PORGY : a Visual Analytics Platform for System Modelling and Analysis Based on Graph Rewriting

Bruno Pinaud*, Oana Andrei**, Maribel Fernández***
Hélène Kirchner****, Guy Melançon*, Jason Vallet*

*Université de Bordeaux, LaBRI, France, {prénom.nom}@u-bordeaux.fr
**School of Computing Science, University of Glasgow, UK
***King's College London, UK
****Inria, France, helene.kircher@inria.fr

Abstract. PORGY is a visual environment for rule-based modelling based on port graphs and port graph rewrite rules whose application is steered by rewriting strategies. The focus of this demonstration is the visual and interactive features offered by PORGY, which facilitate an exploratory approach to model, simulate and analyse different ways of applying the rules while recording the model evolution, as well as tracking and plotting system parameters.

1 Introduction

We propose PORGY [1] (Fernández, Kirchner, and Pinaud, 2016) a general visual modelling framework (Fig. 1) based on graph rewriting (or graph rewriting) for complex systems. PORGY is based on the use of *port graphs with attributes* to represent system states. In a port graph, edges connect to nodes at specific points, called ports. Nodes, ports and edges describe system components and their relationships, while attributes encapsulate the data values associated with each entity. We use graph transformations based on port graph rewrite rules to describe the evolution of the system.

Graph transformations are usually specified by means of rules (Ehrig, Engels, Kreowski, and Rozenberg, 1997) and have been implemented in a variety of modelling tools, e.g., BioNetGen (Faeder, Blinov, and Hlavacek, 2009) or RuleBender (Smith, Xu, Sun, Faeder, and Marai, 2012). Such tools integrate visualisation with modelling and simulation of rule-based biochemical models with an emphasis on visual model exploration and integrated execution of simulations. States are represented by graphs describing the system components; their interactions are defined by rules governed by associated rate constants, which determine how frequently the rules apply. BioNetGen explicitly uses the structure of port graphs, while the other tools use graph-based structures with labels.

Generally speaking, port graph rewrite rules are graphical representations of transformations in the system, thus they provide a direct, visual mechanism to observe the system's behaviour. In addition to port graphs and rewrite rules our modelling approach includes *strategy expressions* to steer rule applications. Strategies allow using operators to combine graph

1. `http://porgy.labri.fr`

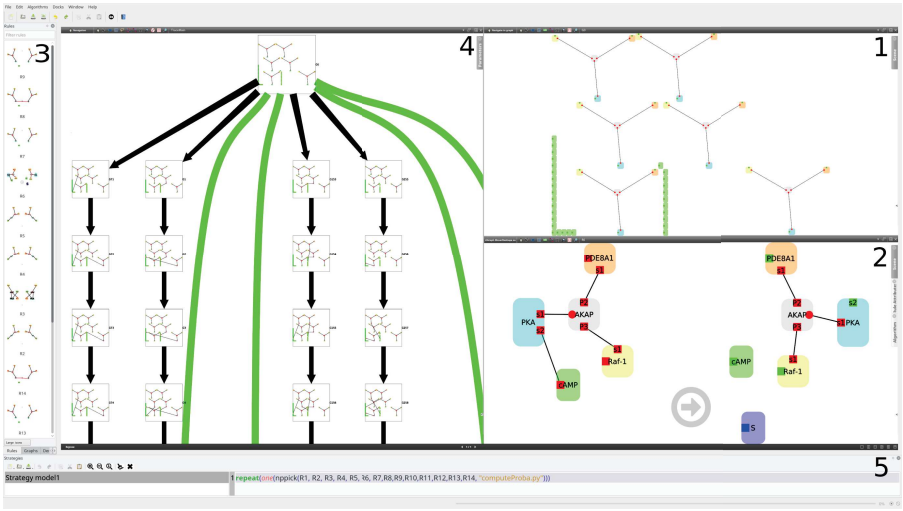

FIG. 1 – *Overview of* PORGY*: (1) editing a graph; (2) editing a rule; (3) all available rules; (4) the derivation tree, a complete trace of the computing history; (5) editing a strategy.*

rewriting rules, as well as operators to define the location where rules should, or should not, apply. Often more than one transformation is possible at a given state, in which case instead of a single transformation step, we may have several alternatives to choose from, and in turn, generate several different sequences of transformations. The various transformation sequences are organised as a tree structure, which we call *Derivation Tree* (DT).

In order to support the tasks involved in the study of a graph rewriting system, PORGY provides facilities to view each component at the same time (rules, strategy, any state of the rewritten graph, DT), to perform on-demand rewriting (strategy-based or rule-based) with drag-and-drop mechanisms, to synchronise the different views to track the evolution of system properties, to explore a DT with all possible derivations at different scales, to track the rewriting process throughout the whole DT, or to plot the evolution of a chosen parameter.

2 The PORGY Model

The design or PORGY is originally inspired from the study of biochemical systems, that is why we detail the use of PORGY for modelling a biochemical network. However, we recently used PORGY as a visual analytics tool for comparing propagation models in social networks (Fernández, Kirchner, Pinaud, and Vallet, 2016).

Elements as Port Graphs. In a port graph, nodes have explicit connection points called *ports*, edges are attached to ports; nodes, ports and edges are labelled by sets of attributes. We represent each model elements (proteins) as a node whose ports represent binding sites which can be linked to other binding sites.

