Adding Formal Verification to occam-π

Peter H. WELCH[a], Jan B. PEDERSEN[b], Fred R.M. BARNES[a],
Carl G. RITSON[a] and Neil C.C. BROWN[a]

[a] *School of Computing, University of Kent, UK*
[b] *School of Computer Science, UNLV, USA*

phw@kent.ac.uk, matt@cs.unlv.edu, frmb@kent.ac.uk, cgr@kent.ac.uk, nccb@kent.ac.uk

Abstract. This is a proposal for the formal verification of occam-π programs to be managed entirely within occam-π. The language is extended with qualifiers on types and processes (to indicate relevance for verification and/or execution) and assertions about refinement (including deadlock, livelock and determinism). The compiler abstracts a set of CSP_m equations and assertions, delegates their analysis to the FDR2 model checker and reports back in terms related to the occam-π source. The rules for mapping the extended occam-π to CSP_m are given. The full range of CSP_m assertions is accessible, with no knowledge of CSP formalism required by the occam-π programmer. Programs are proved just by *writing* and *compiling* programs. A case-study analysing a new (and elegant) solution to the *Dining Philosophers* problem is presented. Deadlock-freedom for colleges with *any* number of philosphers is established by verifying an induction argument (the base and induction steps). Finally, following guidelines laid down by Roscoe, the careful use of *model compression* is demonstrated to verify directly the deadlock-freedom of an occam-π college with 10^{2000} philosphers (in around 30 seconds). All we need is a universe large enough to contain the computer on which to run it.

Keywords. concurrency, formal verification, model checking, occam-pi, CSP, FDR.