Distinguishing Context Dependent Events in Quotients of Causal Stories

Sébastien Légaré, Jean Krivine, Jérôme Feret

Distinguishing Context Dependent Events in Quotients of Causal Stories

Sébastien LÉGARÉ[1,2], Jean KRIVINE[3] and Jérôme FERET[1,2]
[1] Département d'informatique de l'ÉNS, ÉNS, CNRS, Université PSL, 75005 Paris, France
[2] INRIA, Centre de recherche INRIA de Paris, 75012 Paris, France
[3] CNRS, Université Paris Diderot, IRIF, 75013 Paris, France

Corresponding author: jerome.feret@ens.psl.eu

Abstract *Causality analysis of rule-based models allows the reconstruction of the causal paths leading to chosen events of interest. This potentially reveals emerging paths that were completely unknown at the time of creation of a model. However, current implementations provide results in the form of a collection of stories. For large models, this can amount to hundreds of story graphs to read and interpret for a single event of interest. In this work, we hence develop a method to fold a collection of stories into a single quotient graph. The main challenge is to find a trade-off in the partitioning of story events which will maximize compactness without loosing important details about information propagation in the model. The partitioning criterion proposed is relevant context, the context from an event's past which remains useful in its future. Each step of the method is illustrated on a toy rule-based model. This work is part of a longer term objective to automatically extract biological pathways from rule-based models.*

Keywords *Causality, Rule-based modelling, Visualisation, Graph folding, Concurrency*

1 Introduction

Rule-based modelling [1,2] is well suited to the construction of large models characteristic of systems biology. Rules distinguish themselves from reactions by focussing only on the part of molecules that changes during a transition, rather than fully defining the species involved. This lets rule-based models avoid combinatorial explosion issues and obviates the need to reduce models toward a predefined goal. Subsequent analysis can then reveal emerging properties that were initially completely unknown by the modellers.

Causality analysis is one of the most interesting analyses to perform on rule-based models. It allows the reconstruction of the causal paths, also called stories, leading to a chosen event of interest [3]. Considering a protein involved in some pathology for instance, it would provide a quantitative account of each upstream molecule's contribution to its activation. However, current implementations [3,4] present the results of causality analysis as a collection of individual stories. In a large systems biology model, this can amount to hundreds of story graphs to read and interpret for a single event of interest.

The goal of this work is to develop a compact representation that allows the visualisation of all the paths leading to an event of interest on a single graph. To do so, we fold a collection of stories by merging events that are deemed equivalent. The broader the definition of equivalence between events, the more compact the folded representation is. Yet, folding too much may merge events which are similar but play different roles in the propagation of information. Thus, a trade-off has to be found. Here, we refine each event in stories with some contextual information about its past, modulated by the usefulness of that context in its future. We use this additional information to define the quotient of events in families of stories. As a result, we obtain a quotient that remains compact but provides a better insight on the way information is processed in the models.

2 Initial Setting

2.1 Kappa Rule-Based Model

The toy model below is written in Kappa language [5] and will be used to illustrate the method developed in this work. Such toy model does not represent the kind of large systems that can typically benefit from rule-based modelling. However, it features events whose context is not trivially determined and is hence well suited to highlight the details of the methodology.

```
%agent:  A(x)
%agent:  B(y)
%agent:  C(x y z{u,p})
%agent:  D(z s{u,p})

'A binds C'  A(x[./1]), C(x[./1]) @ 0.01
'B binds C'  B(y[./1]), C(y[./1]) @ 0.01
'A phos C'   A(x[1]), C(x[1] z{u/p}) @ 1
'B phos C'   B(y[1]), C(y[1] z{u/p}) @ 1
'C binds D'  C(z[./1]{p}), D(z[./1]) @ 0.01
'A phos D'   A(x[1]), C(x[1] z[2]), D(z[2] s{u/p}) @ 1
'B phos D'   B(y[1]), C(y[1] z[2]), D(z[2] s{u/p}) @ 1

%init:  100 A()
%init:  100 B()
%init:  100 C()
%init:  100 D()

%obs:  'Dphos' |D(s{p})|
%mod:  [true] do $TRACK 'Dphos' [true];
```

