PEP Version 2.0

Christian Stehno

Fachbereich Informatik, Carl-von-Ossietzky-Universität Oldenburg,
D-26111 Oldenburg, Germany

Abstract

The PEP tool provides an integrated development and verification environment for parallel systems. Many improvements have been made since the last version from 1998. Most importantly the new version 2.0 supports Time Petri nets and uses the faster Tcl/Tk 8.0 language. Starting with this release PEP is free software. Downloadable versions are available for Linux and Solaris.

1 Overview of PEP

The PEP tool is a **P**rogramming **E**nvironment based on **P**etri nets[1]. It supports the development, simulation and verification of parallel programs with a variety of tools and formalism through its graphical user interface.

While PEP is mainly an integrated development environment it includes a complete Petri net tool featuring editors and simulators for high and low-level nets and a large base of analyzing and verification tools based on those nets. In addition there exists a 3D simulator and a C code generator with interfaces to the net models of PEP. A complete overview of the capabilities of PEP can be found in section 2.

The PEP tool has been developed in a project supported by the DFG and in cooperation of the universities of Hildesheim, Oldenburg, Humboldt-Universität Berlin and Technische Universität München in 1993-1998. Since then it has become a program used world-wide with hundreds of users. Since the official end of the project the development stopped for almost a year, while it resumed in the last month bringing this latest version.

2 Components of the PEP tool

This section describes the tools of PEP and their interconnections.

The complete structure of the PEP tool consists of three layers shown in figure 1. Links between these layers are top-down meaning from higher to lower levels of abstraction. The borders of these layers are not strict and many tools encompass more than one layer. In the figure they are drawn in their substantial level.

[1] http://parsys.informatik.uni-oldenburg.de/~pep

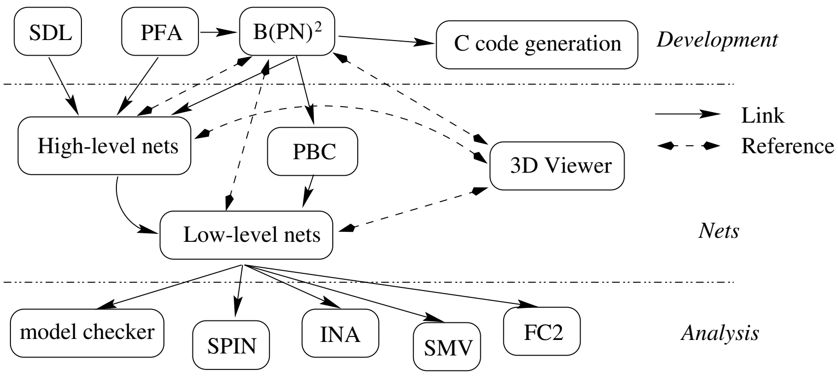

Figure 1: The components of the PEP tool

2.1 Development level

This level contains the input modules on the highest level of abstraction. They are frequently used for large systems specification and design, due to their compactness and simplicity. In addition to an editor for the parallel programming language B(PN)^2 [1], there are also editors for the Specification and Description Language SDL [5] and for parallel finite automata [6]. The first two provide their own textual editor with keyword highlighting and macro substitution, while the latter uses a graphical editor with a point-and-click interface similar to the net editors.

2.2 Net level

The net level provides the Petri net tool part of PEP. It contains editors and simulators for high and low-level nets but goes beyond this. As a result of using the net classes supported by PEP, the net editor contains a composition interface for nets to produce new nets from known with special operators in a way similar to programming. A related approach exists for the Petri net algebra PBC.

All systems from the development level can be compiled into a Petri net representing their semantics. Along with this transformation, references are constructed between representations providing, the ability of PEP to simulate not only the nets but also all models directly connected to a net and vice versa.

The 3D visualisation is a special case for the reference model. It provides a VRML description of a 3D world and links action nodes of this world with the net. Simulating the net results in an animation of the active parts, and also, interaction inside the VRML model gives state changes in the net model.

2.3 Analysis level

The analysis level includes Petri net specific verification checks and model checking of all the models used with PEP.

Nets can be checked for the most common net properties with integrated checkers and the INA tool[2], e.g. for safety. The behaviour of a net can be examined in many ways. They range from state space analysis to model checking with linear and branching time logics. The models used for these purposes include many different approaches, too. They include partial order representation, BDDs[3] and more proprietary models like Promela[4] and communicating automata[5].

3 Improvements

New features of the current version include:

- Time Petri nets are supported in a number of tools
- Improved and reviewed GUI using Tcl/Tk 8.0
- Improved I/O interface for PEP file formats
- PEP becomes free software
- New analyzer for linear programming verification
- Completely rewritten interface to INA and new versions of external tools.

3.1 Time Petri nets

The large interests in real time systems in the last years resulted in a special needs for tools which support models with timing information. Together with the development of the theory we implemented and updated many of the tools of PEP to support Time Petri nets.

