Snoopy—a unifying Petri net framework to investigate biomolecular networks

Christian Rohr1,2,3,∗, Wolfgang Marwan1,2 and Monika Heiner3

1Magdeburg Centre for Systems Biology (MaCS), Otto von Guericke University, 2Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg and 3Data Structures and Software Dependability, Computer Science Department, Brandenburg University of Technology, Cottbus, Germany

Abstract

To investigate biomolecular networks, Snoopy provides a unifying Petri net framework comprising a family of related Petri net classes. Models can be hierarchically structured, allowing for the mastering of larger networks. To move easily between the qualitative, stochastic and continuous modelling paradigms, models can be converted into each other. We get models sharing structure, but specialized by their kinetic information. The analysis and iterative reverse engineering of biomolecular networks is supported by the simultaneous use of several Petri net classes, while the graphical user interface adapts dynamically to the active one. Built-in animation and simulation are complemented by exports to various analysis tools. Snoopy facilitates the addition of new Petri net classes thanks to its generic design.

Availability: Our tool with Petri net samples is available free of charge for non-commercial use at http://www-dssz.informatik.tu-cottbus.de/snoopy.html; supported operating systems: Mac OS X, Windows and Linux (selected distributions).

Contact: snoopy@informatik.tu-cottbus.de

Received on December 14, 2009; revised on February 1, 2010; accepted on February 2, 2010

1 INTRODUCTION

Petri nets combine an intuitive, bipartite graphical representation with formal semantics, allowing for model execution to experience the net behaviour as well as exhaustive analyses to prove certain net properties. The rich body of Petri net theory comprises structural and dynamic analysis techniques, coming up with conclusions on behavioural properties under any timing constraints.

Petri nets enjoy an explicit notion of concurrency. They provide a flexible modelling language and allow the unambiguous representation of mass flow and biochemical or genetic regulation mechanisms at arbitrary abstraction level within one model, ranging from molecular via cellular to multi-cellular level, describing, for example, developmental processes in multi-cellular pattern formation.

Most importantly, Petri nets may serve as umbrella formalism integrating qualitative and quantitative (i.e. stochastic, continuous or hybrid) modelling and analysis techniques, supported by quite a number of reliable tools, developed by the international community of computational methods. Accordingly, Snoopy is set up as unifying

∗To whom correspondence should be addressed.
This net class extends QPN by assigning a stochastic waiting time to transitions. The standard notion of Petri nets are weighted, directed and bipartite graphs. They do not involve any timing aspects. Thus they allow a purely qualitative modelling of biomolecular networks. Tokens may represent molecules or abstract concentration levels. The class of extended Petri nets enhances standard Petri nets by four special arc types: read arcs, inhibitor arcs, equal arcs and reset arcs. They go always from places to transitions and can carry weights.

The built-in animation of the token flow may give first insight in the dynamic behaviour and the causality of the model. Snoopy visualizes the token flow, allowing for an easy comprehension. The animation can be triggered manually or be done in automatic mode with different firing strategies (single/intermediate/maximal step).

Snoopy supports a similar animation by a standard web browser.

Additionally, there is export to numerous external analysis tools of the Petri net community (http://www.informatik.uni-hamburg.de/TGI/PetriNets), among them the model checkers BDD-CTL and IDD-CTL (Heiner et al., 2009a).

3.3 SPN

This net class extends QPN by assigning a stochastic waiting time to transitions. The waiting time is specified by a firing rate function that can be any arithmetic function including the transition's pre-places (as integer variables) and user-defined (real-valued) parameters. Pre-places can be connected with a transition by a modifier arc. Then, they may modify the transition's firing rate, but do not have an influence on the transition's enabledness.

Popular kinetics (mass-action semantics and level semantics) are supported by pre-defined function patterns. Each transition gets its own rate function, making up together a list of rate functions. Moreover, several rate functions lists and parameter lists as well as multiple initial markings can be maintained, allowing for quite flexible models and their systematic evaluation by series of related computational experiments.

The underlying semantics is a continuous time Markov chain (CTMC); so, the simulation follows the standard Gillespie algorithm. Simulation results are available as tables and can be visualized in diagrams, showing the evolution over time of the token numbers on selected places or the firing times of selected transitions. The export to IDD-CSL (Schwarick and Heiner, 2009) and PRISM (http://www.prismmodelchecker.org) opens the doors to standard Markov analyses and CSL model checking.

Additionally, there are the four special arc types as for QPN, and three special transition types: immediate transitions (zero waiting time), deterministic transitions (deterministic waiting time, relative to the time point where the transition gets enabled) and scheduled transitions (scheduled to fire, if any, at single or equidistant absolute points of the simulation time). The unrestricted use of these extended features destroys the Markov property, but the simulation algorithm can be easily adapted. Simulation traces can be exported (as averaged/single/exact traces) and evaluated by simulative model checking of PLTLc with the Monte Carlo Model Checker MC2 (http://www.brc.dcs.gla.ac.uk/software/mc2), cf. Heiner et al., 2009b).

4 CONCLUSIONS

The tool is written in the programming language C++ using the library wxWidgets (http://www.wxwidgets.org). We constantly pay special attention to a platform-independent realization. Snoopy is now available for Mac OS X, Windows and Linux.

Snoopy supplies several further Petri net classes and graph types; see Heiner et al. (2008b) and Snoopy’s website. There one also finds examples for case studies, where Snoopy has been used. Snoopy is still evolving and we are open for suggestions.

ACKNOWLEDGEMENTS

Substantial contributions to Snoopy’s development have been done by former staff members and numerous student projects at Brandenburg University of Technology, chair Data Structures and Software Dependability; see Snoopy’s website for more information.

Conflict of Interest: none declared.

REFERENCES


