Proceedings of the
Second International Workshop on
Bidirectional Transformations
(BX 2013)

A Bidirectional Collaboration Framework for Bio-Model Development

John Roger Wilson-Kanamori, Soichiro Hidaka

18 pages

A Bidirectional Collaboration Framework for Bio-Model Development

John Roger Wilson-Kanamori[1]**, Soichiro Hidaka** [2]

[1] j.r.wilson-kanamori@sms.ed.ac.uk
School of Informatics
University of Edinburgh
United Kingdom

[2] hidaka@nii.ac.jp
National Institute of Informatics
Japan

Abstract: High-level graph data structures have gained favour in representing biological knowledge in a computationally executable form, but the information contained therein must remain accessible to all users no matter their background. Bidirectional graph transformations may be used to synchronise and maintain the consistency of these graph data structures as they evolve through the process of creating and refining a bio-model knowledge base. We outline a bidirectional collaboration framework by which users with vastly differing backgrounds may contribute to the development and evolution of such a knowledge base, and examine a simple example to illustrate its merits. We also identify avenues for further research necessary to refine the framework. No prior biological knowledge is assumed.

Keywords: Bidirectional graph transformations, collaborative development, biological modelling.

1 Introduction

Bidirectional graph transformations, whether based on a functional approach [HHI+10], a Triple Graph Grammar (TGG) approach [SK08], or otherwise, are mechanisms for synchronising and maintaining the consistency of two or more related graph data structures. They hold remarkable promise for applications in such fields as model-driven software development. The functional approach has been applied to resolving the mismatch problem between user and template code evolving throughout the development lifecycle, by providing a means of traceability between the two [YLH+12]. The TGG approach has been used to integate software development tools [AKK+08] amongst other applications.

An alternative domain in which bidirectional graph transformations may play a major role in the future is the computational modelling of biological and biochemical interactions. Standardised modelling languages representing such interactions in an executable form have greatly contributed to the establishment of fields such as systems and synthetic biology in modern research. Recent approaches [BS11] define such models in terms of 'high-level' graph data structures such as port graphs [AK07] and site graphs [DL04], rendered stochastically simulable via

suitably adapted graph rewriting algorithms. The term 'high-level' is used here in the sense that there exists a clearly defined user-friendly hierarchy to the entities involved in the graph, and this is reflected in both static and dynamic aspects of the model.

Such paradigms facilitate the communication of ideas between modellers and practical biologists, and the simulation and analysis of executable bodies of knowledge with formally encoded assumptions, whilst abstracting from the combinatorial complexity involved. However, sharing this information between such fundamentally different approaches - experimental and computational - is a difficult task due to the disparity in backgrounds and methodologies. Furthermore no concrete framework as yet exists to aid collaboration in the creation and further development of biological models, unlike for example the integrated environments provided for domain experts and software engineers in model-driven software development. Bidirectional graph transformations may provide at least a partial solution to these problems.

We thus ask: how may we apply bidirectional graph transformations to the aforementioned biological modelling scenario to create a collaborative framework, what are the possible merits of doing so, and what future work is necessary before such an approach might become mainstream? Ultimately we are successful in applying a subset of bidirectional graph transformations to the problem, illustrated by a simple example, and in identifying several avenues for further refinement.

We approach the scenario by:

1. Defining a model data structure for the knowledge base shared by domain experts including experimental biologists and computational modellers,

2. Defining bijective conversions between the above data structure and 'low-level' graph representations - the latter amenable to bidirectional transformations via the existing tool GRoundTram [HHI⁺11],

3. Defining bijective translations between the knowledge base and user-friendly domain specific representations,

4. Allowing users to query the knowledge base from these domain specific representations to produce a refined view suitable for manipulation, and

5. Transforming from the queries defined in (4) to UnQL+ (an extension to UnQL [BFS00] with graph editing constructs) queries in GRoundTram so that the domain expert queries may be bidirectionalised via low-level graphs and UnQL+ queries upon them.

We establish bijective conversions and translations between the different data structures involved in the framework, thus enabling their bidirectional transformation via GRoundTram. Figure 1 displays a graphical overview of the framework, in which the region surrounded by the dotted line is revisited as Figure 3a and further explained in Section 3.

In this paper we formally define (1) and sketch an overview of (4), assuming implementation of (2), (3), and (5). We have designed informal algorithms for all, but only the first (converting between high-level and low-level representations) is implemented.

This approach may be compared with another data synchronisation strategy that uses bidirectional transformations, namely, the scenario outlined in [FGM⁺07]. As one possible usage
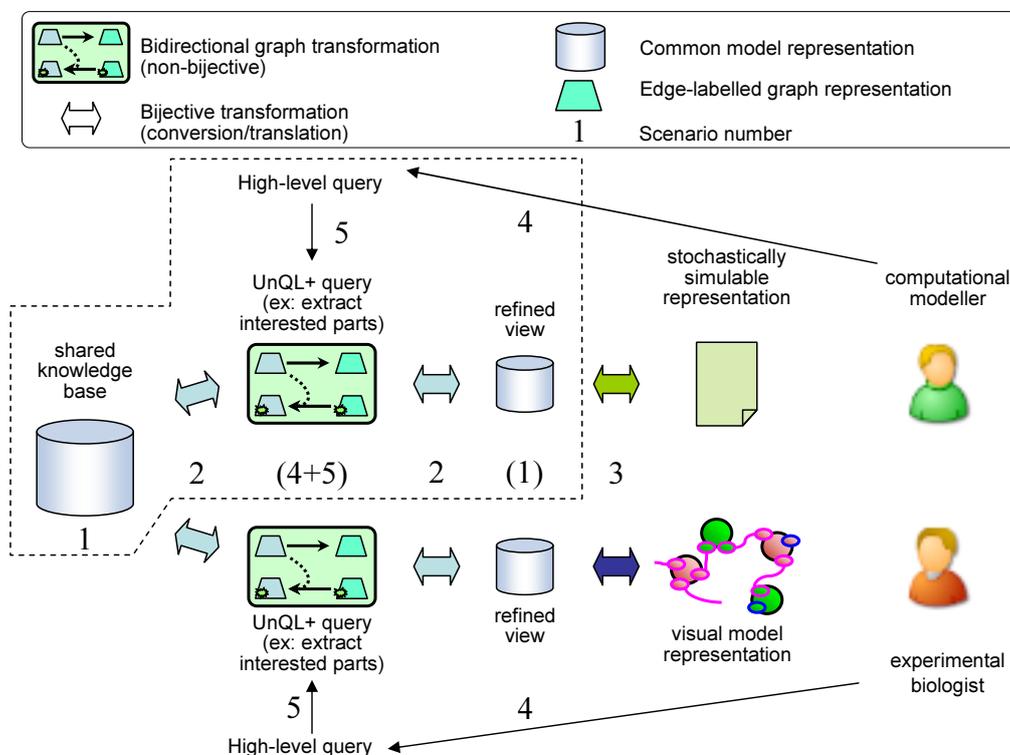
Figure 1: An overview of the proposed collaboration framework.

