

-
- Due Monday November 17 at 5:00 pm. Slip it under my office door or email it to me.
 - Your answer will be judged on accuracy, readability, and elegance. Write neatly or type (font size 10 or more). Use only letter-size sheets with reasonable margins and spacing. **Write on only one side.**
-

Problem 1 [15 points] The distributed mutual exclusion problem is the mutual exclusion problem with the additional constraint that processes can interact only by passing messages over reliable (fifo) channels. On the next page is an attempted solution (from Lamport's CACM paper, "Time, Clocks, and Event-ordering"). It is essentially a distributed version of the Bakery algorithm.

Here are conventions followed in the program:

- For notational convenience, we allow an lc event to do multiple outputs to different systems.
- The type $0..N-1$ consists of integers $0, 1, \dots, N-1$.
- The `Msg` type is a list of message tuples. Each tuple is just like a record, e.g., `(REQ, int t)` is a record with two fields, a constant field `REQ` and an integer field `t`.
- For any type `X`, the type `Sequence X` defines sequences of elements of `X`. For any `q` of type `Sequence X`, we have the following:
 - `empty(q)`: returns true if `q` is empty, otherwise returns false.
 - `remv(q)`: removes the message from head of `q` and returns it; it is undefined if `q` is empty.
 - `apnd(q, m)`: appends `m` to the tail of `q`.

Problem 1a [11 points] Prove or disprove whether system `XX` satisfies $\Box S_0$, where

$$S_0 : [\forall 0..N-1 j, k, j \neq k :: \neg(u[j].st = E \wedge u[k].st = E)]$$

- If you decide to prove, provide a list of predicates, say `Y`, such that (the conjunction of) the predicates satisfies the invariance rule and implies S_0 . **Just give `Y` and any auxiliary variables used in `Y`, and claim it satisfies the invariance rule and implies S_0 . Do not give any details of how `Y` satisfies these conditions.**
- If you decide to disprove, provide a counter-example execution.

Problem 1b [4 points] Prove or disprove whether system `XX` satisfies

$$P_0 : [\forall 0..N-1 i :: (u[i].st = H \rightsquigarrow u[i].st = E)]$$

- If you decide to prove, provide a total-ordering (F, \prec) where `F` is a function of the system variables (and any auxiliary variables you define) such that the following hold:
 - $u[i].st = H \wedge F = j \succ 0$ unless $u[i].st = E \vee F \prec i$
 - $u[i].st = H \wedge F = j \succ 0 \rightsquigarrow u[i].st = E \vee F \prec i$
 - $u[i].st = H \wedge F = 0$ unless $u[i].st = E$
 - $u[i].st = H \wedge F = 0 \rightsquigarrow u[i].st = E$

Just give (F, \prec) and any auxiliary variables used in `F`, and claim it satisfies the above conditions. Do not give the details of how (F, \prec) satisfies the conditions.

- If you decide to disprove an assertion, provide a counter-example execution.

```

system-program XX( int N ) {
  type Msg = { (REQ, int t), (REL, int t), (ACK, int t) }; // messages
  type Msgq = Sequence Msg;
  for every i in 0..N-1 do { Process u[i] := StartProcess(X(i)) }
}

system-program X(int i) {
  type Status = (T,H,E); // thinking, hungry, eating
  Status st := T;
  int c := 0; // logical clock of u[i]
  int[N] req := 0; // req[j] is 0 or timestamp of last received req from u[j]
  int[N] ack := 0; // ack[j] is last received timestamp from u[j]
  Msgq[N] inq := < >; // inq[j] is the channel from u[j] to u[i]

  xc-event snd(0..N-1 j, Msg m) { // append message m to tail of inq[j]
    ec true
    ac apnd(inq[j], m)
  }

  lc-event rec(0..N-1 j) { // receive message m from u[j]
    ec ¬empty(inq[j]) // message at head of inq[j]
    ac Msg m := remv(inq[j]);
    if m = (REQ, t) then {
      req[j] := t;
      ack[j] := t;
      c := max(c,t)+1 ;
      u[j].snd(i, (ACK, c) )
    }
    else if m = (REL, t) then {
      req[j] := 0;
      ack[j] := t;
      c := max(c,t)+1
    }
    else if m = (ACK, t) then {
      ack[j] := t;
      c := max(c,t)+1
    }
  }
}

lc-event bH() { // become hungry
  ec st = T
  ac st := H;
  c := c+1;
  req[i] := c
  for j in 0..N-1, j≠i do { u[j].snd(i, (REQ, c) ) };
}

lc-event bE() { // become eating
  ec st = H ∧ [∀0..N-1 j, j≠i :: req[i]<ack[j] ∧ ( req[j]=0 ∨ req[i]<req[j] ∨ (req[i]=req[j] ∧ i<j) )]
  ac st := E
}

lc-event bT() { // become thinking
  ec st = E
  ac st := T;
  c := c+1;
  req[i] := 0
  for j in 0..N-1, j≠i do { u[j].snd(i, (REL, c) ) };
}

progress-assumptions {
  Wfair(rec(j), bE(), bT()) // bH() has no fairness
}

