Comp-304 : Behavior Diagrams
Lecture 13

Alexandre Denault
Original notes by Hans Vangheluwe
Computer Science
McGill University
Fall 2007

Behavior Diagrams

- Structure Diagrams focused on describing the static composition of components.
- Interaction Diagrams focused on describing the communication between the various components.
- Behavior Diagrams focus on describing the behavior of
  - the whole application
  - a particular process in the application
  - a specific component

Different Formalism

- We will look at different formalisms:
  - Finite State Automaton
  - Activity Diagram
  - State Charts

Finite State Automaton

- A finite automaton is the set of
  - Set of states
  - Input alphabet
  - Rules for changing state
  - Start State
  - Accept State

Formal definition, from Sipser's Theory of Computation

Example : Automatic Door

Automatic Sliding Door

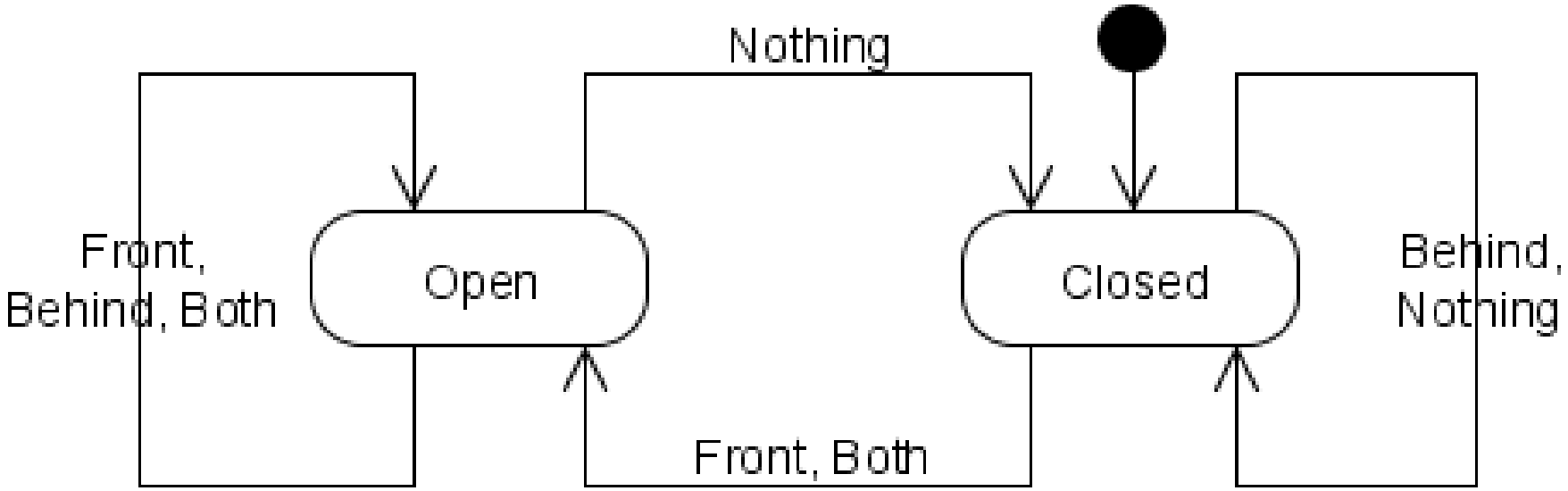
Specification

- 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 or 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 or 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 as closed.

