Behaviour Diagrams

Behaviour Diagrams

- Structure Diagrams are used to describe the static composition of components (i.e., constraints on what intstances may exist at run-time).
- Interaction Diagrams are used to describe the communication between the various components.
- Deployment Diagrams are used to describe the mapping between software artifacts and deployment targets.
- Behaviour Diagrams are used to describe the behaviour
  - Of the whole application
  - Of a particular process in an application
  - Of a specific object in an application

Different Formalisms ...

- We will look at different formalisms/languages for describing behaviour:
  - Finite State Automata
  - Activity Diagrams
  - Statecharts

Finite State Automaton

- A finite automaton consists of
  - Set of states
  - Input alphabet (of input “events”)
  - Rules for changing state
  - Start State (exactly 1)
  - Accept State(s) (when used for language recognition)

Formal definition, from Sipser's Theory of Computation

Example : Automatic Door

Automatic Sliding Door

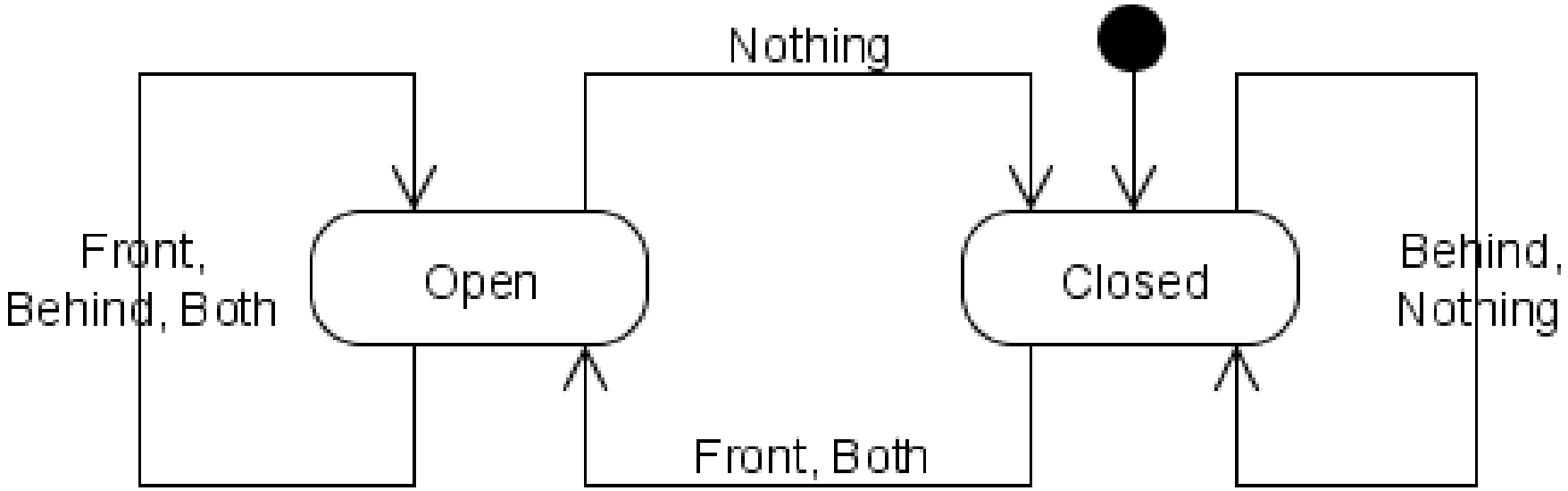
Specification

- The automatic door can be opened or closed.
- The sensor at the top of the door can send 4 types of signals:
  - Nobody : There is nobody in front nor behind the doors.
  - Front: There is somebody in front of the doors.
  - Behind: There is somebody behind the doors.
  - Both: There is somebody in front nor behind the doors.
- The door behaves as follows:
  - The door opens when somebody is in front of the doors.
  - The door closes only when nothing is in front or behind the doors.
- The door starts off closed.