For a description of the Kappa syntax, see [5]. Briefly, bonds are noted as a shared number between brackets. For example, A(x[1]), C(x[1]) means that A and C are bound through their respective site x. A dot between brackets, like A(x[.]), means that a site is free of any bond. States are given between braces. For example, C(z{p}) means here that site z of C is phosphorylated. In the definition of rules, a forward slash indicates an edit. What appears before the slash is a precondition, the binding or state value that a site must take to allow the firing of a rule. After the slash is the new binding or state value taken by a site after the firing of the rule. Brackets and braces without a forward slash inside them represent required preconditions that are unchanged by the firing of the rule.

Fig. 1 shows a sketch of the toy model, which can be described as follows. Kinases A and B can bind to protein C, each through a different binding site. Both A or B can phosphorylate site z of protein C once they are bound. Protein C can bind to D, but only after C was phosphorylated. A or B can then phosphorylate D as well, but only if they are in a same molecular complex as D. The phosphorylation of protein D is set as the event of interest. This simplified model is representative of scaffolding as it can occur in cell signalling. Protein C for instance could contain SH2 domains [6] to recruit kinases which would in turn phosphorylate multiple residues within a complex.

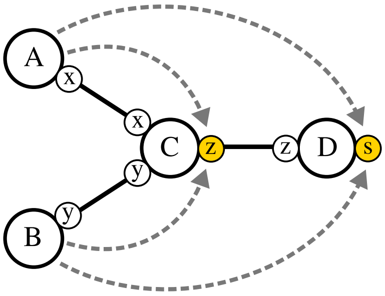

Fig. 1. Sketch of the toy model used to illustrate the method developed in this work.

2.2 Stories

Traces of computation from rule-based models can be sampled by the means of stochastic simulators [7,8]. Yet, such traces are not convenient to understand the mechanisms of signal processing because they contain satellite events that are unnecessary. They also describe precisely the order in which events occurred even when these events are causally independent. However, traces may be post-processed in order to extract relevant information. Event structures [9] abstract away the interleaving order between causally independent events. Then, irrelevant events may be discarded by using operational research techniques [3] or heuristic approaches [4]. The remaining events are the necessary steps required to reach the event of interest. Their representation in the form of a graph is called a story.

Fig. 2 shows the four stories corresponding to the four possible ways of obtaining phosphorylated `D` from the initial conditions of the toy model. Story 1 represents the case where kinase `A` phosphorylates both `C` and `D`. Story 2 is similar, but uses kinase `B` instead. Stories 3 and 4 represent cases where kinase `A` phosphorylates `C` and then kinase `B` phosphorylates `D`, or the other way around.

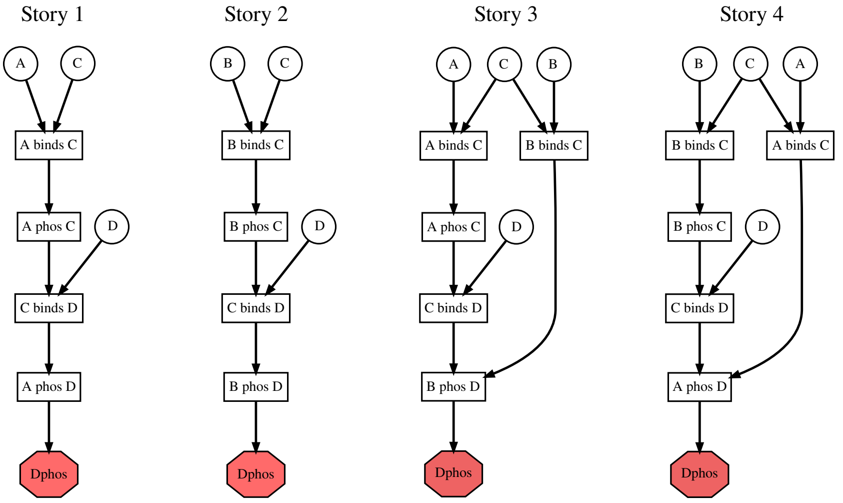

Fig. 2. The four possible stories to the phosphorylation of protein `D` in the toy model. Events are shown as rectangular nodes. Introduction nodes are shown as circles and represent the involvement of a new individual protein in the story. The event of interest appears as a red octagonal node. Edges represent precedence relationships. All graphs in this work were built with Graphviz [10].

