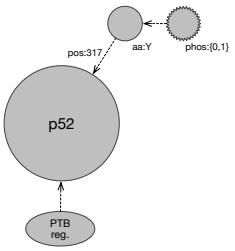


Fig. 8. Definition of a gene product

The gene *Shc1* has a residue with three admissible values for `pos`. We represent the *p52* splice variant, where `pos`=317, of *Shc1* as in Fig. 8. We use these rewriting rules in §3.2 in the back-end of `KAMI` that generates Kappa models. The full current hierarchy of `KAMI` is shown in Fig. 9.

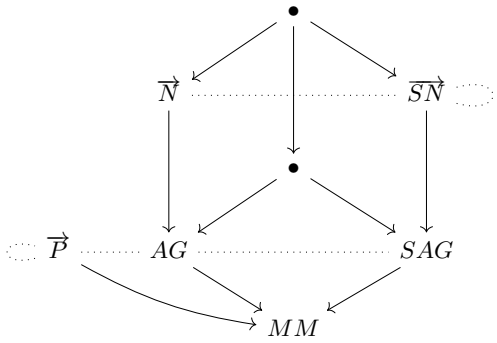


Fig. 9. The full 'hierarchy type' of `KAMI` including (semantic) nuggets and proteins

3 The KAMI bio-curation tool

In the previous section, we have seen how the generic framework of graph hierarchies, as provided by ReGraph, can be exploited to build a knowledge representation (KR) suitable for PPIs. Importantly, an update of the KR is defined by a step of graph rewriting defined in the terms of the KR's 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.

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.

In this section, we describe the current front- and back-end to the KAMI bio-curation tool: the front-end takes input—either directly from the user via a GUI or through INDRA[9] statements[10]—and constructs, then applies, the appropriate step of graph rewriting. As we will explain, the system can exploit domain-specific background knowledge—in the form of semantic nuggets—to identify whether or not the input speaks of an interaction that already exists in the KR. We also very briefly describe the back-end of KAMI which takes a collection of protein definitions and calculates the instantiation of nuggets to that collection of gene products, *i.e.* the *contextualization* of our representation to the 'cell type' defined by the given collection of proteins.

3.1 Knowledge input and aggregation

Given an (INDRA) input such as 'EGFR phosphorylates Shc1 on Y317' or 'Grb2's SH2 domain binds Shc1 phosphorylated on Y317', we need to compute the rewriting rule(s) required to insert this knowledge into KAMI's hierarchy. This problem is an instance of the standard problem in semantics—given an input, calculate its denotation—with a slight twist: the computed rules depend on the *current state* of the hierarchy. Indeed, given such an incoming input, depending on the current state, we may need to perform a significant update or there may be nothing to do at all as the input is subsumed by what the KR already contains.

[9] https://github.com/sorgerlab/indra

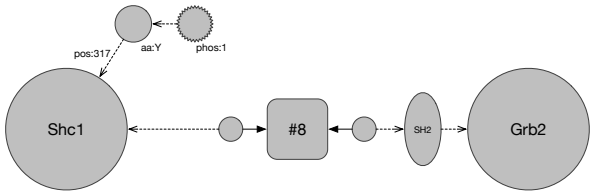
[10] http://indra.readthedocs.io/en/latest/modules/statements.html

The key task in computing update rules concerns identifying whether, or not, (i) each entity mentioned in the input already exists in the KR; and (ii) the (inter)action in question already exists in the KR. The first question can be resolved fairly easily using *grounding*: several standard names/IDs exist for genes (UniProt, HGNC, &c.) and regions/domains (PFAM, InterPro, &c.). The current version of `KAMI` takes inputs in the form of `INDRA` statements[11] which include such grounding information—at least for genes—as meta-data; however, it should be a straightforward task to obtain grounding in cases where `INDRA` does not provide it or, in the future, where we intend to use less pre-processed input formats.

`KAMI` contains a module, called the *gene anatomizer*, which takes a UniProt ID (or similar) and interrogates various databases (principally InterPro) to construct a representation of the gene and all its (significant) regions, including grounding information. By including all regions, not just those mentioned in an input, we often enable stronger inference during the construction of a rewriting rule: knowing that Grb2 has only one SH2 domain means that it *must* be the one referred to in the above input. Moreover, the anatomizer need only be run once on any given gene; the results are maintained in the action graph and can be reused freely.

The second *identification* problem, for interactions, has sharper teeth: to the best of our knowledge, no system of grounding for PPIs exists to date[12]. This problem cannot be solved automatically in general: even if an input speaks of 'A binds B' and we already have a binding action between A and B, we *cannot* immediately infer that they refer to the same action as A and B may be able to bind in multiple ways. However, we can exploit background knowledge in some cases to establish that an input speaks of an existing interaction.

For example, given an input of the form 'Grb2's SH2 domain binds Shc1 phosphorylated on Y317', `KAMI` would first construct a proto-nugget:



[11] We chose to use `INDRA` for now as it also provides us with import from BioPAX [8] and a number of NLP systems. However, there is no obstacle to providing direct import to `KAMI` from such sources; indeed, doing so would avoid losing certain kinds of information that are not represented in the current version of INDRA, *e.g.* regions.

