

- 
- 2 problems amounting to 50% of Exam 1.
  - Total time 120 mins.
  - No book, no notes.
  - Your answer will be judged on accuracy, readability, and elegance.
  - Hand in only your final answer. Write on only one side.
  - Keep your rough work separate.
- 

- **NOTE: “Being invariant” versus “satisfying invariance rule”:** “X is invariant”, or  $\square X$ , means X holds at every state of every execution, i.e., at every reachable state. “X satisfies invariance rule” means that X holds initially and, for any state (reachable or not) satisfying X and any event, executing the event in the state results in a state that satisfies X. So “X satisfies invariance rule” implies “X is invariant”, but the converse is not true. For example, in problem 1 of Exam 1b,  $S_1$  satisfies the invariance rule, and  $nb + nc \leq na$  is invariant but it does not satisfy the invariance rule.

### Problem 1 [10 points]

```
system-program X( int N ) { // N ≥ 1
  // each labelled statement is atomic
  // initialization
  Semaphore full := 0;
  Semaphore empty := N;
  int x := 0; int na := 0; int nb := 0; int nc := 0;
  Process a := StartProcess(A()); Process b := StartProcess(B()); Process c := StartProcess(C());
  // end of initialization

  function void A() {
    forever do {
      1: na++;
      2: P(empty);
      3: x++;
      4: V(full);
    }

  function void B() {
    forever do {
      1: P(full);
      2: x--;
      3: V(empty);
      4: nb++;
    }

  function void C() {
    forever do {
      1: P(full);
      2: x--;
      3: V(empty);
      4: nc++;
    }

  progress-assumptions { Wfair(*) }
}
```

Assume that  $N \geq 1$  and that each labelled statement is atomic. For each assertion below, prove or disprove whether  $X()$  satisfies the assertion:

- (a)  $\square nb + nc \leq na$
- (b)  $\square nb + nc \geq na - N$
- (c)  $\square nb + nc \geq na - 4N$
- (d)  $[\forall \text{int } i :: na = i \rightsquigarrow na = i + 1]$
- (e)  $[\forall \text{int } i :: nb = i \rightsquigarrow nb = i + 1]$

If you decide to prove an assertion  $\square X$ , provide a list of predicates, say  $Y$ , such that (the conjunction of) the predicates satisfies the invariance rule and implies  $X$ . **Just give  $Y$  and any auxiliary variables used in  $Y$ . Do not give details of how  $Y$  satisfies the conditions.**

If you decide to prove a progress assertion, provide a sequence of leads-to assertions, each justified by a leads-to via rule application or by the closure of previous assertions **In the former case, just mention the relevant event; don't give details. In the latter case, just mention the relevant assertions; don't give details.**

If you decide to disprove an assertion (safety or progress), provide a counter-example execution.

**Problem 2 [5 points]** This problem is about whether the termination detection algorithm for diffusing computations still works if message transmission is not instantaneous.

First, we summarize the algorithm for the case of instantaneous message transmission:

- The computation involves processes  $u[0], \dots, u[N - 1]$ , where  $u[0]$  is the environment process.
- Process  $u[i]$  maintains the following:
  - deficits of its incoming edges ( $dfct[0], \dots, dfct[N - 1]$ )
  - total incoming deficit (tid)
  - total outgoing deficit (tod)
  - identity of the incoming engagement edge if engaged (engager).
- Process  $u[i]$  sends a signal on an incoming edge if
  - (1) the edge's deficit exceeds one, or
  - (2) the edge is not the incoming engagement edge and its deficit exceeds zero, or
  - (3) the edge is the incoming engagement edge, tid equals one, tod equals zero, and the process is not active.
- The computation satisfies the following:
 
$$S_0 : \square u[0].tod = 0 \Rightarrow [\forall 1..N-1, i :: u[i] \text{ not active}]$$

$$P_0 : [\forall 1..N-1, i :: u[i] \text{ not active}] \rightsquigarrow u[0].tod = 0$$

Now consider the case of non-instantaneous message transmission. Specifically, let there be a fifo channel from every  $u[i]$  to every other  $u[j]$ . When  $u[i]$  sends a message to  $u[j]$ , the message is appended to the tail of the channel. When  $u[j]$  receives a message from  $u[i]$ , it gets the message at the head of the channel. Each  $u[i]$  still follows the same rules as before.

