Shared memory review

K. V. S. Prasad
Dept of Computer Science
Chalmers Univ
Thu 18 Sep 2014

WARNING

- There's a lot more detail in the book than the lectures can possibly cover
  - And the slides are a tiny "trailer" of the lectures
- So: WORK with the TEXTBOOK

- A little good news:
  - You may skip the Ada, BACI and Promela sections if you wish

Readers-writers with protected object

- Look again at slide 7.16 (175/363)
- Protected objects
  - Mutex as for monitors
    - Only one operation at a time
  - Condition variables (queues) -> barriers on ops
    - Separate queue for each barrier
    - Implicit wait managed by implementation
  - No explicit waits or signals in code
  - All barriers rechecked on any exit

Invariants for the R-W problem

- R >= 0
  - Does not follow from just the protected object
  - Need to look at user code
- W >= 0, in fact, W<=1
  - Easier to see in prot. obj. because of boolean
- (R>0 -> W=0) and (W=1 ->R=0)
  - If R>0 then at least one reader is at p2 or p3. They got there because W=0. Can W change while R>0. No, by mutex, and by reader code.
  - W:=1 only if R=0 barrier passed, etc.

Lemmas needed for monitor version

- If readers waiting, then W=1
- If writers waiting, then R>0 or W=1

At least these are immediate from the code of the protected object.

Can either readers or writers starve? Both can, up to implementation to be fair.

How bad can the proofs get?

- Very (hard, not necessarily long)
  - If the decent protected objects still need so much thought, …
    - The less structured methods are surely messier
      - Hence the search for syntactically expressed structure
- So maybe sane people should skip proofs?
  - No, they should skip programming instead
    - Particularly of safety- or mission-critical systems
- If you don't want to study hard and always be very thorough and careful
  - Please don't become a surgeon!

Monitor fr dining philosophers

- Look again at slide 7.11 (170/363)
- Invariants
  - Not empty(OKtoEat[i]) -> (fork[i]<2)
  - Sum (i=0..4) (fork[i]) = 10 – 2*E
    - Where E is the number of eating philosophers
- Deadlock implies E=0 and all enqueued.
  - impossible

Intro to functional programming

- Computation = rewriting preserving value
- Rewrite lhs of equation by rhs
  - Set of equations is program
- Read-eval-print loop
- No assignment
  - Value of var not time-dependent