Snoopy - A Tool to Design and Animate/Simulate Graph-Based Formalisms

Monika Heiner Ronny Richter Martin Schwarick

Brandenburg University of Technology at Cottbus
Dep. of CS, Data Structures and Software Dependability
Postbox 10 13 44, 03013 Cottbus, Germany
snoopy @ informatik.tu-cottbus.de
http://www-dssz.informatik.tu-cottbus.de/software/snoopy.html

Abstract

We sketch the fundamental properties and features of Snoopy, a tool to model and execute (animate, simulate) hierarchical graph-based system descriptions. The tool comes along with several pre-fabricated graph classes (in particular some kind of Petri nets and other related graphs), and facilitates a comfortable integration of further graph classes due to its generic design. To support an aspect-oriented model engineering, different graph classes may be used simultaneously. Our tool runs on Windows and Linux operating systems, and is available free of charge for non-commercial use.

1 Preliminaries

Snoopy [Webd] is a generic and adaptive tool for modelling and animating/simulating of hierarchical graph-based formalisms. While concentrating our development as far on several kinds of Petri nets and related graph classes, the generic design of Snoopy facilitates also a comfortable extension by new graph classes. The simultaneous use of several graph classes is supported by the dynamic adaptation of the graphical user interface to the active one. So it is possible to treat qualitative and quantitative models of the system under investigation side by side. For example you can start with a qualitative Petri net model and check some essential properties with external analysis tools. To get a basic understanding of the causal dependencies it is possible to play the token game. Later you can easily move on to related quantitative models, deterministic, stochastic or continuous ones, for a deeper understanding of the time dependencies governing your system. These quantitative models can be simulated with internal or external tools. This integrating approach was employed in [GH06], [GHL07]. To support this style of model engineering it is possible to convert between different graph classes, obviously with loss of information in some directions.

Snoopy runs on Windows and Linux operating systems; it is available free of charge for non-commercial use, and can be obtained from our website [Webd]. The source code is available on request.

2 Graph Independent Features

Snoopy provides for all graph classes some consistently available generic features, e.g. edit (copy, paste, cut), and layout (mirror, flip, rotate, and automatic layout by Graphviz [Weba]), as well as some graphical file export for documentation purposes (eps, Xfig, FrameMaker).

Graph constraints permit only the creation of syntactically correct models of the implemented graph classes.

The construction of large graphs is supported by a general hierarchy concept of subgraphs (represented as macro nodes) and by logical (fusion) nodes, which serve as connectors of distributed net parts. Additionally, colours or different shapes of individual graph elements may be used to highlight special functional aspects (compare figure 1).

A generic interaction mode allows a communication between different graphs. Some events in one graph can trigger commands (colouring, creating or deleting of graph elements) in another graph, even if they are of different graph classes [Dub07].

Furthermore, a dynamic colouring of graph elements for the visualization of paths or node sets (e.g. P/T-invariants, structural deadlocks, traps) is available [Win06].

A digital signature by md5 hash ensures the structure and the layout of the graph separately, which increases the confidence in former analysis results during model development [Dub05].

