
Downloaded from https://kar.kent.ac.uk/14440/ The University of Kent's Academic Repository KAR

The version of record is available from

This document version UNSPECIFIED

DOI for this version UNSPECIFIED

Licence for this version UNSPECIFIED

Additional information

Versions of research works

Versions of Record
If this version is the version of record, it is the same as the published version available on the publisher's web site. Cite as the published version.

Author Accepted Manuscripts
If this document is identified as the Author Accepted Manuscript it is the version after peer review but before type setting, copy editing or publisher branding. Cite as Surname, Initial. (Year) 'Title of article'. To be published in Title of Journal, Volume and issue numbers [peer-reviewed accepted version]. Available at: DOI or URL (Accessed: date).

Enquiries
If you have questions about this document contact ResearchSupport@kent.ac.uk. Please include the URL of the record in KAR. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies).
Communicating Complex Systems

Technical paper for ICECCS 2006

Peter H. Welch, Frederick R.M. Barnes and Fiona A. Polack

P.H. Welch and F.R.M. Barnes are members of the Computing Laboratory, University of Kent, Canterbury, Kent, CT2 7NF, England. email: {phw,frmb}@kent.ac.uk.
F.A. Polack is a member of the Department of Computer Science, University of York, Heslington, York, YO10 5DD, England. email: fiona@cs.york.ac.uk.

Abstract: This paper outlines a new language and run-time technology offering formalised design and efficient implementation for highly concurrent dynamic systems. It is based on a careful combination of ideas from Hoare’s CSP (giving compositional semantics, refinement and safety/liveness analysis) and Milners’s pi-calculus (giving dynamic network construction and mobility). We have been experimenting with systems developing as layered networks of self-organising neighbourhood-aware communicating processes, with no need for advanced planning or centralised control. The work reported is part of our TUNA (‘Theories Undepinning Nanite Assemblies’) project, in partnership with colleagues from the Universities of York and Surry, which is investigating formal approaches to the capture of safe emergent behaviour in highly complex systems. A particular study modelling artificial blood platelets is described. The software technology scales to millions of processes per processor and distributes over common multiprocessor clusters.

Keywords: concurrency, modelling, emergent behaviour, safety, occam-pi, CSP, pi-calculus
Communicating Complex Systems

Peter H. Welch, Frederick R.M. Barnes and Fiona A. Polack

Abstract—This paper outlines a new language and run-time technology offering formalised design and efficient implementation for highly concurrent dynamic systems. It is based on a careful combination of ideas from Hoare’s CSP (giving compositional semantics, refinement and safety/liveness analysis) and Milner’s π-calculus (giving dynamic network construction and mobility). We have been experimenting with systems developing as layered networks of self-organising neighbourhood-aware communicating processes, with no need for advanced planning or centralised control. The work reported is part of our TUNA (‘Theories Undepinning Nanite Assemblies’) project, in partnership with colleagues from the Universities of York and Surry, which is investigating formal approaches to the capture of safe emergent behaviour in highly complex systems. A particular study modelling artificial blood platelets is described. The software technology scales to millions of processes per processor and distributes over common multi-processor clusters.

Index Terms—concurrency, modelling, emergent behaviour, safety, occam-π, CSP, π-calculus

I. INTRODUCTION AND MOTIVATION

... to be completed ...

II. OVERVIEW OF OCCAM-π

Software systems programmed in occam-π are built as layered networks of communicating processes. Processes in occam-π are self-contained, interacting with their external environment through communication channels and a few other synchronisation mechanisms (all built from CSP events). Combined with strict parallel-usage and alias checks performed by the compiler, this gives occam-π programs strong safety guarantees — that can be established formally using Hoare’s CSP [2].

The following subsections provide an overview of the different occam-π mechanisms relevant to this work. The language encompasses much more, however, and we give some hints ...

A. Processes, channels and networks

... to be completed ...

B. Mobile channels, processes and barriers

... to be completed ...

How we use the occam-π mechanisms to build simulations – the model of self-contained neighbourhood-aware autonomous agents that roam around a virtual world we create for them. The facilities in occam-π allow us to do interesting things, like changing the ‘world’ dynamically, including lazy simulation [3]. How we use mobile barriers to time-step the simulation, as well as providing fairly cheap support for visualisation (phased barriers) [4]. Elaborate on how directly such design mirrors the low-level layers of processes (e.g. platelets, chemicals, physical space) in nature and their low-level interactions ... with benefits on the reality of the modelling and its efficiency (processor activity only to support active processes and their immediate environment).

Then talk about the blood-clot simulation and/or implementation of Steve’s CSP models.

III. SIMULATING COMPLEX SYSTEMS

... to be completed ...

IV. FORMAL DESIGN AND ANALYSIS

... to be completed ...

Something about basic processes, channels and networks. Some integrators in figure 1. Maybe this can be assumed?

Fig. 1. Serial and parallel integrators

B. Mobile channels, processes and barriers

... to be completed ...

How we use mobile channels (for dynamic construction of process networks), mobile processes (for distributing computation) and mobile barriers (for N-way synchronisation).

C. Dynamic process creation

... to be completed ...

Something about FORKING and FORK (for dynamic growing and shrinking of systems).

IV. FORMAL DESIGN AND ANALYSIS

... to be completed ...

Something about the initial design of CSP/B models and their safety/liveness analysis using model checkers. How
these are turned into executable occam-pi (or JCSP) systems.

Something on the CSP models we have for mobile barriers and channels?

V. Conclusions and Future Work

... to be completed ...

What we’ve done and where we’re going.

References