[12] A notable side-effect of the `KAMI` project will be precisely to provide such a grounding.

It would then use grounding meta-data to resolve *Shc1*, its residue Y317, the *phos* state of Y317, *Grb2* and its SH2 domain to existing nodes in the action graph. What about the remaining nodes—the two binding sites and the action? Given that the proto-nugget matches the semantic nugget of Fig. 7, its action is identified as an *SH2-pY* binding. The constraints imposed by the semantic action graph now require that the binding site of the SH2 domain and the *SH2-pY* action[13] be identified with those in the action graph already, giving rise to the following updated action graph.

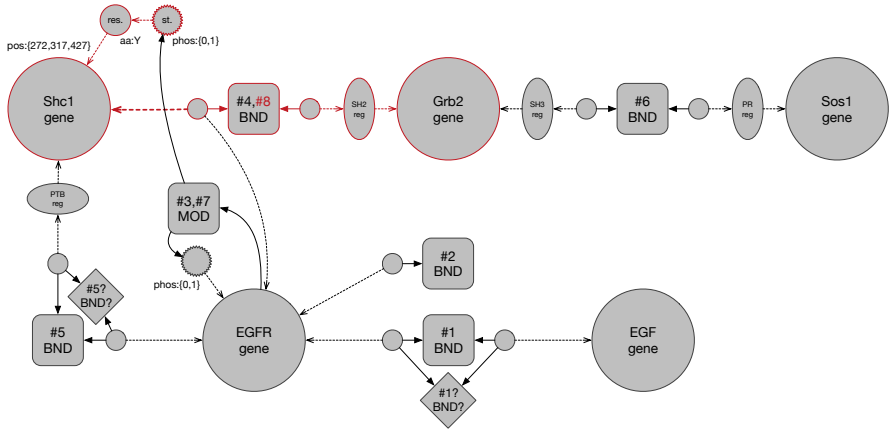
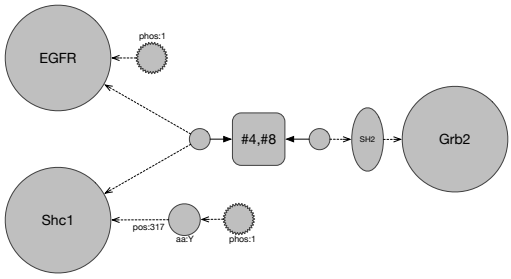


Fig. 10. The updated action graph

Moreover, the two nuggets for *Grb2*'s SH2 domain will also be merged, giving rise to a *disjunctive* nugget expressing '*Grb2*'s SH2 domain binds phosphorylated *EGFR* or *Shc1* phosphorylated on Y317'.



[13] This also implies that the second binding site must be identified with that belonging to *EGFR* as binding actions have at most two binding sites—a constraint, elided until now, enforced by the meta-model—*i.e.* this site is a template with an instance in *EGFR* and another in *Shc1*.

The ability to express such disjunctive statements means that a nugget corresponds to a shared *mechanism*—a family of PPIs—so any update, concerning *Grb2*, of a family member of this nugget—for example, that some mutation in the SH2 domain abrogates binding to *EGFR*—would apply at the level of the *mechanism*: *Grb2* binds *Shc1* in the same way as it binds *EGFR*; *therefore* the mutation also abrogates *Grb2*'s binding to *Shc1*. This solution of the update problem, discussed in the introduction, is a special case of what biologists call *by similarity* inference; but it occurs in `KAMI` not through logical 'inference' but by the merging of nodes.

3.2 Model instantiation and output

The back-end of `KAMI` performs two tasks. Firstly, given a collection of nuggets and their action graph, and given a collection of rewriting rules defining gene products, it applies those rewriting rules to the action graph[14]. This rewriting step is then propagated to all nuggets from which we can easily determine, for each gene product and each nugget, whether or not the nugget still applies. For example, a nugget testing for a certain value of an `aa` attribute of a residue would not apply to an instance of that gene that assigns a different value to that attribute. We detect this because the original nugget no longer matches the transformed one.

This effects a transformation from a gene- and mechanism-based level of representation to a protein- and PPI-based level: it *contextualizes* the knowledge with respect to the given collection of gene products. The second step now amounts to a standard parsing task: the contextualized knowledge is translated into Kappa. Each gene product defines a distinct agent type and the rules are read off by 'multiplying out' disjunctions, *e.g.* 'A_1 or A_2 binds B_1 or B_2' gives rise to four distinct rules.

4 Current and future work

We have presented an overview of the aims and 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 documenting the curation process. The tool is based on solid theoretical foundations, discussed to some extent in [2,12], that will be further developed in the long version of the present paper.

[14] Unlike normal updates, this does not rewrite the action graph in-place; instead, it copies the relevant part of the action graph and rewrites *that* in-place.

