Finite State Automata

COMP370

Intro to Computer Architecture

Stepping Back from the Hardware

- Finite State Automata (FSA) are a means of modeling behavior
- They are used in programming and in hardware design.
- We will first talk about Finite State Automata concepts and then learn to create machines that implement them.

Combinatorial vs. Sequential

Two types of "combination" locks

Combinatorial
Success depends only on the values, not the order in which they are set.

Sequential
Success depends on the sequence of values (e.g., R-13, L-22, R-3).

State of Sequential Lock

- Our lock example has four different states, labeled A-D:

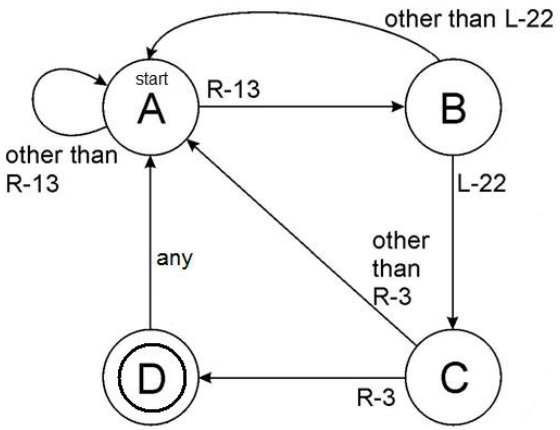
A: The lock is not open, and no relevant operations have been performed.

B: The lock is not open, and the user has completed the R-13 operation.

C: The lock is not open, and the user has completed R-13, followed by L-22.

D: The lock is open.

Combination Lock FSA



FSA Components

- A finite state machine (FSM) or finite state automaton (FSA) (plural: *automata*) is a model of behavior composed of:
  - States
  - Transitions
  - Actions

States

- A state is the current position in a sequence of steps.
- States are often represented as circles in diagrams of a FSA.
- The current state is the "memory" of the system.
- An FSA can only be in one state at a time.
- During the use of an FSA, the system changes from state to state.

Start and Final States

- There is one start state. The FSA is in the start state at the beginning.
- There may be one or more final states. If the FSA is evaluating input, a final state indicates correct input.

Transitions

- A transition is a change from one state to another state (or possibly back to the same state).
- Transitions are usually indicated by arrows in a FSA diagram.
- Transitions are triggered by some input. If there is no input, there is no transition.
- If there is no transition specified for an input, we assume the FSA goes to a terminal failing final state.

Actions

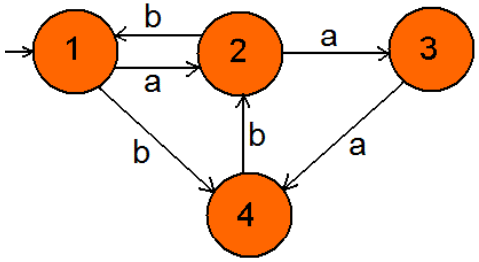
- Some FSA will take an action or output something during execution. Some FSA do not take any actions or only an action at a final state.
- **Mealy Machine** output is associated with the transition
- **Moore Machine** output is associated with the state

FSA Use

- Finite State Automata are used in hardware, software and documentation.
- Compilers and many input parsers use FSA
- Sequential circuits are often modeled with an FSA.
- Documentation for network protocols is often described (and implemented) using FSA.

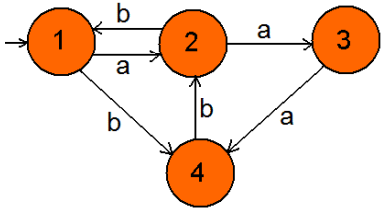
FSA Execution

- The FSA changes state with input.



After input of **a b b** the FSA will be in state 4.
After input of **a a a b** the FSA will be in state 2

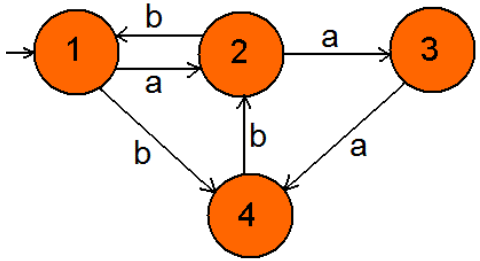
In what state will the FSA be after input of "b b a a"?



