Abstraction and Model Checking in the PEPA Plug-in for Eclipse

Michael J. A. Smith [†]
Department of Informatics and Mathematical Modelling
Danmarks Tekniske Universitet
Lyngby, Denmark
Email: mjas@imm.dtu.dk

***Abstract*—The stochastic process algebra PEPA is widely used for performance modelling, and a large part of its success is due to its rich tool support. As a compositional Markovian formalism, however, it suffers from the state space explosion problem, where even small models can lead to very large Markov chains. One way of analysing such models is to use abstraction — constructing a smaller model that bounds the properties of the original.**

We present an extension to the PEPA plug-in for Eclipse that enables abstracting and model checking of PEPA models. This implements two new features. The abstraction view provides a graphical interface for labelling and aggregating states of individual PEPA components. The model checking view provides an interface for constructing CSL properties, which are then verified with respect to the specified abstraction. We have an internal CSL model checker for CTMDPs, so the tool can be used as a stand-alone.

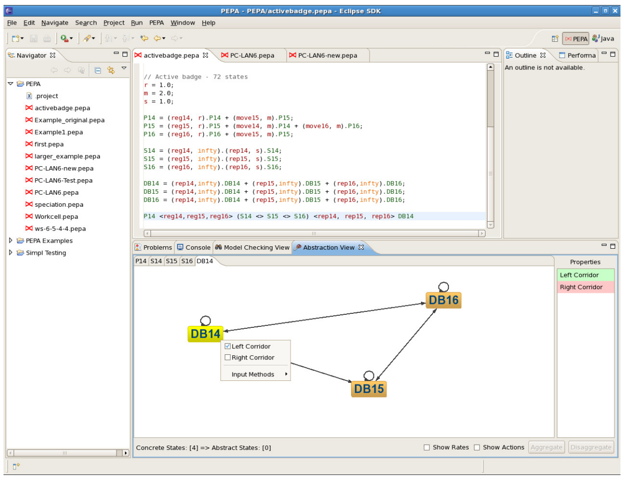
I. Introduction

The Performance Evaluation Process Algebra (PEPA) [5] is a language for modelling systems in which a number of interacting components run in parallel, and whose behaviour is stochastic. The core semantics of PEPA is in terms of Continuous Time Markov Chains (CTMCs), and an alternative semantics in terms of Ordinary Differential Equations (ODEs) has also been developed. PEPA has been applied in practice to a wide variety of systems, and its success as a modelling language has been largely down to its extensive tool support. Most recently, the *PEPA Plug-in Project* [11] has integrated a range of analysis techniques — based on both numerical solution and simulation — into a single tool built on top of the Eclipse platform [1].

As with all compositional Markovian formalisms, however, PEPA suffers from the state space explosion problem. A model can have an underlying state space that is exponentially larger than its description, meaning that it can be infeasible to analyse. Fluid flow approximation using PEPA's ODE semantics can solve this problem if we are only interested in the *average* behaviour of the system over time. However, if we want to reason over *all* possible behaviours of the model — for example, the probability that an error occurs withing some time interval — then we must consider the CTMC semantics.

In this paper, we present a new extension to the PEPA plug-in, in which a model can be *abstracted* by combining, or aggregating, states. To safely over-approximate the behaviour

[†] This work was funded by a Microsoft Research European Scholarship.



```
// Active badge - 72 states
r = 1.0;
m = 2.0;
s = 1.0;

P14 = (reg14, r).P14 + (move15, m).P15;
P15 = (reg15, r).P15 + (move14, m).P14 + (move16, m).P16;
P16 = (reg16, r).P16 + (move15, m).P15;

S14 = (reg14, infty).(rep14, s).S14;
S15 = (reg15, infty).(rep15, s).S15;
S16 = (reg16, infty).(rep16, s).S16;

DB14 = (rep14,infty).DB14 + (rep15,infty).DB15 + (rep16,infty).DB16;
DB15 = (rep14,infty).DB14 + (rep15,infty).DB15 + (rep16,infty).DB16;
DB16 = (rep14,infty).DB14 + (rep15,infty).DB15 + (rep16,infty).DB16;

P14 <reg14,reg15,reg16> (S14 <> S15 <> S16) <rep14, rep15, rep16> DB14
```

Fig. 1. The PEPA plug-in, showing the abstraction interface

of the original model (for any aggregation of its states), we use two abstraction techniques — *abstract CTMCs* [9] (a type of Markov decision process with infinite branching), and *stochastic bounds* [4]. We provide a model checker for the three-valued Continuous Stochastic Logic (CSL) [2], which computes from the abstraction a *safe bound* of the probability of a quantitative property holding in the original model — if the actual probability is p, then the model checker will return an interval $I = [p_1, p_2]$ such that $p \in I$.

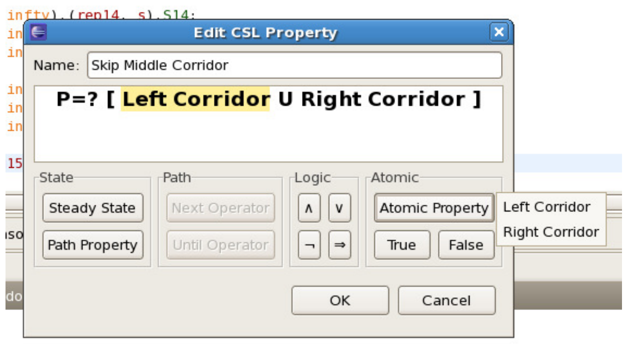
The current version of the PEPA plug-in can be downloaded from `http://www.dcs.ed.ac.uk/pepa/tools/plugin`, and provides two new views:

1) The **Abstraction View** is a graphical interface that shows the state space of each sequential component in a PEPA model. It provides a facility for labelling states (so that they can be referred to in CSL properties), and for specifying which states to aggregate.
2) The **Model Checking View** is an interface for constructing, editing, and model checking CSL properties. The property editor provides a simple way to construct CSL formulae, by referencing the labels given to states in the abstraction view. It ensures that only syntactically well-formed CSL formulae can be constructed.