The development of KAMI continues in earnest. The most immediate goals concern providing additional background knowledge, principally for the binding domains—PTB, SH3, WW, PDZ, &c.—and other enzymatic domains commonly implicated in signalling. This additional knowledge will already substantially increase the ability of the front-end to aggregate effectively through the merging of nodes. However, a further powerful source of background knowledge concerns closely related genes or, better, *conserved regions* of genes that typically share mechanisms. This could be captured by the merging of *region* nodes; in this way, we would extend the power of the system to identify automatically potential merging to a far wider class of (binding) actions.

In the longer term, we intend to broaden KAMI's current, very much mechanistically-oriented representation to incorporate *phenomenological* aspects. These will come in essentially two kinds: phenomenological *states*, such as 'activation' of an enzymatic domain; and *actions* that typically express the overall effect of an entire cascade of mechanistic actions. In a way somewhat analogous to the refinement of semantic templates outlined above, the tool must be able to support the gradual refinement of phenomenological knowledge about signalling—of which there is a great deal in the bio-medical literature—into its mechanistic 'implementation'.

In this way, we hope that KAMI can become an authentic 'tool for discovery' that provides 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.

Related work. Our work bears a superficial similarity to the INDRA project developed in the Sorger Lab at Harvard Medical School [10]. 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; and *statements* have none of the disjunctive flavour of nuggets—and therefore fails to solve the 'update problem'.

Indeed, INDRA sets out to solve a different problem: its aim is not the decontextualization of knowledge but the (semi-)automation of model construction. In line with this, INDRA does not seek a transparent and semantically rigorous curation procedure; instead it invests in a battery of techniques—some based on background knowledge, others on heuristics—to infer conflicts and other relationships between INDRA statements. The outcome of this *assembly* procedure is an executable model, either ODEs or rule-based, but whose provenance and built-in assumptions remain rather opaque since no meaningful audit trail can be provided.

References

1. Paolo Baldan. *Modelling concurrent computations: from contextual Petri nets to graph grammars.* PhD thesis, Department of Computer Science, University of Pisa, 2000.
2. Adrien Basso-Blandin, Walter Fontana, and Russ Harmer. A knowledge representation meta-model for rule-based modelling of signalling networks. *EPTCS*, 204:47–59, 2016.
3. Andrea Corradini, Tobias Heindel, Frank Hermann, and Barbara König. Sesquipushout rewriting. In *International Conference on Graph Transformation*, pages 30–45. Springer, 2006.
4. Vincent Danos, Jérôme Feret, Walter Fontana, Russ Harmer, Jonathan Hayman, Jean Krivine, Chris Thompson-Walsh, and Glynn Winskel. Graphs, rewriting and pathway reconstruction for rule-based models. In *Foundations of Software Technology and Theoretical Computer Science*, 2012.
5. Vincent Danos, Jérôme Feret, Walter Fontana, Russ Harmer, and Jean Krivine. Rule-based modelling of cellular signalling. In *CONCUR 2007 – Concurrency Theory: 18th International Conference*, pages 17–41. Springer, 2007.
6. Vincent Danos, Jérôme Feret, Walter Fontana, Russ Harmer, and Jean Krivine. Rule-based modelling and model perturbation. In *Transactions on Computational Systems Biology XI*, pages 116–137. Springer Berlin Heidelberg, 2009.
7. Vincent Danos, Russ Harmer, and Glynn Winskel. Constraining rule-based dynamics with types. *MSCS*, 23(2):272–289, 2013.
8. Emek Demir et al. The BioPAX community standard for pathway data sharing. *Nature biotechnology*, 28(9):935–942, 2010.
9. John Gerhart. 1998 Warkany lecture: Signaling Pathways in Development. *Teratology*, 60(4):226–239, 1999.
10. Benjamin M. Gyori, John A. Bachman, et al. From word models to executable models of signaling networks using automated assembly. *bioRxiv*, 2017.
11. Russ Harmer. Rule-based modelling and tunable resolution. *EPTCS*, 9:65–72, 2009.
12. Russ Harmer. *Rule-based meta-modelling for bio-curation.* Habilitation à Diriger des Recherches, ENS Lyon, France, 2017.
13. Leonard A. Harris et al. BioNetGen 2.2: advances in rule-based modeling. *Bioinformatics*, 32(21):3366–3368, 2016.
14. Kevin A. Janes et al. A systems model of signaling identifies a molecular basis set for cytokine-induced apoptosis. *Science*, 310(5754):1646–1653, 2005.
15. Jonathan Laurent. Causal analysis of rule-based models of signaling pathways. Master's thesis, École Normale Supérieure, Paris, France, 2015.
16. Rodrigo Machado, Leila Ribeiro, and Reiko Heckel. Rule-based transformation of graph rewriting rules: towards higher-order graph grammars. *Theoretical Computer Science*, 594:1–23, 2015.
17. Antoni Mazurkiewicz. Introduction to trace theory. *The Book of Traces*, pages 3–41, 1995.
18. Evan J. Molinelli et al. Perturbation biology: inferring signaling networks in cellular systems. *PLoS Computational Biology*, 9(12):e1003290, 2013.
19. Sven Nelander et al. Models from experiments: combinatorial drug perturbations of cancer cells. *Molecular Systems Biology*, 4(1):216, 2008.