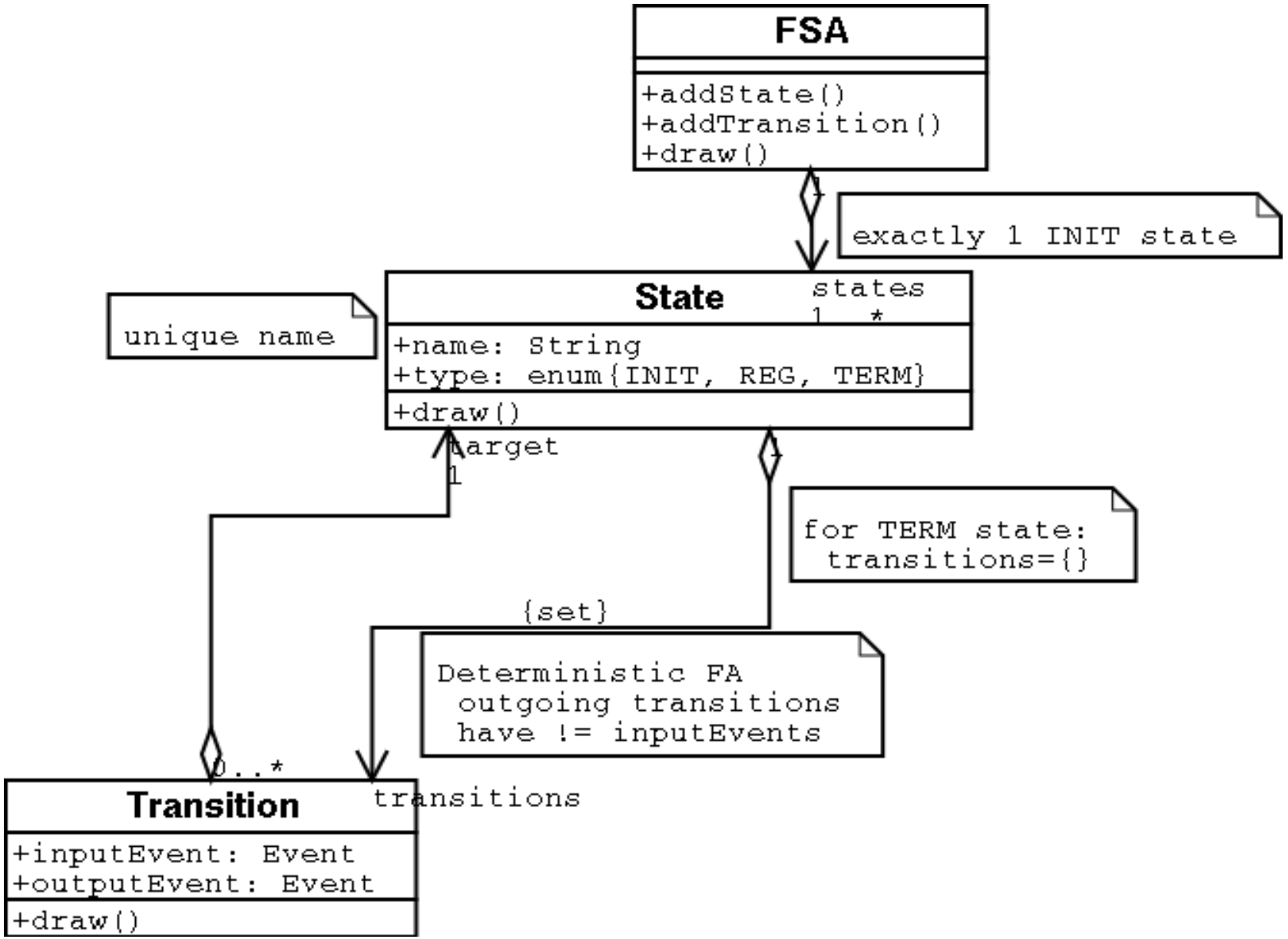
Formal Specification

- The automatic door can be *opened* or *closed*. (state)
- The sensor at the top of the door can send 4 types of signals: (input alphabet)
  - Nobody : There is nobody in front nor behind the doors.
  - Front: There is somebody in front of the doors.
  - Behind: There is somebody behind the doors.
  - Both: There is somebody in front and behind the doors.
- The door behaves as follows: (transition)
  - The door opens when somebody is in front of the doors.
  - The door closes only when nothing is in front or behind the doors.
- The door starts off as closed. (start state)