```

Problem 1a [11 points] We prove that it satisfies

$$S_0 : [\forall 0..N-1 i, j, i \neq j :: \neg(u[i].st = E \wedge u[j].st = E)]$$

Notation: Below, i, j, k range over $0..N-1$. For any quantity x , we shorten $u[i].x$ to x_i . The sequence of REQ and REL messages in $\text{inq}_i[j]$ is denoted by $\text{inqr}_i[j]$. The sequence of timestamps of messages in $\text{inq}_i[j]$ is denoted by $\text{inqt}_i[j]$.

$$S_1 : \langle \text{ack}_j[k] \rangle \circ \text{inqt}_j[k] \circ c_k \text{ is non-decreasing}$$

$$S_2 : c_j \geq \text{ack}_j[k] \geq \text{req}_j[k]$$

$$S_3 : \begin{aligned} & \text{(a) } (\text{inqr}_k[j] = \langle \rangle) \vee \\ & \text{(b) } (\text{inqr}_k[j] = \langle (\text{REQ}, \text{req}_j[j]) \rangle) \wedge \text{st}_j = H \vee \\ & \text{(c) } (\text{inqr}_k[j] = \langle (\text{REL}, *) \rangle) \wedge \text{st}_j = T \vee \\ & \text{(d) } (\text{inqr}_k[j] = \langle (\text{REL}, *), (\text{REQ}, \text{req}_j[j]) \rangle) \wedge \text{st}_j = H \end{aligned}$$

$$S_4 : \text{st}_j = E \Rightarrow (\text{req}_j[j], j) = \min(\{(\text{req}_k[k], k) : \text{st}_k \neq E\})$$

$$S_5 : \text{st}_j \neq T \wedge \text{req}_j[k] \neq 0 \wedge (\text{req}_j[j], j) < (\text{req}_j[k], k) \Rightarrow \text{st}_k = H \wedge \text{req}_k[k] = \text{req}_j[k]$$

$$S_6 : \text{st}_j \neq T \wedge \text{req}_j[k] = 0 \wedge \text{req}_j[j] < \text{ack}_j[k] \Rightarrow (\text{st}_k = T \wedge c_k \geq \text{req}_j[k]) \vee (\text{st}_k = H \wedge \text{req}_k[k] = \text{req}_j[k])$$

$S_{1..6}$ satisfies the invariance rule and implies S_0 . (In fact, each of S_1, S_2, S_3 satisfies the invariance rule.)

Grading:

- 0 pt: Disproving assertion, non-assertional proof (including proof by contradiction).
- 3 pt: Predicates covering the relationship between the clock of a process, its timestamps on outgoing messages, and its timestamps at remote nodes (i.e., $S_{1,2}$ above).
- 3 pt: Predicates covering the relationship between the state of a process (T, H, E) and its REQ and REL messages in transit (i.e., S_3 above).
- 5 pt: Predicates covering the relationship between the state of a process, the timestamp of its request (if any), and its timestamps at remote nodes (i.e., $S_{4..6}$ above).

Problem 1b [4 points] We prove that it satisfies

$$P_0 : [\forall 0..N-1 i :: (u[i].st = H \rightsquigarrow u[i].st = E)]$$

Let F be number of pending requests whose extended timestamps are smaller than $u[i]$'s request's extended timestamp, i.e.,

$$F = |\{j :: \text{st}_j \neq T \wedge (\text{req}_j[j], j) < (\text{req}_i[i], i)\}|$$

This F satisfies the required conditions.

Grading:

- 4 pt: Describing an adequate F .