of lenses, this scenario utilises a common abstract view created by multiple lenses over multiple concrete data storages. Each concrete data storage can maintain, in addition to data that is synchronised with other concrete data storages, its own data that does not participate in the synchronisation.

In contrast to this we adopt domain specific abstract views over a common concrete high-level data structure, converting bijectively from this high-level data structure to a simpler format that enables bidirectional transformations. We also provide bijective mapping between domain specific representations and the common high-level data structure, thus allowing multiple user groups and representations to utilise the bidirectional transformations in a manner best suited to their needs. We believe that this approach is applicable to any scenario in which collaboration is paramount, rather than particular to biological modelling.

We first proceed in Section 2 by further elucidating the problem outlined above. We provide an overview of the collaboration framework in Section 3, followed by a usage scenario and implementation details regarding update operations on the framework (Section 4). Sample update operations are made available online as a benchmark. We conclude in Section 5 by summarising the future directions envisaged. No prior biological or biochemical knowledge is presumed.

2 The Problem

Let us assume a collaborative biological modelling scenario in which user groups with differing backgrounds (for example experimental biologists and computational modellers) work together to develop a consensus model of a typical biochemical network. In this case, model development involves not only the initial creation of a working model, but also verification and validation of the encoded knowledge via biological experiments and computational techniques such as parameter refinement. This requires all sets of users to repeatedly revisit their work, incrementally revising at every step.

Modelling in a biological context has two major features, the first of which is the confluence of multiple disciplines with vastly different backgrounds. If we conceptualise a high-level model data structure (MDS) representing a 'knowledge base' shared amongst all users involved in the collaborative effort, then broadly speaking each user group will prefer a domain specific 'user view' or representation upon the underlying information. Formally, this would involve well-defined algorithms maintaining a one-to-one correspondence between the knowledge base and the different domain specific representations. In our scenario, one representation might be a graphical model visualisation for experimental biologists, allowing definition of constraints, conditions, and interactions in an intuitive interface whilst abstracting from the underlying mechanics and implementation. Alternatively, we might consider code for computational modellers, amenable to simulation and analysis but not necessarily intuitively usable by other users. Similar but distinct representations might even exist within the same user group; for example, different versions of model code (that offer different but individually useful capabilities) or instances of the model with differing simulation parameters.

The presence of multiple users and multiple representations of the same underlying graph-like model data structure, and the desirability of modifying the source knowledge base via any of the above, raises questions regarding the integrity of said knowledge base. This is doubly so once we start allowing the creation of abstract views upon these representations via the use of graph querying, allowing the user to isolate only the components of the model that are of their interest. Bidirectional transformations may thus be employed to ensure that any modifications reflected back to the source knowledge base are done so correctly regardless of the user's choice of representation and view.

Another consideration arising due to the collaborative nature of biological modelling is the issue of conflicting modifications upon the knowledge base, requiring domain expert knowledge to identify and resolve. Although in future this may result in the integration of a version control system into the framework, for now we assume that the knowledge base is 'locked' during each edit.

The second peculiarity of biological models lies in the high degree of combinatorial complexity involved, which make many practical models nonamenable to traditional mathematical techniques. For example, a single protein may have any number of different configurations in terms of bindings with other proteins and internal state (i.e. phosphorylation or methylation), each of which would have to be tracked separately in a system of ordinary differential equations. This is addressed by representing the knowledge involved using recently developed modelling paradigms such as the rule-based modelling language Kappa introduced in Subsection 2.1, which allow an intuitive but appropriately abstract representation of biological interactions (the mod-
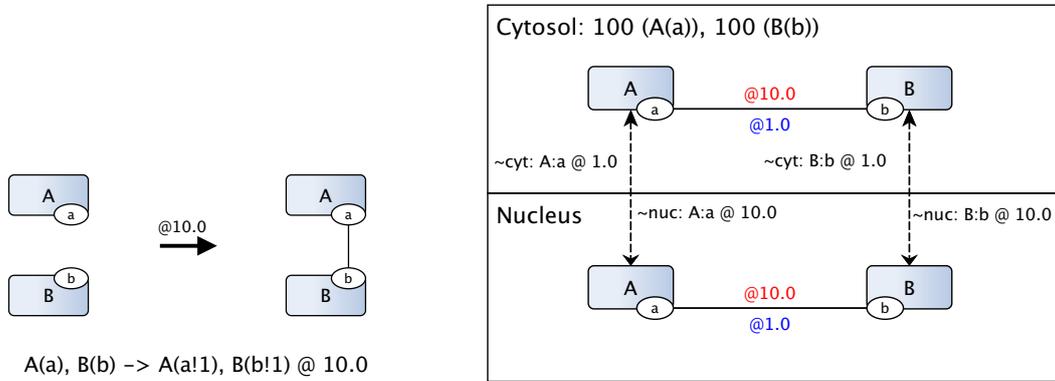
(a) A typical Kappa binding rule, displayed in both graphical and textual form.

(b) The visualisation of a simple Kappa binding and transport model.

Figure 2: Illustrations of the Kappa stochastic rule-based modelling language.

eller need be concerned only with the interesting part of the system under study). Knowledge bases created in these paradigms may often be represented in some form of high-level graph formalisation, for example the Systems Biology Graphical Notation (SBGN) [LHM+09].