Diagrams


Nothing
Front,
Behind, Both
Open
Closed
Behind,
Nothing
Front, Both

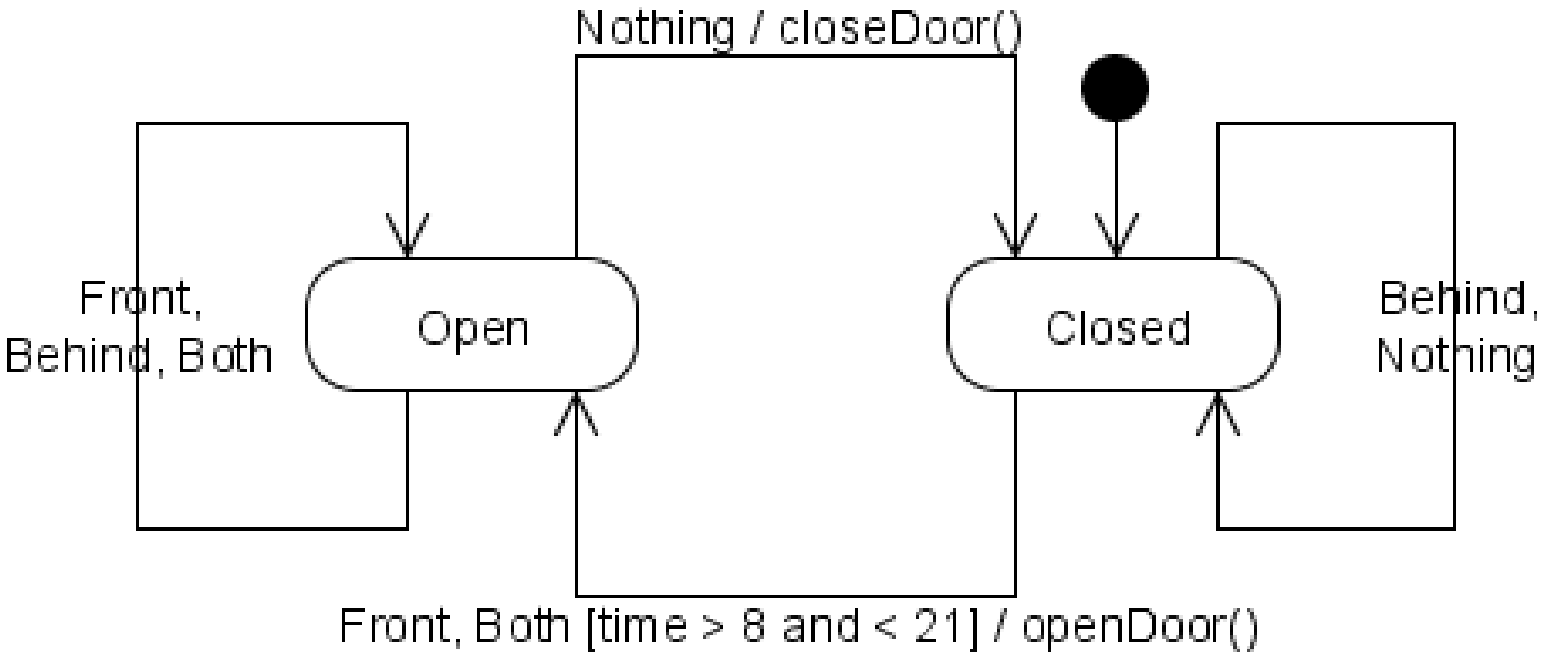
Class Diagram of FSA



Describing Reactive System Behaviour: Output and Guards

- We can extend this by adding the notions of output and guards.
  - Both of these additions can be found on the transition arrow.
- When a transition is triggered, it can broadcast an output event (or perform an action).
- Conditions can be imposed on transitions by adding guards.
  - A transition can then only take place if the guard evaluates to true.

Example



Note: total state (modal ++)

Executing/Simulating an DFA

```
# initialize the state
 currentState = getInitState()

 # as long as there is input
 while environment.inputRemaining():

   # get input event from the environment
   currentEvent = environment.getInput()

   # find applicable transition from currentState
   currentTransition = None
   for transition in currentState.transitions:
     if transition.inputEvent == currentEvent:
       currentTransition = transition
       break # assumes determinism!
   if currentTransition == None:
     print "unrecognized event, rejecting input"
     sys.exit() # or ignore: pass

   # generate output event
   environment.putOutput(transition.outputEvent) # could be action

   # update the current state
   currentState = transition.target

 if currentState.type == TERM:
  print "input accepted"
 else:
  print "input rejected"
```

Non Deterministic vs. Deterministic

- A non-deterministic FSA (NFA) is a finite state automaton where there exists a least one state from which multiple transitions can be triggered by the same event.