Interactions (reactions) as rewrite rules. A *rewrite rule* $L \Rightarrow R$ consists of two subgraphs, L and R, linked together with a special node ($\Rightarrow$) that encodes the correspondence between

the elements of L and R (Fig 1, panel 2). Let G be a port graph such that there is a port graph morphism g from L to G. By replacing the subgraph $g(L)$ of G by $g(R)$ and connecting it with the rest of the graph, we obtain a port graph G' representing a result of a *rewriting step* of G using $L \Rightarrow R$. Rewriting is intrinsically non-deterministic since several subgraphs of a port graph may be rewritten under a set of rules.

Strategy for Rule Application. A strategy consists of either a rule or a composition of operators over a rule set. Rule applications can be governed by probabilities. Beyond the different choices for rule application, many other choices have to be made to control rewriting: choose where to apply a rule in a graph, define a sequence of rules which are correlated, iterate a rule or a sequence of rules, etc. The full strategy language is presented in Fernández et al. (2016).

Derivations and Derivation Tree. A *derivation* is a sequence of rewriting steps. Each step involves the application of a rule at a specific position in the graph. Navigating along derivations helps understanding how a specific state has been reached. In general, several derivations are possible from any state, giving rise to the notion of derivation tree DT (Fig. 1, Panel 4).

3 Experimentation and analysis

PORGY has been designed with the Visual information-seeking mantra of Shneiderman (1996) in mind: *Overview first, zoom and filter, then details on demand.* PORGY is built on top of the open-source visualisation framework TULIP [2] as a set of C++11 TULIP plugins.

Overview First. To understand the behaviour of non-deterministic systems, it is often useful to execute several times the same rewrite program on the same input to look for potential variations. These can be seen as branches in the DT. Although it is often a large data structure, it provides an indexed representation of the system evolution where each node represents one system state, and an edge is the application of a rule or a strategy. PORGY allows us to analyse the derivation tree and work with it at different levels. For instance, Small Multiples (SMs) allow seeing consecutive graph states like a comic-strip.

Zoom and Filter. One may be interested in plotting the evolution of a parameter computed out of each intermediate state. An interactive scatter plot can be built as in Fig. 2. Moreover, all graphical views are synchronised. For instance, if some interesting points are selected inside the scatter plot, they are also immediately selected inside the corresponding branch of the derivation tree. The synchronisation is also valid for graph elements.

Details on Demand. We can investigate further the selected nodes by zooming in and seeing distinctly the graphs. Hovering the mouse pointer over an edge allows to see which elements were changed by the application of the rule. The modified elements are emphasised in the picture, to clearly display which ones have evolved.

4 Conclusion

We have illustrated some key features of PORGY, an open-source general-purpose modelling and analysis environment. Domain-specific versions of PORGY can be easily implemented by extending or refining the features presented here.

2. `http://tulip.labri.fr`

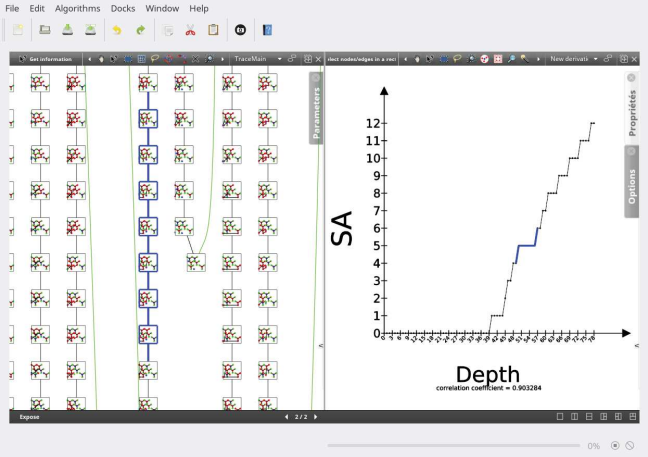

FIG. 2 – *A scatter plot showing the evolution of the* SA *protein concentration and the corresponding derivation tree. The points selected in the scatter plot (in blue) are automatically selected in the derivation tree.*

References

Ehrig, H., G. Engels, H.-J. Kreowski, and G. Rozenberg (1997). *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1-3*. World Scientific.

Faeder, J., M. Blinov, and W. Hlavacek (2009). Rule-Based Modeling of Biochemical Systems with BioNetGen. In I. V. Maly (Ed.), *Systems Biology*, Volume 500 of *Methods in Molecular Biology*, pp. 113–167. Humana Press.

Fernández, M., H. Kirchner, and B. Pinaud (2016). Strategic Port Graph Rewriting: An Interactive Modelling and Analysis Framework. Research Report, Inria.

Fernández, M., H. Kirchner, B. Pinaud, and J. Vallet (2016). Labelled Graph Rewriting Meets Social Networks. In *RTA'16*, Volume 9942 of *LNCS*, pp. 1–25. Springer.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. of the IEEE Symp. on Visual Languages*, pp. 336–343.

Smith, A. M., W. Xu, Y. Sun, J. R. Faeder, and G. Marai (2012). RuleBender: integrated modeling, simulation and visualization for rule-based intracellular biochemistry. *BMC Bioinformatics 13*(8).

Résumé

PORGY est un environnement interactif utilisé pour la modélisationde systèmes obtenus à partir de règles de réécriture, pilotés à l'aide de stratégies et basées sur des graphes utilisant des nœuds à ports. Cette démonstration présente quelques uns des aspects de visualisation analytique proposés par PORGY. Cette dernière facilite la modélisation du système, sa simulation ainsi que l'analyse des résultats à différentes échelles.