Does this non-instantaneous transmission version of the algorithm still satisfy  $S_0$  and  $P_0$ ?

If your answer is yes, give an assertional proof. Your proof should give the invariance and leads-to proof rule applications but no further details. You can assume that you already have an assertional proof that the instantaneous transmission version of the algorithm satisfies  $S_0$  and  $P_0$ .

If your answer is no, give a counter-example execution.

**Problem 1 [10 points]** We prove the assertions in (a), (c), (d), and disprove the assertions in (b), (e).

The given program is a producer-consumer problem, with one producer (a) adding to  $x$  and two consumers (b, c) removing from  $x$ . Thus  $x$  equals  $na - nb - nc$ , except that it lags by 1 in each of the following cases: (1) a has updated  $na$  but has not yet updated  $x$  (i.e., a on 2, 3); (2) b has updated  $x$  but has not yet updated  $nb$  (i.e., b on 3, 4); (3) b has updated  $x$  but has not yet updated  $nb$  (i.e., b on 3, 4).

Also full equals  $x$  except that it lags by 1 in each of the following cases: (1) a has updated  $x$  but has not yet updated full (i.e., a on 4); (2) b has updated full but has not yet updated  $x$  (i.e., b on 2); (3) c has updated full but has not yet updated  $x$  (i.e., c on 2).

Finally, empty equals  $N - x$  except that it lags by 1 in each of the following cases: (1) a has updated empty but has not yet updated  $x$  (i.e., a on 3); (2) b has updated  $x$  but has not yet updated empty (i.e., b on 3); (3) c has updated  $x$  but has not yet updated empty (i.e., c on 3).

This leads to the following predicates, where  $1[P]$ , for a predicate  $P$ , denotes the indicator function of  $P$ , i.e., equals 1 if  $P$  holds and 0 otherwise:

$$S_0 : X.\pi\text{set} = \{a, b, c\}$$

$$S_1 : x = na - nb - nc - 1[a \text{ on } 2, 3] - 1[b \text{ on } 3, 4] - 1[c \text{ on } 3, 4]$$

$$S_2 : x = \text{full} + 1[a \text{ on } 4] + 1[b \text{ on } 2] + 1[c \text{ on } 2]$$

$$S_3 : N - x = \text{empty} + 1[a \text{ on } 3] + 1[b \text{ on } 3] + 1[c \text{ on } 3]$$

$S_{0..3}$  satisfies the invariance rule, so  $\square S_{0..3}$  holds. (In fact,  $S_0, S_1, S_2, S_3$  each satisfy the invariance rule.)

(a)  $\square nb + nc \leq na$  holds because  $S_{0..3}$  implies  $nb + nc \leq na$ .

(Details:  $S_1$  can be written as  $nb + nc = na - x - 1[a \text{ on } 2, 3] - 1[b \text{ on } 3, 4] - 1[c \text{ on } 3, 4]$ .  $S_2$  implies  $x \geq \text{full}$ , which implies  $x \geq 0$ . These two imply  $nb + nc \leq na$ .)

(b)  $\square nb + nc \geq na - N$  does not hold. Below is a counter-example execution:

- state:  $na = nb = nc = x = \text{full} = 0, \text{empty} = N$
- transitions: a.1..a.4  $N$  times
- state:  $na = x = \text{full} = N, nb = nc = \text{empty} = 0$
- transitions: a.1
- state:  $na = N + 1, x = \text{full} = N, nb = nc = \text{empty} = 0$

(c)  $\square nb + nc \geq na - 4N$  holds because  $S_{0..3}$  implies  $nb + nc \geq na - 4N$ .

(Details:  $S_3$  implies  $x \leq N - \text{empty}$ , which implies  $x \leq N$ . This and  $S_1$  (as in part (a)) imply  $nb + nc \geq na - N - 3$ , which implies  $nb + nc \geq na - 4N$ , for  $N \geq 1$ .)

(d)  $\forall i \int na = i \rightsquigarrow na = i + 1$  holds.