- NFAs can always be transformed into a DFAs.

Regular Expressions

- Regular expressions can be “compiled into” finite state automata

What is a Regular Expression?

- A (text) pattern that describes (matches) a set of strings, according to certain syntax rules.
- As such, a Regular Expression specifies a language
- Examples of regular expressions include:
  - Text starting with the letter “a” and finishing with the letter “z”.
  - Text with at least one number, but not starting with the letter “a” or “b”.
  - Text with a letter repeated three times in a row.
  - Text contains the string “abc” exactly three times.

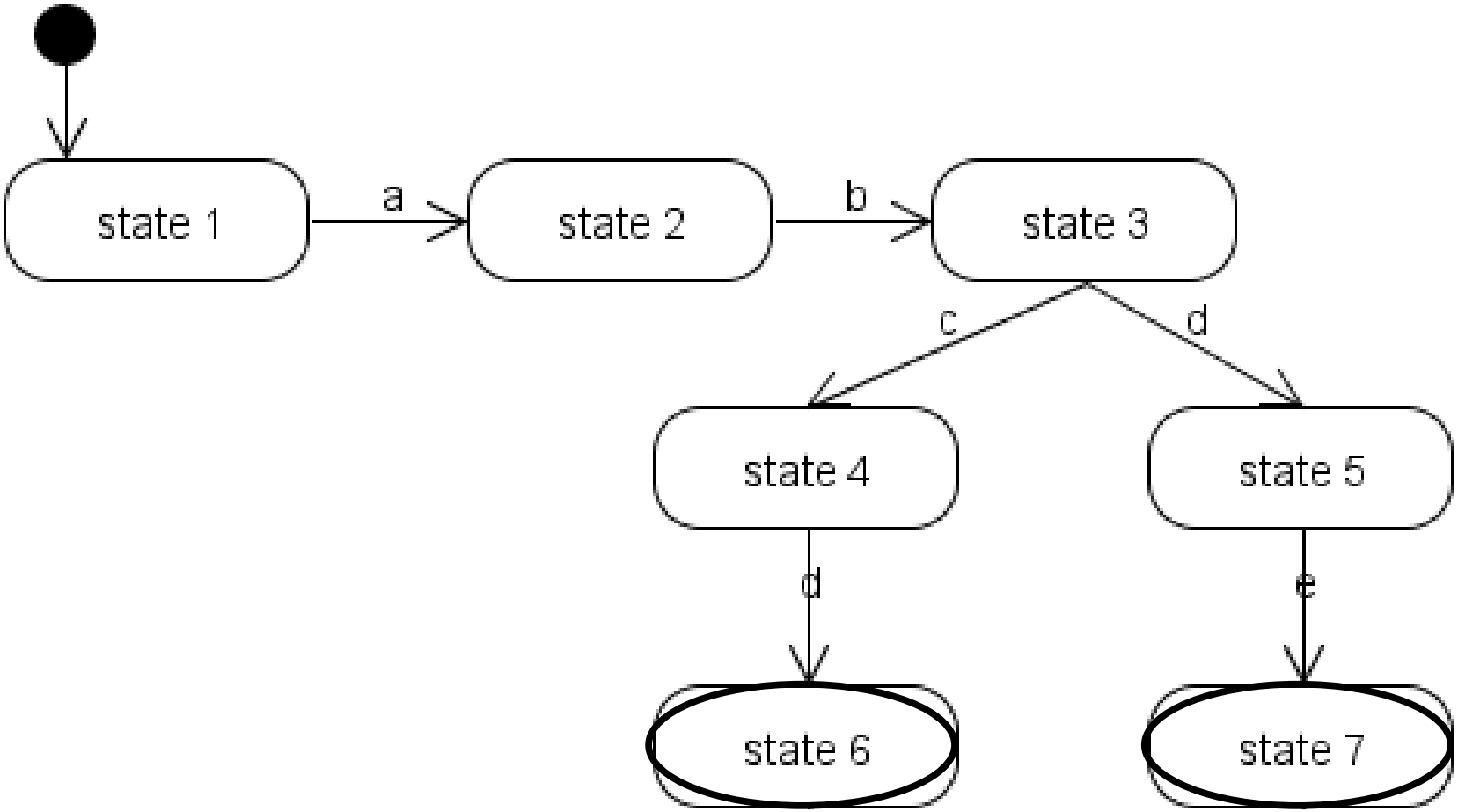
RegEx Constructs

- Most Regular Expression Language offer the following constructs.
  - Sequence: abc (really shorthand for the sequence 'a' 'b' 'c')
  - Alternatives: john|bob
  - Grouping: b(o|a)b
  - Quantification:
    - ? : 0 or 1 : (514)?555-5555
    - * : 0 or more : abc*
    - + : 1 or more : abc+
  - Escape character to allow use of meta-characters: \?, *, ...
  - Substitution:
    - P = a*b+
    - Q = {P}xyz{P}

From RegEx to FSA

ab((cd)|(de))