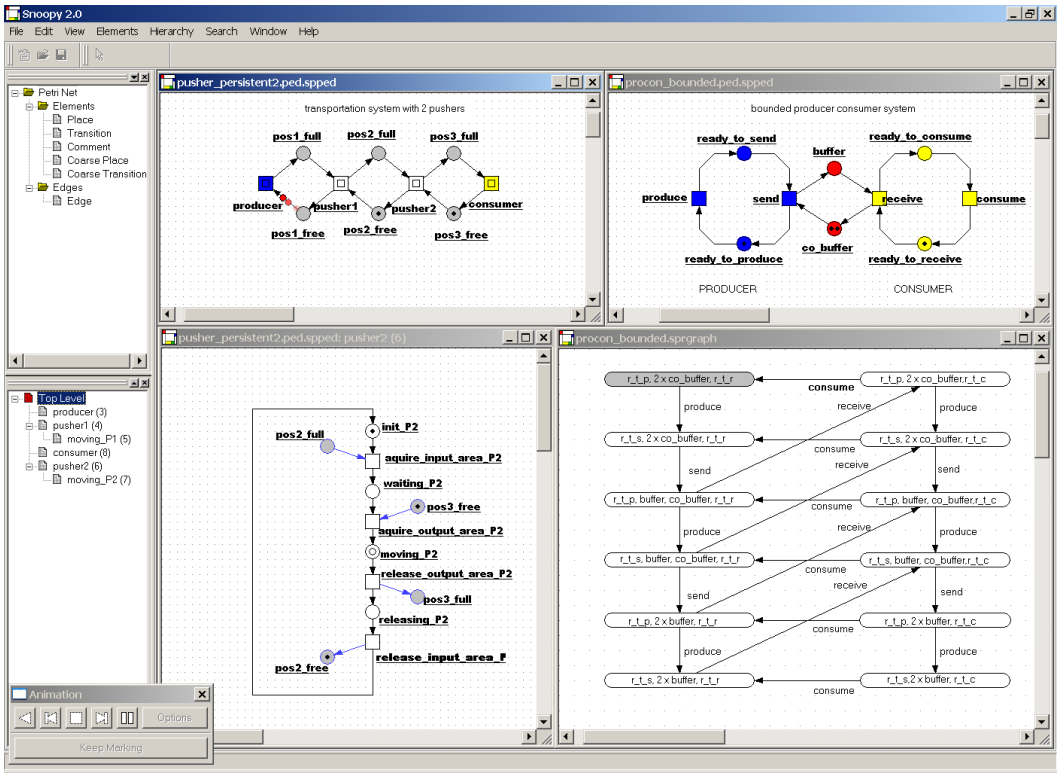

Figure 1: Snoopy Screenshot.

3 Realized Graph Classes

3.1 Reachability Graph

This simple graph class supports just one node and one arc type, besides comment nodes. The graph nodes can carry a name, a description and a Petri net state, which consists of a list of places and their markings. The arcs may be labelled by arbitrary character strings.

3.2 Petri Net

This bipartite graph class allows qualitative modelling by the standard notion of place/transition Petri nets. An animation by the token game gives first insights into the dynamic behavior and the causality of the model. The token game may be played step-wise or fully automated in forward or backward direction with different firing rules (maximal, minimal or intermediate steps). For modelling of large systems the hierarchy concept and fusion nodes have been proven to be useful.

The generic interaction manager [Dub07] allows to construct the reachability graph driven by the token flow animation of the Petri net. Furthermore, the export to a wide range of external analysis tools is available, among them INA, Lola, Maria, MC-Kit, Pep, Prod, Tina (see [Webc] for tool descriptions) as well as to our own toolbox Charlie [Sch06b]. Additionally, an import of the APNN file format supports advanced model sharing with other Petri nets tools.

3.3 Extended Petri Net

This graph class enhances place/transition Petri nets by some special arc types: read arcs, reset arcs, equal arcs and inhibitor arcs. A token flow animation and exports to external analysis tools are available, as in standard Petri nets. The special arc types are accepted only in the export to the APNN file format, which supports these graph elements.

3.4 Time Petri Net

This class introduces place/transition Petri nets with time. Up to now, time durations or time intervals can be assigned to transitions. The net analysis is supported by an export to INA.

3.5 Continuous Petri Net

Continuous Petri nets [Sch06a] may be considered as a structured approach to write systems of ordinary differential equations (ODEs), which are commonly used for a quantitative description of biochemical reaction networks (cf. section 4). For simulation several numerical algorithms (12 stiff/unstiff solvers, among others Runge-Kutta and Rosenbrock) are implemented, as well as an export to SBML [Webe] to use external analysis tools, popular in the systems biology community. Moreover, the ODEs defined by a continuous Petri net can be generated in LaTeX style.

3.6 MTBDD

For teaching purposes and documentation of small case studies we implemented multi-terminal binary decision diagrams.

3.7 Fault Tree and Extended Fault Tree

Fault trees describe the dependencies of component-based systems in failure conditions and are commonly used in risk management of systems with high dependability demands. Snoopy allows the qualitative and quantitative evaluation of fault trees. Several dependability measures may be computed, for example reliability, probability of system failure, and mean time to failure [Kur07].

3.8 Miscellaneous

The generic design of our graph tool allows an uncomplicated extension by new graph classes. For example EDL signatures (a formalism to describe patterns of computer network attacks) have been realized in [Roh07]. Snoopy is also involved in the tool chain of the approach for embedded system design presented in [GBC07]. You might want to find your own favorite graph class in a future version of this paper - we are open for suggestions and cooperations.

