Fourth International Conference on the Quantitative Evaluation of Systems

The ipclib PEPA Library

Allan Clark
Laboratory for Foundations of Computer Science
School of Informatics
University of Edinburgh

Abstract

PEPA[6] is a popular stochastic process algebra which allows a compositional approach to stochastic model description. The ipc compiler translates a given PEPA model into a format suitable for processing by the Hydra[5] Markovian response-time analyser. The ipc software has undergone some improvements which have led to its refactoring as a library for handling PEPA models.

1. Introduction

PEPA allows the compositional description of Markov Chains. The grammar for the standard description of PEPA components is given by:

$$P ::= (a, \lambda).P \mid P + P \mid P \underset{L}{\bowtie} P \mid P/L$$

A model is represented by a series of definitions which describe the sequential behaviour of named components. These named components are then combined together in a main system equation which represents the interaction between the various components in a model. Full details of the PEPA stochastic process algebra can be found in [6].

The HYpergraph-based Distributed Response Time Analyser or Hydra[5] is a software tool which may be used to compute transient distributions and response-time densities and quantiles in Markov chains with state spaces in the order of 10^7 states.

The ipc tool[2] compiles PEPA models to the descriptions of Markov chains accepted by Hydra. In addition ipc fully supports the calculation of apparent rate synchronisation as defined in [6]. This procedure is described in [3].

The ipc tool has been adapted to cooperate with the Condor[4] distributed computing platform. Many static-analyses of PEPA models have been incorporated into the ipc tool in order to avoid the needless solving of erroneous models. There are a number of transformations which take place within the compiler to prepare the model to be translated to a Markov chain. Also there are a number of new features on top of the vanilla PEPA models which can now be utilised. These include; process arrays, immediate actions and functional rates.

Current development of ipc has shown a trend towards more flexibility. In particular allowing analysis methods outside the realm of Hydra, for example stochastic simulation, as an alternative to Markovian analysis.

To enhance maintainability and the re-use of ipc software the code is to be reorganised as a PEPA library.

2. The ipclib library

Work on this library has begun and there are now several components. These include: A PEPA model parser module, a static-analysis module, a PEPA to LaTeX translation module and modules implementing tranformations over PEPA models including simplifications such as the removal of process arrays and the hiding operator using renaming. Additionally a stochastic probe[1] translator. This module also implements the combining of the translated probe with the model to be analysed.

The ipclib contains utility programs which operate over PEPA models. These programs utilise some part of the library to provide a command-line interface to the user. The pepaprobe utility is the most sophisticated of the tools developed thus far. The input is a PEPA model and a number of probes given as `--probe` options. These probes are performance measurement specification probes in the style of [1] and describe, using a high-level regular-expression like language, additional components to be combined with the input model. Such probes are intended to be observational and therefore do not change the original behaviour of the given model. The output is the input model transformed according to all of the given probes.

Two significant enhancements to the specification of probes have been made. The first change is that each probe may be specified as local to a given process within the whole system of the model. This means that the probe may observe actions from an individual component as being distinct from those same actions performed by other compo-

0-7695-2883-X/07 $25.00 © 2007 IEEE
DOI 10.1109/QEST.2007.20

nents in the model.

The second change is to labels. Originally labels could only be either `start` or `stop` to specify that the probe has entered or exited a passage to be measured. These were essentially communication messages passed from the measurement or observation probe to a control or master probe. Since such untimed communication is not available in standard PEPA, these `start` and `stop` signals were previously implemented via renaming. With the addition of immediate actions, untimed communication is possible which simplifies the translation of labelled actions. Because of this the labels are generalised to be any name the user wishes. These can then be used as communication signals between user defined probes. In addition labels may be attached to the end of any probe rather than attached to a specific activity.

As an example the probe:
$Client1\ (a, b, a, b : start), c : stop$ defines a probe which will be attached to a $Client1$ component. The probe waits for a sequence of a, b, a, b actions before sending a $start$ signal. After this if the probe observes a c action it sends a $stop$ signal (and returns to the original state).

The *without* operator (R/a) allows the probe to be reset to a given position. It means that the probe expects to observe a sequence of activities corresponding to the probe R without at any time witnessing the excluded action a. The probe: $(a, a, a)/b : start, b : stop$ will wait until it observes three a activities without observing a b activity before sending the $start$ communication signal.

Another utility program is pepacheck. This program runs the various static program analyses over the PEPA model input files. These analyses check the model for constructs which are either unsuitable for compilation to a CTMC or are likely to indicate a mistake on the part of the the modeller. These include, but are not limited to; the use of an undefined process name, a process which is defined but not used and a cooperation over a set of actions in which one or both sides do not perform all of the actions.

This could be called externally as part of another PEPA tool or used to periodically check a library of PEPA models.

The pepalatex tool accepts as input a PEPA model and outputs a translation into LaTeX format.

3. Conclusions and Future Work

The re-organisation of the core of the ipc code into reusable library modules has inabled considerable simplification of the code and interfaces between separate tasks. This has increased the maintainability of the code and its fitness to accomodate further enhancements. This has so far been highlighted with the addition of the full probe specfication language and indeed enhancements to both the semantics for probes and their user descriptions. This is an original contribution which provides a superset of the features offered by an earlier (distinct) implementation of stochastic probes due to Ashok Argent-Katwala.

In the immediate future the remainder of the ipc code will be ported into ipclib. In the longer-term the modules of ipclib should prove useful in the further development of PEPA tools such as editor environments, teaching aids and the generation of a PEPA component library.

The code for ipclib may be downloaded from: `http://homepages.inf.ed.ac.uk/s9810217/software/`.

4. Acknowledgements

This work has been sponsored by the project SENSORIA, IST-2005-016004. Stephen Gilmore has participated in countless helpful discussions. Jeremy Bradley first created the ipc software tool and has since provided many helpful insights into its workings.

References

[1] A. Argent-Katwala, J. Bradley, and N. Dingle. Expressing performance requirements using regular expressions to specify stochastic probes over process algebra models. In *Proceedings of the Fourth International Workshop on Software and Performance*, pages 49–58, Redwood Shores, California, USA, Jan. 2004. ACM Press.

[2] J. Bradley, N. Dingle, S. Gilmore, and W. Knottenbelt. Derivation of passage-time densities in PEPA models using IPC: The Imperial PEPA Compiler. In G. Kotsis, editor, *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, pages 344–351, University of Central Florida, Oct. 2003. IEEE Computer Society Press.

[3] J. Bradley, N. Dingle, S. Gilmore, and W. Knottenbelt. Extracting passage times from PEPA models with the HYDRA tool: A case study. In Jarvis [7], pages 79–90.

[4] A. Clark and S. Gilmore. Evaluating quality of service for service level agreements. In L. Brim and M. Leucker, editors, *Proceedings of the 11th International Workshop on Formal Methods for Industrial Critical Systems*, pages 172–185, Bonn, Germany, Aug. 2006.

[5] N. J. Dingle, W. J. Knottenbelt, and P. G. Harrison. HYDRA: HYpergraph-based Distributed Response-time Analyser . In H. R. Arabnia and Y. Man, editors, *PDPTA'03, Proceedings of the 2003 International Conference on Parallel and Distributed Processing Techniques and Applications*, volume 1, pages 215–219, Las Vegas, NV, June 2003.

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

[7] S. Jarvis, editor. *Proceedings of the Ninteenth UK Performance Engineering Workshop*, University of Warwick, July 2003.