Editorial

We are pleased to present a Special Issue on Concurrency and Real-Time for Eiffel-like Languages. The papers in this special issue originate from a workshop held in York, U.K., in July 2006.

The workshop was organised by the University of York, the University of Teesside and ETH Zürich. It was sponsored by the ARTIST Network of Excellence on Embedded Systems Design (<http://www.artist-embedded.org>) and supported by Formal Methods Europe. The workshop proceedings (consisting of nine fully refereed and revised papers, three invited papers, and one invited tutorial) appeared as a University of York technical report. After the workshop, presenters were invited to submit substantially revised and extended papers for inclusion in this special issue. As a result, four papers were accepted and are included herein.

Two of the papers are concerned with the relationship between contracts (pre- and postconditions of routines that belong to classes, and invariants of classes themselves) and concurrency, and the semantic issues associated with their formalisation, reasoning, and implementation. “Contracts for Concurrency” by Nienaltowski, Meyer and Ostroff presents a new semantics for contracts that applies equally well to sequential and concurrent object-oriented programming. The semantics itself is argued to be a generalisation of traditional correctness semantics for object-orientation, and proof techniques are presented. “Beyond Contracts for Concurrency”, by Ostroff, Torshizi, Huang and Schoeller, shows how theorem proving that is useful for sequential object-oriented programming can also be used in a concurrent setting, using a Hoare-style logic. For system properties (such as liveness), a virtual machine is described that makes it feasible to use model checking to verify these properties.

“Flexible Access Control Policy for SCOOP” by Nienaltowski addresses a challenging problem within concurrent object-oriented programming: how to reduce the amount of locking that is necessary to ensure correctness, so as to increase the amount of parallelism. Nienaltowski presents a formal lock passing mechanism (which is implemented in the current version of concurrent Eiffel) and demonstrates how it improves expressiveness and reduces the risk of deadlock.

Finally, “Cameo: an Alternative Model of Concurrency for Eiffel” by Brooke and Paige presents a new concurrency model for Eiffel-like languages, where it is argued that the new model provides increased clarity and parallelism but also increases overhead due to lock management. A CSP formalisation of the model is provided.

We thank John Cooke for supporting us in publishing the collection of papers contained in this journal, and for his help through the reviewing and editorial process. We also thank the participants of the workshop, the workshop program committee, and all of the reviewers involved in selecting the papers.

Richard F. Paige
Phillip J. Brooke
Jin Song Dong
Jim C. P. Woodcock

Published online 16 June 2009