Snoopy - a unifying Petri net framework to investigate biomolecular networks

Christian Rohr\textsuperscript{1,2}, Wolfgang Marwan\textsuperscript{1} and Monika Heiner\textsuperscript{2}

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

ABSTRACT

Summary: 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 being 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

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 modeling 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, e.g., 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 Petri net framework, comprising a family of related Petri net models, sharing structure, but being specialized by their kinetic information: qualitative (time-free) place/transition Petri nets (QPN) as well as quantitative (time-dependent) Petri nets such as stochastic Petri nets (SPN) and continuous Petri nets (CPN). This provides the grounds to investigate one and the same model in various complementary ways (Heiner et al., 2008a).

To move easily between the qualitative, stochastic and continuous paradigms, models can be converted into each other; obviously with loss of information in some directions (cf. arrows labelled with Abstraction in Fig. 1). Different net classes can be used simultaneously.

2 MAIN CHARACTERISTICS

The Snoopy software has three distinguished characteristics. (1) It is extensible; its generic design facilitates the implementation of new Petri net classes. (2) It is adaptive by supporting the simultaneous use of several models, with the graphical user interface adapting dynamically to the net class in the active window. (3) It is platform-independent.

Snoopy supplies two features for the design and systematic construction of larger Petri nets. Logical nodes (places/transitions) serve as connectors and are used, e.g., for species involved in several reactions. Macro transitions (macro places) help to hide transition-bordered (place-bordered) subnets in order to design hierarchically structured nets.

Further features consistently available for all Petri net classes include: editing (copy, paste, cut), coloring of individual net elements and of computed node sets (e.g. place/transition invariants, traps, structural deadlocks, Parikh vectors), layouting (automatic layouting, mirror, flip, rotate), graphical export to eps, Xfig and FrameMaker, and print. The graphical editor prevents the construction of syntactically incorrect models.

Snoopy offers built-in animation (QPN) and simulation (SPN, CPN). Supplementary, Petri nets of these three net classes can be exported to various analysis tools, among them Charlie (http://www-dssz.informatik.tu-cottbus.de/charlie.html) and META-TOOL (http://pinguin.biologie.uni-jena.de/bioinformatik/networks).

Snoopy provides also import and export of the standard exchange format SBML, Level 2, Version 3 (http://sbml.org).

*to whom correspondence should be addressed
3 PETRI NET CLASSES

3.1 Petri Nets (QPN)

The standard notion of Petri nets are weighted, directed, bipartite graphs. They do not involve any timing aspects. Thus they allow a purely qualitative modeling 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-tools of the Petri net community (http://www.informatik.uni-hamburg.de/TGI/PetriNets), among them the model checkers BDD-tools of the Petri net community (http://www.informatik.uni-hamburg.de/TGI/PetriNets), among them the model checkers BDD-tools of the Petri net community (http://www.informatik.uni-hamburg.de/TGI/PetriNets), among them the model checkers BDD-tools of the Petri net community (http://www.informatik.uni-hamburg.de/TGI/PetriNets), among them the model checkers BDD-tools of the Petri net community.

3.2 Stochastic Petri nets (SPN)

This net class extends QPN by assigning a stochastic waiting time to transitions. The waiting time is specified by a firing rate function, which 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, 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-CTL (Heiner et al. 2009) and PRISM (http://www.prismmodelchecker.org) opens the doors to standard Markov analyses and CSL model checking.

3.3 Continuous Petri Nets (CPN)

Continuous Petri nets offer a graphical way to specify unambiguously systems of ordinary differential equations (ODEs). The real-valued tokens denote concentrations. The continuous rate functions have to obey similar rules as for SPN. The ODEs generated by a given CPN can be exported to Latex and as plain ASCII text. Snoopy provides 12 stiff/unstiff solvers for the numerical integration of CPN. The deterministic simulation traces are available as tables, can be visualized in diagrams, and exported to, e.g., checked against LTLc properties with MC2.

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; we are open for suggestions.

ACKNOWLEDGEMENT

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, and Execute Graph-Based Formalisms. Snoopy’s website for more information.

REFERENCES