Design of Parallel and High Performance Computing
HS 2013
Markus Püschel, Torsten Hoefler
Department of Computer Science
ETH Zurich

Homework 4
Out: 2013-10-24
Revision: 1

Locks

Simple Spin Lock

Prove that the lock given below violates one or more of the neccessary lock properties. Use sequential consistency.

```
volatile int lock = 0;

void lock() {
  while(lock != 0) {/*wait here*/}
  lock = 1;
}

void unlock() { lock = 0; }
```

Give an alternative for a two thread lock. Prove (using sequential consistency) that it provides mutual exclusion and deadlock freedom.

Filter Lock

Prove that the filter lock (given below) provides mutual exclusion for n threads. Use sequential consistency.

```
volatile int level[n] = {0,0,...,0};
volatile int victim[n];

void lock() {
  for (int l=1; l<n; l++) {
    level[tid] = l;
    victim[l] = tid;
    while ((∃k != tid) (level[k] >= l && victim[l] == tid)) {};
  }
}

void unlock() { level[tid] = 0; }
```

Bakery Lock

Prove that the bakery lock (given below) provides mutual exclusion for n threads. Use sequential consistency.

```
volatile int flag[n] = {0, 0, ..., 0};
volatile int label[n] = {0, 0, ...,0};
void lock() {
  flag[tid] = 1;
  label[tid] = max(label[0], ...,label[n-1]) + 1;
  while ((∃k != tid)(flag[k] && (label[k],k) ≪ (label[tid],tid))) {};
}

public void unlock() { flag[tid] = 0; }
```

Note: (a,b) $\ll$ (c,d) behaves like a $<$ c, unless a=c, in which case it becomes b $<$ d.

Better than Dijkstra?

The following is an excerpt of the journal "Communications of the ACM", Vol. 9, No. 1, page 45, 1966. Does the given algorithm ensure mutual exclusion?

Comments on a Problem in Concurrent Programming Control

Dear Editor:

I would like to comment on Mr. Dijkstra's solution [Solution of a problem in concurrent programming control. *Comm ACM 8* (Sept. 1965), 569] to a messy problem that is hardly academic. We are using it now on a multiple computer complex.

When there are only two computers, the algorithm may be simplified to the following:

Boolean array $b(0; 1)$ **integer** $k, i, j,$
comment This is the program for computer i, which may be either 0 or 1, computer $j \neq i$ is the other one, 1 or 0;
$C0$: $b\ (i)$:= **false**;
$C1$: **if** $k \neq i$ **then begin**
$C2$: **if not** $b(j)$ **then go to** $C2$;
else $k := i$; **go to** $C1$ **end**;
else critical section;
$b(i)$:= **true**;
remainder of program;
go to $C0$;
end

Mr. Dijkstra has come up with a clever solution to a really practical problem.