State Machine Slicing for Optimizing Test Case Generation for UML-RT Models

Reza Ahmadi
Supervisor: Prof. Dr. Juergen Dingel

Outline

- Unit Testing UML-RT Models
- Using Slicing for Testing UML-RT Models
- A Quick Tool Demo
- Summary and Future Work

Unit Testing UML-RT Models

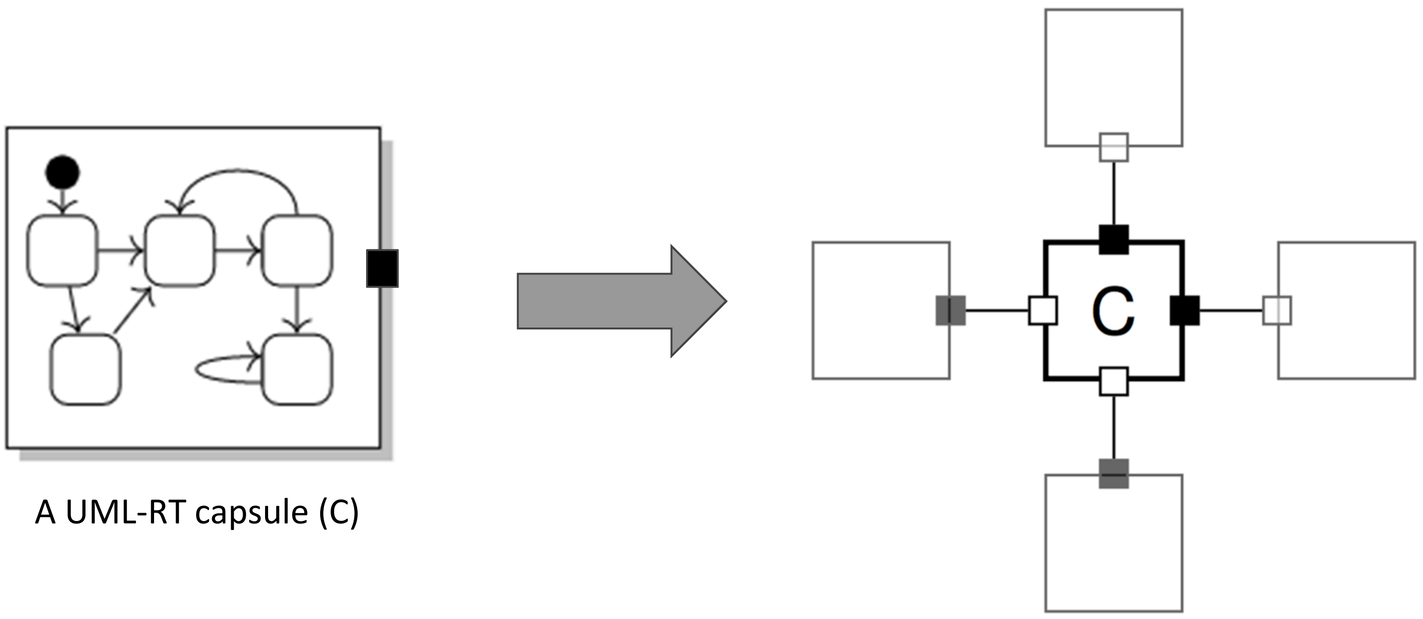

Fig 1. A UML-RT Capsule Communicates Other Capsules Connected to it

Extracting a Capsule and Driving it by Test Inputs

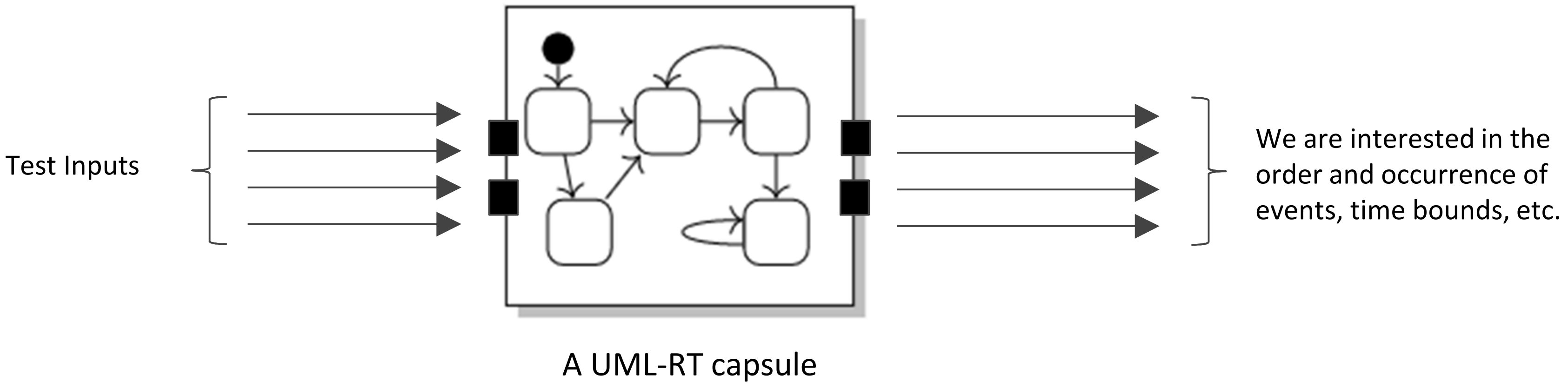
Test Inputs
We are interested in the order and occurrence of events, time bounds, etc.
A UML-RT capsule

