

## Applying Formal Verification, SS 2012

### Functional Verification of Concurrent Programs

When writing down solutions in ASCII, you may use  $x$  instead of  $^{\circ}x$  and  $x'$  instead of  $^ax$  in two-state assertions. It is also permissible to write just  $x$  instead of  $\acute{x}$  in single-state assertions. You can write  $\wedge$  as  $\&$ ,  $\neg$  as  $!$ , etc.

#### Assignment 1

Below is a proof outline for an implementation of Peterson's mutual exclusion algorithm (Eike Best, "Semantics of Sequential and Parallel Programs", p. 217). The proof outline is correct and interference-free.

Explain why this specification guarantees mutual exclusion of the two processes in the critical section.

```
record Petersons_mutex_1 =
  pr1 :: nat
  pr2 :: nat
  in1 :: bool
  in2 :: bool
  hold :: nat
```

```
lemma Petersons_mutex_1:
  "||- .{pr1=0 & !in1 & pr2=0 & !in2 }.
  COBEGIN .{pr1=0 & !in1}.
  WHILE True INV .{pr1=0 & !in1}.
  DO
    .{pr1=0 & !in1}. < in1:=True,, pr1:=1 >;
    .{pr1=1 & in1}. < hold:=1,, pr1:=2 >;
    .{pr1=2 & in1 & (hold=1 & hold=2 & pr2=2)}.
  AWAIT (!in2 & !(hold=1)) THEN pr1:=3 END;;
  .{pr1=3 & in1 & (hold=1 & hold=2 & pr2=2) }.
  < in1:=False,, pr1:=0 >
  OD .{pr1=0 & !in1}.
  ||
  .{pr2=0 & !in2}.
  WHILE True INV .{pr2=0 & !in2}.
  DO
    .{pr2=0 & !in2}. < in2:=True,, pr2:=1 >;
    .{pr2=1 & in2}. < hold:=2,, pr2:=2 >;
    .{pr2=2 & in2 & (hold=2 & (hold=1 & pr1=2))}.
  AWAIT (!in1 & !(hold=2)) THEN pr2:=3 END;;
  .{pr2=3 & in2 & (hold=2 & (hold=1 & pr1=2))}.
  < in2:=False,, pr2:=0 >
  OD .{pr2=0 & !in2}.
  COEND
  .{pr1=0 & !in1 & pr2=0 & !in2}."
```

```

apply oghoare
— 104 verification conditions.
apply auto
done

```

## Assignment 2

Fill in the blanks to obtain a valid rely-guarantee formula. A proof is *not* required.

Remember: Angle brackets  $\langle \cdot \rangle$  denote atomic blocks.

```

record Example2 =
  x  :: nat
  c_0 :: nat
  c_1 :: nat

lemma Example2:
  "⊢ COBEGIN
    (⟨ ˆx:=ˆx+1;; ˆc_0:=ˆc_0 + 1 ⟩,
      {_____,
       {_____,
       {_____,
       {_____}})
    ||
    (⟨ ˆx:=ˆx+1;; ˆc_1:=ˆc_1+1 ⟩,
      {_____,
       {_____,
       {_____,
       {_____}})
  COEND
  SAT [ {ˆx=0 ∧ ˆc_0=0 ∧ ˆc_1=0},
        {ˆx=ˆa x ∧ ˆc_0=ˆa c_0 ∧ ˆc_1=ˆa c_1},
        {True},
        {ˆx=2} ] "

```