Note that stories represent concurrency, or parallelism. When a given node has more that one incoming edge, those edges share an *and* relationship. In story 3 for instance, events `"C binds D"` *and* `"B binds C"` are both required to enable `"B phos D"`. Alternative paths cannot be represented within a story. They are rather shown as distinct stories.

3 Method

The story folding method presented in this work is implemented in the Python package KappaPathways [11]. The package is currently under development and still requires the implementation of additional functionalities before fully fledged pathway extraction from rule-based models can be performed.

3.1 Quotient of stories with concurrency

The main goal of this work is to build a compact representation of the results obtained from causality analysis of rule-based models. To do so, we seek to fold any arbitrarily large collection of stories into a single quotient graph. The first intuitive way to partition story events is according to the Kappa rules that they represent. That is, looking at the four stories from Fig. 2, all nodes with the same label are merged together.

Fig. 3, *left* shows the quotient obtained by simply partitioning story events according to their corresponding Kappa rule. While this graph does summarize some of the information from all the stories, it also clearly introduces ambiguities. For instance, readers could be mislead into thinking that completing event "A binds C" is sufficient to reach "A phos D", while in reality "C binds D" is also necessary. This ambiguity arise because when a node in the quotient graph has more than one incoming edge, it is impossible to know whether each edge comes from distinct stories or concurrent branches of a same story.

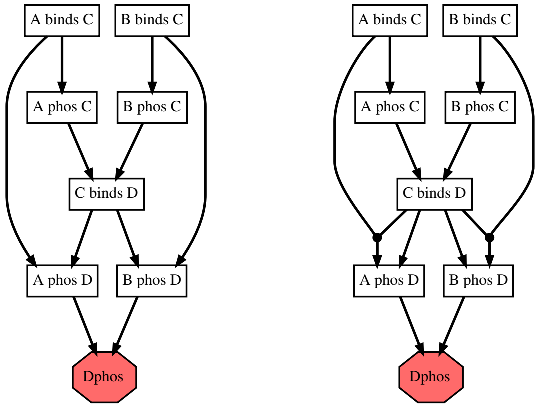

Fig. 3. Quotients of the four stories from Fig. 2. *Left*) Quotient obtained by partitioning event nodes according to their corresponding Kappa rule. *Right*) Quotient with hyperedges representing concurrency. Introduction nodes were removed for clarity.

Fig. 3, *right* illustrates a simple solution to the representation of concurrency in the quotient graph. Before the folding operation, any story edges that have the same target are regrouped in a single hyperedge with many sources and one target. The ensuing quotient allows a clear distinction between concurrent and alternative paths.

Still, the graph from Fig. 3, *right* remains misleading. It suggests paths that are not coherent with the model. For instance, the path "A phos C" →"C binds D" →"B phos C" seems allowed by itself. According to the model, it is instead only possible if "B binds C" also occurs concurrently. This inconsistent interpretation of the quotient graph tells us that a partitioning of events simply based on Kappa rules folds the stories too much. The next section presents additional event partitioning criteria that solve those ambiguities.

3.2 Relevant context

Looking back at stories 1 and 2 from Fig. 2 provides a hint into why folding too much produces a quotient with paths that are inconsistent with the model. Those two stories both pass through a same type of event, namely "C binds D". However, their preceding events are different. Although the latest modification that both stories went through is the same, the accumulated state of the molecules involved up to that point is different. They may hence have different futures open to them. That is, not only the events themselves matter, but also the context that was built up in their past. Still, not all past context may be relevant. A good trade-off must be found in the information that is kept about the context of each event. Too much information leads to a blow-up in the quotient graph that may become unreadable or even too costly to compute, whereas too few information may not be discriminant enough. In this work, we propose that the appropriate amount of information to keep corresponds to the relevant context.

