Finite State Machines

Reading: Finite-State Machines Packet

Caffeine Break

• When purchasing soda from a vending machine we insert nickels (n), dimes (d), and quarters (q) (assume no paper money)
• We select a soda (s) when we have enough money
• We can model a vending machine’s behavior using a finite state machine (FSM)
  – Abstract model of a system (physical, biological, mechanical, electronic, or software)
  – Consists of inputs, states, and transitions
  – States can be initial or final
• For a vending machine, what are the possible inputs, states, and transitions?
Vending Machine FSM Break Down

- Inputs: n, d, q, s
- State: circles/bubbles labeled with the amount of money in the machine for a single purchase
  - Initial State: 00 (i.e. no money is in the machine)
  - Final States: states with a total input greater than or equal to 30 cents – represented by double circles
- Transitions: arrows labeled with the inputs that caused the change of state

Assume a soda costs 30¢
Potential Valid Input: n, d, q, s
Potential Invalid Input: n, s
Finite State Machines

• Key components
  – Finite (countable) number of states
    • Represent internal “memory”
    • Store information about what has happened before
  – Transitions
    • Response of the system to its environment (input)
    • Depends on current state and current input
    • Often, but not always, results in a change of state

ATM FSM

• Let's create an FSM for an ATM machine
• What are user inputs?
• What states can the machine be in?
  – States do not always have to have a label, which means a higher level of abstraction
    • a state could represent a UI prompting the user for the amount of money to withdrawal
• What are the transitions generated by inputs?
  – Transitions can also represent high level abstractions of user inputs
FSM and Programming Languages

- FSMs may be used as a precise description of programming language syntax
  - Used in compilers
- Example: Real Constants in Pascal
  - A floating-point number in Pascal is written as a string of digits that contain a decimal point and optionally a sign
  - There must be at least one digit before and after the decimal point
  - Floating-point data may also be represented using Pascal scientific notation
    - Sign, following by a floating-point number, followed by \( E \), another sign, and an integer (+ signs may be omitted)
Real Constants in Pascal

- What states does the FSM move through with the following inputs? Are these valid inputs (i.e. do they lead to a final state)?
  - 0.9
  - 9.
  - .9
  - 9.0
  - -9.3E15
  - 1.0.0

Text Processing FSMs

- Let’s start with a simple example – suppose our alphabet consists only of ‘a’ and ‘b’
- We can write an FSM that will search for the string “abba”
- An input string like “aababbbabba…” contains “abba” if we reach state 4
Text Processing FSMs (2)

<table>
<thead>
<tr>
<th>Input</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>b</th>
<th>b</th>
<th>b</th>
<th>a</th>
<th>b</th>
<th>b</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Does our FSM recognize the string “abba” in “aababbbabba”?

Text Processing and Code!

- Consider the following
  - Word: maximal sequence of zero or more upper- and zero or more lower-case letters
  - For the sequence “Go State!”
    - Is “State!” a word?
    - Is “tat” a word?
    - How about “irregardless”?
  - \( wc \) is the word count, initially 0
  - \( lc \) is the line count, initially 0
  - \( cc \) is the character count, initially 0
- Create an FSM that recognizes “words” and counts the number of words, lines, and characters
  - Inputs, States, Transitions
  - Other data we need to track?
Word FSM

- Contains both inputs (\n, A-Za-z, and other) and actions (++lc, ++cc, ++wc)
- What do states 0 and 1 represent?
- Why no final state?

Transition Table for Word FSM

<table>
<thead>
<tr>
<th>State</th>
<th>A-Za-z</th>
<th>\n</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1: ++wc, ++cc</td>
<td>0: ++lc, ++cc</td>
<td>0: ++cc</td>
</tr>
<tr>
<td>1</td>
<td>1: ++cc</td>
<td>0:++lc, ++cc</td>
<td>0: ++cc</td>
</tr>
</tbody>
</table>
Translating an FSM into Code

• Standard idiom: while-switch idiom
  – While loop gets each character
  – Switch statement inside the while loop executes the code based on current state

Switch Statements

• Switch statement is a control structure similar to if/else
• Cases are executed based on an integer value
• If the calculated value does not match any listed values then no case code is executed, unless…
• The optional default case is implemented
• Remember to include break after the statements for each case!!! (unless you explicitly intend to fall through)

    switch (<integer expression>) {
        case <value>:
            <statements>
            break;
        case <value>:
            <statements>
            break;
        …
        default: //optional
            <statements>
            <break>
    }
Switch Examples

```java
Scanner in = new Scanner(System.in);
System.out.println("Place?");
int place = in.nextInt();

switch (place) {
    case 1:
        System.out.println("1st!");
        break;
    case 2:
        System.out.println("2nd!");
        break;
    case 3:
        System.out.println("3rd");
        break;
}
```

```java
Scanner in = new Scanner(System.in);
System.out.println("Place?");
int place = in.nextInt();

switch (place) {
    case 1:
    case 2:
    case 3:
        System.out.println("Medal!");
        break;
    default:
        System.out.println("No medal");
        break;
}
```

WordCounter

- Take a look at the code implementation in the FSM reading (Figure 7 on Page 9)
Practice FSM

- Create an FSM that would recognize the acronym “lol” (case insensitive) in a String of input (like a text message)
  - By word, I mean that “lol” would be surrounded by whitespace, end with punctuation (non-letter and non-number), or be the only text in a String
    - Lol - YES
    - Lollipop - NO
    - LOL. - YES
    - Alol - NO

- Implement your FSM in a method `containsLol()` that accepts a String of text and returns true if the String contains “lol”
- We’ll copy method implementations into a test program and see if they pass

References

- “Finite-State Machines,” by Suzanne Balik and Matthias Stallmann