Motivation

• The era of multicore architectures in computer hardware brings an unprecedented need for parallel programming. We are interested in multithreaded programs with shared data, where programmers use lock primitives to control access to the shared data.
• Mutual exclusion locks (mutexes) are used to protect shared data from inconsistent concurrent updates. Improper use of mutexes can lead to the “deadly embrace” problem and Circular-Mutex-Wait (CMW) deadlocks.
• In our on-going project, called Gadara, the objective is to control the execution of multithreaded programs in order to avoid deadlocks by using techniques from discrete control theory (see Fig. 1).

Main Contributions

• We propose a systematic program control architecture (see Fig. 1) based on Discrete Control Theory, which combines the strengths of offline analysis and control synthesis with online knowledge and control to dynamically avoid CMW deadlocks in concurrent programs.
• We define a special class of Petri nets, called Gadara nets, to formally model lock allocations and releases in multithreaded programs (see Fig. 2).
• We establish necessary and sufficient conditions for liveness and reversibility of Gadara nets, which connect these behavioral properties of the dynamic system to a certain structure in the net, called siphon.
• We develop an efficient control synthesis strategy for Gadara nets, based on structural analysis. The strategy is both correct and maximally permissive with respect to the goal of liveness enforcement for Gadara nets.

Main Properties of Gadara Nets

Theorem 1:
Program is deadlock-free                                  (Goal)
⇔ Gadara net is live                                      (Behavioral Property)
⇔ Gadara net cannot reach a problematic siphon          (Structural Property)

Iterative Control Of Gadara nets (ICOG)

Properties of ICOG:

Theorem 2: ICOG is correct and maximally permissive with respect to the goal of liveness enforcement for Gadara nets.

Theorem 3: ICOG converges in a finite number of iterations.

Experiments

• Gadara has been tested on a set of real-world programs, e.g., OpenLDAP, BIND, Apache. Gadara identifies and avoids both previously known and unknown deadlocks while adding performance overheads ranging from negligible to 10%.
• In benchmark tests, Gadara successfully avoids injected deadlock faults, imposes negligible to modest performance overheads (at most 18%).

References