The PEPA Plug-in Project

Mirco Tribastone
Laboratory for Foundations of Computer Science
The University of Edinburgh
mtribast@inf.ed.ac.uk

Abstract

We present a GUI-based tool supporting the stochastic process algebra PEPA with modules for performance evaluation through Markovian steady-state analysis, fluid flow analysis, and stochastic simulation.

1. Introduction

PEPA is a stochastic process algebra that extends classical process algebras with the association of exponentially distributed random variables to actions [1]. The Continuous Time Markov Chain underlying a PEPA model can be employed to carry out performance evaluation. However, this technique is subject to the well-known state space explosion problem, which makes the model intractable because of its size. To tackle the problem, recent work has been focused on providing fluid flow approximation of the state space which leads to an underlying mathematical representation in the form of a system of first-order ordinary differential equations (ODEs) [2].

The PEPA Plug-in Project is built on top of the Eclipse technology [3] and consists of a collection of plug-ins for the Eclipse IDE. As such, it is available on all the platforms supported by the framework, including Windows XP, various Linux distributions and Mac OS X on which it has been successfully tested. It is deployed as an Eclipse *feature* and it is available for download at `http://homepages.inf.ed.ac.uk/mtribast/`. The software comprises two main components. *PEPAto* is designed to be an Application Programming Interface (API) providing core services for the execution of PEPA tasks. *Eclipse UI* provides a rich graphical user interface to such services, built on top of the Eclipse/SWT API. It features an editor supporting graphical annotations for problems encountered during the modelling process, and user-friendly tools for the navigation of the state space and the extraction of performance measures from the model.

2. The PEPAto Library

PEPAto is centred around the in-memory representation of a PEPA model—the abstract syntax tree. This can be obtained either via parsing of PEPA descriptions (usually represented by text files with the `pepa` extension), or programmatically using the API. The syntax for such descriptions is compliant with the original design of the language. A backward compatible extension is also provided to deal with *arrays*, i.e. a parallel composition of independent copies of sequential components. An array can be specified as $P[n]$, where P denotes sequential component and n is an integer indicating the number of copies. Arrays are not just an abbreviation for the *cooperation* operator, they are considered in their canonical form [4], which may considerably reduce the state space size of the underlying Markov chain.

PEPAto has a module for static analysis, used for checking the well-formedness of a model and detecting potential problems prior to inferring the derivation graph of the system. Problems include unused declarations of rates and components, potential local deadlock, transient states of sequential components, unnecessary declaration of types in action sets of the *cooperation* and *hiding* operators, and self-containing processes, i.e. unguarded component uses giving rise to non-well-founded definition of processes.

Markovian analysis is supported by four tools. The *Derivator* generates the underlying Continuous Time Markov Chain of the model. The *Steady-state Analyser* calculates the long-run probability distribution over the state space. A number of solvers is supported, including a direct solver, the Conjugate Gradient, Bi-conjugate Gradient, and Generalised Minimum Residual methods. Then, the *Throughput Analyser* determines the action throughput in steady state and the *Utilisation Analyser* calculates the percentage of time each atomic component spends in each local state in steady state.

The *ODE Analyser* generates the system of ODEs underlying the continuous approximation of the state space of the model. It provides transient and, in the limit, steady state measures via computation of the time series of the compo-

0-7695-2883-X/07 $25.00 © 2007 IEEE
DOI 10.1109/QEST.2007.34

nents. Available solvers include the adaptive step-size fifth order Dormand-Prince solver and the IMEX stiff solver.

An alternative way to reason about performance evaluation of PEPA models is provided by the *Stochastic Simulation* module, which is based on Gillespie's and Gibson-Bruck's algorithms. The module has a number of options such as simulation time, number of replications, and components of interest, of which it provides time series evolution with confidence intervals. Detection of *rare events* is also supported.

3. The Eclipse User Interface

The Eclipse User Interface provides an editor as well as a number of *views* which collectively lay out in the *PEPA Perspective* of the Eclipse IDE. When the content of the edited file is changed, the description is parsed and static analysis is carried out. The resulting abstract syntax tree is shown in the *AST View* for debugging purposes. The messages from static analysis are shown in the Eclipse *Problems* view and markers are displayed in the editor site.

The plug-in contributes a top-level menu item which makes accessible all the PEPA tasks. As state space derivation may be a long-running operation, it is scheduled in a background thread which keeps the user interface responsive. The generated state space is shown in the *State Space View* which displays it in a tabular form, the table having as many columns as the number of sequential components of the models. An additional column is displayed when the steady state solution is obtained in order to show the long-run probability of each state. The view can also launch the *Single Step Explorer*, a tool for navigating the state space.

The State Space View features a novel user-friendly rule-based mechanism for filtering the state space. There are four rules available: (1) Filter states which have a sequential component in a given local state; (2) Filter states which have unnamed sequential components; (3) Filter states which have long-run probability above/below a given threshold (4) Filter states which match a given pattern. The rules can be grouped together and the boolean operators AND, OR, and NOT can be applied to them. A set of rules can be assigned a name and saved to disk for persistence across user sessions.

The *Graph View* is provided to show bar charts for action throughput, pie charts for component utilisation, and time series for stochastic simulation and ODE analysis. The charts can be exported into a number of formats such as PNG and SVG. Data serialisation is available via the CSV format.

A relevant tool is the *Sensitivity Analysis Tool* which lets the user determine the impact that changes in the values of the rates or the number of components in arrays have on performance metrics such as throughput of an action, utilisation of a particular local state, or overall probability of a subset of the state space as filtered by a given rule. The experiments are scheduled in a background thread and results are presented in the form of charts in the Graph View.

4. Related Work

PEPA is supported by a suite of tools, most notably the PEPA Workbench [5] and ipc—the Imperial PEPA Compiler [6]. Our contribution can be regarded as an enhancement over the former, which supports Markovian steady-state analysis only and does not feature static analysis tools. The ipc tool is a complementary solution because of its unique capability of providing passage time analysis.

5. Acknowledgement

The author would like to thank Stephen Gilmore for very valuable comments and suggestions on this work.

This work has been sponsored by the project SENSORIA, IST-2005-016004.

References

[1] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[2] J. Hillston. Fluid flow approximation of PEPA models. In *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems*, Torino, Italy, September 2005.

[3] Eclipse Foundation. Eclipse home page. `http://eclipse.org`.

[4] S. Gilmore, J. Hillston, and M. Ribaudo. An efficient algorithm for aggregating PEPA models. *IEEE Transactions on Software Engineering*, 27(5):449–464, May 2001.

[5] S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In *Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, May 1994.

[6] J.T. Bradley, N.J. Dingle, S.T. Gilmore, and W.J. Knottenbelt. Derivation of passage-time densities in PEPA models using IPC: The Imperial PEPA Compiler. In *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, October 2003.