Tool support for the design/development of concurrent programs
Computing Science – School of Computing Science

Overview

The aim of this research is to design and provide tool support for the design of software that executes concurrently. Before the tool support itself can begin, there is significant research to be done to choose the best notation(s) and method(s) that will be supported.

Society’s reliance on technology requires efficient and effective delivery of services; concurrent programs unlock the power of multi-core hardware but this can be at the expense of programming errors being introduced leading to expensive or even fatal outcomes. Concurrent programs are becoming increasingly important because the “end of Moore’s law” is leading to ever more processors on a chip and the need to write programs that are concurrent. Such programs are extremely difficult to get right and formal approaches are becoming recognised as key, but their adoption is retarded because of the lack of tool support.

Cliff Jones’ recent research is on concurrency. Chin Wei Ngan is an expert on tool support for concurrency. Leo Freitas has a wide range of expertise with theorem proving tools such as Isabelle and Z-Eves. He has collaborated with Jones (and AI researchers in Edinburgh) on the “AI4FM” project; a book on the outcomes of the project is in preparation.

Jones has met Chin Wei Ngan and discussed concurrency research on two visits to Singapore.

Methodology

There are two major formal directions to the design of shared-variable concurrent programs: Separation Logic (SL) and Rely/Guarantee (R/G). The later was proposed by Jones (two original R/G references each have over 500 citations) and the paper with Yatapanage builds key links with SL.

Recent research with Hayes (Queensland) has yielded an algebraic version of R/G more suitable for mechanisation that enables machine-assisted derivation of correct concurrent algorithms. Freitas is an expert tool builder and has a PhD student who has looked at establishing some of the key laws of the algebraic R/G; Hayes has supervised an Australian honours project on a further generalisation. The time is ripe to look at the mechanisation of program development steps using the new approach. (Jones has an EPSRC grant Taming Concurrency and it currently hosting a six-month sabbatical by Hayes who leads an ARC project along similar lines.)
Wei Ngan’s expertise is in mechanisation of verification processes. His team has built an automatic verification framework, (HIP/SLEEK) which supports automated reasoning of inductive predicates and user-supplied lemmas. This framework is leveraged on state-of-the-art theorem prover/solvers, such as Z3, Omega and Coq. One goal is to explore the limits of proof mechanization for concurrent programming.

The student will need to use theorem proving systems to develop concurrent programming tools needed by end users. This will involve:

- Specialist training in the proof tools
- Training with libraries related to concurrency verification, such as R/G algebraic laws
- Development of front-end tools for end users unskilled with theorem provers to employ the concurrent program derivation facilities

**Timeline**

M1-6: Learn about formal techniques: algebraic R/G, SL, refinement calculus, etc.

M6-18: Training in theorem proving systems is notoriously hard. In the first year report, the student should be able to provide a survey of the use of provers for R/G and concurrent verification. This should be of interest to a conference on the subject. The first visit to NUS will involve a review of their (then) current tools; at least one Newcastle academic will accompany the student.

M18-24: Development of available proof libraries to be used by concurrent programming experts not familiar with proof tools. This should result in submission to the prestigious archive of formal proofs as well as publication. The second visit to NUS will involve presentations on work to date.

M24-30: Creation of a repository of programming patterns and examples amenable to formal verification within the proof libraries available. The second year report will include detailed examples of the use of such patterns.

M30-36: Proof engineering for higher levels of automation. This will include auxiliary lemmas, generic proof methods, automation, etc. The third visit will be extended to expose the structure of the planned thesis and get feedback thereon.

M36-48: Write up and submission. All members of the supervisory team will review the thesis.

**Training & Skills**

1. Regular (initially weekly) supervisory meetings with both Newcastle academics – giving access to tool building experts in both countries. We will bring NUS into discussions via Skype once a month.
2. Training in semantics and R/G (Jones’ and Freitas’ UG course).
3. Visits to Wei Ngan’s team in Singapore at least once per year.
4. The student will be sent to conferences and workshops to learn the state of the art.

**References & Further Reading**

The link between R/G and SL is explored in:


The new algebraic approach to R/G is in:


One of many publications on the AI4FM project is:


**Further Information**

Cliff Jones – ext. 88183 – cliff.jones@ncl.ac.uk
Leo Freitas – ext 88036 – leo.freitas@ncl.ac.uk
Chin Wei Ngan - chinwn@comp.nus.edu.sg