ALGORITHMIC PROOF GENERATION FOR CSP-CASL-PROVER

Liam O'Reilly

At the last BCTCS we suggested an architecture for a theorem prover for the language Csp-Casl [4]. This tool is based on the interactive theorem prover Isabelle/HOL [2]. Csp-Casl integrates the process algebra Csp [5] with the algebraic specification language Casl [1], forming a specification language tailored to the description of distributed systems.

Meanwhile in [3], we have realised this architecture to the point that we have algorithms which produce the theorems as well as the proof scripts needed in Csp-Casl-Prover. Tests in various scenarios including a proper industrial setting demonstrate that our architecture is feasible.

In this talk we will report these results and discuss alternatives to our original implementation strategy, namely instead of starting from a shallow encoding of CASL in Isabelle/HOL to start from a semi-deep encoding, or even a deep encoding of Casl.

References

[1] P. D. Mosses, editor. Casl *Reference Manual*. LNCS 2960. Springer, 2004.

[2] Tobias Nipkow, Lawrence C. Paulon, and Markus Wenzel. *Isabelle/HOL*. LNCS 2283. Springer, 2002.

[3] Liam O'Reilly, Yoshinao Isobe, and Markus Roggenbach. Integrating Theorem Proving for Processes and Data. In Magne Haveraaen, John Power, and Monika Seisenberger, editors, *CALCO-jnr 2007*. University of Bergen, to appear.

[4] Markus Roggenbach. CSP-CASL - a new integration of process algebra and algebraic specification. *Theoretical Computer Science*, 354(1):42–71, 2006.

[5] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1998.

Liam O'Reilly
Department of Computer Science, Swansea University, Swansea, UK
csliam@swansea.ac.uk