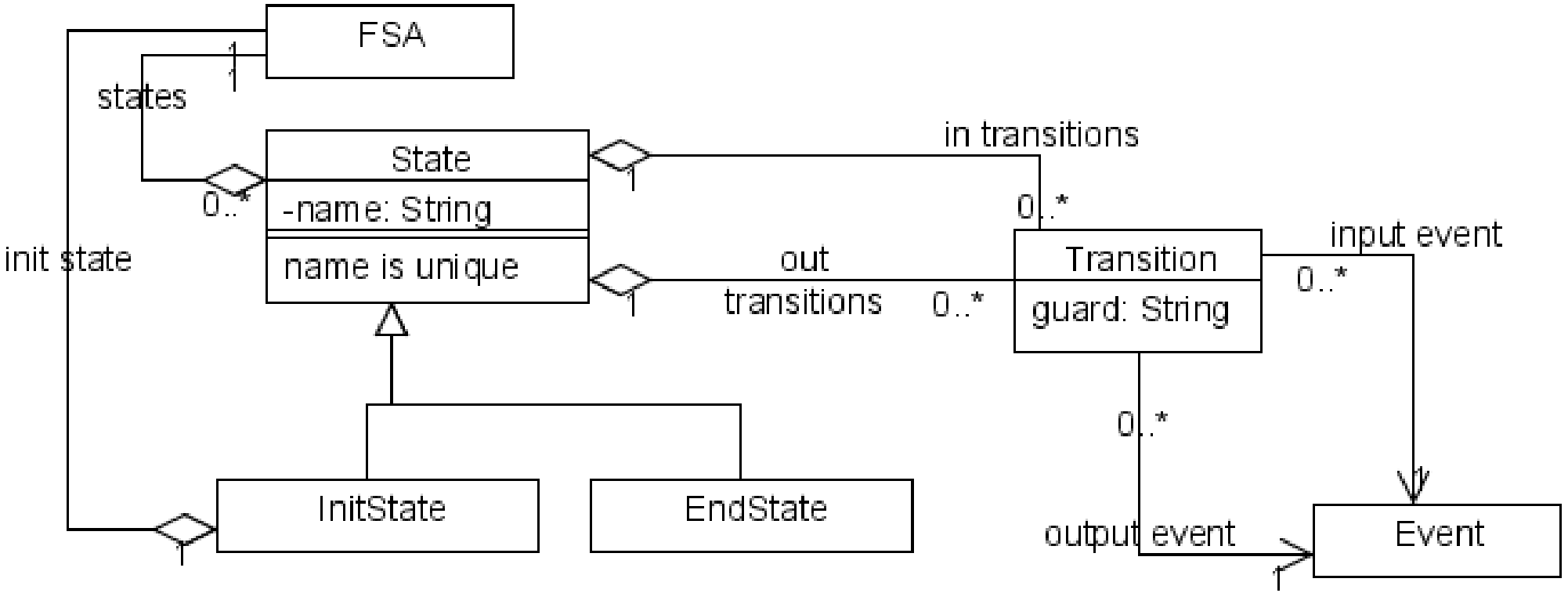
Specification

- 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 or 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 or 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)