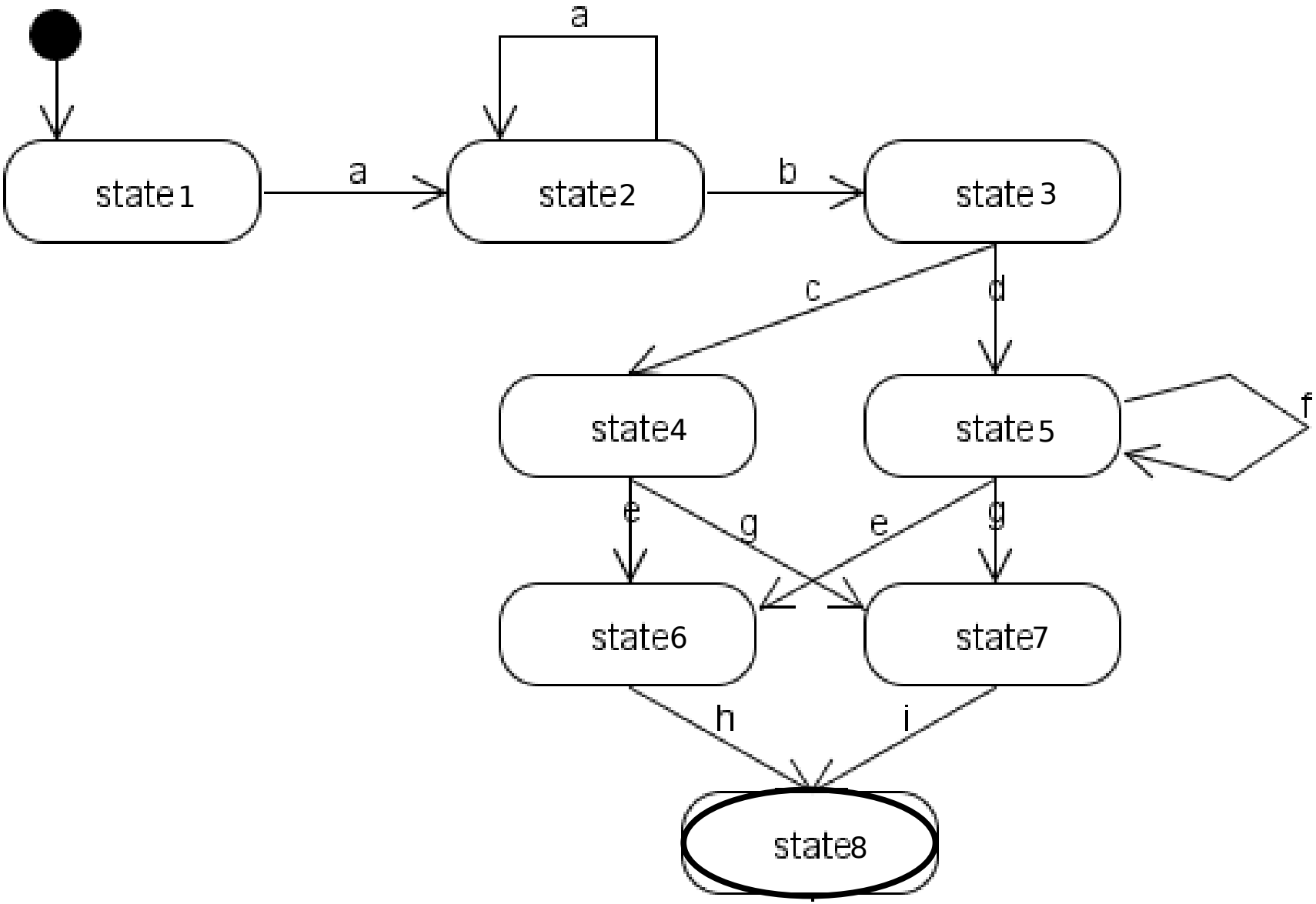
Solution



RegExp

a+b(c|df*)(eh|gi)