Intuitively, all operations upon these knowledge bases - for example the refinement of model parameters or the addition of components to the network - should broadly correspond to well-known database query and manipulation operations: SELECT, (in-place) UPDATE, INSERT, and DELETE. Unfortunately, while high-level graph formalisations are suitable for displaying complex information in a natural manner, they are not amenable to the query-based approaches that make their low-level counterparts suitable as database representations and as subjects to bidirectional transformations. To resolve this, we propose to distil high-level graph data structures to an equivalent edge-labelled digraph representation via a bijective transformation.

The process of bridging the gap between high-level and low-level graph data structures is non-trivial, but is greatly simplified by utilising rooted directed edge-labelled graphs and the existing bidirectional graph transformation framework GRoundTram. This allows a straightforward representation of hierarchical high-level entities, their attributes, and the labelled relationships between them. We are then able to adopt graph querying languages such as UnQL+ to manipulate and query the digraph data structure (DDS).

The remainder of this section contains a brief introduction to the modelling language adopted in this paper for the model data structure, and its representations as model code (Subsection 2.1) and the visualisation thereof (Subsection 2.2). Readers familiar with the Kappa stochastic rule-based modelling language, or unconcerned with the details of the user-specific representations, are encouraged to skip ahead to the framework overview in Section 3.

2.1 Kappa Model Syntax

In this paper we focus on the stochastic rule-based modelling language Kappa [DL04], a formal site graph rewriting language that represents proteins and other biological entities as agents. Agents have sets of sites that can be used to hold internal state or to bind and interact with other agents to create a vast interconnected system of hierarchical entities. Agent-agent interactions (Figure 2a) are defined by rules that specify the molecular context required for an interaction, along with a rate constant that controls its activity. Since these rules abstract away from all but the important aspects of the interaction, they alleviate the inherent combinatorial complexity of non-trivial biological models.

A Kappa model of a biological system is a collection of compartments, agents, rules and their associated rate constants, and an initial population of agents on which these rules act. Example 1 serves to illustrate such a model in its textual code form.

Example 1 *Example Kappa model code for a simple binding and transport model:*

```
### Agents:
%agent: A(a,loc˜nuc˜cyt)
%agent: B(b,loc˜nuc˜cyt)

### Rules:
# A-B interaction
'Binding cyt' A(a,loc˜cyt), B(b,loc˜cyt)
                    -> A(a!1,loc˜cyt), B(b!1,loc˜cyt) @ 10.0
'Unbinding cyt' A(a!1,loc˜cyt), B(b!1,loc˜cyt)
                    -> A(a,loc˜cyt), B(b,loc˜cyt)     @ 1.0
'Binding nuc' A(a,loc˜nuc), B(b,loc˜nuc)
                    -> A(a!1,loc˜nuc), B(b!1,loc˜nuc) @ 10.0
'Unbinding nuc' A(a!1,loc˜nuc), B(b!1,loc˜nuc)
                    -> A(a,loc˜nuc), B(b,loc˜nuc)     @ 1.0
# Transport
'Unbound A nuc transport' A(a,loc˜cyt) -> A(a,loc˜nuc) @ 10.0
'Unbound B nuc transport' B(b,loc˜cyt) -> B(b,loc˜nuc) @ 10.0
'Unbound A cyt transport' A(a,loc˜nuc) -> A(a,loc˜cyt) @ 1.0
'Unbound B cyt transport' B(b,loc˜nuc) -> B(b,loc˜cyt) @ 1.0

### Initial Conditions:
%init: 100 (A(a,loc˜cyt))
%init: 100 (B(b,loc˜cyt))
```

The model begins by describing two agents (abstractly named *A* and *B* for sake of simplicity), each with a single binding site (*a* and *b* respectively) and located in either of two cellular compartments (the nucleus and the cytosol, i.e. the inner and outer regions of a complex cell). Note that in the sample code, the compartments are represented abstractly as states of the 'loc' site.

An alternative approach would be to utilise the Spatial Kappa language extension [Ste] to define the compartments explicitly. The code thus generated may be treated as an alternate representation of the underlying model data structure. In other words it provides another representation of the knowledge base, albeit requiring an extra layer of processing to take advantage of the toolset used in the creation and simulation of Example 1.

The model then specifies two sets of rules that the agents must obey: they may bind to and unbind from one another when in the same compartment, and may transfer between the compartments (represented by modifying the state of the 'loc' site) whilst not bound. The rates of the rules are weighted such that the agents aggregate in the nucleus in a bound form. Finally, the initial conditions of the model are given: a hundred each of *A(a)* and *B(b)*, unbound and located in the cytosol.

This code may then be executed in a Kappa simulation based on a suitably generalised Gillespie algorithm, inducing stochastic dynamics on the initial populations. These stochastic trajectories are obtained by generating a continuous time Markov chain: at any given time a rule may apply to a given state in a number of ways, which is multiplied by the rate of the rule to define the rule's activity in that state of the system, which in turn determines the likelihood that this rule will fire next. Various forms of static analysis, such as causal story tracking, may also be performed upon the model.

Kappa has been used in practice to create biological models ranging from simple MAPK cascades [DFF$^+$07] to elaborate dynamic repair schemes [DFF$^+$09] and synthetic constructs [SW11] [Moo12].

2.2 Model Visualisations

Graphical representations, most prominently SBGN, have developed as a medium to represent mathematically and computationally encoded models in a user-friendly format. They foster the efficient storage, exchange, and reuse of biological information due to the simplicity of their syntax and semantics and the unambiguity with which they can represent networks of interactions.

SBGN is not tied to a particular form of modelling, but conversely this means that its various forms often fail to express models in a succinct but ambiguous manner. The Process Description language suffers from information bloat and is unsuitable for the large (often infinite) reaction networks present in rule-based models. The Activity Flow language in turn is too simplistic to unambiguously specify the knowledge bases we are interested in. The Entity Relationship language most closely corresponds to our requirements, in that we may describe Kappa agents as entities and the rules between them as relationships, but even so doesn't fully capture the information contained in a Kappa model.