For the time being PEP supports timing information within the algebra PBC, on the low-level nets, within all programs that support the finite prefix (which can be build for Time Petri nets), and within the INA tool. The net editor supports time information only on the editing level. Simulation is planned to be added in the next release.

With the latest version of the unfolder PEP is able to create the finite prefix of a net according to [9, 3] also for Time Petri nets [2, 10]. The finite prefix contains the complete information on the behaviour of nets presented in a compact way using a partial order. By this the concurrency of the net can be represented without explicit serialisation. The PEP model checker and many of the analysis

[2]INA: http://www.informatik.hu-berlin.de/lehrstuehle/automaten/ina
[3]SMV: http://www.cs.cmu.edu/ modelcheck/smv.html
[4]SPIN: http://netlib.bell-labs.com/netlib/spin/whatispin.html
[5]FC2: http://www-sop.inria.fr/meije/verification/

tools use the unfolding for their calculation. With the generation of the finite prefix for Time Petri nets all of these tools can be used on timed nets without change.

Complementing the tools using the finite prefix, the INA tool provides some other checks for Petri nets. PEP now also produces time information files for INA if desired, so all analysis of the INA tool for Time Petri nets is available in PEP.

3.2 Updated GUI using Tcl/Tk 8.0

With this release PEP has undergone a major change with respect to its graphical user interface. Since PEP now uses the common Tcl/Tk 8.0, it is much easier to install than in the previous version. The interpretation of the scripts is now compiled on the fly and gives a better performance and usability.

Many parts of the GUI were rewritten using standard Tcl/Tk features and avoiding system calls. Thus, PEP becomes more portable and more tightly integrated having less system dependencies. The interface of PEP is smoother and cleaner, reducing the costs of new implementations for tool programmers.

3.3 New I/O library

Most of the tools of PEP use a file interchange format and interface to communicate with each other. Although many formats are used by more than one tool, most of them provided their own input and output routines. This made a change of the format complicated and yielded many errors in different implementations. For some of the formats the new PEP I/O library provides all necessary functions, ensuring compatibility between all tools. As of this version, only the low level net formats are supported by the library, but the next release will hopefully include many more.

The use of the library simplifies the use of data generated by PEP and will help programmers implement new tools for PEP. It also reduces the size of the PEP tool, as the library is provided for all the tools of PEP using shared code.

3.4 Changed license policy

From now on, PEP is distributed as a free program in OpenSource instead of forcing users to ask for a license file. We hope to attract both new programmers and new users by this step. It simplifies the usage of PEP and gives programmers the chance to work not only with the GUI part of PEP which was available as source already.

3.5 New linear programming analyser

Programmed and provided by Viktor Khomenko, the new analyser implements the algorithm presented in [7]. This checker enhances the abilities of the already integrated LP tool of PEP by generating counter examples for its results and supporting a larger variety of solvable equation systems. In addition, it speeds up calculations in a dramatic way through better use of the information provided by the finite prefix. By the use of the finite prefix, the tool is also able to check Time Petri nets (see above).

3.6 Updates for external tools

The GUI for the INA tool was renewed. With the work of Stephan Roch, the usability of INA through the graphical user interface has changed completely. The tool is now quite stable and provides a compact and easy to use look and feel.

New versions of the external tools SPIN and SMV are used in this version of PEP. This should speed up the calculations within those tools, since both have improved their speed. New features are not supported yet, however, unless they are used automatically.

References

[1] E. Best and R. P. Hopkins. B(PN)^2– a Basic Petri Net Programming Notation. *Proc. of PARLE*, Springer, *LNCS 694*

[2] B. Bieber and H. Fleischhack: Model Checking of Time Petri Nets Based on Partial Order Semantics. In *Proc. of Concur '99*, 1999

[3] J. Esparza and S. Römer and W. Vogler. An Improvement of McMillan's Unfolding Algorithm. In *Proc. of TACAS'96*, 1996.

[4] J. Esparza: Model Checking Using Net Unfoldings, *Science of Computer Programming*, Volume 23, pages 151-195, Elsevier, 1994

[5] H. Fleischhack and B. Grahlmann. A Compositional Petri Net Semantics for SDL. In *Proc. of ATPN'98, LNCS 1420*, Springer, 1998

[6] B. Grahlmann, M. Moeller, and U. Anhalt. A New Interface for the PEP Tool – Parallel Finite Automata. In *Proc. of AWPN'95*, AIS 22. FB10 Universität Oldenburg, 1995

[7] V. Khomenko and M. Koutny. LP Deadlock Checking Using Partial Order Dependencies. *Proc. of CONCUR'2000, LNCS 1877*, Springer-Verlag, 2000

[8] M. Koutny. A Compositional Model of Time Petri Nets. In *Proc. of Application and Theory of Petri Nets 2000, LNCS 1825*, Springer, 2000

[9] K. McMillan. Symbolic model checking: An approach to the state explosion problem. Kluwer Academic Publishers, 1993

[10] C. Stehno. Entfaltung von Zeit-Petrinetzen, Integration in das PEP-System und Optimierungen (in German). Master thesis. Universität Oldenburg, 2000