Diagrams


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

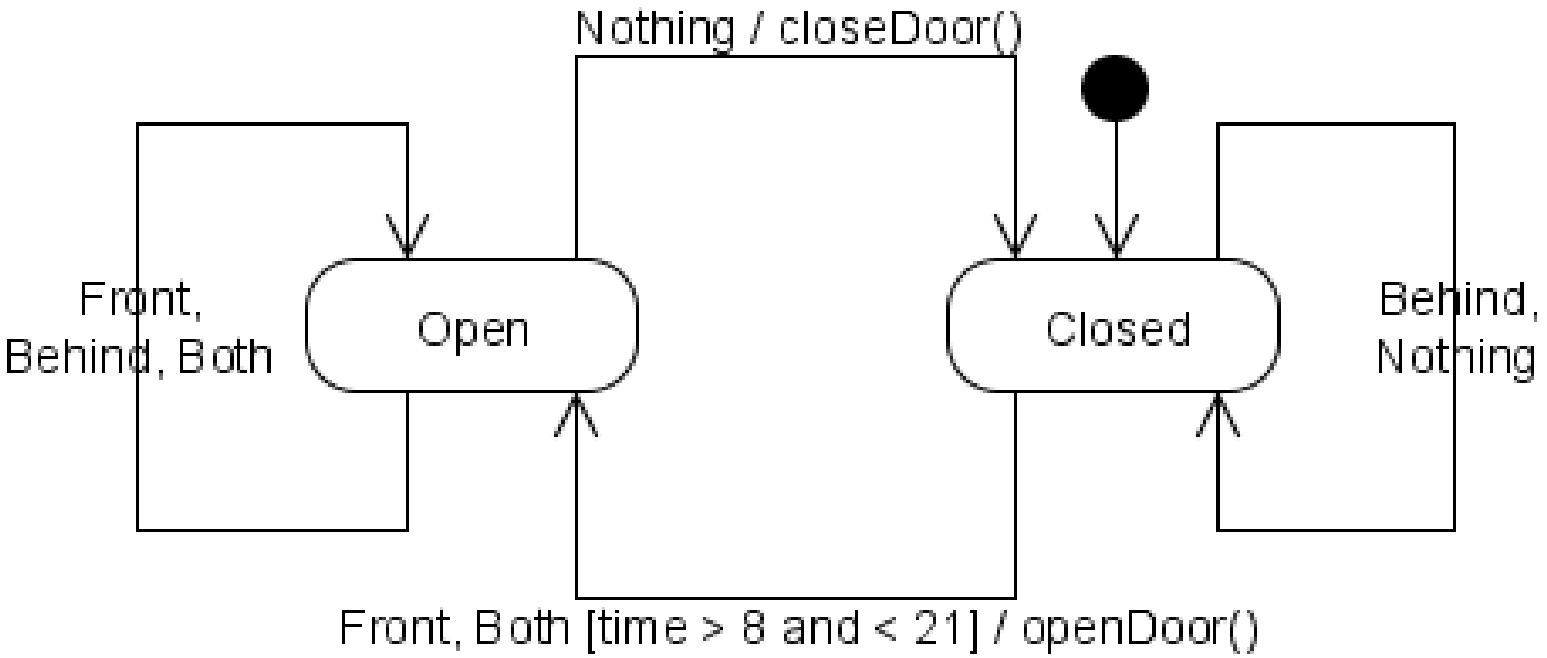
Class Diagram of FSA



Output and Guards

- We can extend typical FSA 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 send an output event to another component.
- Conditions can be imposed on transitions by adding guards.
  - A transition can then only fire if the transition is true.

Example


Nothing / closeDoor()
Front,
Behind, Both
Open
Closed
Behind,
Nothing
Front, Both [time > 8 and < 21] / openDoor()

Non Deterministic vs Deterministic

- A non-deterministic FSA (NFA) is a finite state automaton where there exists a least one state where multiple transitions can be triggered by the same event.
- Since all NFA can be transformed into a DFA (although this might cause a combinatorial explosion), we mostly consider the case of DFAs.

Equivalence with FSA

- Regular expressions and finite state automaton are equivalent in the descriptive power.
  - Any FSA can be converted into a regular expression.
  - Any regular expression can be converted into a FSA.

What is a Regular Expression?

- A text pattern that describes or matches a set of strings, according to certain syntax rules.
- 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.