The relevant context of a given event consists in the context from its past which remains useful in its future. More precisely, this corresponds to the locally relevant context, which appears relevant within a same story. We also define the globally relevant context as the context from an event's past which is found useful in the future of other stories which pass through an event corresponding to the same Kappa rule. The information about context can be extracted from the trace that was used to produce the stories. The following steps describe how we obtain context for each event and determine its relevance.

3.2.1 Edit nodes

First, for each story, the modifications that are performed by every event are extracted from the trace as illustrated on Fig. 4. Each individual modification is called an edit and is represented by a separate node connected to the event which it originates from. Edit nodes are labelled according to the modification that they correspond to using the Kappa syntax. Edges are then added from edit nodes to the downstream events which require them later in the story. Edges that do not correspond to any edge from the original story are referred to as transitive edges.

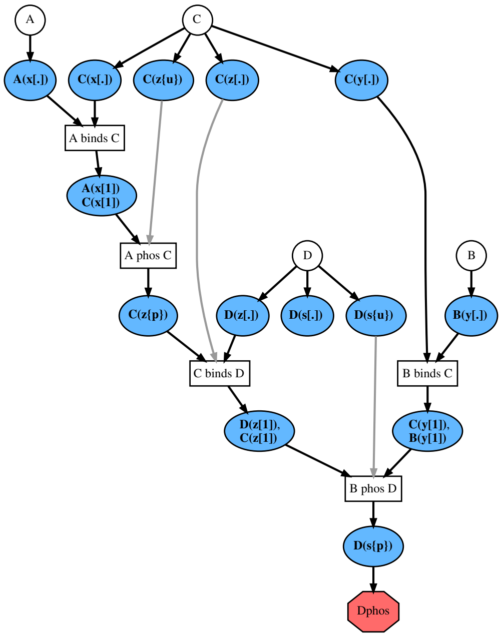

Fig. 4. Addition of edit nodes to story 3 from Fig. 2. Edit nodes are shown in blue and their labels represent modified sites in Kappa syntax. Transitive edges are shown in gray.

3.2.2 Locally relevant context

Second, the locally relevant context is determined as illustrated on Fig. 5. The following substeps are performed iteratively from top to bottom on the graph with added edit nodes. a) The current node is selected as the first edit node from the top of the graph whose locally relevant context was not already computed, excluding edits coming from introduction nodes. b) The past context of the current node is gathered. It first consists in the immediate upstream edit nodes from the current node, ignoring transitive edges. Neighboring nodes are also added as the edit nodes coming from the same event or introduction node as any upstream node. Lastly, the context that was found for those upstream and neighbor nodes during previous iterations is added to the past context of the current node. c) The locally relevant context is found as any past context node that remains useful in the future of the current node. On the graph, this corresponds to any past context node that have at least one path to a node which is reachable from the current node. The path should however not pass through the current node or an edit node that is incompatible with the edit from the past context node.

Fig. 5, *left* shows the first iteration of the steps described above on story 3 from Fig. 2. It provides the locally relevant context associated with event `"A binds C"`. Two elements of context are found relevant, as displayed by the two green edges and the inscription `z[.]{u}` on the label of the highlighted current node. This indicates that the locally relevant context is that protein `C` must have its site `z` free of any bond, and also unphosphorylated. Those two edits from the past are useful for future events `"C binds D"` and `"A phos C"`, respectively. Also note the red dashed edges which indicate past context nodes whose path to the future of the current node was blocked by incompatible edits.

Fig. 5, *right* shows the second iteration of locally relevant context determination. It now focusses on the context of event `"A phos C"`. Only the edit node coming from event `"A binds C"` counts here as an immediate upstream node since the edge to `C(z{u})` is transitive. `C(z[.])` and `C(z{u})` are nevertheless subsequently added as past context because the were found as relevant during the previous iteration.

