Parsing and Validation of full CSP-M Specifications using Haskell and Prolog (Abstract)*

Marc Fontaine and Michael Leuschel
Heinrich-Heine Universität Düsseldorf
Universitätsstr. 1, D-40225 Düsseldorf
{fontaine,leuschel}@cs.uni-duesseldorf.de

CSP is a process algebra defined by Hoare. The first semantics associated with CSP was a denotational semantics in terms of traces, failures and divergences. Later an operational semantics was added [7]. CSP has been applied in many applications, notably for security protocols [6].

The most widely used tools today are FDR [2] and ProBE [3]. These tools use a syntax called machine readable CSP (CSP-M) which is the combination of Core-CSP with a rudimentary functional language.

In earlier work we presented CIA [4] a CSP parser and interpreter that supported full Core-CSP and some of the CSP-M extensions. CIA is written in Prolog which made it easy to integrate into the ProB[1] animator and model checker [5], making it the first tool for model-checking combined CSP and B specifications [1].

Our new work is motivated by an industrial application which demands support for full CSP-M syntax, linked with B specifications. Furthermore, an important requirement of our industrial partner was compatibility with the FDR and Probe tools, while at the same time fixing some shortcomings (e.g., the fact that the main process as well as all of its subprocesses in isolation have to be finite state) and obvious bugs of those tools.

This is challenging as the semantics of CSP-M is more or less implicitly defined by its implementations in FDR and ProBe. Although, Scattergood [8] describes a formal semantics of CSP-M, there still is a considerable gap between this formal semantics and what is implemented in FDR and ProBE.

Other motivations for our re-implementation of CSP-M are for example the desire to:

- make it possible to link full CSP-M specifications with B [1] for property validation
- make it possible to generate state-space graphs or gather statistics for CSP-M models
- explore the state-space of a CSP-Process lazily and be able to animate infinite state CSP-M specifications.

Each of these proved to be difficult to achieve with the existing tools. In this talk we will describe:

- a new parser for CSP-M, implemented in Haskell using combinator parsing,
- a type checker for CSP-M, implemented by compiling CSP-M into (non-executable) Haskell and using the Haskell type checker to detect typing errors in CSP-M specifications,
- an animator and model checker for CSP-M, which is a complete re-implementation of [4] in Prolog, with an additional pre-compilation phase to efficiently and correctly handle nested let-expressions and other CSP-M constructs which introduce local variables,
- empirical results, showing that our tool is, for some specifications at least, much faster and more robust than the existing ProBE and FDR tools.

[1] ProB is a completely different tool than ProBE.

Submitted to Programmiersprachen und Rechenkonzepte, 2007, Bad Honnef.

1. REFERENCES

[1] M. Butler and M. Leuschel. Combining CSP and B for specification and property verification. In *Proceedings of Formal Methods 2005*, LNCS 3582, pages 221–236, Newcastle upon Tyne, 2005. Springer-Verlag.

[2] Formal Systems (Europe) Ltd. *Failures-Divergence Refinement — FDR2 User Manual.*

[3] Formal Systems (Europe) Ltd. *Process Behaviour Explorer (ProBE User Manual).* available at `http://www.fsel.com/probe_manual.html`.

[4] M. Leuschel. Design and implementation of the high-level specification language CSP(LP) in Prolog. In I. V. Ramakrishnan, editor, *Proceedings of PADL'01*, LNCS 1990, pages 14–28. Springer-Verlag, March 2001.

[5] M. Leuschel and M. Butler. ProB: A model checker for B. In K. Araki, S. Gnesi, and D. Mandrioli, editors, *FME 2003: Formal Methods*, LNCS 2805, pages 855–874. Springer-Verlag, 2003.

[6] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *IEEE Symposium on Foundations of Secure Systems*, 1995.

[7] A. W. Roscoe. *The Theory and Practice of Concurrency.* Prentice-Hall, 1999.

[8] J. B. Scattergood. *Tools for CSP and Timed-CSP.* PhD thesis, Oxford University, 1997.