4. Test Generation from Finite State Models

Model-Based Testing

Andrea Polini

Software Engineering II – Software Testing
MSc in Computer Science
University of Camerino

December 4th, 2014

Models in the Design Phase

Design Phase

- Between the requirements phase and the implementation phase "*The last you start the first you finish*"
- Produce models in order to clarify requirements and to better formalize them
- Models can be the source of test set derivation strategies

Various modeling notations for behavioral specification of a software system have been proposed. Which to use depends on the system you are developing and the aspects you would like to highlight:

- Finite State Machines
- Petri Nets
- Statecharts
- Message sequence charts

Finite State Machines

FSM

A finite state machine is a six-tuple $<\mathscr{X}, \mathscr{Y}, \mathscr{Q}, q_0, \delta, \mathscr{O}>$ where:

- $\mathscr{X}$: finite set of input symbols
- $\mathscr{Y}$: finite set of output symbols
- $\mathscr{Q}$: finite set of states
- $q_0 \in \mathscr{Q}$: initial state
- δ: transition function ($\mathscr{Q} \times \mathscr{X} \to \mathscr{Q}$)
- $\mathscr{O}$: output function ($\mathscr{Q} \times \mathscr{X} \to \mathscr{Y}$)

Typical extensions:

- Transition and output functions can consider strings
- Definiton of the set of accepting states $\mathscr{F} \subseteq \mathscr{Q}$
- Non determinism

Properties of FSM

Useful properties/concepts for test generation

- Completely specified (input enabled)
  - $\forall(q_i \in \mathcal{Q}, a \in \mathcal{X}).\exists q_j \in \mathcal{Q}.\delta(q_i, a) = q_j$
- Strongly connected
  - $\forall(q_i, q_j) \in \mathcal{Q} \times \mathcal{Q}.\exists s \in X^*.\delta^*(q_i, s) = q_j$
- V-equivalence (distinguishable)
  - Let M_1 and M_2 two FSMs. Let $\mathcal{V}$ denote a set of non-empty string on the input alphabet $\mathcal{X}$, and $q_i \in \mathcal{Q}_1$ and $q_j \in \mathcal{Q}_2$. q_i and q_j are considered $\mathcal{V}-$*equivalent* if $\mathcal{O}_1(q_i, s) = \mathcal{O}_2(q_j, s)$. If q_i and q_j are $\mathcal{V}-$*equivalent* given any set $\mathcal{V} \subseteq \mathcal{X}^+$ than they are said to be *equivalent* ($q_i \equiv q_j$). If states are not equivalent they are said to be *distinguishable*.

Properties of FSM....cntd

Useful properties/concepts for test generation...cntd

- Machine equivalence
  - M_1 and M_2 are said to be *equivalent* if $\forall q_i \in \mathscr{Q}_1 . \exists q_j \in \mathscr{Q}_2 . q_i \equiv q_j$ and viceversa.
- k-equivalence
  - Let M_1 and M_2 two FSMs and $q_i \in \mathscr{Q}_1$ and $q_j \in \mathscr{Q}_1$ and $k \in \mathbb{N}$. q_i and q_j are said to be $\mathscr{K} - equivalent$ if they are $\mathscr{V} - equivalent$ for $\mathscr{V} = \{s \in X^+ | \mid s \mid \leq k\}$
- Minimal machine
  - an FSM is considered *minimal* if the number of its states is less than or equal to any other *equivalent* FSM

Conformance Testing

Conformance Testing

Relates to testing of communication protocols. It aims at assessing that an implementation of a protocol conform to its specification.
Protocols generally specify:

- Control rules (FSM)
- Data rules

The Testing Problem

Testing problems ingredients:

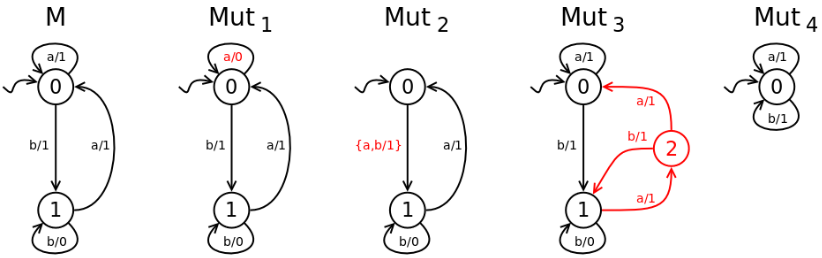
- Reset inputs ($\mathscr{X} = \mathscr{X} \cup \{Re\}$, and $\mathscr{Y} = \mathscr{Y} \cup \{null\}$)
- Testing based on requirements checks if the implementation conforms to the machine on a given requirement.
- The testing problem is reconducted to an equivalence (nevertheless finite experiments). Is the SUT (IUT) equivalent to the machine defined during design?
- Fault model for FSM – given a fault model the challenge is to generate a test set T from a design M_d where any fault in M_i of the type in the fault model is guaranteed to be revealed when tested against T
  - Operation error
  - Transfer error
  - Extra-state error
  - Missing-state error

Mutation of FSMs

Mutant

A mutant of an FMS M_d is an FSM obtained by introducing one one or more errors one or more times.

- Equivalent mutants: mutants that could not be distinguishable from the originating machine



The Testing Problem

Fault coverage

Techniques to measure the goodness of a test set in relation to the number of errors that it reveals in a given implementation M_i.

- N_t: total number of first order mutants of the machine M used for generating tests.
- N_e: Number of mutants that are equivalent to M
- N_f: Number of mutants that are distinguished by test set T generated using some test generation method.
- N_l: Number of mutants that are not distinguished by T

The fault coverage of a test suite T with respect to a design M is denoted by $FC(T, M)$ and computed as follows:

$$\begin{aligned} FC(T, M) &= \text{Number of mutants not distinguished by T /} \\ &\quad\ \text{Number of mutants that are not equivalent to M} \\ &= (N_t - N_e - N_f)/(N_t - N_e) \end{aligned}$$

Characterization Set

Let $M = < \mathscr{X}, \mathscr{Y}, \mathscr{Q}, q_1, \delta, \mathscr{O} >$ an FSM that is minimal and complete. A characterization set for M, denoted as $\mathscr{W}$, is a finite set of input sequences that distinguish the behaviour of any pair of states in M.