CALIFORNIA INSTITUTE OF TECHNOLOGY
Computing and Mathematical Sciences

CS 142

R. Murray, M. Chandy — **Homework Set #6** — Issued: 13 Nov 2019
Fall 2019 — Due: 20 Nov 2019

Note: Please keep track of the number of hours that you spent on this homework set (including reading) and enter this when you turn in your homework on Moodle.

1. Consider two UNITY program F and G and let $F \,[\!]\, G$ represent the union of the two programs. We write P in F to say that a property P holds in a program F.

   Show the following equivalences:

   (a) P **unless** Q in $F \,[\!]\, G \equiv (P$ **unless** Q in $F \wedge P$ **unless** Q in $G)$.

   (b) (FP of $F \,[\!]\, G) \equiv$ (FP of $F) \wedge$ (FP of G)

2. Consider two UNITY program F and G and let $F \,[\!]\, G$ represent the union of the two programs. We write $P \text{in} F$ to say that a property P holds in a program F. Prove the following statements formally or provide counterexamples with formal program definitions.

   (a) P **ensures** $\neg P$ in $F \implies P$ **ensures** $\neg P$ in $F \,[\!]\, G$ for any G.

   (b) $(P \leadsto Q$ in $F) \wedge ($**stable**(P) in $G) \implies P \leadsto Q$ in $F \,[\!]\, G$

3. Consider the following program:

   Program *GCD*
   var x, y : integers
   initially $x = X \wedge y = Y \wedge x > 0 \wedge y > 0$
   assign
   $x > y \;\rightarrow\; x := x - y$
   $[\!]$ $x < y \;\rightarrow\; y := y - x$

   (a) Write a program that uses superposition to count the number of actions that are executed before the *GCD* program reaches a fixed point. Write down your program formally similar to the one in the Lecture 7.1 slide 7.

   (b) Show that the program you write down is correct. In other words, show that your program terminates and the count is correct. You do not need to prove that the original program *GCD* terminates.

4. You are given a distributed system in which agents are organized in a ring. Agents are numbered i for $0 \leq i < N$ where $N > 1$. The only channels in the system are from agent i to agent $(i + 1) \mod N$.

   The system has a number of indivisible tokens. An agent can delete a token that it holds but cannot create tokens. Let n_i be the number of tokens held by agent i and $ch_{i,j}$ the state of (i.e. messages in transit in) the channel from agent i to agent j. The actions of agent i are:

(a) **Agent i sends a token to an agent j**

$$n_i > 0 \rightarrow n_i = n_i - 1 || ch_{i,j} := ch_{i,j}.append(token)$$

(b) **Agent j receives a token from an agent i**

$$ch_{i,j}.head = token \rightarrow n_j := n_j + 1 || ch_{i,j} := ch_{i,j}.remove(token)$$

(c) **Agent i deletes a token**

$$n_i > 0 \rightarrow n_i := n_i - 1$$

Appending a message to a channel appends the message to the tail of the sequence of messages in flight. Likewise, removing a message is receiving the message at the head of the channel, i.e., the earliest message in flight in the channel. $ch.head = x$ is the predicate: the message at the head of channel ch is x.

Proposed algorithm. Here is a proposal for an algorithm by which agent 0 obtains an upper bound on the number of tokens in the system. Agent 0 sends a special message called a *marker* on its outgoing channel. A marker message has a field called *count*; this field is 0 in the marker message sent by agent 0. When an agent, other than agent 0, receives a marker the agent adds the number of tokens it holds to the marker message and sends the updated marker to the next agent in the ring. When agent 0 receives the marker it claims that an upper bound on the number of tokens in the system is the *count* field of the marker message that it received plus the number of tokens that it received after it sent the marker out and before receiving the marker back.

Agent 0 has zero tokens initially and has the following (local) variables:

(a) *initiated* which is false initially and is made true when the agent initiates the algorithm.

(b) *terminated* which is false initially and is made true when the agent terminates the algorithm.

(c) r which is 0 initially and is the number of tokens that the agent receives after it initiates and before it terminates the algorithm.

(d) c which (the designer of the algorithm claims) is an upper bound on the number of tokens in the system.

$$invariant\ terminated\ \Rightarrow\ c \geq \sum_j (n_j + m_j)$$

where m_j is the number of tokens in flight in the channel from agent j.

Is the algorithm correct? Prove or give a counterexample.

Next the actions of the agents are given in detail; you can skip this section if the algorithm is clear from the description above.

(a) **Agent 0 initiates the algorithm** Agent 0 initiates the algorithm by sending a marker with a count of 0 on its outgoing channel.

$$\neg initiated \rightarrow initiated := true || ch_{0,1}.append(marker(0))$$

(b) **Agent i, $i \neq 0$, receives a marker with count k** When an agent other than agent 0 receives a marker with a count of k it sends a marker with a count of $k + n_i$ on its outgoing channel.

$$ch_{i-1,i}.head = marker(k) \rightarrow$$
$$ch_{i,i+1} := ch_{i,i+1}.append(marker(k + n_i))||ch_{i-1,i} := ch_{i-1,i}.remove(marker(k))$$

(c) **Agent 0 receives a token** The only change is that the agent increments r if the algorithm has been initiated but hasn't yet terminated.

$$initiated \wedge (\neg terminated) \wedge (ch_{-1,0}.head = token) \rightarrow$$
$$r := r + 1||n_0 := n_0 + 1||ch_{-1,-0} := ch_{-1,0}.remove(token)$$

(d) **Agent 0 receives the marker with count k** The agent sets c and terminates the algorithm

$$initiated \wedge (\neg terminated) \wedge (ch_{-1,0}.head = marker(k)) \rightarrow$$
$$c := r + k||terminated := true||ch_{-1,-0} := ch_{-1,0}.remove(marker(k))$$