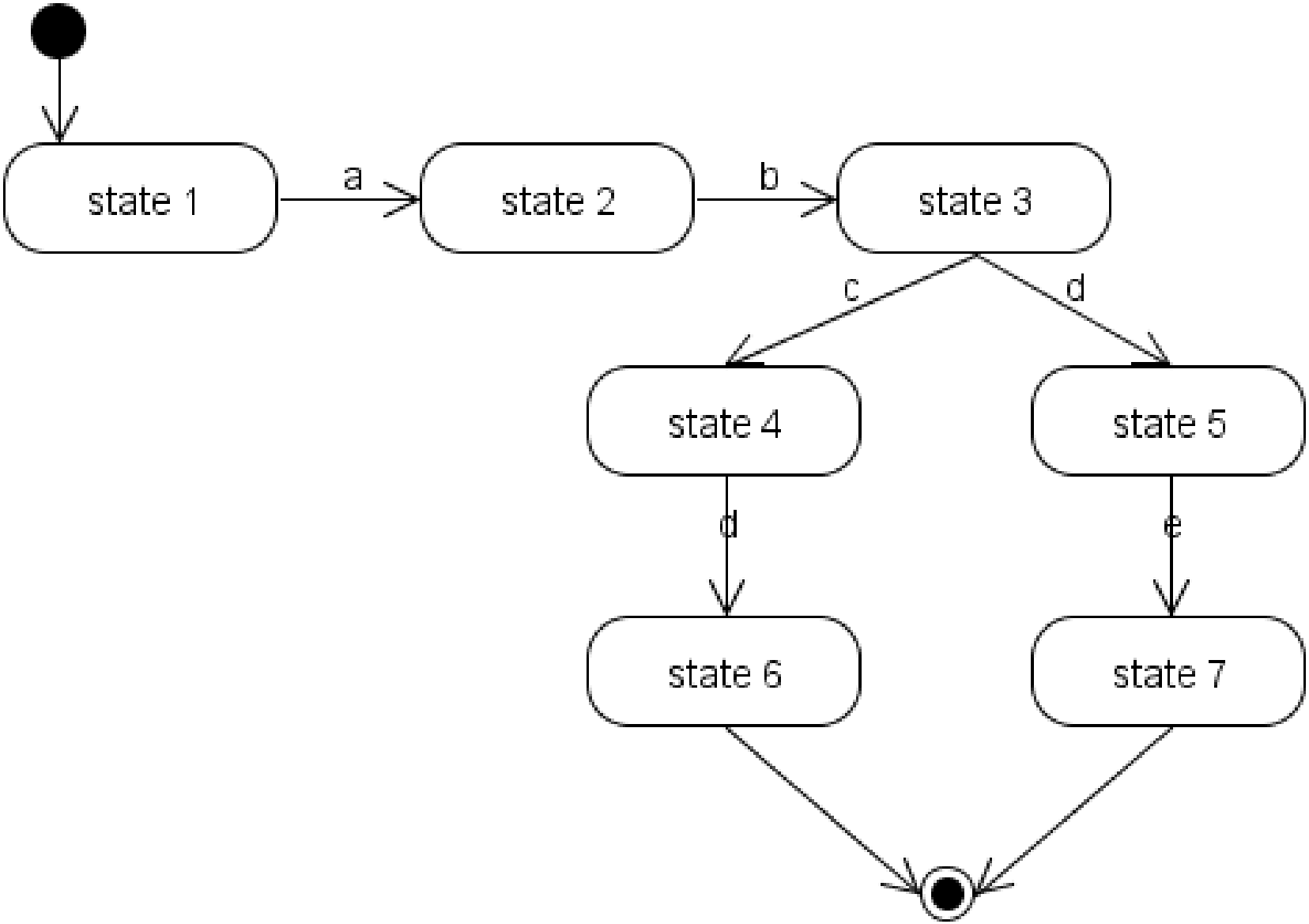
RegEx Constructs

- Most Regular Expression Language offer the following constructs.
  - Alternation: john|bob
  - Grouping: b(o|a)b
  - Quantification:
    - ? : 0 or 1 : (514)?555-5555
    - * : 0 or more : abc*
    - + : 1 or more : abc+

