Compositional Proofs of Concurrent Programs

Lawrence C. Paulson

Project GR/M75440/01, funded by the Engineering and Physical Sciences Research Council (EPSRC), was undertaken to provide a better understanding of *compositional reasoning*, that is, how to verify systems built from components. It is a continuation of project GR/K57381, *Mechanising Temporal Reasoning*.

A fundamental issue is the representation of program states. The choices are between strongly- and weakly-typed representations and between formalizing a single, universal state representation or giving each component an individual state representation. Sidi Ehmety and I have investigated two strongly-typed approaches: local, polymorphic records [6] and abstract states [4]. We have produced a weakly-typed proof environment for UNITY based on ZF set theory and used it to formalize most of the proofs in Charpentier and Chandy [1].

We have formalized other compositional theories, such as existential and universal properties [2, 3]. Our findings here are positive: the proofs for both papers are simple. In one case five pages of informal proofs are reduced to a few lines of Isabelle/HOL proof script [5]. We have mechanized another theory of compositional reasoning: the *progress sets* of Meier and Sanders. A critical evaluation of the theories we investigated is now available [7].

UNITY's simplicity makes it ideal for the fundamental research undertaken here. UNITY proofs are traditionally carried out on paper, and a continuing theme of this research is the surprises that occur when these proofs are attempted using computer assistance. Although UNITY is too simple to apply to large-scale industrial verification, this is our ultimate aim. Large concurrent systems can only be verified with machine assistance. Formal methods researchers are divided between those who perform proofs on paper and those who advocate computer-based verification tools. Many of the people in the first group wish to use tools, provided their traditional proof style is respected. Part of the tool builder's job is to understand which aspects of their proof style are essential. Some of the obstacles to mechanization originate in the conventions, notations and implicit assumptions of the pencil-and-paper community.

This project has investigated most of the proposals for compositional reasoning in the UNITY literature. It has shown that while temporal reasoning about program components remains difficult, compositional reasoning is not as hard as was previously thought.

References

[1] K. Mani Chandy and Michel Charpentier. An experiment in program composition and proof. *Formal Methods in System Design*, 20(1):7–21, 2002.

[2] Michel Charpentier and K. Mani Chandy. Examples of program composition illustrating the use of universal properties. In José Rolim, editor, *Parallel and Distributed Processing*, LNCS 1586, pages 1215–1227, 1999. Workshop on Formal Methods for Parallel Programming: Theory and Applications.

[3] Michel Charpentier and K. Mani Chandy. Theorems about composition. In R. Backhouse and J. Nuno Oliveira, editors, *Mathematics of Program Construction: Fifth International Conference*, LNCS 1837, pages 167–186. Springer, 2000.

[4] Sidi O. Ehmety and Lawrence C. Paulson. Representing component states in higher-order logic. In Richard J. Boulton and Paul B. Jackson, editors, *TPHOLs 2001: Supplemental Proceedings*, number EDI-INF-RR-0046 in Informatics Report Series, pages 151–158. Division of Informatics, University of Edinburgh, September 2001. Online at `http://www.informatics.ed.ac.uk/publications/report/0046.html`.

[5] Sidi O. Ehmety and Lawrence C. Paulson. Program composition in Isabelle/UNITY. In *Parallel and Distributed Processing*. IEEE, 2002. Workshop on Formal Methods for Parallel Programming: Theory and Applications; text on CD-ROM.

[6] Lawrence C. Paulson. Mechanizing a theory of program composition for UNITY. *ACM Transactions on Programming Languages and Systems*, 25(5):626–656, 2001.

[7] Lawrence C. Paulson. Mechanizing compositional reasoning for concurrent systems: Some lessons. Technical report, Computer Laboratory, University of Cambridge, 2003. On the Internet at `http://www.cl.cam.ac.uk/users/lcp/papers/UNITY`.