More recently, efforts have been made in both Kappa (unpublished work, [WK]) and its sibling language BioNetGen [CHB$^+$11] to combine contact maps (site graph extensions of protein-protein interaction maps familiar to many biologists, representing agent-agent interactions in the model) with enriched model knowledge in the form of distilled dynamic information. One major goal of such approaches is to graphically describe an entire rule-based model such that it is both locally and globally comprehensible.

Current graphical representations of Kappa models have tentatively developed into the notion of 'rule-annotated contact maps'. Such a visualisation is based on the static elements present

(i.e. the compartments and agents and how they are related), similar to a standard Kappa contact map. It also displays model dynamics by mapping the preconditions of the rules (categorised five ways based on effect - creation, deletion, binding/unbinding interaction, state modification, and inter-compartment transport) to the static structure, and annotating them with their preconditions and rates.

A sample visualisation is shown in Figure 2b, based on the simple binding and transport model introduced in Example 1. Each edge corresponds to one or more rules; in turn, each annotation upon an edge describes the precondition (if any) and rate to one of these rules, such that the rule can be fully reconstructed from the annotation. The exact notation and representation remains a work in progress and subject to future refinement, but it suffices to naturally display the model considered in this paper.

3 The Collaboration Framework

In the previous section, we described two alternate representations (amongst many theoretically possible) of the same underlying biological knowledge base. Our challenge lies in synchronising these representations such that they may be utilised as a component for a structured framework. To do this, we define a new formal data structure to encapsulate the key underlying knowledge encoded in both model code and visualisations thereof, the preliminary formalisation of which is given in Definition 1.

Definition 1 A Kappa model data structure (KMDS) may be expressed as a tuple **(C,V,S,I,T)** where:
  C is a set of named compartments, each containing an initial population of agent expressions.
  V is a set of agents and nodes, each a child of a compartment from **C**:
    $\mathbf{V}_a$ is a set of agents.
    $\mathbf{V}_n$ is a set of nodes (currently used purely for graphical purposes).
  S is a set of sites, each a child of an agent from $\mathbf{V}_a$.
  I is a set of states, each a child of a site from **S**.
  T is a set of transitions (rules subdivided into five different categories):
    $\mathbf{T}_c$ is a set of creation rules representing a transition from a node to an agent with rate and conditions.
    $\mathbf{T}_d$ is a set of deletion rules representing a transition from an agent to a node with rate and conditions.
 $\mathbf{T}_t$ is a set of transport rules representing a transition from an agent in one compartment to an agent in another compartment with forward and backward rate and conditions.
    $\mathbf{T}_i$ is a set of interaction rules representing a transition from a site to a site with forward and backward rate and conditions.
 $\mathbf{T}_m$ is a set of modification rules representing a transition from a state to a state with forward and backward rate and conditions.
A concrete example of such a model data structure is given later in Example 2.

The entities form a hierarchy linking compartments down to states (nodes in this KMDS are simply used to represent 'empty' agents, such that creation transitions are graphically denoted
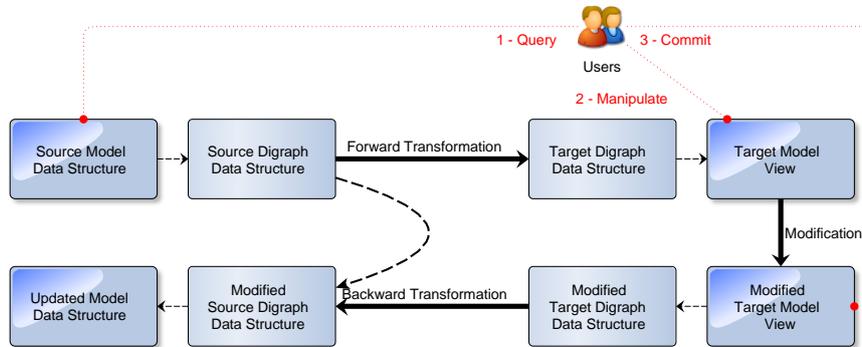
(a) A closeup of the bidirectional transformations involved in the framework.
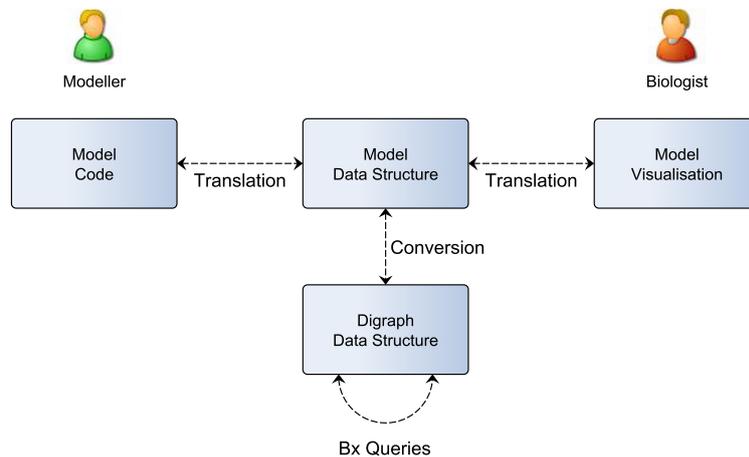
(b) The data structures involved in the framework, emphasising the different representations that collaborators may utilise upon the same underlying knowledge.

Figure 3: Users interact on the top level with the appropriate representation for their task and expertise; these representations correspond to a model data structure which can be translated to its digraph equivalent. Queries and bidirectional transformations are performed on the digraph data structure and passed back to the top level.

from such an 'empty' agent to a 'real' agent, and deletion transitions vice versa). Each transition corresponds to a set of rules linking together two of these entities within the model. Creation transitions govern the creation of new agents, deletion transitions the removal of existing agents, and transport transitions allow agents to move from one compartment to another (and possibly back again). Interaction transitions control the binding and unbinding of agents with one another, such as shown earlier in Example 1, and are often associated with modification transitions that switch between the internal states of a site. The conditions associated with a transition detail the rule preconditions, and are simply represented as strings since they hold no dynamic meaning within the data structure; similarly the attached rates simply store the relevant part of the model.

The present definition covers only a subset of all models possible in Kappa and its language extensions; for example, its compartments have no explicit notion of structural neighbourhood (rather, this is implicitly inferred via the transport rules), and it is unable to deal with rules with multiple effects (for example, both modification and unbinding in a single rule). As new representations are defined on the underlying knowledge base, it may become necessary to adjust the model data structure to better represent the knowledge contained within.

