Mutex review

None of the mutual exclusion algorithms presented so far scores full marks.

Selected problems:

- have a $\mathcal{O}(n^2)$ pre-protocol (Peterson)
- rely on special instruction (e.g. xc, ts, etc.)
- use unbounded ticket numbers (e.g. bakery)
- sacrifice eventual entry (e.g. fast)

Szymanski's Algorithm

- has none of these problems,
- enforces *linear wait*,
- requires at most $4p - \lceil \frac{p}{n} \rceil$ writes for p CS entries by n competing processes, and
- can be made immune to process failures and restarts as well as read errors occurring during writes.

How does he do it?

Idea

"The prologue is modeled after a waiting room with two doors. [...] All processes requesting entry to the CS at roughly the same time gather first in the waiting room. Then, when there are no more processes requesting entry, waiting processes move to the end of the prologue. From there, one by one, they enter their CS. Any other process requesting entry to its CS at that time has to wait in the initial part of the prologue (before the waiting room)." Szymanski, 1988, in ICCS

Phases of the pre-protocol

1. announce intention to enter CS
2. enter waiting room through door 1; wait there for other processes
3. last to enter the waiting room closes door 1
4. in the order of PIDs, leave waiting room through door 2 to enter CS

Shared variables

Each process `i` exclusively writes a variable called `flag`, which is read by all the other processes. It assumes one of five values:

- **0** denoting that `i` is in its non-CS,
- **1** declares `i`'s intention to enter the CS
- **2** shows that `i` waits for other processes to enter the waiting room
- **3** denotes that `i` has just entered the waiting room
- **4** indicates that `i` left the waiting room

Algorithm 1.8: Szymanski's algorithm (n processes, process i)

integer array flag[1..n] $\leftarrow$ [0,. . . ,0]

forever do

p1: *non-critical section*

p2: flag[i]:=1

p3: **await** $\forall$j. flag[j] $<$3

p4: flag[i]:=3

p5: **if** $\exists$j. flag[j] $=$ 1 **then**

p6: flag[i]:=2

p7: **await** $\exists$j. flag[j]=4

p8: flag[i]:=4

p9: **await** $\forall$j$<$i. flag[j] $<$2

p10: *critical section*

p11: **await** $\forall$j$>$i. flag[j] $<$2 or flag[j] $>$3

p12: flag[i]:=0

How to implement the atomic tests

The atomic tests can be implemented by loops. The order of the tests is crucial for the mutual exclusion property. But which order? Szymanski's original paper is unclear on the matter.

See Promela Code samples (and your homework ;).

How to prove mutual exclusion

This is reasonably hard. So hard indeed that even Turing Award winners (Manna and Pnueli) published about solving the problem (with non-atomic tests), using the "one big invariant" method. See the de Roever book pp.157–164 for a proof using the Owicki-Gries method on (parameterized) transition diagrams (with atomic tests).

What is hard about the proof? Finding the assertions.

What now?

- You should be making progress on Assignment 0 (due Monday) and Homework 2 (due Friday).
- You can (soon) find Promela code on the website for most of the algos discussed today.
- New questions about critical sections will be up soon, due Friday next week.