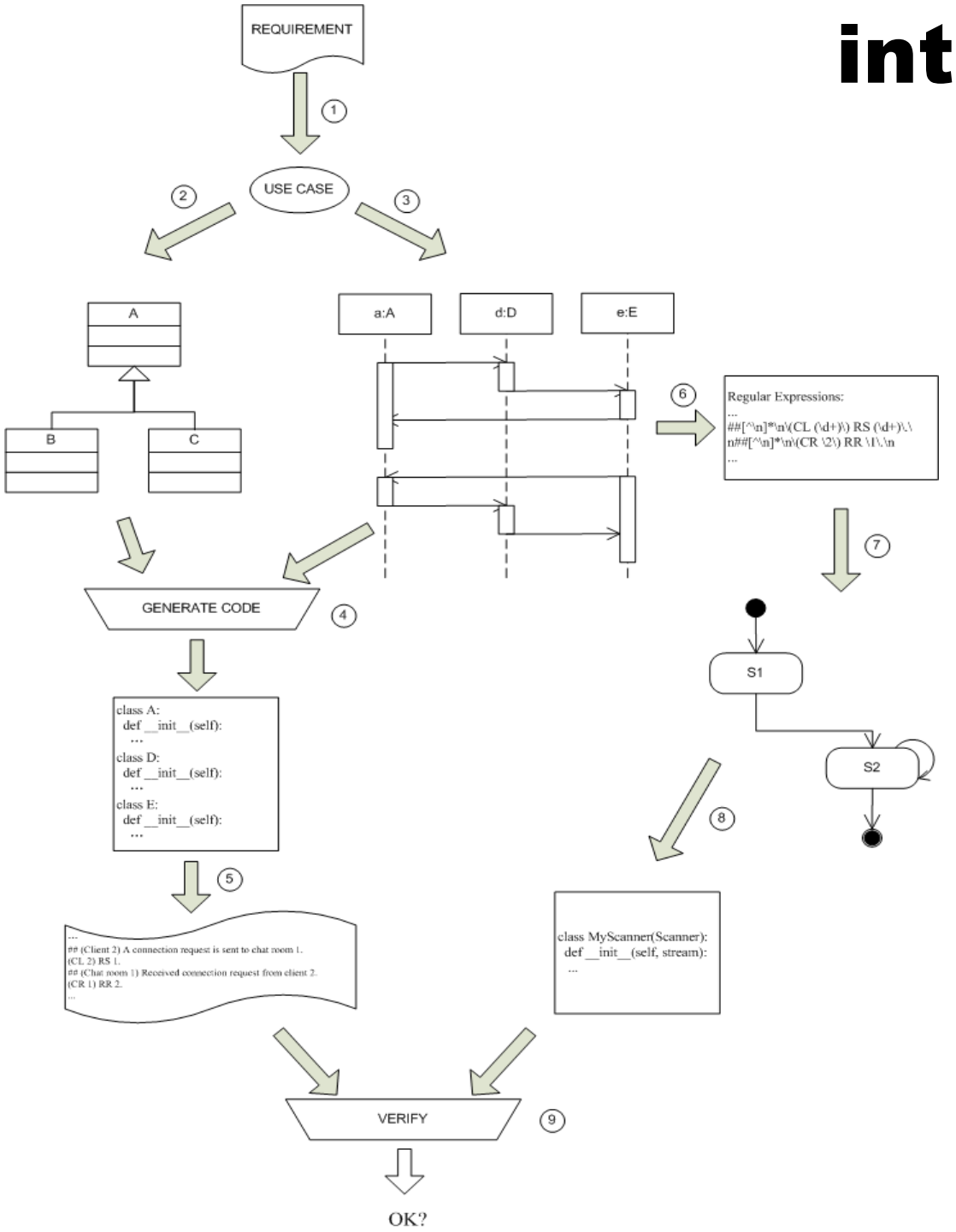
Corresponding FSA



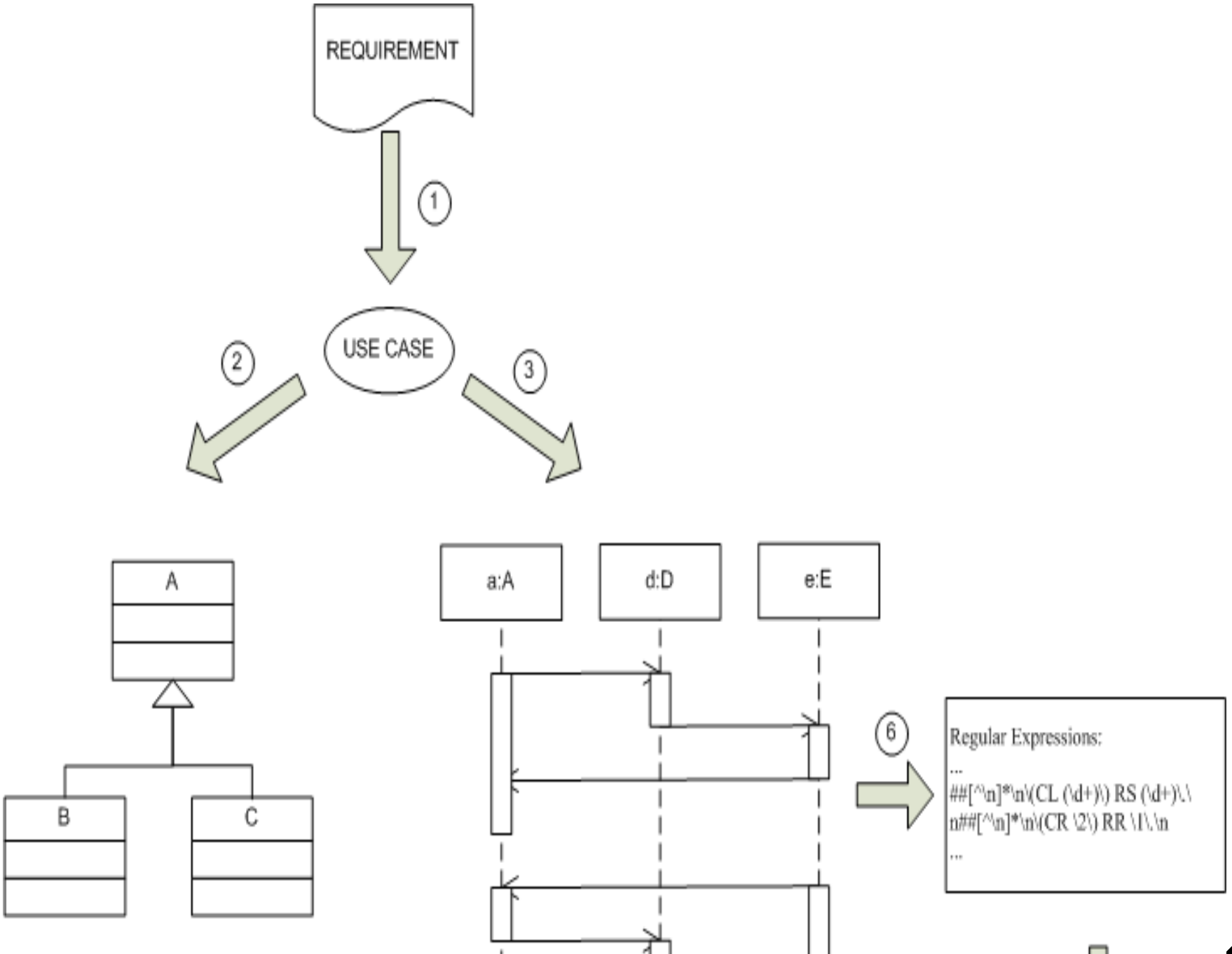
Real-world Examples

