Deadlock checking by a behavioral effect system for lock handling

Ka I Pun[a], Martin Steffen[a], Volker Stolz[a,b]

[a] *Dept. of Computer Science, University of Oslo*
[b] *United Nations University—International Institute for Software Technology (UNU-IIST)*

Abstract

Deadlocks are a common error in programs with lock-based concurrency and are hard to avoid or even to detect. One way for deadlock prevention is to deadlocks. Often static approaches try to confirm that the lock-taking statically analyze the program code to spot sources of potential adheres to a given order, or, better, to infer that such an order exists. Such an order precludes situations of cyclic waiting for each other's resources, which constitute a deadlock.

In contrast, we do not enforce or infer an explicit order on locks. Instead we use a *behavioral* type and effect system that, in a first stage, checks the behavior of each thread or process against the declared behavior, which captures potential interaction of the thread with the locks. In a second step on a global level, the state space of the behavior is explored to detect potential deadlocks. We define a notion of *deadlock-sensitive simulation* to prove the soundness of the abstraction inherent in the behavioral description. Soundness of the effect system is proven by subject reduction, formulated such that it captures deadlock-sensitive simulation.

To render the state-space finite, we show two further abstractions of the behavior sound, namely restricting the upper bound on re-entrant lock counters, and similarly by abstracting the (in general context-free) behavioral effect into a coarser, tail-recursive description. We prove our analysis sound using a simple, concurrent calculus with re-entrant locks.

Keywords: concurrency, deadlock prevention, static analysis, behavioral type and effect systems, simulation relation, abstraction