4 Case Studies

Snoopy has been used for a wide range of case studies, technical as well as biochemical ones, such as the control software of a production cell [HDS99], the metabolism of the potato tuber [KJH05], or various signal transduction networks [HKW04], [Neu04], [GH06], [GHL07] – just to mention a few due to space limitations.

5 Implementation

Snoopy was started in 1997 as a student's project [Men97], [Fie04] and is still under development and maintenance. It is based on the experience gathered by its predecessor PED, which it replaces. The tool is written in the programming language C++ with use of the Standard Template Library. A crucial point of the development is its platform-independent realisation, so Snoopy is now available for Windows and Linux operating systems. For this purpose, the graphical user interface employs the framework wxWidgets [Webf]. The object-oriented design (compare Figure 2) uses several design patterns, thus special requirements may be added easily. Due to a strict separation of internal data structures and graphical representation it is straightforward to extend Snoopy by a new graph class.

6 Future Work

In [GHL07], an integrative approach for biochemical network modelling has been proposed, using qualitative as well as stochastic and continuous Petri nets. This technology will be supported by the extension developed in [Leh07]. The import

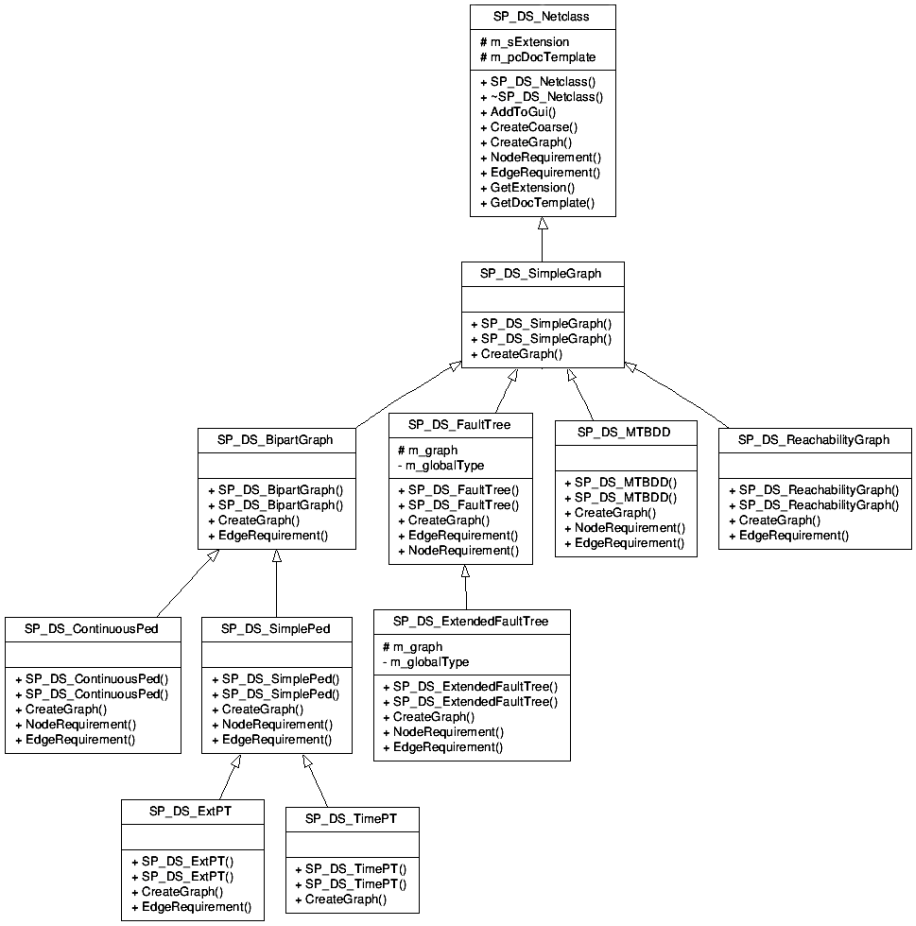

Figure 2: Extract of Snoopy's class hierarchy as generated by Doxygen.

of biochemical network models in the KEGG and SBML data formats is about to be released [Sch07], which will allow the direct re-use and re-engineering of models from the systems biology community. An ongoing student's project works on managing and executing animation sequences, especially counter examples produced by external analysis tools.