On the other hand, it contains the minimally necessary information for reconstructing both model code and visualisations; furthermore, models constructed in this format may be converted to an edge-labelled digraph representation in a straightforward manner. It thus acts as the pivotal data structure for our proposed collaboration framework as depicted in Figure 3a.

We assume the presence of a well-defined set of representations on the model data structure (MDS) which form the basis of user interaction with the collaboration framework. We also assume that the MDS and its corresponding representations may be converted to a digraph data structure (DDS) for the purposes of graph querying and bidirectional transformation (Figure 3b). The algorithms involved in translating (at the top level) and converting (between different levels) between these data structures are assumed to be bijective and thus information preserving. We then allow bidirectional transformations upon the low-level data structure, which utilise the aforementioned bijective algorithms in providing a user-friendly representation.

Let us illustrate the data structures with a concrete example, again adopting the Kappa model introduced earlier in Example 1 and its corresponding visualisation shown in Figure 2b. In this model two agents A and B, each with a single site, may bind and unbind from one another without restriction (solid lines) and transport back and forth between the two compartments cytosol and nucleus whilst in their unbound form (dotted lines). The initial conditions specify 100 each of A and B in the cytosol, and the rates are weighted such that the agents will aggregate in the nucleus in a bound form. The visualisation and the code both correspond to a unique underlying MDS describing this model, detailed below in Example 2.

Example 2   The Kappa model data structure (MDS) for the simple binding and transport model:

$$
\begin{aligned}
\mathbf{C} &= \{(c_1, \text{cytosol}, [100(A(a)), 100(B(b))], [], [\text{Agent } a_1, \text{Agent } a_2]), \\
&\quad (c_2, \text{nucleus}, [], [], [\text{Agent } a_3, \text{Agent } a_4])\} \\
\mathbf{V}_a &= \{(a_1, A, \text{Comp } c_1, [\text{Site } s_1]), (a_2, B, \text{Comp } c_1, [\text{Site } s_2]), \\
&\quad (a_3, A, \text{Comp } c_2, [\text{Site } s_3]), (a_4, B, \text{Comp } c_2, [\text{Site } s_4])\} \\
\mathbf{S} &= \{(s_1, a, \text{Agent } a_1, []), (s_2, b, \text{Agent } a_2, []), \\
&\quad (s_3, a, \text{Agent } a_3, []), (s_4, b, \text{Agent } a_4, [])\} \\
\mathbf{I} &= \{\} \\
\mathbf{T}_i &= \{(\text{Comp } c_1, \text{Agent } a_1, \text{Site } s_1, \text{Agent } a_2, \text{Site } s_2, [], 10.0, [], 1.0), \\
&\quad (\text{Comp } c_2, \text{Agent } a_3, \text{Site } s_3, \text{Agent } a_4, \text{Site } s_4, [], 10.0, [], 1.0)\} \\
\mathbf{T}_t &= \{(\text{Comp } c_1, \text{Agent } a_1, \text{Comp } c_2, \text{Agent } a_3, \sim cyt{:}A{:}a, 1.0, \sim nuc{:}A{:}a, 10.0), \\
&\quad (\text{Comp } c_1, \text{Agent } a_2, \text{Comp } c_2, \text{Agent } a_4, \sim cyt{:}B{:}b, 1.0, \sim nuc{:}B{:}b, 10.0)\}
\end{aligned}
$$

The MDS corresponds directly to Figure 2b. Two compartments, one of which houses the initial population of the model, each contain two possible agents and no nodes. Each of these
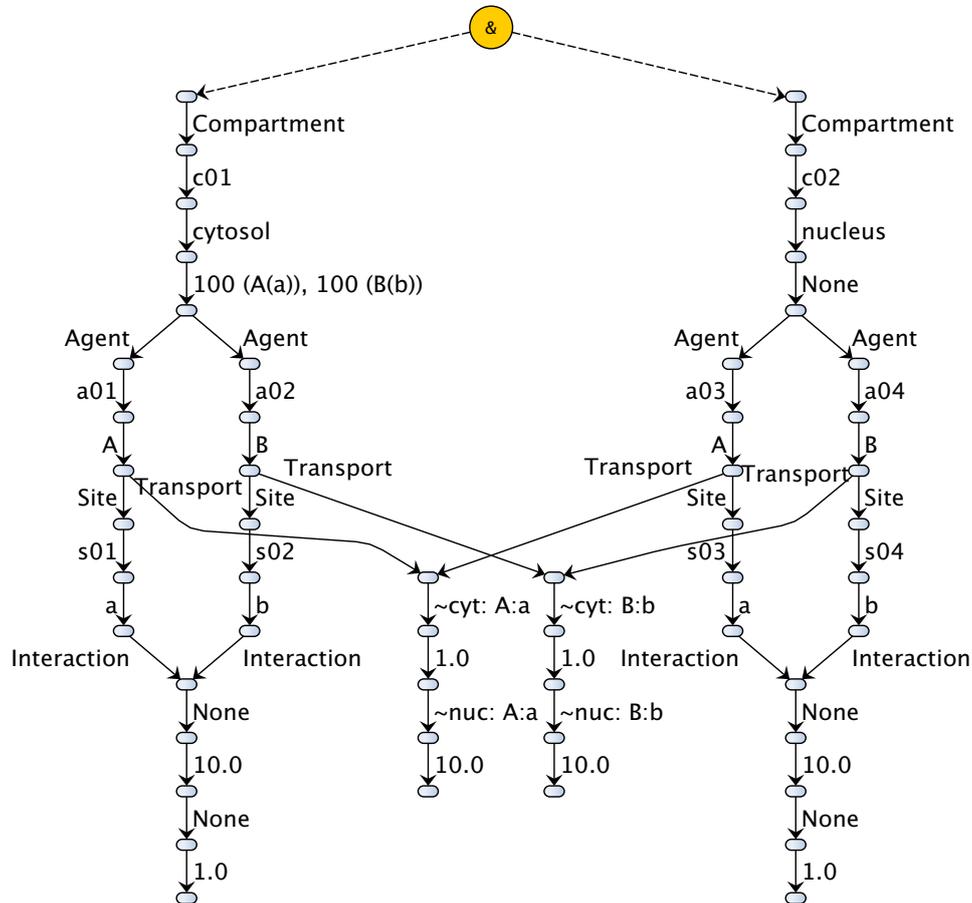
Figure 4: The digraph data structure for the simple binding and transport model previously described in Figure 2b, Example 1, and Example 2. Note the hierarchical tree-like structure of the entities, linked together by transitions.

agents has a single site which does not hold any state. The MDS contains four transitions, each corresponding to two rules (forward and backward): two interaction transitions controlling the binding and unbinding at the specified agent sites within a compartment, and two transport transitions controlling the movement of the specified agents between the compartments with the given conditions.

