Parallel Programming
0024

Week 06

Thomas Gross

Spring Semester 2010

Apr 15, 2010

Classroom exercise

Consider this program (fragment) [PingPong] for thread A (myid == 0) and thread B (myid == 1)

```
// thread A
public void run() {
      while (true) {
   A1: non_critical section
   A2:   while ( !(signal.turn == 0)){}
    A3: critical_section
    A4: signal.turn = 1;
        }
    }
```

Class room exercise, continued

```
// thread B
public void run() {
      while (true) {
   B1: non_critical section
   B2:   while ( !(signal.turn == 1)){}
     B3: critical_section
     B4: signal.turn = 0;
         }
     }
```

Your task (now!)

Show that these threads will never be both in their critical section at the same time.

You should prove this property in a manner that’s similar to the proof given in class.

Some thoughts on how to proceed

We introduced already labels for statements and produced two distinct versions for thread A and thread B.

Now you should formulate the invariant.

Invariant(s)

(i) at(A3) -> turn == 0

(ii) at(B3) -> turn== 1

(iii) not [at(A3) AND at(B3)] m

Proof strategy

Proof by induction on the execution sequence.

Base case: does (i) hold at the start of the execution of the program (threads at A1 and B1)

Induction step: Assume that (i) holds. Will execution of an additional step invalidate (i)?

Proof (i)

at(A1): condition (i) is false => do not care about signal

at(A2): condition (i) is false => do not care about signal

at(A3): condition (i) is true => turn == 0, follows from the fact that turn was 0 at(A2) AND the transition from A2->A3 did not change value of turn

at(A4): condition (i) is false ==> do not care about turn

Now, we consider:

at(B1) : no change to turn

at(B2) : no change to turn

at(B3) : no change to turn

at(B4) : changes turn to 0

=> Invariant 1 is true

Proof (ii)

Same way (please do it if you had trouble with proof of i)

Proof (iii)

Induction start trivial.

Proof of induction step by contradiction.

Assume thread A entered CS (A3) at time t1

Assume thread B entered CS (B3) at time t2, where t2 = t1 + delta

--> CONTRADICTION: since we are in A3 signal MUST be 0 (cannot be 0 and 1 at the same time)

Assume thread B entered CS (B3) at time t1

Assume thread A entered CS (A3) at time t2, where t2 = t1 + delta

--> CONTRADICTION: since we are in B3 signal MUST be 1 (cannot be 0 and 1 at the same time)

Classroom exercise (based on 3rd variation)

```
class Turn {
    // 0 : wants to enter exclusive section
    // 1 : does not want to enter ...

    private volatile int flag = 1;

     void request() { flag = 0;}
     void free() { flag = 1; }
     int read() { return flag; }
}
```

Worker

```
class Worker implements Runnable {

    private int myid;
    private Turn mysignal;
    private Turn othersignal;

    Worker(int id, Turn t0, Turn t1) {
        myid = id;
        mysignal = t0;
        othersignal = t1;
    }
```

Worker

```
    public void run() {
        while (true) {
         mysignal.request();

            while (true) {
                if (othersignal.read() == 1) break;
            }
      // critical section
      mysignal.free();
        }
    }
}
```

Master

```
class Synch3b {
    public static void main(String[] args) {
      Turn gate0 = new Turn();
      Turn gate1 = new Turn();

      Thread t1 = new Thread(new Worker(0,gate0, gate1));
      Thread t2 = new Thread(new Worker(1,gate1, gate0));
      t1.start();
      t2.start();
    }
}
```

Worker

```
public void run() {

    while (true) {
     mysignal.request();

        while (true) {
            if (othersignal.read() == 1) break;
        }
  // critical section
  mysignal.free();
    }
}
```

Worker 0

```
  public void run() {
  while (true) {
A1:
A2:    s0.request();


A3:   while (true) {
            if (s1.read() == 1) break;
          }
A4:   // critical section
A5:   s0.free();
        }
  }
```

Worker 1

```
    public void run() {
    while (true) {
B1:
B2:    s1.request();


B3:   while (true) {
            if (s0.read() == 1) break;
          }
B4:   // critical section
B5:   s1.free();
       }
   }
```

Mutual exclusion

Show that this solution provides mutual exclusion.

Invariants

(i) s0.flag == 0 equivalent to (at(A3) V at(A4) V at(A5))

(ii) s1.flag == 0 equivalent to (at(B3) V at(B4) V at(B5))

(iii) not (at(A4) ∧ at(B4))

Induction

Show with induction that (i), (ii), and (iii) hold.

***At the start*, s0.flag==1 and at(A1) – ok**

Induction step:

assume (i) is true. Consider all possible moves

A1 → A2

A2 → A3

A3 → A3

A3 → A4

A4 → A5

A5 → A1

Let’s look at them one by one:

Induction step

A1 → A2 : no effect on (i) – ok

A2 → A3 : (i) holds (s0.flag == 0 and at(A3)) – ok

A3 → A3 : (i) holds, no change to s0.flag, at(A3) – ok

A3 → A4 : (i) holds, no change to s0.flag, at(A4) – ok

A4 → A5 : (i) holds, no change to s0.flag, at(A5) – ok

A5 → A1 : (i) holds, s0.flag == 1 and at(A1) – ok

Note that the "– ok" is based on the observation that no action by Thread Worker 1 will have any effect on s0.flag

So (i) holds.

Your turn

Show that (ii) holds as well.

Sorry if you think this is trivial. You're right.

Proving (iii)

Recall

(iii) not (at(A4) ∧ at(B4))

Use … induction.

***At the start*, at(A1) and at(B1), so (iii) holds.**

***Induction step*: assume (iii) holds and consider possible transitions.**

Assume at(A4) and consider B3 → B4 (while Worker0 remains at A4!)

no other transition is relevant or possible

But since s0.flag==0 (because of (i)), a transition B3 → B4 is not possible, so (iii) remains true.

…

Same argument applies if we start with the assumption at(B4).

So no transition will violate (iii).

Of course this sketch of a proof depends on the fact that no action by Worker0 (Worker1) will modify any of the state of Worker1 (Worker0).

Any Questions?