FIT5171 Applied Session 11–12
System Testing and Finite State Machine

Week 11 – 12, 2024

1. Finite state machines can be very useful in system testing by capturing system functionality in terms of states, events, transitions, actions and guards. Suppose when we are creating an invitation, we need three inputs: a name, a location and a recipient.

   Draw a finite state machine depicting this scenario. Hint: Think about what states represent.

2. Finite state machines can be very useful in testing by capturing system functionality in terms of states, events, transitions, actions and guards. Suppose we want to implement a Web service that allows users to create proposals programmatically. A user will specify the user name, password and the details of the proposals to be created. The system creates the proposal upon successfully validating the provided information. A timeout function that puts a limit on how long the service can take to create an proposal will also be implemented. That is, the system only waits for a certain amount of time to receive user inputs. When the time is up, the system will go back to the state where it is ready to accept another request.

   Draw a system-level finite state machine depicting this scenario. If you are thinking of input validation, do not break it down into validation of different inputs now. Since the system is a long-running system, we will assume that there is no final state. That is, after an proposal is successfully created, the system goes back to the state where it is ready to create another proposal.

3. Formal verification techniques, including model checking and theorem proving, work on mathematical abstractions of software systems. Systems and properties to be verified are expressed in certain mathematical logics. Inference is then performed to formally verify the truthfulness of the property.

   Discuss the pros and cons of these formal verification techniques and compare them against testing.