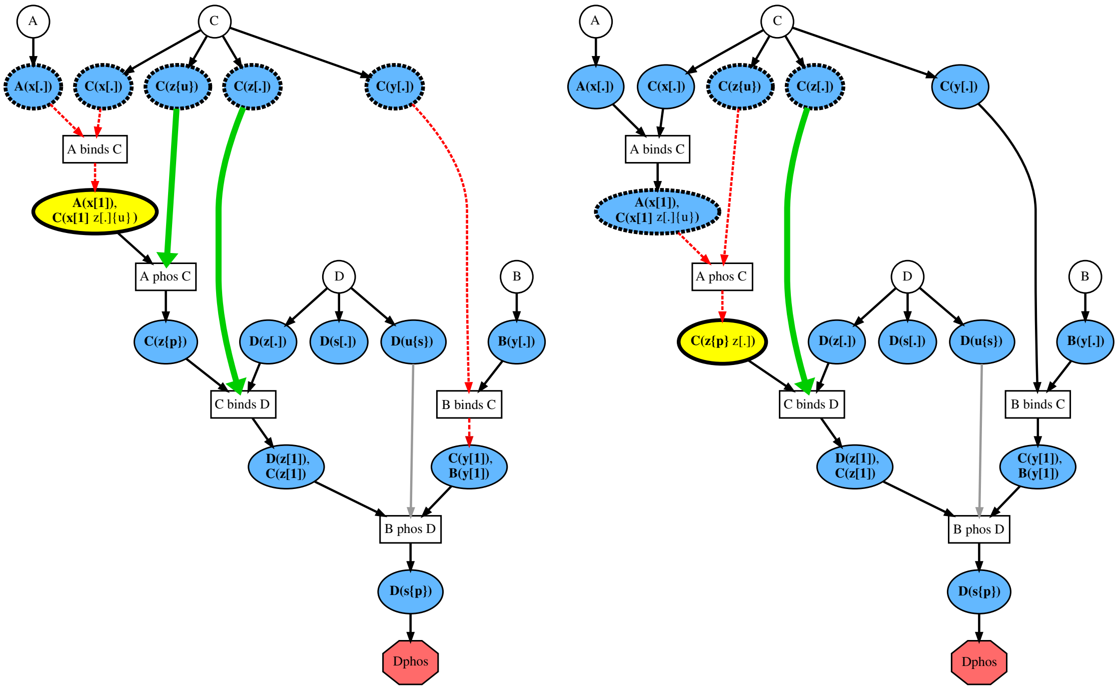

Fig. 5. Highlight of the nodes and edges involved in the determination of the locally relevant context. The first (*left*) and second (*right*) iterations are shown on story 3 from Fig. 2. The current node of each iteration is shown in yellow with bold border. Past context nodes have a bold dashed border. Paths from past context nodes to any node reachable by the current node are shown with thick green edges. Paths that are blocked by an edit node that is incompatible with the edit from the past context node are shown with dashed red edges. The locally relevant context obtained at the end of each iteration is written in thin font on the label of the current node.

3.2.3 Globally relevant context Third, the globally relevant context is evaluated. The goal is to harmonize what is considered relevant across all stories. Suppose an element of context which exists in the past of two different stories, but is found irrelevant in the first story and locally relevant in the second. Then, the first story must be revised, knowing that this given element of context is actually relevant when considering the whole system.

To do so, the total possible context of each edit is first computed. It corresponds to any context that was found locally relevant across all the edit nodes that represent a same edit among all stories. Then, the globally relevant context of a given edit node is found as the elements from the total context which exist in the past of that edit node within the story where it is found. Finally, elements of context are removed if they are found globally relevant across all edit nodes that represent a same edit.

Fig. 6 shows the four stories now with edit nodes containing the globally relevant context associated with each event. It turns out that the only relevant context is whether protein `C` was bound to kinase `A` or `B` when it got phosphorylated. Looking back at the toy model, this is precisely what was expected. Note that this context, the binding to `A` or `B`, was not locally relevant in story 3 as seen on Fig. 5. It was instead added as globally relevant from stories 1 and 2. Also, the locally relevant context `C(z[.])` from Fig. 5 was removed since it is equally present in all four stories and is hence not discriminating.

3.3 Quotient of contextualized stories

It is now possible to fold the contextualized stories from Fig. 6. Using edit nodes and their context defined in the previous, we can now partition the nodes in a way that will preserve the information propagation dictated by the model. Edit nodes across all stories are merged if they have the same edit and the same relevant context. Event nodes are merged if they correspond to the same Kappa rule and their target edit nodes also have the same relevant context.

