Generating Complete and Finite Test Suite for **ioco**

Is It Possible?

Adenilso Simao
Alexandre Petrenko

adenilso@icmc.usp.br

April/06/2014
9th Workshop on Model-Based Testing
Grenoble, France

Introduction

Introduction (II)

Introduction (III)

Introduction (IV)

Context

- Model Based Testing
  - Modeling
  - Test Generation
  - Test Execution
  - Test Analysis

Modeling

- Many different models
  - State based
    - Finite State Machines (FSM)
    - Input/Output Transition System (IOTS)

Test Generation

- Tests should be
  - Sound
    - Every implementation that fails is indeed faulty
  - Exhaustive (for a given fault domain)
    - Every fault implementation fails
- Test generation can be
  - On the fly
  - Preset
  - Adaptive

Test Execution

- Interaction with system
- For FSM, it is straightforward
- For IOTS, it can be
  - Synchronous
    - Controllability problems
    - Input/Output Conflict
  - Asynchronous
    - Decidability problems

Problem Statement

- Is it possible to generate complete (sound and exhaustive) test suites for IOTS?
  - A la Finite State Machines
- Conformance relation
  - **ioco**

Problem Statement (II)

- It is known that:
  - It is decidable for FSMs
    - Many methods available
  - It is undecidable whether two IOTSs are equivalent, when interacting via FIFO queues (Hierons, 2012)
    - In the general case
- But,
  - Is there a subclass of IOTS models for which can generate complete test suites?
    - Under which assumptions?

Existing solutions

- I/O conflicts yield uncontrollable tests
  - Solved via FIFO queues (or ignored)
- Nondeterministic test suite generation
  - Theoretically complete, but unbounded

Summary of Results

- Test generation method which produces tests that are
  - Finite (and bounded)
  - Controllable
  - Sound
  - Exhaustive for a given class of faults

Input Eager IOTS

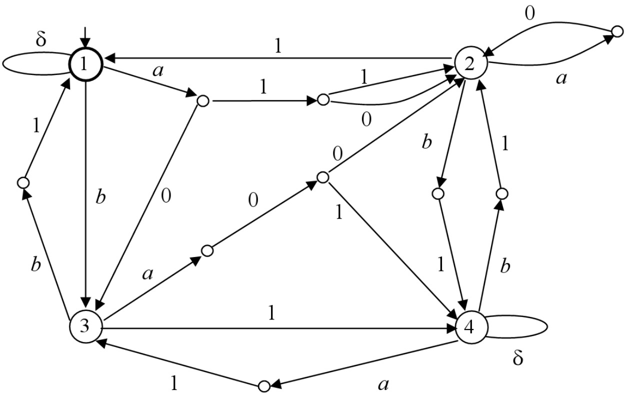
- The key assumption
  - The implementation is assumed to be eager for inputs
  - It solves conflicts favoring inputs over outputs

Input/Output Transition Systems

Definition

$(S, s_0, I, O, h_{\mathbb{S}})$, where S is a finite set of states and $s_0 \in S$, is the initial state, I and O are disjoint sets of input and output actions, respectively, and $h_{\mathbb{S}} \subseteq S \times (I \cup O) \times S$ is the transition relation.

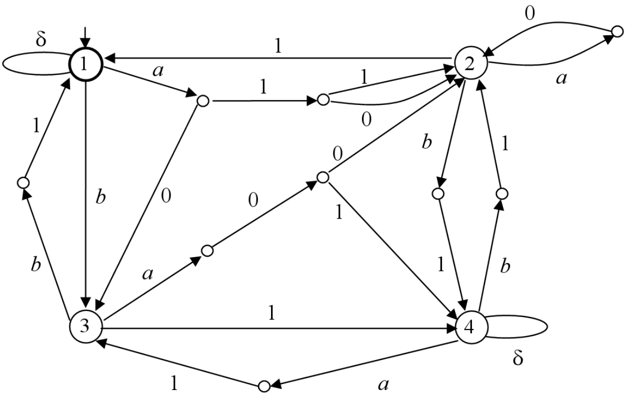
Example



Input/Output Conflicts

- Input states
  - Only inputs are enabled
  - Quiescent
- Bridge traces
  - From input state to input states
- Quasi-stable state
  - There is a conflict between inputs and outputs

Input/Output Conflicts (II)



Generating Complete Tests

- HSI-method for FSMs
  - Uses sets of distinguishing input sequences,
    - Harmonized state identifiers, one per state
    - Any two identifiers share an input sequence which distinguishes the two states
  - Appended to state and transition covers
    - Checks that every state of the implementation corresponds to some state of the specification
    - Checks that every transition of the implementation corresponds to a transition of the specification
- Complete for a given fault domain
  - All implementations which correspond to an FSM with at most a given number of states

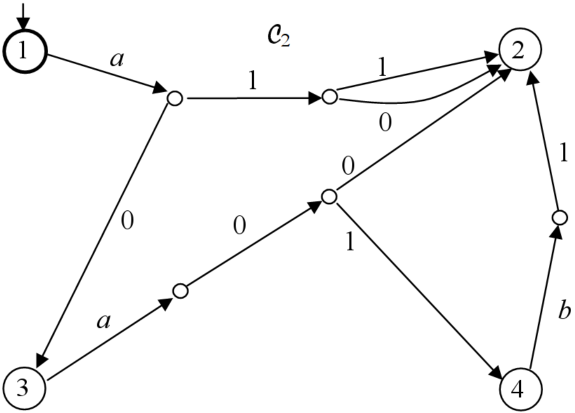
Generating Complete Tests (II)

- Inspired by HSI-method
  - Fault domain
    - All (input eager) IOTS with at most as many input states as the specification
- State reachability
- Transition coverage
  - Bridge traces
    - $Cov(s, x)$
    - (s, x)-cover
- State distinguishability

