Design of Parallel and High Performance Computing
HS 2014
Torsten Hoefler, Markus Püschel
Department of Computer Science
ETH Zurich

Homework 6
Out: 2014-10-30
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.