Lecture 7
Finite State Machines

Kenneth M. Anderson
Foundations of Software Engineering
CSCI 5828 - Spring Semester, 2000

Today’s Lecture

• Explore Finite State Machine issues
• Present a FSM-like language called SDL
• Discuss Homework 2

Finite State Machines (FSMs)

• Formal Definition
  \[ M = \{Q,I,\delta\}, \text{ where} \]
  \[ Q \text{ is a finite set of states} \]
  \[ I \text{ is a finite set of inputs} \]
  \[ \delta \text{ is a transition function} \]
  \[ \delta : Q \times I \rightarrow Q \]
  \( \delta \) can be a partial function

Finite State Machines (FSMs)

• Graph Representation
  – Nodes represent states
  – Arcs are directed and labeled with elements of \( I \)
  – Arc labeled \( i \) goes from state \( q_1 \) to state \( q_2 \)
    iff \( \delta(q_1, i) = q_2 \)
An Example

- \( Q = \{ q_1, q_2, q_3 \} \)
- \( I = \{ i_1, i_2, i_3, i_4 \} \)
- \( \delta = \)

<table>
<thead>
<tr>
<th>( q_1 )</th>
<th>( q_2 )</th>
<th>( q_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i_1 )</td>
<td>( i_2 )</td>
<td>( i_3 )</td>
</tr>
</tbody>
</table>

Finite State Machines (FSMs)

- Execution Model
  - Machine in some state
  - Input causes state change according to \( \delta \)
- Common Extensions
  - \textit{Start} states and \textit{stop} states
  - Output generated upon state transition
    - \( \delta : Q \times I \rightarrow Q \times O \)

Example

- \( O = \{ o_1, o_2, o_3, o_4 \} \)

Advantages of FSM Model

- Simple
- Obvious graphical representation
- Easy to Build Support Tools
  - Transformers
    - Transform FSM Model into other representations
  - Analyzers
    - Will this FSM run forever? Is it possible for it to halt? Are the state sequences infinite?
Shortcomings of FSM Model

- Theoretical Limit on Computational Power
  - FSM has no “memory”
  - Using states as memory is inefficient
    - Consider modeling a cruise control system with states that model car speed
    - 8-bit register = $2^8 = 256$ states!
- State Space Explosion for Large Problems

“No Memory” Problem

We would like to apply all of the defined test cases but a finite state machine cannot guarantee that $D^N = A^N$

Shortcomings, continued

- Inherently Synchronous
  - FSM in single, global state at each time instant
- State Space Explosion for Composed FSMs
  - States are multiplicative
  - On next slide, imagine composing the producer/consumer/buffer state machines into one FSM. See the result in the textbook on page 173, Figure 5.16

Producer/Consumer Example

2-unit Buffer
FSMs as Recognizers

Levels of Complexity

- Turing Machine
  - Unbounded tape
- Linear-Bounded Automata
  - Bounded tape
- Push-Down Automata
  - stack
- Finite State Machines
  - limited computational power but its simple to understand and program
- Programming Languages
  - Execution Semantics
- Context Sensitive Langs.
  - Language Semantics
- Context Free Grammars
  - Syntax
- Regular Expressions
  - Lexical Structure

An FSM-Based Tool: SDL

- Used Widely for Telephony Applications
- Extended FSMs
  - Modularity
  - Channel
- Tools
  - Analysis
  - Simulation
  - Code-generation

Homework 2

- Use a Finite State Machine to describe the possible state transitions for our home security system
  - Use only the notations presented in this lecture
- Retrieve the assignment from the Website
  - You may turn in hardcopy in class, or send the diagram via e-mail as a postscript or PDF attachment