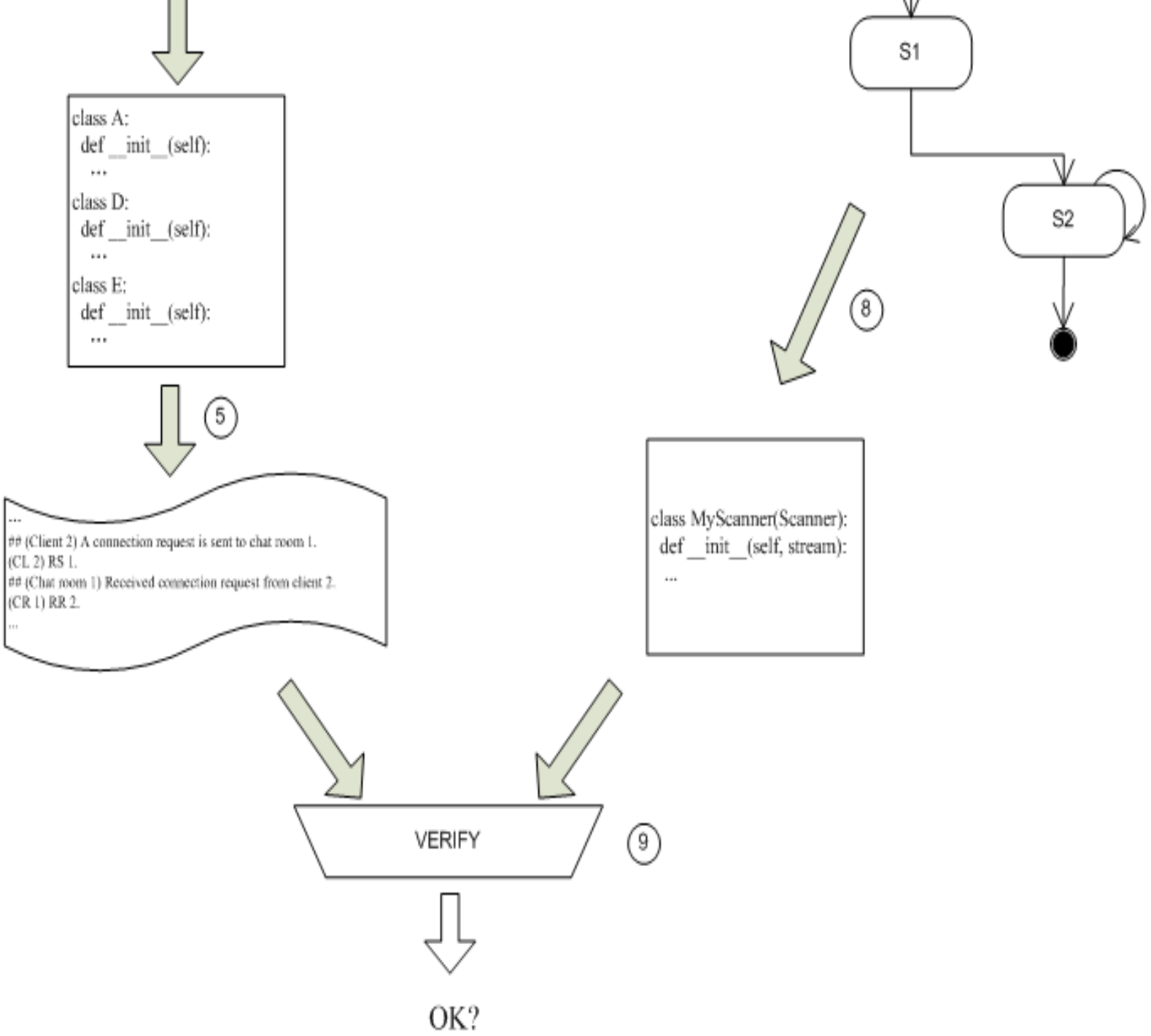
http://msdl.cs.mcgill.ca/people/hv/teaching/SoftwareDesign/COMP304B2003/assignments/assignment3/solution/

The Big Picture: testing interactions



From Requirement





To automated Testing