Slaying vampires: Improved static analysis for IPC

Stephen Gilmore*

June 3rd, 2004

Abstract

The Imperial PEPA Compiler (IPC) [BDGK03] provides PEPA modellers with model analysis capabilities which were unavailable in the previous generation of PEPA tools. IPC extends PEPA modelling capabilities both in kind (passage-time analysis) and dimension (orders of magnitude increase in model state space size). In this paper we report on improved error detection and reporting in the implementation of IPC.

1 Introduction

A swift blow with a hammer will drive a wooden stake into the heart of a vampire and slay him as he slumbers in his coffin. Errors in performance models should be dealt with just as swiftly and as brutally. Catching errors early in the performance modelling process means that the modeller does not waste time trying to analyse an incorrect model or, worse, trying to interpret the analysis results obtained.

As the name suggests, IPC is a compiler. Detecting errors in the input to a compiler is the responsibility of its *parser*, a component of the front end which accepts source files, performs lexical analysis, grammar checking, static analysis and syntax tree building. Unfortunately the PEPA parser used in early releases of IPC is too lax, passing on to the back end inputs which should have been rejected as being ill-formed. This is not a good idea because the back end cannot report errors in terms of the input model (because it never sees the input directly, only via the filtered representation in the abstract syntax tree).

What constitutes an error in a PEPA model? Some are unambiguous; a model which fails to parse according to the PEPA grammar is obviously to be rejected. What do we do about the following?

*Laboratory for Foundations of Computer Science, The University of Edinburgh, Edinburgh EH9 3JZ, Scotland. Email: `stg@inf.ed.ac.uk`

1. Components which are defined in the model file but not used in the PEPA system equation.
2. Cooperation sets which include redundant activities, not performed by either the left or right co-operand.

It might be possible to argue that the above are quite reasonable uses of the PEPA language as follows.

1. Components which are defined but not used have no effect on the model and are presumably included in the file as a library of model definitions to be used in some models and not others.
2. Redundant activities are not harmful, if removed from a coopertion set they would not change the statespace or transition system generated from the model.

It is also possible to argue more pessimistically about these cases.

1. Components which are defined but not used indicate that something has been missed out of the system equation.
2. Redundant activities in cooperation sets have probably been included in the wrong set and should be somewhere else in the model, where they would materially change its behaviour.

The forthcoming "Peter Cushing" release of IPC—named after the actor who gave the definitive performance of the vampire slayer Van Helsing—will treat these cases as errors in contrast to the previous "James Clerk Maxwell" and "Jonny Wilkinson" releases which permitted all of them.

2 Improved error reporting in IPC

While working on better static analysis for IPC we took the opportunity to improve the diagnostic error reporting from the parser which, following the model of the ML edition of the PEPA Workbench [GH94], was thoroughly wretched. Poor syntax error reporting and diagnostic error messages were perhaps forgiveable when the limited power of the PEPA Workbench tool meant that only small model inputs could be accepted. In contrast the PEPA models processed by IPC can be sizeable and so vague error messages will impede the productive use of the tool because the modeller will spend too much time trying to track down the missing semicolon or extra commas in the input model. In addition it is more common to find larger PEPA models now because they are generated from models in other languages such as PEPA nets [GHRK03] or from tools such as AMoGeT [BCGH04].

As an example of the improved error reporting we consider the following flawed PEPA input model, with a syntax error caused by omiting the full stop character used to separate an activity prefix from the process term which follows it.

```
% File: tests/bad/test51.pepa
% Syntax error, missing full stop in prefix
P = (a, r)P;
(P <> P)
```

When presented with this flawed input model the previous releases of IPC would report the error thus:

```
** The Imperial PEPA compiler:
ipc v1.00-rc1 101203 ("Jonny Wilkinson" Release)
DNAmaca generator: Jeremy Bradley; PEPA parser: Stephen Gilmore
Incorporating Sven Panne's port of the GNU GetOpt library
Produced on location in/between Edinburgh, Leeds, London and Ludlow
Mail queries and problems to: jb@doc.ic.ac.uk

Compiling: test51.pepa -> test51.mod

Fail: PepaParser: Unparsed :-
 = (a, r)P;
        (P <> P)
```

In constrast the Peter Cushing release of IPC reports the above error thus:

```
** The Imperial PEPA compiler:
ipc v1.01-rc0 260504 ("Peter Cushing" Release)
DNAmaca generator: Jeremy Bradley; PEPA parser: Stephen Gilmore
Incorporating Sven Panne's port of the GNU GetOpt library
Produced on location in/between Edinburgh, Leeds, London and Ludlow
Mail queries and problems to: jb@doc.ic.ac.uk

Compiling: test51.pepa -> test51.mod

Fail:  at line 3, column 10 of PEPA input file
! P = (a, r)P;
!          ^
! PepaParser: Period expected between closing bracket and 'P'
```


Acknowledgements

Stephen Gilmore is supported by the DEGAS (Design Environments for Global ApplicationS) project IST-2001-32072 funded by the FET Proactive Initiative on Global Computing and by the ENHANCE (Enhancing the Performance Predictability of Grid Applications with Patterns and Process Algebras) project funded by the UK Engineering and Physical Sciences Research council under grant number GR/S21717/01.

References

[BCGH04] A. Benoit, M. Cole, S. Gilmore, and J. Hillston. Evaluating the performances of skeleton-based high level parallel programs. In M. Bubak, D. van Albada, P. Sloot, and J. Dongarra, editors, *The International Conference on Computational Science (ICCS 2004)*, LNCS, pages 299–306. Springer Verlag, 2004.

[BDGK03] 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 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, October 2003. IEEE Computer Society Press.

[GH94] 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*, number 794 in Lecture Notes in Computer Science, pages 353–368, Vienna, May 1994. Springer-Verlag.

[GHRK03] S. Gilmore, J. Hillston, M. Ribaudo, and L. Kloul. PEPA nets: A structured performance modelling formalism. *Performance Evaluation*, 54(2):79–104, October 2003.