Forge 4

Due: 11:59 PM Saturday, February 29, 2020

Your solutions for this assignment should terminate within 30 seconds (don't panic if it takes longer! Please reach out to a TA though).

Stencil Files

We have provided two stencil files for the two Forge problems. They can be found in `/course/cs1950y/pub/forge4`. You **must** follow the stencil.

Problem 1: Handshakes

There are five people (`Person`) at a party. There was some handshaking (`handshakes`), though it's not necessary that every pair of people shook hands. Of course, nobody shook their own hand, handshakes go both ways, and a pair of people shook hands no more than once.
Make the above definitions precise by filling in the `basicDefinitions` pred.

After you've done this, model the following scenarios:

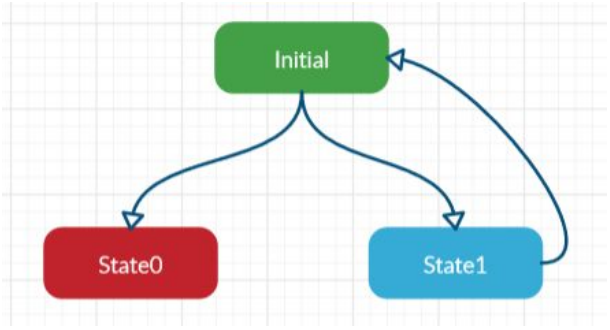
A. At the end of the party, everyone said they had shaken a different number of hands than everyone else. Is this actually possible? Express a constraint such that Forge will either find an instance of such a scenario, or show that no such instance exists.
B. Like before, everyone said they had shaken a different number of hands than everyone else. It turns out that exactly one person named `Liar` lied about how many hands they had shaken. Express this constraint. In a comment, state how many hands `Liar` actually shook (if there are multiple solutions, give only one).
C. Is there another solution in which `Liar` shook a different number of hands than the answer you found in (B)? `check` whether there is no other solution by writing another pred, `noOtherSolution`.

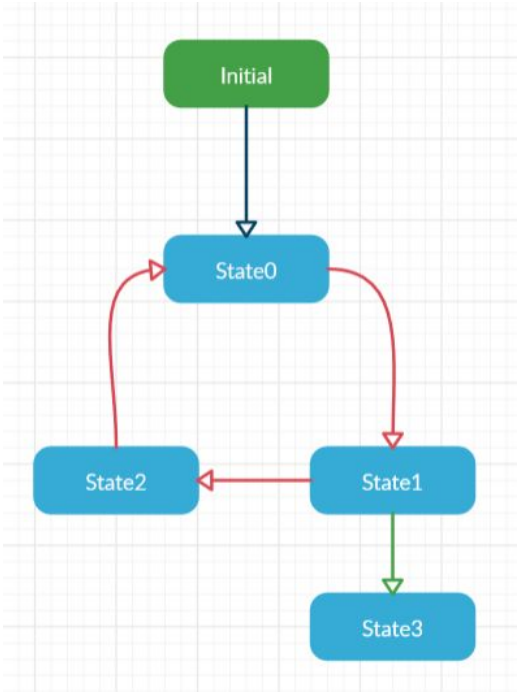
Problem 2: State Machine

A *state machine* is a directed graph that models how a system moves from `State` to `State` as it executes. It has one or more marked `Initial` (or starting) states, and edges connecting each state to its successor(s). An initial state can be the successor of another state. Construct an Forge model of a state machine, then write predicates that constrain Forge to produce instances of the following types of machine:

A. a *deterministic* machine, in which there is one initial state and each state has at most one successor
B. a *nondeterministic* machine, in which there are multiple initial states, or where at least one state has more than one successor, or both
C. a machine with at least one state that is *not reachable* from any initial state.

D. a machine where all states are *reachable* from some initial state (it need not be the same initial state for each one)
E. a *connected* machine in which every state is reachable (along the successor relation) from every other state
F. a machine with a *deadlock*: a machine with a state that is reachable from an initial state, but has no successors.
G. a machine with a *livelock*: a machine where there exists some cycle reachable from an initial state and a state (the "livelocked" state) reachable from the cycle that's not part of the cycle. Note that this livelocked state cannot be reached at any point before reaching the cycle or in the cycle itself.





Above, left: State0 is the deadlocked state in this machine.
Above, right: states 0, 1, and 2 form a cycle that could prevent State3 from ever being reached (livelock).

Problem 3: Case Study

Forge and its predecessor Alloy can be used to model real protocols and systems to find bugs. Please read *Using lightweight modeling to understand chord* and write a one-page response to it. In your response, please discuss the following:

- What result does the paper show?
- Describe in your own words three of the invariants that fail with the Chord ring election.
- Based on your past experiences in CS, what's another problem or system that it might be valuable to model, and why?

Note that we are not looking for a complete technical understanding of the Chord protocol. Instead, focus on how bounded model checkers were used and what sorts of counterexamples they exposed in the protocol.

Submit your response as a pdf named `case_study.pdf`. **Do NOT put your name or any other identifying information in the pdf!** If you do, you will lose points.

Handing In

Run `cs1950y_handin forge4` from a directory holding your 2 Forge files (`handshakes.rkt`, `state_machine.rkt`) and your case study as a pdf (`case_study.pdf`). You should receive an email confirming that your handin has worked.