From this MDS we may proceed to construct the corresponding edge-labelled DDS depicted in Figure 4: in essence, we parse the entities to form a tree-like structure, and then link together corresponding entities according to the transitions present. Once again the exact format of the DDS remains flexible and subject to refinement. However, one distinct advantage of its current state is that it clearly reflects the tree-like hierarchy of entities (compartments, agents / nodes, sites, states), simplifying the traversal of edges to locate a particular node. This advantage is further elaborated in Section 4, where we examine operations on the DDS via UnQL+.

Of course, we are by no means limited to Kappa model code and visualisations as our representations upon the system; the opportunities for new representations are limitless, provided they can translate from the MDS. Neither are we limited to Kappa and its model data structure as our top-end user interface. Given an alternative MDS with similar representations upon the underlying knowledge base, along with an algorithmic means of translating between the MDS and a corresponding DDS (amenable to querying via UnQL+ and bidirectionalisation via GRound-Tram), it would be completely feasible to replace the top-end interface in its entirety. We believe that this modularity is a major advantage in future applications for the framework.

Our next step is to define the update operations possible via bidirectional graph transformation on the DDS.

4 Update Operations

Let us continue with our running example: a simple binding and transport model, described in the Kappa modelling language, under collaborative development by two or more users. In this section, we describe the sequence of events that occurs when one of these users wishes to modify the model under the framework.

4.1 A Sample Update Scenario

The user starts with a complete representation of the source model data structure, for example the visualisation shown in Figure 2b. From this, he or she queries for a subset of the information encoded in the model to create a view, up to and including its entirety (i.e. an identity query). In future iterations of this framework, we expect this query to automatically translate to UnQL+ from a query upon the model data structure; for now, we abstract from this translation and focus on querying the DDS in UnQL+. For example, let us assume that our user is an expert in the cytosolic compartment of the model. It is a simple matter to construct a graph query to select only the subset of the model that is of interest.

A forward transformation interprets the query to create a target DDS from the source. This is translated bijectively to a user-friendly view shown in Figure 5a.

The user is then able to modify the (subset of the) model as he or she desires. For example, let us suppose that our user wishes to add a third agent *C(c)* to the compartment, which binds competitively with the existing agent *A(a)* for the attention of *B(b)* and thus obstructs the accumulation of bound *A-B* pairs in the nucleus. The corresponding changes to the model are shown in Figure 5b.
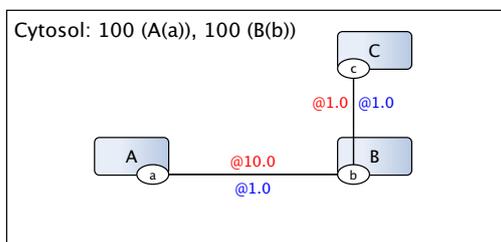
Finally, the user commits the modifications to the backward transformation mechanism of GRoundTram. The modified target model is bijectively translated once more into its DDS and integrated back to the original source, creating an updated source model (Figure 5c) for future use.

Throughout the process, it is our desire that the user remains completely unconcerned with the underlying DDS structures or the queries made upon them (Figure 6).

The advantages of the framework as proposed are threefold: the ability to conduct UnQL+ graph queries on high-level graph data structures as long as the conversion (from MDS to DDS) is
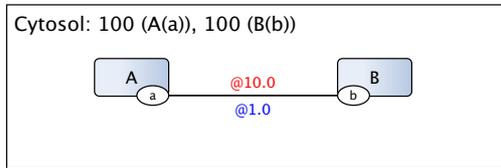
(a) A target view on the model described in Figure 2b after undergoing a simple query (forward transformation) eliminating all but the cytosolic compartment.

(b) The target view after modification. The user has added a new agent to the model, competitively binding with the existing agent *A(a)* at *B(b)*.
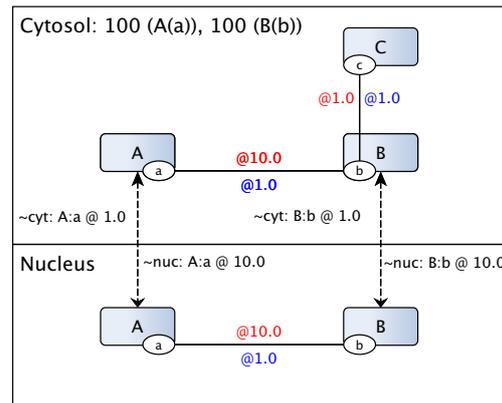
(c) After committing the modifications, the source model data structure is updated such that the new source reflects the changes made.

Figure 5: A series of rule-annotated contact maps depicting how a model evolves through one update iteration of the framework. Note that the modification made is meaningless in the context of simulating the model - another separate update is required to add agent *C(c)* to the initial conditions of the compartment!

well-defined; the guaranteed correctness of reflecting the target modification back to the source, no matter what view the modification is made on; and the opportunities for traceability and structured support (in terms of the manipulation options available to the user) for an iterative development methodology.

4.2 Update Implementation

In the remainder of this section we give further details of the data querying and manipulation options available to the user throughout the process, working on the DDS level. These include queries to select pertinent information from the model as a whole (Subsubsection 4.2.1), in-place refinement of model data (Subsubsection 4.2.2), and insertion (Subsubsection 4.2.3) and deletion (Subsubsection 4.2.4) of model entities and relations. Together the three manipulation operations provide all the functionality that a user may wish to employ in modifying the model; in essence, complicated modifications may be built from combinations of these atomic operations (for example, moving the contents of one compartment to another may involve a deletion followed by an insertion).