State Reachability

- Guarantees that s is reached in any conforming implementation
- Preamble $\mathcal{C}_s$ for state s
  - A submachine of the specification
    - Single-input
    - Acyclic
    - Output-preserving
- We propose an algorithm for computing preamble

State Reachability (II)



State Reachability (III)

- Input state cover
  - Set of preambles, one for each input state
- Transition cover
  - Preambles of the state cover followed by bridge traces

Transition Cover

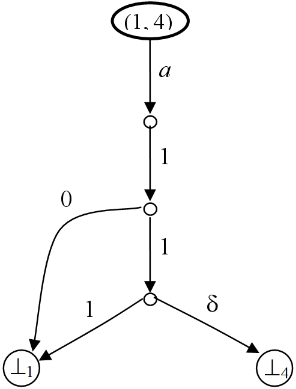
- A *transition cover* V of $\mathcal{S}$ is the set of preambles of an input state cover chained with (s, x)-covers

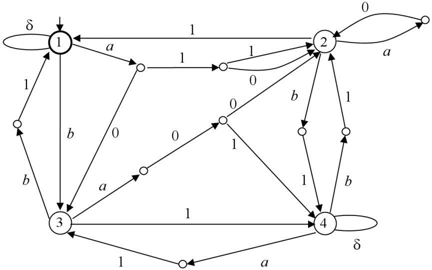
State Distinguishability

- Given two input states s_1 and s_2, how to decide in which of the states the implementation is in
- Separator $\mathcal{R}(s_1, s_2)$
  - Single-input acyclic machine
  - Traces are disjoint
  - Two sink states, one for each state (s_1 and s_2)
- We propose an algorithm for generation of a separator
  - Based on product machine
- Minimality
  - There is a separator for each pair of states

Separators

$\mathcal{R}(1, 4)$

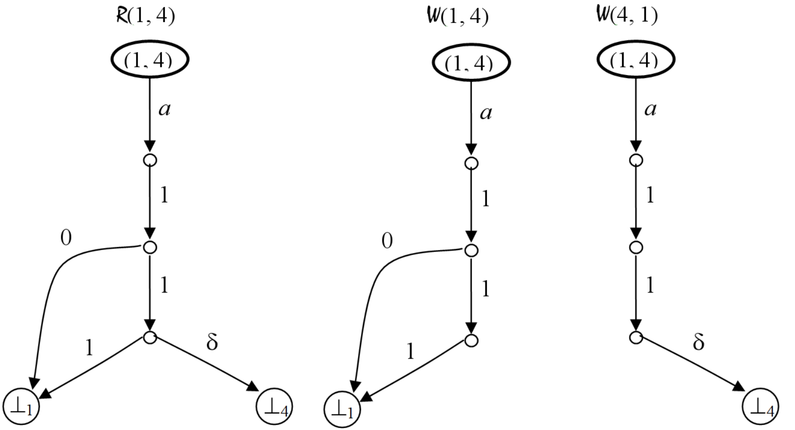




Distinguishers

- Distinguisher $\mathcal{W}(s_1, s_2)$
  - Derived from $\mathcal{R}(s_1, s_2)$
  - Removing one of the sink states

Distinguishers (II)



State Identifiers

- State identifier $\mathcal{ID}(s)$
  - Set of distinguishers
  - One for each input state different from s

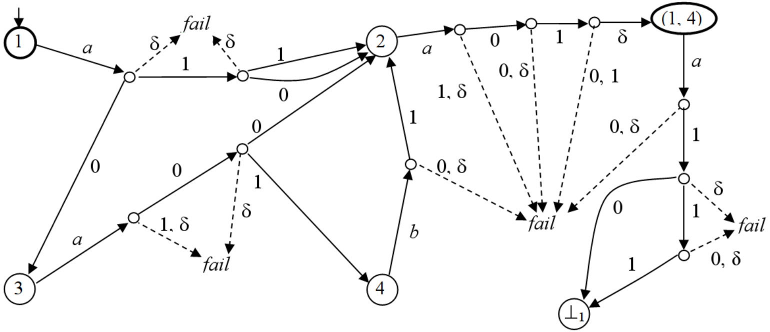
Harmonized State Identifiers

- Each pair of state identifiers has common prefix
  - Long enough to distinguish the states
- HSI

Complete Test Suite

- The set of IOTSs obtained by chaining each IOTS from the input state cover and transition cover with a corresponding harmonized state identifier
  - $D = \{\mathcal{T} @_s \mathcal{R} \mid s \in sink(\mathcal{T}), \mathcal{T} \in (Z \cup V), \mathcal{R} \in \mathcal{ID}(s)\}$, where
  - $sink(\mathcal{T})$ is the set of sink states of $\mathcal{T}$
  - Z is a state cover.
  - V is a transition cover
- Complete with fail state

Example



- Test Case $TC(\mathcal{C}_2 @_2 Cov(2, a) @_1 \mathcal{W}(1, 4))$.

Conclusion

- Generating Complete and Finite Test Suite for **ioco**: Is It Possible?
  - (qualified) Yes, it is.
  - For some IOTS
    - All input states are reachable
    - All input states are distinguishable
    - There are harmonized state identifiers
  - Under certain assumptions
    - There is no more input states in the implementation than in the specification
    - The implementation is eager for inputs
- We have proposed a method similar to HSI for FSM

Future Work

- Relax the constraints and assumptions

Generating Complete and Finite Test Suite for **ioco**

Is It Possible?

Adenilso Simao
Alexandre Petrenko

adenilso@icmc.usp.br

April/06/2014
9th Workshop on Model-Based Testing
Grenoble, France