Formalization of a type and effect system for deadlock checking using Coq & Ott

Peter Brottveit Bock[1,2] and Ka I Pun[2], Martin Steffen[2], and Volker Stolz[2]

[1] IT University Copenhagen, Danemark
[2] University of Oslo, Department of Informatics, Norway

Abstract

Deadlocks are a common problem in programs with lock-based concurrency and are hard to avoid or even to detect. One way for deadlock prevention is to statically analyse the program code to spot sources of potential deadlocks.

We reduce the problem of deadlock checking to race checking, another prominent concurrency-related error for which good (static) checking tools exist. The transformation uses a type and effect-based static analysis, which analyses the data flow in connection with lock handling to find out control-points which are potentially part of a deadlock. These control-points are instrumented appropriately with additional shared variables, i.e., race variables injected for the purpose of the race analysis.

The formalization of a type and effect system of that kind is rather intricate, containing series of definitions, inductive rule systems, intermediate lemmas and proofs, that all must fit together to prove ultimately that the formalization is "adequate", in this case that it correctly is able to guarantee absence of deadlock.

Despite careful design of the systems and meticulous proof-reading, the ultimate correctness of such arguments hinges on many subtle details of the type system, the semantics etc. A machine checked proof, using a theorem prover, increases the confidence in the formal validity of the result. So, besides sketching the effect system for statically checking of absence of deadlock, we report on the a machine-checked formalization of the system and parts of the proofs, using Coq, a well-known type-theoretic theorem prover, and OTT, a "domain-specific language for semanticists".