We make available online (http://www.prg.nii.ac.jp/projects/bxbio/) a set of sample queries and manipulations upon the simple binding and transport model introduced in this paper, allow-
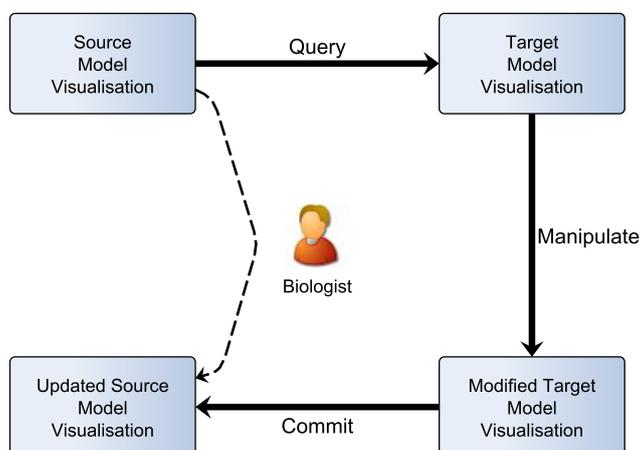
Figure 6: How a sample user may view the collaboration framework, completely abstracting away from the underlying implementation and the bidirectional graph transformation tools there.

ing readers to better visualise the framework application of bidirectional transformations to the model DDS. We stress that, at this stage, this is not a comprehensive evaluation of the framework, merely a preliminary proof of concept to demonstrate the feasibility of applying graph querying languages and bidirectional graph transformation to the field of biological modelling.

4.2.1 Selection

Queries on the DDS may be formulated via UnQL+ queries and tested using the unidirectional GRoundTram module *unqlplus*. It is often useful to apply them independently of the graph transformations, for example to generate a new non-modifiable user view on the knowledge base; alternatively, they may be incorporated into the various refinement, insertion, and deletion operations defined below. At first, we limit the possible queries to two basic types:

- Selecting a compartment, along with entities that exist and transitions that occur within its boundaries (including transport rules that map an agent in the compartment to a corresponding partner in a neighbouring compartment, although we do not display the partner itself).

- Selecting a specific agent, along with any sites, states, and transitions associated with it.

The queries take advantage of the tree-like DDS structure to locate the target entity, but their expressive power is currently constrained by limitations upon the ability of GRoundTram to provide bidirectional transformations upon their results. For example, although it is possible to define a query to eliminate the transport rules entirely from a single compartment, we are currently unable to reflect insertions upon the target view (i.e. the addition of new agents to the compartment) back to the source in an efficient manner. In the future we wish to provide a wider, more complicated range of queries without compromising the functionality described in the following sections.

4.2.2 Refinement

Apart from the data queries described above, the remaining user update operations will consist of data manipulation. Simple in-place refinement of edge labels - the renaming of entities within the model, the definition of new initial conditions, modifications made upon the rule preconditions, or rate parameter refinement - is possible via basic bidirectional transformations using the in-place update modules of GRoundTram. From a high-level user viewpoint, operations in this category would correspond to model modifications that simply edit pre-existing information, regardless of the representation chosen.

4.2.3 Insertion

The primary reason for limiting selection upon the DDS is the high (Recursively Enumerable) complexity involved in reflecting subgraph insertion from a queried target to its source. Insertion by in-place modification (and thus, without RE complexity) is possible only if (1) edits are made directly on a subgraph of the source extracted by the query using the graph variable reference, and (2-1) the subgraph is not traversed by the query, or (2-2) the inserted subgraph does not include any subgraph that would be extracted by the query.[1] However, a subset of UnQL+ queries (for example, involving the 'delete' construct) recursively copy said source to obtain their desired result, and hence any modifications made upon the target violate condition (1). By limiting queries to those detailed above it is possible to reflect all necessary insertions as may be feasibly desired by users (compartments, agents to an existing compartment, sites to an existing agent, states to an existing site, all forms of transitions) upon a modified target back to an updated source.

With regards to the characteristics of the chosen DDS representation, the current structure ensures that entities will always be added to a single parent (or the root, in case of a compartment) and will always be a single-stranded graph terminating in a leaf; transitions will have two parents but are otherwise similar in form. No cycles are formed, which also acts to simplify the subgraph insertion process and keep it within the operating capabilities of the current GRoundTram implementation. Once again, the GRoundTram in-place update modules are used.

From a high-level user viewpoint, update operations in this category would include the construction and integration of new structural or dynamic information, such as the addition of new agents and their behaviour into the model. The previously explained example in Subsection 4.1 is an illustration of the technique that may be used to add a new agent to a visualisation upon a Kappa model.

4.2.4 Deletion

Similar to subgraph insertion, subgraph deletion requires careful balancing of the expressive power of UnQL+ against the current implementation of GRoundTram. Again, the current DDS structure ensures that deletion will always be to a leaf (i.e. if we delete an agent, we wish to delete all sites, states, and transitions associated with it). Care is required to ensure that all associated

---

[1] Conditions (1) and (2-1) may be explained by [Voi09] where the query is polymorphic with respect to the extracted subgraph, i.e., the query does not depend on the content of the subgraph and just sends to the target as is. However, we heve not yet formalized the exact correspondence.

transitions are also eliminated. The GRoundTram module dedicated to deletion allows us to successfully reflect the deletion of both entities and transitions from a target view.

From a high-level user viewpoint, update operations in this category would include the removal of existing structural or dynamic information from the model. For example, in a visualisation as detailed in this paper, this would simply involve the deletion of an entity and all associated transitions.

5 Conclusion

We have presented in this paper an adaptation of a pre-existing bidirectional graph transformation framework, with the aim of facilitating the collaborative creation and development of biological models. We demonstrate the advantages of such an approach, including the querying of representations of high-level graph data structures using UnQL+ and the traceable reflection of user modifications back to their underlying source using bidirectional transformations, and we hint at the modularity with which it may be applied to further fields.

