P.H.Welch and Fred Barnes

Communicating Processes, Safety and Dynamics: the New **occam**

Abstract

This presentation reports continuing research on language design, compilation and kernel support for highly dynamic concurrent reactive systems. The work extends the **occam** multiprocessing language, which is both sufficiently small to allow for easy experimentation and sufficiently powerful to yield results that are *directly* applicable to a wide range of industrial and commercial practice.

Classical **occam** was designed for embedded systems and enforced a number of constraints – such as statically pre-determined memory allocation and concurrency limits – that were relevant to that generation of application and hardware technology. Most of these constraints have been removed in this work and a number of new facilities introduced (*channel structures, mobile channels, channel ends, dynamic process creation, the extended rendezvous and process priorities*). These significantly broaden **occam**'s field of application and raise the level of concurrent system design directly supported. Four principles were set for modifications/enhancements of the language. They must be useful and easy to use. They must be semantically sound and policed (ideally, at compile-time) to prevent misuse. They must have very lightweight and fast implementation. Finally, they must be aligned with the concurrency model of the original core language (i.e. they must not damage its safety and must not add significantly to the ultra-low overheads for management of that concurrency). These principles have all been observed.

The presentation gives several examples to illustrate the new capabilities and show how safety (e.g. against race hazards) has been preserved in this new dynamic and mobile environment. Micro-benchmark figures for overheads are reported that remain (well) below 100 nanoseconds on an 800 MHz. Pentium III. Other examples are taken from projects testing out these mechanisms in more complex applications (such as the **occWeb** webserver and the *Raw Metal occam eXperiment*, **RMoX**).

Also mentioned are a number of other additions and extensions to the **occam** language that correct, tidy up or complete facilities that have long existed. These include fixing the **PRI ALT** bug, allowing an unconditional **SKIP** guard as the last in a **PRI ALT**, replicator **SKIP** sizes, run-time computed **PAR** replication counts, **RESULT** parameters and abbreviations, nested **PROTOCOL** definitions, in-line array constructors and parallel recursion. All these enhancements are available in the latest release (1.3.3) of **KRoC**, freely available (GPL/open source) from:

www.cs.ukc.ac.uk/projects/ofa/kroc/

Commercial support is available via:

www.quickstone.com