

Extracting Finite-State-Machine from Code

Presented by: Hui Shen

05/06/2008

Outline

- Related work
- Goal
- Algorithm
- Implementation
- Conclusion

Related work

- Some related work about extracting UML notations from existing code
 - Class diagram
 - Sequence diagram
- Some tools have UML feature
 - NetBeans IDE
 - RED project

Related work

- Extracting Finite-State-Machine (FSM) from Code is harder than extracting sequence diagram and other static diagrams.
- Not too many works about extracting FSM
 - Bandera for model checking. Output model checking code.

Goal

- Extracting finite-state-machine from existing code.
 - Input: A function and a variable to track.
 - Output: State machine of the variable.
- Software Understanding

Algorithm

- For FSM, there are two important components.
 - State
 - Transition
- For state
 - Each assignment of variable must be a state.

Algorithm

- For transition
 - Between two simple assignments
 - In control structure (“if”, “while”, “switch”)
 - Between two control structures

Implementation

- Build abstract syntax tree of code
- Find each variable declaration and build state
- For each state, find the previous state(s) and build transitions.

Implementation

```
PROGRAM test
VAR state AS INT ;

BEGIN
  state := 0 ;

  IF state == 0 THEN
    state := 1 ;
  ELSE state := 2 ;
  END ;

  WHILE state <= 3 DO
    state := state + 1 ;
  END ;

  SWITCH state
  CASE 1 :   state := 5 ;
  CASE 2 :   state := 6 ;
  DEFAULT : state := 7 ;
  END ;

  state := 8 ;

END
```

Test case



Output of test case

Conclusion

- The project extracts finite state machine from existing code (single function) successfully.

Reference

- Bandera : Extracting Finite-state Models from Java Source Code. James C. Corbett, Matthew B. Dwyer, John Hatcli, Shawn Laubach, Corina S. Pasareanu, Robby, Hongjun Zheng
- OMG. UML 2.0 Infrastructure Specification. Object Management Group www.omg.org 2007