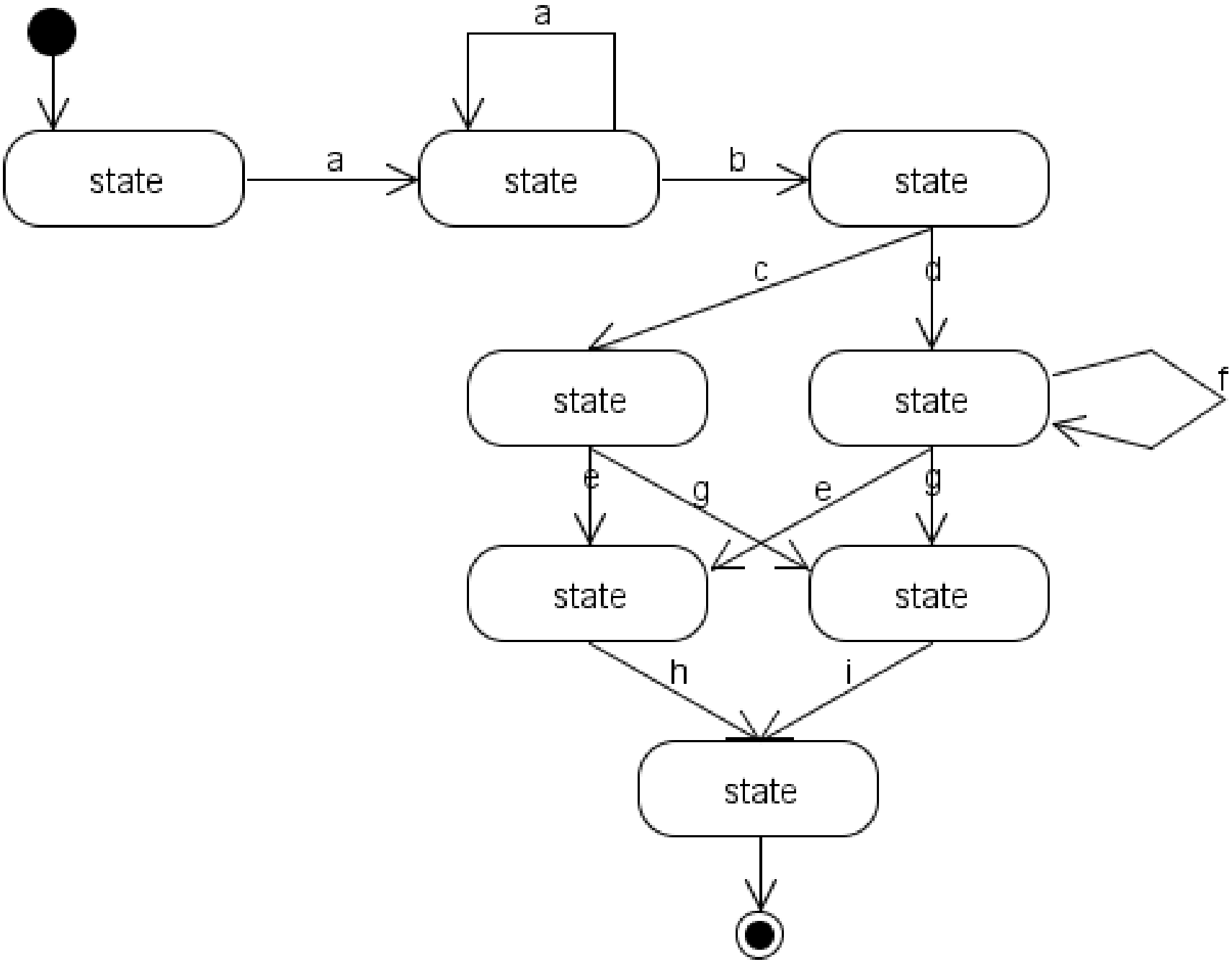
From RegEx to FSA

ab((cd)|(ed))