1. 1
2. 2
3. 3
4. 4

Undefined Transitions

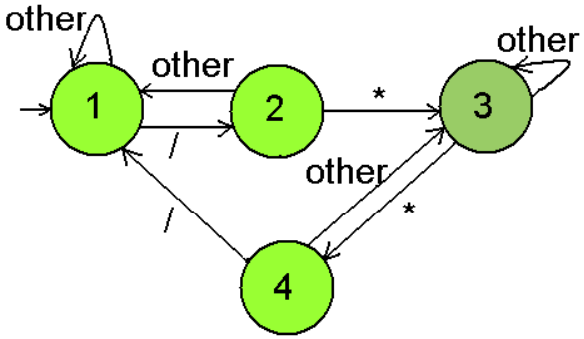
- We will assume the FSA stops at undefined transitions



After input of **b a** the FSA will stop.

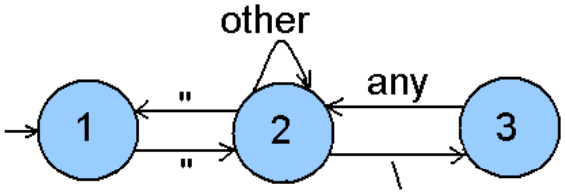
Input Parsing

This FSA detects Java /* comments */



Another Java Example

This FSA finds the quote strings in a Java program

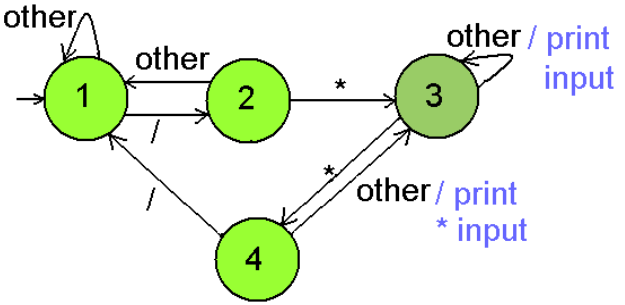


Mealy Machine

- A Mealy Finite State Automata performs some action when a transition occurs.
- The actions do not change the state of the FSA
- Actions are usually related to what the FSA is supposed to accomplish.
- An action might do something with the input.
- Not every transition needs to perform an action.

Input Parsing

This FSA prints the text of Java /* comments */

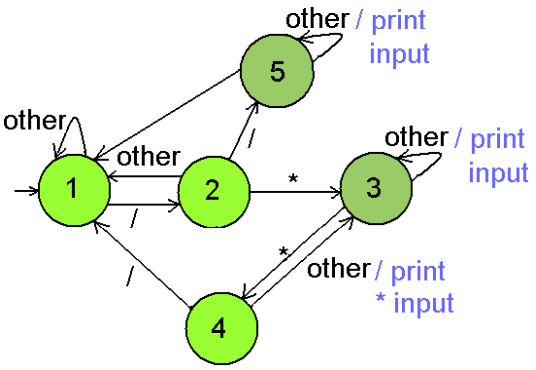


Write an FSA that will print the text of // format Java comments.

- Java comments start with two slashes and end with and end of line character '\n'

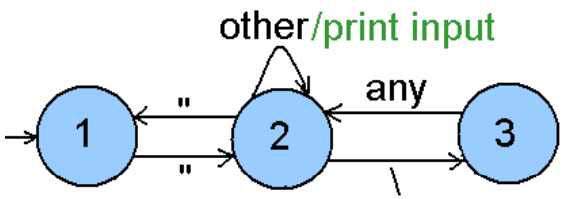
Combining FSA

Different FSA can be combined



Another Mealy Machine

This FSA prints the text of Java quote strings.



Combine the Quote and Comment FSA

- Using the FSA for printing quote strings and the FSA for printing comments, combine them to print both quotes and comments.

Back to combinatorial logic

Removing Leading Zeroes

- Consider a system with a four digit display (four 7 segment displays).
- The input is 4 groups of 4 binary bits representing the four decimal digits.
- Using binary-to-7-segment display devices and other gates, prevent leading zeroes from being displayed.

Without Leading Zero Removal