Testing

- Primary validation technique used in industry
- Part of system development life-cycle
- Expensive, error prone, time consuming (for Real-Time Systems)
- UPPAAL model can be used to generate test specifications
Model-Based Testing

- **Input:** Timed automata model of system and environment
- **Test-suite generation algorithm**
- **Output:**
  - Test input/output sequence $\sigma_{fp} = i_0, i_1, i_2, \ldots$
  - Test suite $T = \{\sigma_1, \ldots, \sigma_n\}$, minimized w.r.t global time or global length

Testing Verdict

- **Test specification** $\sigma_{fp} = i_0, i_1, i_2, \ldots$
- **Test in/output** $\delta_{fp} = i_0, o_0, i_1, o_1, i_2, i_3, \ldots$
- **Test Verdict:**
  - OK, if $\delta_{fp} = i_0, o_0, i_1, o_1, i_2, i_3, \ldots$ run of system model
  - NOK, otherwise
Testing Real-Time Systems

- Test input sequence $\sigma^p = \epsilon_{0,i_0}, \epsilon_{1,i_1}, \epsilon_{2,i_2}, \ldots$
- Test in/output $\delta^p = \epsilon_{0,i_0}, \epsilon_{1,o_0}, \epsilon_{1,i_1,o_1}, \ldots$
- Test Verdict:
  - OK, if $\delta^p = \epsilon_{0,i_0}, \epsilon_{1,o_0}, \epsilon_{1,i_1,o_1}, \ldots$ run of system model
  - NOK, otherwise
- Timed Automata?

Controllable Timed Automata

- **Input Enabled**: all inputs can always be accepted
- **Output Urgent**: enabled outputs will occur immediately
- **Determinism**: two transitions with same input/output leads to the same state
- **Isolated Outputs**: if an output is enabled, no other output is enabled
Test Purpose Generation

- Generate input sequence for a single test purpose
  - Reachability analysis
  - Example: “fast” output can be sent by system

- TP2.OK reachable?
- Test input: start!.delay(0).stop!.delay(0).fast?

Coverage Based Test Generation

- Systematic testing
- Coverage measurement
- Examples:
  - Location coverage,
  - Edge coverage,
  - Definition/use pair coverage
Coverage Based Test Generation

- Systematic testing
- Cover measurement
- Examples:
  - Location coverage,
  - Edge coverage,
  - Definition/use pair coverage
Coverage Based Test Generation

- Systematic testing
- Cover measurement
- Examples:
  - Location Coverage,
  - Edge Coverage,
  - Definition/Use Pair Coverage

Generated by modified model-checking algorithm in UPPAAL
Location Coverage

- Test sequence traversing all locations
- Encoding:
  - Enumerate locations \( l_0, \ldots, l_n \)
  - Add an auxiliary variable \( l_i \) for each location
  - Label each ingoing edge to location \( i \): \( l_i := \text{true} \)
  - Mark initial visited \( l_0 := \text{true} \)
- Check: \( EF( l_0 = \text{true} \land \cdots \land l_n = \text{true} ) \)

Edge Coverage

- Test sequence traversing all edges
- Encoding:
  - Enumerate edges \( e_0, \ldots, e_n \)
  - Add auxiliary variable \( e_i \) for each edge
  - Label each edge \( e_i := \text{true} \)
- Check: \( EF( e_0 = \text{true} \land \cdots \land e_n = \text{true} ) \)
Definition/Use Pair Coverage

Dataflow coverage technique

- Def/use pair of variable $x$:

  $x:=0$  \[ \rightarrow \]  \hspace{1cm} \text{definition} \hspace{1cm} \text{no defs} \hspace{1cm} x := A$  \[ \rightarrow \]  \hspace{1cm} \text{use}

- Encoding:
  - $v_d \in \{ \text{false} \} \cup \{ e_p, ..., e_n \}$, initially false
  - Boolean array $du$ of size $|E| \times |E|$ at definition on edge $i$: $v_d := e_i$
  - At use on edge $j$: if($v_d$) then $du[v_d,e_j] := \text{true}$

- Check:
  - $\text{EF( all } du[i,j] = \text{true})$
Test Suite Generation

- In general a set of test cases is needed to cover a test criteria
- Add global reset of SUT and environment model and associate a cost (of system reset)

\[ \sigma = \epsilon_0, i_0, \ldots, \epsilon_1, i_1, \text{reset} \epsilon_2, i_2, \ldots, \epsilon_0, i_0, \text{reset}, \epsilon_1, i_1, \epsilon_2, \ldots \]

- Test suite \( T = \{ \sigma_i, \ldots, \sigma_n \} \) with minimum cost

CoVer Tool

- Extension of UPPAAL
  - Graphical user interface
  - Timed Automata models
  - Symbolic on-the-fly algorithms
- Graphical input language for coverage criteria, a.k.a. Observers
- Generation of test specifications
Application w. Ericsson

- Automatic generation of test programs
- Systematic tests with coverage