Finally, a PNML [Webb] import and export will be implemented, as soon as a (preliminary) final standard will be available.

References

[Dub05] Matthias Dube. Signing and Verifying SNOOPY files. Technical Report, Universit degli Studi di Milano, Dipartimento di Informatica e Comunicazione, 2005.

[Dub07] Matthias Dube. Design and Implementation of a Generic Concept for an Interaction of Graphs in Snoopy (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., 2007.

[Fie04] Markus Fieber. Design and Implementation of a Generic and Adaptive Tool for Graph Manipulation (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., 2004.

[GBC07] Luis Gomes, Joao Barros, and Aniko Costa. Petri Nets Tools and Embedded Systems Design. In *Proc. Workshop on Petri Nets and Software Engineering (PNSE07) at Int. Conf. on Application and Theory of Petri Nets (ICATPN '07 Siedlce)*, pages 214–219, 2007.

[GH06] David Gilbert and Monika Heiner. From Petri Nets to Differential Equations - an Integrative Approach for Biochemical Network Analysis;. In *Proc. 27th ICATPN 2006, LNCS 4024*, pages 181–200. Springer, 2006.

[GHL07] David Gilbert, Monika Heiner, and Sebastian Lehrack. A Unifying Framework for Modelling and Analysing Biochemical Pathways Using Petri Nets. In *Proc. CMSB 2007, LNCS/LNBI 4695*, pages 200–216. Springer, 2007.

[HDS99] Monika Heiner, Peter Deussen, and Jochen Spranger. A Case Study in Design and Verification of Manufacturing Systems with Hierarchical Petri Nets. *Journal of Advanced Manufacturing Technology*, 15:139–152, 1999.

[HKW04] Monika Heiner, Ina Koch, and Jürgen Will. Model Validation of Biological Pathways Using Petri Nets - Demonstrated for Apoptosis. *BioSystems*, 75:15–28, 2004.

[KJH05] Ina Koch, Björn H. Junker, and Monika Heiner. Application of Petri Net Theory for Modeling and Validation of the Sucrose Breakdown Pathway in the Potato Tuber. *Bioinformatics*, 21(7):1219–1226, 2005.

[Kur07] Anja Kurth. Fault Trees in Snoopy (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., 2007.

[Leh07] Sebastian Lehrack. Stochastic Petri Nets in Snoopy (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., in preparation, 2007.

[Men97] Thomas Menzel. Design and Implementation of a Framework for Petri Net Oriented Modelling (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., 1997.

[Neu04] Gerry Neumann. Modeling of Biochemical Processes with Petri Nets; Hemostasis vs. Fibrinolysis vs. Inhibitors (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., 2004.

[Roh07] Christian Rohr. Design and Implementation of an Editor for Visualizing and Debugging of EDL Signatures (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., 2007.

[Sch06a] Daniel Scheibler. A Tool to Design and Simulate Continuous Petri Nets (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., 2006.

[Sch06b] Martin Schwarick. A Tool to Analyze Petri Nets (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., 2006.

[Sch07] Daniel Schrödter. Re-engineering of Biochemical Networks (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., submitted, 2007.

[Weba] Website Graphviz. Graph Visualization Software. `http://www.graphviz.org` last visit: 08/2007.

[Webb] Website Petri Net Markup Language. PNML Framework Release 1.2.0. `http://www-src.lip6.fr/logiciels/mars/PNML/`, last visit: 08/2007.

[Webc] Website Petri Nets World. The Home Site. `http://www.informatik.uni-hamburg.de/TGI/PetriNets/`, last visit: 08/2007.

[Webd] Website Snoopy. A Tool to Design and Animate/Simulate Graphs. `http://www-dssz.informatik.tu-cottbus.de/software/snoopy.html`.

[Webe] Website Systems Biology Markup Language. The Home Site. `http://sbml.org/index.psp`, last visit: 08/2007.

[Webf] Website wxWidgets. A Toolkit for cross-platform GUI Application. `http://www.wxwidgets.org`, last visit: 08/2007.

[Win06] Katja Winder. Invariant-based Structural Characterization of Petri Nets (in German). Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Dept., 2006.