Figure 1 shows a screenshot of the abstraction view in use.

II. Abstraction and Model Checking of PEPA

Our motivation behind the abstraction view is to allow the modeller to experiment with different aggregations of states in a fast and straightforward way. Since we support *compositional* abstractions, and the number of states in a single sequential PEPA component is usually small, it is possible to do so graphically. The user simply selects a number of states in a component, and clicks the “aggregate” button. The states can be separated again by clicking “disaggregate”. The same graphical interface is also used to compositionally label sets of states in the model.


Name: Skip Middle Corridor

P=? [Left Corridor U Right Corridor]

Fig. 2. The CSL property editor

To construct CSL formulae, we provide an editor, which is illustrated in Figure 2. Labels from the abstraction view are made available as atomic properties, and we support both the path and steady state CSL operators[1]. Once a property is specified, it can be verified in the model checking view, using the currently specified abstraction. If the result is not precise enough, a different abstraction can be constructed, and the property checked again. Importantly, the model checker always returns a *safe* probability interval in the case of quantitative properties, and a *safe* answer of ‘true’, ‘false’, or ‘maybe’ for Boolean properties, with respect to the original model.

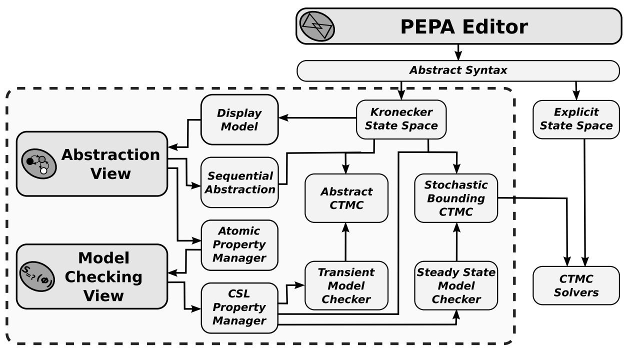


Fig. 3. Architecture of the abstraction and model checking features

Figure 3 shows an outline of the architecture of part of the PEPA plug-in, with our extension shown within the dashed box. At the heart of our implementation, we use a Kronecker representation for PEPA models, similar to that described in [6]. We have developed a compositional application of abstract Markov chains, in the context of PEPA models, which we use for verifying CSL path formulae. Our model checker is based on the algorithm in [3], for computing time-bounded reachability properties of uniform Continuous Time Markov Decision Processes (CTMDPs). For CSL steady state formulae, we generate lumpable and stochastic bounding PEPA models compositionally [10], which are then solved using existing CTMC solvers.

III. Conclusions

We have presented an extension to the PEPA plug-in for Eclipse, which allows compositional abstraction and CSL model checking of PEPA models. To put this in the context of other tools, PRISM [7] supports CSL model checking of (non-abstracted) PEPA models[2], and MRMC [8] supports model checking of the CSL time-bounded until operator on CTMDPs, but not the abstraction of higher-level formalisms.

In the near future, we plan to extend our tool with support for the derived CSL operators (F, G, etc.) and the PEPA aggregate combinator, and to provide the ability to export to MRMC. The biggest limitation at present is that properties and abstractions cannot be saved and loaded alongside PEPA models, and we hope to implement this functionality soon.

Acknowledgements

We would like to thank Jane Hillston and Stephen Gilmore for their feedback and support while developing the tool, and Mirco Tribastone for his advice on Eclipse development.

References

[1] The Eclipse platform. `http://www.eclipse.org`.
[2] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Proceedings of CAV ’00*, volume 1855 of *LNCS*, pages 358–372. Springer, 2000.
[3] C. Baier, H. Hermanns, J.-P. Katoen, and B.R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theoretical Computer Science*, 345(1):2–26, 2005.
[4] J.-M. Fourneau, M. Lecoz, and F. Quessette. Algorithms for an irreducible and lumpable strong stochastic bound. *Linear Algebra and its Applications*, 386:167–185, 2004.
[5] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
[6] J. Hillston and L. Kloul. An efficient Kronecker representation for PEPA models. In *Proceedings of PAPM-PROBMIV ’01*, volume 2165 of *LNCS*, pages 120–135. Springer, 2001.
[7] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proceedings of TACAS ’06*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.
[8] J.-P. Katoen, M. Khattri, and I.S. Zapreevt. A Markov reward model checker. In *Proceedings of QEST ’05*, pages 243–244. IEEE Computer Society, 2005.
[9] J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. Three-valued abstraction for continuous-time Markov chains. In *Proceedings of CAV ’07*, volume 4590 of *LNCS*, pages 316–329. Springer, 2007.
[10] M.J.A. Smith. Compositional stochastic bounding of PEPA models, 2010. In submission. Available from: `http://lanther.co.uk/papers/PEPA_abstraction.pdf`.
[11] M. Tribastone. The PEPA plug-in project. In *Proceedings of QEST ’07*, pages 53–54. IEEE Computer Society Press, 2007.

[1]Note that we do not support the time-bounded next operator.

[2]PRISM does not support PEPA’s active-active synchronisation.