- (1)  $na = i \wedge a \text{ on } 1, 3, 4 \rightsquigarrow na = i + 1$  (via a.1, a.3, a.4)
- (2)  $na = i \wedge a \text{ on } 2 \wedge \text{empty} > 0 \rightsquigarrow na = i \wedge a \text{ on } 3$  (via a.2)
- (3)  $a \text{ on } 2 \wedge \text{empty} = 0 \rightsquigarrow \text{full} + 1[b \text{ on } 2, 3] + 1[c \text{ on } 2, 3] = N$  ( $\square S_{2,3}$  (add  $S_2$  and  $S_3$ ))
- (4)  $\text{full} + 1[b \text{ on } 2, 3] + 1[c \text{ on } 2, 3] = N \rightsquigarrow \text{empty} > 0$  (via b, c)
- (5)  $na = i \wedge a \text{ on } 2 \wedge \text{empty} = 0 \rightsquigarrow na = i + 1$  (3,4)
- (6)  $na = i \rightsquigarrow na = i + 1$  (1,2,5)

(e)  $\forall i \int nb = i \rightsquigarrow nb = i + 1$  does not hold. Below is a counter-example (complete) execution:

- (1) state:  $na = nb = nc = x = \text{full} = 0, \text{empty} = N$
- (2) transitions: a.1..a.4
- (3) state:  $na = x = \text{full} = 1, nb = nc = \text{empty} = N - 1$
- (4) transitions: c.1..c.4
- (5) Repeat 1,2,3,4 infinitely.

**Grading:**

- 2 pt: Predicate relating  $x$ ,  $na$ ,  $nb$ ,  $nc$ , process control (i.e.,  $S_1$ ).
- 1 pt: Predicate relating  $x$ , full, process control (i.e.,  $S_2$ ).
- 1 pt: Predicate relating  $x$ , empty, process control (i.e.,  $S_3$ ).
- 1 pt for each of part a, b, c, e.
- 2 pt: part d.

**Problem 2 [5 points]** Let  $M$  denote the non-instantaneous transmission version of the algorithm. We will prove that  $M$  satisfies  $S_0$  and  $P_0$ .

Let  $z[i, j]$  denote the sequence of messages and signals in the channel from  $u[i]$  to  $u[j]$ . Let  $m[i, j]$  be the number of messages in  $z[i, j]$ . Let  $s[i, j]$  be the number of signals in  $z[i, j]$ . There are several ways to proceed.

The straightforward way is to modify the assertional proof of the instantaneous transmission version:

- Redefine  $\langle i, j \rangle$  to be an engagement edge if  $(u[j].tid \neq 0 \wedge u[j].engager = i) \vee [j].tid = 0 \wedge m[i, j] > 0$ . (In the instantaneous version, the second disjunct is missing.)
- Augment the list of predicates satisfying the invariance rule with the following predicate:  $u[i].tod = \sum_j (u[j].dfct[j] + m[i, j])$ . (In the instantaneous version,  $m[i, j] = 0$ .)

With these changes, the assertional proof of the instantaneous version becomes an assertional proof of the non-instantaneous version.

Another way is to show that  $M$  is a special case of the instantaneous transmission version. Let  $u[i]'$  be the system consisting of  $u[i]$  augmented with its incoming channels, i.e.,  $\{z[j, i] : 0..N - 1\}$ . Then  $\{u[i]'\}$  is an instantaneous transmission computation. If  $u[i]'$  obeyed the rules of the termination detection algorithm, then it would satisfy  $S_0$  and  $P_0$ . But  $u[i]'$  obeys the rules of  $u[i]$ , i.e., with  $u[i].dfct[j]$  instead of  $u[i]'.dfct[j]$  and  $u[i].tod$  instead of  $u[i]'.tod$ . However, note that this is ok because these rules are special cases of the corresponding  $u[i]'$  rules, because  $u[i]'.dfct[j] = u[i].dfct + m[j, i]$  and  $u[i]'.tod = u[i].tod - (\sum_j s[j, i])$ . Hence  $M'$  is a special case of the instantaneous version.

**Grading:**

- 0 pt: Answering NO.
- 1 pt: Answering YES.
- 4 pt: Giving details.