Reliable Software - Assignment 1

Joergen Aleksander Fagervik

2013-30-05

1 General

2 Task 1

2.1 Code

```
int x = 10;
int y = 0;

active proctype t1() {
    do
    :: x != y -> x = x - 1; y = y + 1;
    od
}


active proctype t2() {
    do
    :: x == y -> atomic { x = 7; y = 3; }
    od
}

init {run t1(); run t2();}
```

3 Task 2

3.1 Code

```
int x = 0;

proctype P() {
    int k;
    int N = 5;

    int local_x;

    for (k: 1..N) {
        local_x = x; /* LOAD */
        local_x++;   /* INCREMENT */
        x = local_x; /* STORE */
    }
}

proctype check() {
    if
    :: (_nr_pr == 1) -> printf("x is %d\n", x); assert (x == 3);
    fi
}


init {
        run P(); run P(); run P();

    run check();
}
```

4 Task 3

4.1 Code

```
#define N 3  // Number of people
#define B_MIN 0
#define B_MAX 4

#define NUM_M 300

byte PERSON1 = 0;
byte PERSON2 = 1;
byte PERSON3 = 2;

chan c1 = [NUM_M] of {int};
chan c2 = [NUM_M] of {int};
chan c3 = [NUM_M] of {int};

byte a1 = 5, a2 = 6, a3 = 7;

byte b[N];  // Lower bounds b[i] for each person
byte d[N];  // Candidate dates d[i] for each person
byte f[N];  // Earliest available dates f[i] for each person

// MAIN IDEA: RECEIVE TO YOUR OWN IDS CHANNEL, SEND TO NEXT

active proctype P1(byte id) {
  byte prev_date;

  d[id] = 0;
  f[id] = 0;

  do
  :: true ->
    d[id] = f[id];

    c2 ! d[id];

    c1 ? prev_date;

    if
    :: prev_date > d[id] -> d[id] = prev_date;
    :: else -> skip;
    fi;

    f[id] = (d[id] - b[id] + a1);

    if
    :: f[id] <= prev_date -> break;
    :: else -> skip;
```

```
    fi;
  od;
}

active proctype P2(byte id) {
  byte prev_date;

  d[id] = 0;
  f[id] = 0;

  do
  :: true ->
    d[id] = f[id];

    c3 ! d[id];

    c2 ? prev_date;

    if
    :: prev_date > d[id] -> d[id] = prev_date;
    :: else -> skip;
    fi;

    f[id] = (d[id] - b[id] + a2);

    if
    :: f[id] <= prev_date -> break;
    :: else -> skip;
    fi;
  od;
}

active proctype P3(byte id) {
  byte prev_date;

  d[id] = 0;
  f[id] = 0;

  do
  :: true ->
    d[id] = f[id];

    c1 ! d[id];

    c3 ? prev_date;
```

```
    if
    :: prev_date > d[id] -> d[id] = prev_date;
    :: else -> skip;
    fi;

    f[id] = (d[id] - b[id] + a3);

    if
    :: f[id] <= prev_date -> break;
    :: else -> skip;
    fi;
  od;
}

init {
  // Initialize lower bounds b[i] and intervals a[i]
  byte b1, b2, b3;

  // Loop through possible values of b1, b2, and b3
  for (b1: B_MIN..B_MAX) {
  for (b2: B_MIN..B_MAX) {
  for (b3: B_MIN..B_MAX) {

    // Initialize lower bounds and intervals for each person
    b[0] = b1; b[1] = b2; b[2] = b3;

    run P1(PERSON1);
        run P2(PERSON2);
        run P3(PERSON3);

    // TODO: Fix this loop
    do
    :: _nr_pr > 1 -> 0;
    :: else -> break;
    od;

    // Print the largest possible meeting date and the corresponding lower bo
    if
    :: d[0] > d[1] && d[0] > d[2] -> printf("Largest meeting date: %d, b1: %d
    :: d[1] > d[0] && d[1] > d[2] -> printf("Largest meeting date: %d, b1: %d
    :: d[2] > d[0] && d[2] > d[1] -> printf("Largest meeting date: %d, b1: %d
    :: else -> skip;
    fi;
  }
  }
  }
```

```
}
```

5 Task 4

5.1 Code

```
mtype = {p,v}; //  P for Wait, V for signal

chan S = [0] of {mtype};

byte N = 0;

proctype semaphore(byte n) {
    N = n;
    do
    :: N > 0 ->
        S ? p;
        N--;
    :: N == 0->
        S ? v;
        N++;
    od
}


active[5] proctype c() {
    do
    :: S ! p;
        printf("%d bien here!\n", _pid);
       S ! v;
    od
}

active proctype check() {
    assert (N >= 0);
}


init {
    byte n = 3;
    run semaphore(n);

    // run check();

    /*bit i;
    for (i: 1..5) {
        run c();
    }*/
}
```

6 Task 5

6.1 Code

```
mtype = { p, v }; //  P for Wait, V for signal

chan S = [0] of {mtype};

// TODO: Impl chanl
chan C = [5] of {int};

byte N = 0;

proctype semaphore(byte n) {
    N = n;
    do
    :: N > 0 ->
        S ? p;
        N--;
    :: N == 0->
        S ? v;
        N++;
    od
}

active[5] proctype c() {
    do
    :: S ! p;
        printf("%d bien here!\n", _pid);
       S ! v;
    od
}

active proctype check() {
    assert (N >= 0);
}


init {
    byte n = 3;
    run semaphore(n);

    // run check();

    /*bit i;
    for (i: 1..5) {
        run c();
    }*/
}
```