Fig 2. Extracting a Capsule and Driving it by Test Inputs

Testing Using Symbolic Execution

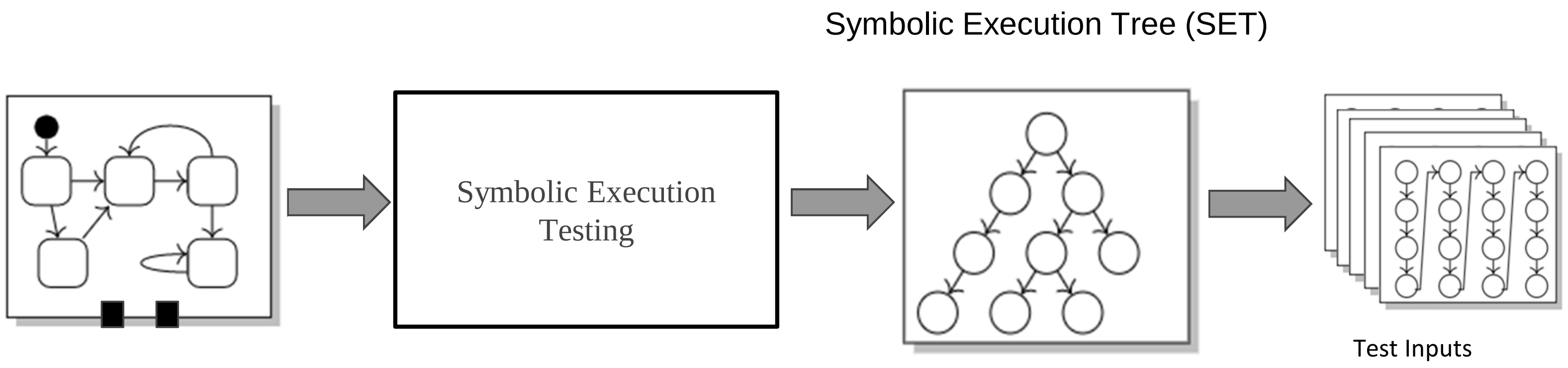

Fig 3. Test case generation Using Symbolic Execution

Other Test Generation Techniques

- Test generation using a test budget (Random/Sequential)
  - Easy to implement and light to execution
  - Many bugs may remain hidden after test budget limit is reached
- Using symbolic execution approaches
  - Can catch bugs hidden deep in the state machine
  - Downside is it needs heavy computations, can simply end up **Path Explosion**

Slicing to the Rescue

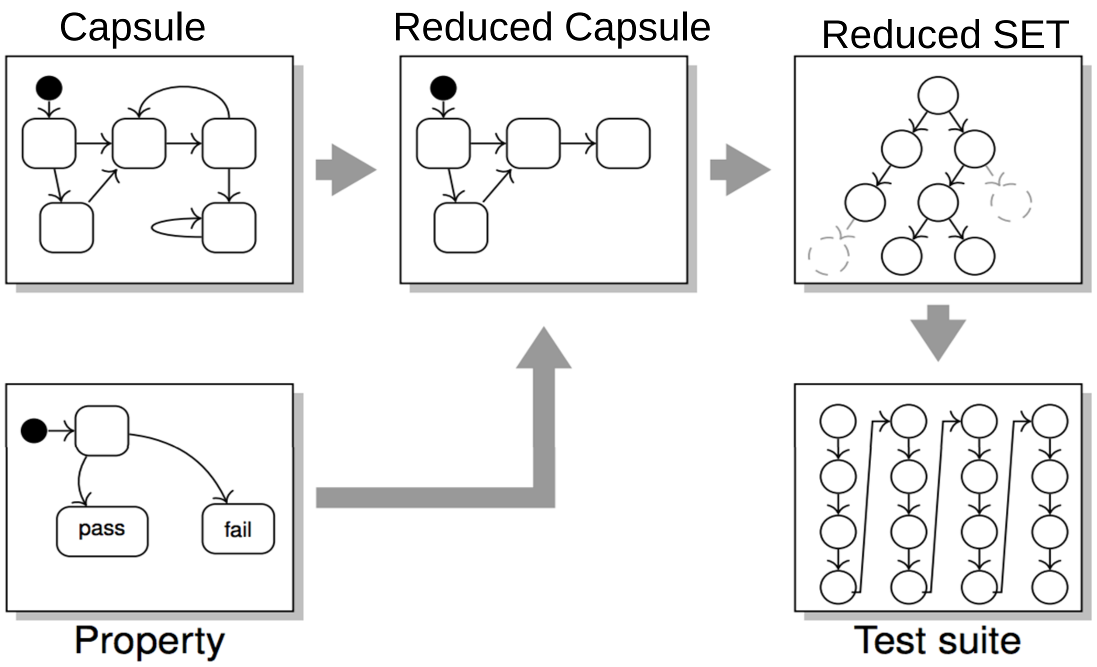

Fig 4. Using Slicing for Optimizing Test Case Generation

Tool Demo!

- Trying our slicing tool to slice a simple state machine
- Observing how slicing can contribute in reducing symbolic execution time of a state machine
    - We do not generate tests from symbolic execution tree

Summary and Future Work

- We slice state machines to reduce the size of state machines
  - For optimizing test case generation
- Current tool works perfectly for a subset of UML-RT, features to be added
  - Support for slicing composite capsules
  - Support for slicing state machines with timers

Thank you