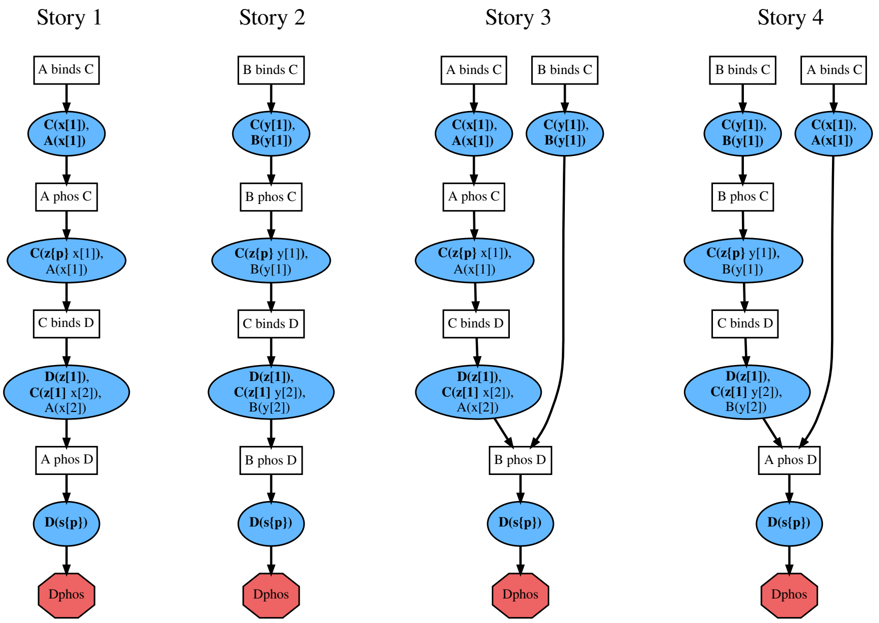

Fig. 6. The four stories from Fig. 2 annotated with relevant context on edit nodes. Introduction nodes were removed for clarity.

Fig. 7, *left* presents the resulting quotient. There are two nodes representing rule `"C binds D"`. One corresponds to the case where `C` was priorly bound to `A`, and to other to the case where it was bound to `B`. There is no way to read this graph by following a path that is not allowed by the model.

Fig. 7, *right* shows a more compact graph obtained by removing the contextual information about events that we have used to get a more precise quotient. We consider this last graph as the correct representation of the paths to the event of interest, as opposed to the two graphs from Fig. 3 which both lead to a wrong interpretation.

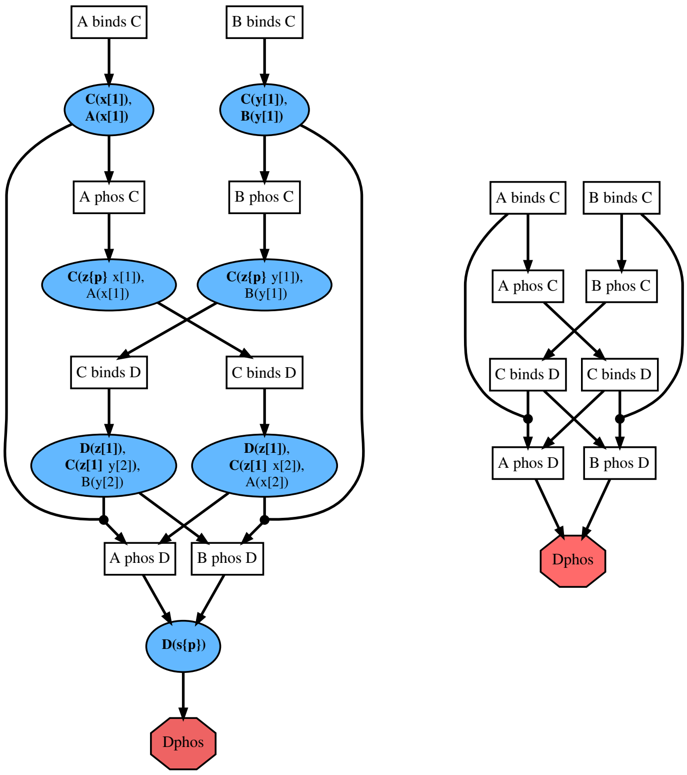

Fig. 7. Quotient graph obtained after considering relevant context. *Left*) Full version of the graph. *Right*) Compact version with edit nodes removed.