GRoundTram's ability to bidirectionalise graph transformations was applied positively throughout the development of the framework. Unidirectional application of UnQL+ queries independently of modifications, and abstraction away from the implementation details of both forward and backward transformations, are essential features of the proposed functionality. On the other hand there is always room for further enhancement: streamlining input files and formats (currently UnCAL files are used interchangably with both UnQL+ query inputs and DOT graph inputs); improving runtime error messages (currently difficult due to missing traceability between UnQL+ queries and their desugared internal UnCAL counterparts); further clarifying the conditions under which the in-place update module may be used for updates beyond simple edge label refinement; and generally improving user experience by providing a more accessible layer of functionality above the edge-labelled digraphs. Related to this, further work is also needed in broadening the range of queries available to the user, and ensuring their compatibility and correctness with the bidirectional graph transformations provided. It is hoped that the GRoundTram toolset grows in parallel with the framework, as new use cases and new issues are brought to light and examined.

From a modelling perspective as well, much work remains to be done before the proposed collaborative framework may be presented to users as a functional tool. Not least on this list is formalising the top-layer representations on the framework, especially the desired visualisations and operations upon them, and the bijective algorithms translating between them and converting between MDS and DDS. Eventually we hope to generalise both the conversion algorithms and the querying language such that, simply by defining an input schema, any possible model data structure may be converted to an equivalent edge-labelled digraph representation and manipulated via bidirectional graph transformations. In particular, we hope to draw upon concurrent work regarding a schema-based formalism for high-level queries upon a low-level digraph data structure [Tao] in translating more complex queries from high-level views to low-level graphs. Work is also required to test the framework on larger and more complicated models as opposed to the simple binding and transport model presented here, and in providing a suitable solution for the issue of conflicting modifications that arises due to the collaborative nature of the framework.

Nonetheless, by combining multiple concurrent representations on a consensus knowledge base with the guaranteed correctness and traceability properties offered by bidirectional graph transformations, we believe that we have succeeded in outlining a foundation and providing a proof of concept for further research. What currently represents one small step towards establishing structured model-driven biological approaches may one day assist in tackling the large questions faced by such fields as synthetic biology and regenerative medicine.

Bibliography

[AK07]   O. Andrei, H. Kirchner. Graph rewriting and strategies for modeling biochemical networks. In *Symbolic and Numeric Algorithms for Scientific Computing, 2007. SYNASC. International Symposium on*. Pp. 407–414. 2007.

[AKK⁺08] C. Amelunxen, F. Klar, A. Königs, T. Rötschke, A. Schürr. Metamodel-based tool integration with moflon. In *Proceedings of the 30th international conference on Software engineering*. ICSE '08, pp. 807–810. ACM, 2008.

[BFS00]   P. Buneman, M. Fernandez, D. Suciu. UnQL: a query language and algebra for semistructured data based on structural recursion. *The VLDB JournalThe International Journal on Very Large Data Bases* 9(1):76–110, 2000.

[BS11]   J. Bachman, P. Sorger. New approaches to modeling complex biochemistry. *Nature methods* 8(2):130, 2011.

[CHB⁺11] L. Chylek, B. Hu, M. Blinov, T. Emonet, J. Faeder, B. Goldstein, R. Gutenkunst, J. Haugh, T. Lipniacki, R. Posner et al. Guidelines for visualizing and annotating rule-based models. *Molecular BioSystems* 7(10):2779–2795, 2011.

[DFF⁺07] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Krivine. Rule-based modelling of cellular signalling. *CONCUR 2007–Concurrency Theory*, pp. 17–41, 2007.

[DFF⁺09] V. Danos, J. Féret, W. Fontana, R. Harmer, J. Krivine. Investigation of a biological repair scheme. *Membrane Computing*, pp. 1–12, 2009.

[DL04]   V. Danos, C. Laneve. Formal molecular biology. *Theoretical Computer Science* 325(1):69–110, 2004.

[FGM⁺07] J. N. Foster, M. B. Greenwald, J. T. Moore, B. C. Pierce, A. Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 29(3):17, 2007.

[HHI⁺10] S. Hidaka, Z. Hu, K. Inaba, H. Kato, K. Matsuda, K. Nakano. Bidirectionalizing Graph Transformations. In *ACM SIGPLAN International Conference on Functional Programming*. Pp. 205–216. ACM, 2010.

[HHI⁺11] S. Hidaka, Z. Hu, K. Inaba, H. Kato, K. Nakano. GRoundTram: An integrated framework for developing well-behaved bidirectional model transformations. In *Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on*. Pp. 480–483. 2011.

[LHM⁺09] N. Le Novère, M. Hucka, H. Mi, S. Moodie, F. Schreiber, A. Sorokin, E. Demir, K. Wegner, M. Aladjem, S. Wimalaratne et al. The systems biology graphical notation. *Nature biotechnology* 27(8):735–741, 2009.

[Moo12]   J. Moore. *Foundation Technologies in Synthetic Biology: Tools for Use in Understanding Plant Immunity*. PhD thesis, University of Edinburgh, 2012.

[SK08]   A. Schürr, F. Klar. 15 Years of Triple Graph Grammars. In *ICGT '08: Proceedings of the 4th international conference on Graph Transformations*. Pp. 411–425. Springer-Verlag, 2008.

[Ste]   D. Stewart. Spatial Biomodelling. http://www.demonsoft.org/SpatialKappa/.

[SW11]   D. Stewart, J. Wilson-Kanamori. Modular Modelling in Synthetic Biology: Light-Based Communication in *E. coli*. *Electronic Notes in Theoretical Computer Science* 277:77–87, 2011.

[Tao]   Z. Tao. BiQuery. http://www.prg.nii.ac.jp/members/stefanzan/biquery.html.

[Voi09]   J. Voigtländer. Bidirectionalization for free!(Pearl). In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages*. Pp. 165–176. ACM, 2009.

[WK]   J. Wilson-Kanamori. Rule-Annotated Contact Maps. Informal notes and talk at University of Edinburgh.

[YLH+12]   Y. Yu, Y. Lin, Z. Hu, S. Hidaka, H. Kato, L. Montrieux. Maintaining invariant traceability through bidirectional transformations. In *Proceedings of the 2012 International Conference on Software Engineering*. Pp. 540–550. 2012.