Parallel Programming WS 2014/2015 - Assignment 2

Kastens, Pfahler
Institut für Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn
Nov 07, 2014

Exercise 1 (Process Interference)

Consider the following process `p1`:

```
{ P1: moneyBag = x }
b1 = 10;
t1 = moneyBag;
t1 = t1 + b1;
moneyBag = t1;
{ Q1: moneyBag = x + 10 }
```

Last time we have shown that the postcondition Q1 holds for the sequential execution of the process if the execution starts in a state described by P1.

a) Consider a second process `p2`:

```
{ P2: moneyBag = x }
moneyBag = moneyBag - 5;
{ Q2: moneyBag = x - 5 }
```

Assume the statement sequences in both `p1` and `p2` are executed as single atomic actions. Show that the concurrent execution of `p1` *does* interfere with `p2`.

b) Weaken the preconditions to preconditions P1’ and P2’ with P1 => P1’ and P2 => P2’, such that

```
{ P1′ } S1 { Q1′ }
{ P2′ } S2 { Q2′ }
```

can be proven.

Hint: Weaken the preconditions to express that each process could be executed after the other one has been executed.

c) Show non-interference using the new pre- and postconditions.

d) Since now the processes do not interfere you can apply the concurrency rule (PPJ-17f) to prove the result of the concurrent execution.

Exercise 2 (LAB: Concurrent Programming in Java)

Write a Java program "Counters" that takes a command line parameter n and creates n threads of class "Counter". Each counter has a name ("counter_i") and counts from 1 to 9 by periodically outputting its name and counter value and then sleeping for a random value between 0 and 99 milliseconds. Make class "Counter" a subclass of "Thread" (PPJ-11).

Example output for "java Counters 3":

```
counter_1 --> 1
counter_3 --> 1
counter_2 --> 1
counter_2 --> 2
counter_1 --> 2
counter_3 --> 2
counter_3 --> 3
counter_1 --> 3
counter_1 --> 4
...
```

Exercise 3 (Homework: Interference)

Two processes p1 and p2 operate on a common variable y. Their atomic actions are indicated by angled brackets.

```
y = 1;
co
   p1: <y = y + 2;> //
   p2: <y = y - 1;> <y = y + 4;>
oc
```

Prove that y = 6 holds after both processes have finished. Derive assertions that hold at each point with observable state in the program. Show non-interference of each of these assertions with all relevant assignments.

Hints: Use weakened assertions (disjunctions) that take all possible interleavings of the atomic actions into account. At the end of the program the post-conditions of both processes hold. Their conjunction should imply that y = 6.