4 Conclusion

This work describes a method to build a compact representation for the visualisation of causality analysis results in rule-based modelling. A collection of stories provided by a current implementation of causality analysis [3,4] is folded into a single quotient graph. The main challenge is to find a partitioning of story events which maximizes compactness of the quotient graph without loosing important details about how information propagates in the model. The appropriate partitioning criterion suggested here is the relevant context, the context from an event's past which remains useful in its future. By folding stories based on their events type and relevant context, a quotient graph is obtained that faithfully represents all the paths to an event of interest.

Scalability of the method is important since its true usefulness lies in its application to large models. While rigorous optimization and benchmarking was not performed yet, we tested the current implementation on a model of human cell signalling comprising about three thousand Kappa rules. This model is considered large because representing it in a reaction-based setting, like in a Petri net, would lead to a combinatorial explosion with millions of nodes. Calculation time of quotient graphs for events of interest from this model is usually of the order of several minutes on an average personal computer. Those graphs are typically easy to visualize with less than fifty ordered nodes. For cases where the graph becomes larger and confusing to read, quantitative data from the model's execution can be used to prune away least impactful paths.

The story folding method reported in this paper is a milestone in a longer term objective to automatically extract biological pathways from rule-based models. Other functionalities are planned to be implemented within the KappaPathways [11] package to reach that goal. Those include counterfactual analysis [12], the representation of inhibitions from negative influences and the ordering of events participating in feedback loops.

Acknowledgements

This work has been partially supported by French ANR project DCore ANR-18-CE25-0007.

References

[1] Vincent Danos, Jérôme Feret, Walter Fontana, Russ Harmer, and Jean Krivine. Rule-based modelling of cellular signalling. In *CONCUR*, pages 17–41. Springer, 2007.

[2] Michael L. Blinov, Jin Yang, James R. Faeder, and William S. Hlavacek. Graph theory for rule-based modeling of biochemical networks. In *Transactions on Computational Systems Biology VII*, pages 89–106. Springer, 2006.

[3] Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, Jonathan Hayman, Jean Krivine, Chris Thompson-Walsh, and Glynn Winskel. Graphs, Rewriting and Pathway Reconstruction for Rule-Based Models. In *FSTTCS*, pages 276–288, 2012.

[4] Jonathan Laurent. KaFlow. `https://github.com/jonathan-laurent/KaFlow`.

[5] Pierre Boutillier, Mutaamba Maasha, Xing Li, Héctor F Medina-Abarca, Jean Krivine, Jérôme Feret, Ioana Cristescu, Angus G Forbes, and Walter Fontana. The Kappa platform for rule-based modeling. *Bioinformatics*, 34(13):i583–i592, 2018.

[6] Melany J. Wagner, Melissa M. Stacey, Bernard A. Liu, and Tony Pawson. Molecular mechanisms of sh2- and ptb-domain-containing proteins in receptor tyrosine kinase signaling. *Cold Spring Harb Perspect Biol.*, 5:a008987, 2013.

[7] V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable simulation of cellular signaling networks. In *APLAS*, pages 139–157. Springer, 2007.

[8] James R. Faeder, Michael L. Blinov, and William S. Hlavacek. Rule-based modeling of biochemical systems with BioNetGen. *Methods in molecular biology*, 500:113–167, 2009.

[9] Glynn Winskel. Event structures. In *Advances in Petri Nets*, pages 325–392. Springer, 1986.

[10] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz and dynagraph – static and dynamic graph drawing tools. In *Graph Drawing Software*, pages 127–148. Springer-Verlag, 2003.

[11] Sébastien Légaré. KappaPathways. `https://github.com/slegare2/KappaPathways`.

[12] Jonathan Laurent, Jean Yang, and Walter Fontana. Counterfactual resimulation for causal analysis of rule-based models. In *IJCAI*, pages 1882–1890. ijcai.org, 2018.