Assignment 4

Comments:
In general, the proofs for this assignment were better than last time, but some still didn't seem to get the idea of a structured, step-by-step induction, and some presented arguments that did not prove (inductively or otherwise) the required properties.

On the other hand, a few people used induction to prove invariants, but the invariants they chose didn't prove the required properties.

A few need to review the definitions of actions, transitions, and traces to better understand the distinctions among them, and that we refer to a trace of a transition, or a trace of a series of transitions, but we ordinarily don't refer to a trace of an action.

Regarding simulation relation, a few people tried to make it a **function**, and missed the distinction between function and relation. And most of the rest seemed to be struggling with exactly how to describe the **relation** between the two channels (including the people who proved that the relation exists).

Refer again to the lecture notes, with automata A and B, and transition (s, π, s') of A, and **execution fragment** (not *transition*) (u, α, u') of B, and note that we do **not** say that $u = f(s)$ and $u' = f(s')$. Instead, we say $u \in f(s)$ and $u' \in f(s')$. So a given member of states(A) does not map exclusively to only one member of states(B); instead, there may be pairings of a member of states(A) to many members of states(B). Or to put it another way, in general, if s is a state of automaton A, f(s) is a **set** of states of automaton B. But the key is that among the various possible pairings, there exist some pairings (execution fragments) that satisfy the properties of the simulation relation, i.e. that for those particular fragments, the starting states of the two automata are the same, their ending states are the same, and the traces of the two fragments are the same. And to show that a simulation relation exists is to show that this relationship is true for **some**, not necessarily all, executions.

1. Prove "Theorem 4.1": If automata A and B have the same external signature and if there is a simulation relation f from A to B, then $traces(A) \subseteq traces(B)$.

2. Define a simulation relation from a Reliable FIFO Channel to a Lossy Channel.

Answer 1:

Induction hypothesis: Theorem 4.1 is true for an execution fragment of A consisting of a single transition -- this is an immediate consequence of Part 2 of the definition of Simulation Relation.

Let s be a reachable state of A and let $u \in f(s)$ be a reachable state of B. Let $[p_1,p_2,...,p_i] = acts(A)$.

Then by Part 2 of the definition of Simulation Relation, if $T=(s,p_1,s')$ is a transition of A, there is an execution fragment α of B beginning with u and ending with some $u' \in f(s')$ such that $trace(\alpha) = trace(T)$. Thus Theorem 4.1 is true for an execution consisting of 1 step.

Induction: if Theorem 4.1 is true for an execution of n-1 steps, then it is true for an execution of n steps.

Assume we have an execution fragment T of n-1 steps going from state s to state s' (s,s' ∈ states(A)) and an execution fragment α going from state u to state u' (u,u' ∈ states(B) and u ∈ f(s) and u' ∈ f(s')) such that trace(α) = trace(T), by Part 2 of the definition of Simulation Relation, if T'=(s',p_n,s'') is a transition of A, then there is an execution fragment α' of B beginning with u' and ending with some u'' /in f(s'') such that trace(α') = trace(T'). Therefore [trace(α) |- trace(α')] is a trace of an execution fragment of B that is equal to the trace [trace(T) |- trace(T')] of the n-step execution fragment T |- T' of A.

Therefore, any trace of A has an equal trace in traces(B) and therefore traces(A) ⊆ traces(B).

Answer 2:
Given a Reliable FIFO Channel with sig(RC) consisting of SendRC(i,j,m) and ReceiveRC(i,j,m) and states(RC) = RC.queue, initially \0, we can define a LossyChannel which differs from ReliableChannel only in that SendLC(i,j,m) sometimes fails to append m to LC.queue, whereas the effect of SendRC(i,j,m) is to always append m to RC.queue.

To define a simulation relation from LC -> RC, the appropriate relation f on states(LC) × states(RC) is equality – i.e. f(RC.queue) = LC.queue.

Now we must show that this relation exhibits the properties of a simulation relation:

- First note that the start state of both LC and RC is the empty queue.

- Next, if we look at execution fragments E_R and E_L beginning at some point where we have RC.queue = LC.queue and the next transition is Receive(i,j,m), after the Receive action we have tail(RC.queue) = tail(LC.queue) so f still holds. And for both channels the trace of this execution fragment is simply {Receive(i,j,m)} so trace(E_R) = trace(E_L).

- Finally, if at some point in an execution we have RC.queue = LC.queue and the next transition is Send(i,j,m), ***if*** LossyChannel does append m to LC.queue, the state of LossyChannel after the transition will be identical to the state of ReliableChannel after such transition, i.e. RC.queue |- m = LC.queue |- m, and for both channels the trace is {Send(i,j,m)} so again, trace(E_R) = trace(E_L).

(Alternatively, of course, LossyChannel might fail to append m to LC.queue, in which case its trace would be {[]}. But to show that a simulation relation exists, we only have to show that the required properties hold for some subset of LossyChannel's executions.)

That the simulation relation holds for execution fragments longer than one step can be seen by induction on the number of steps (same as in question 1).