Solution



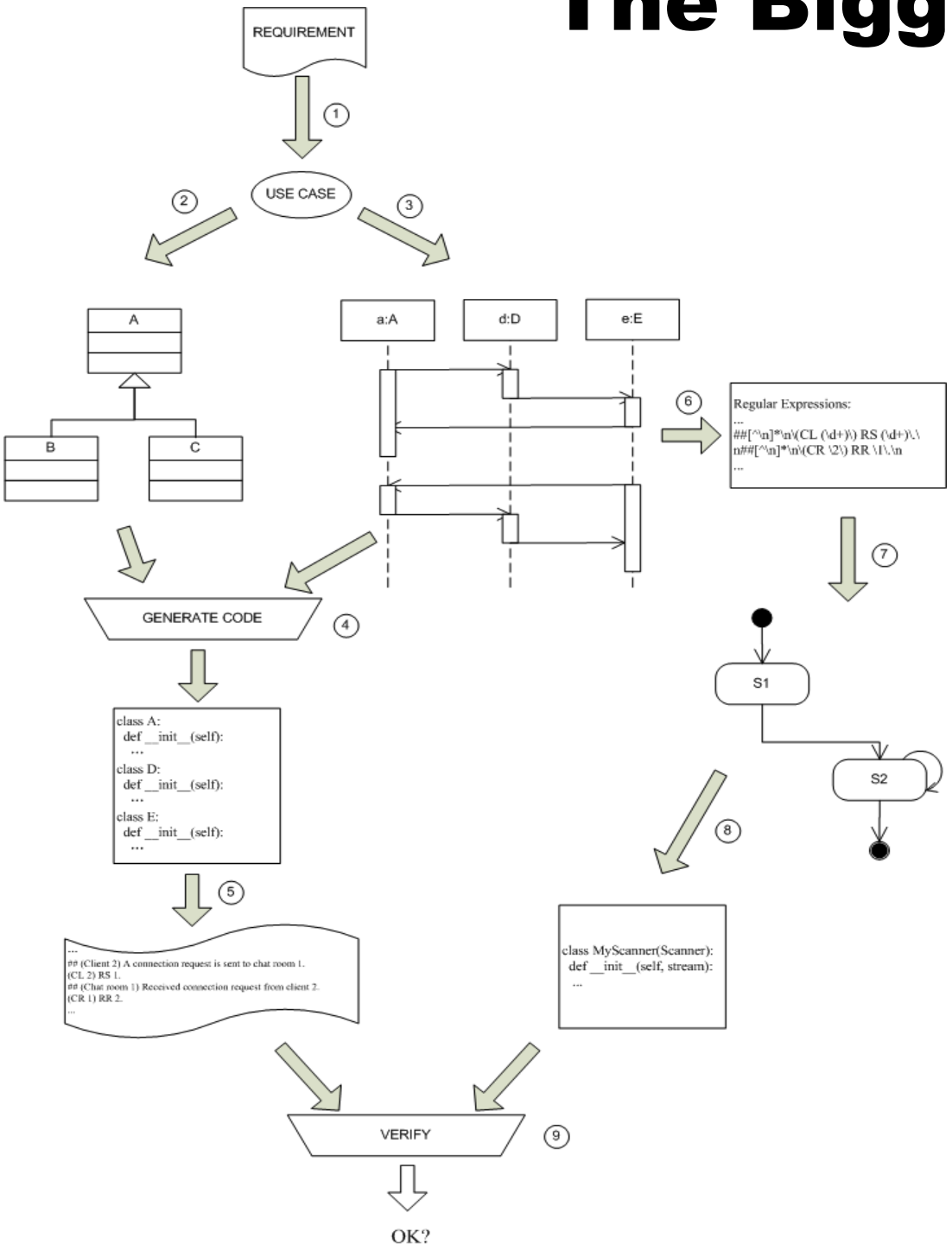
From FSA to RegEx



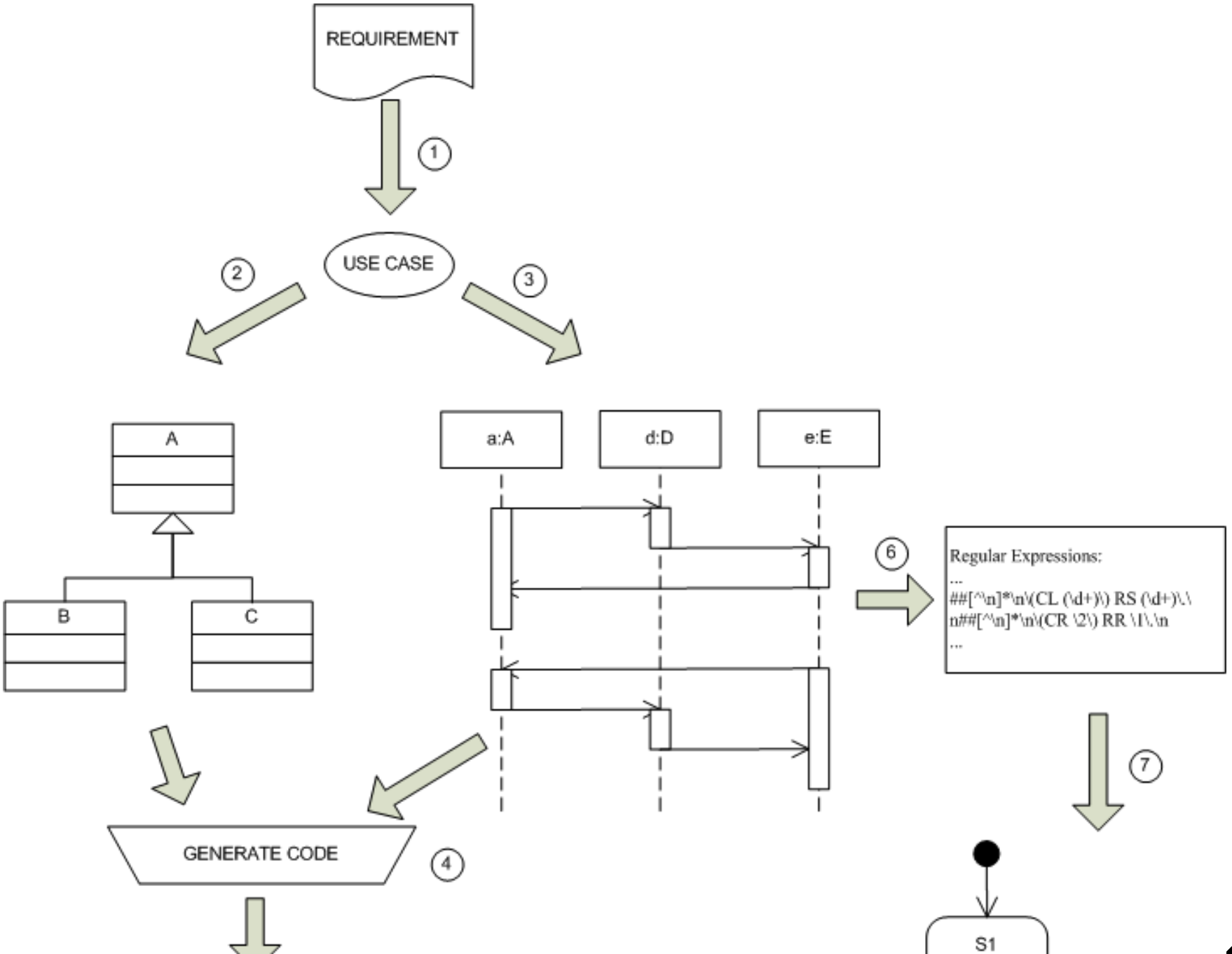
Solution

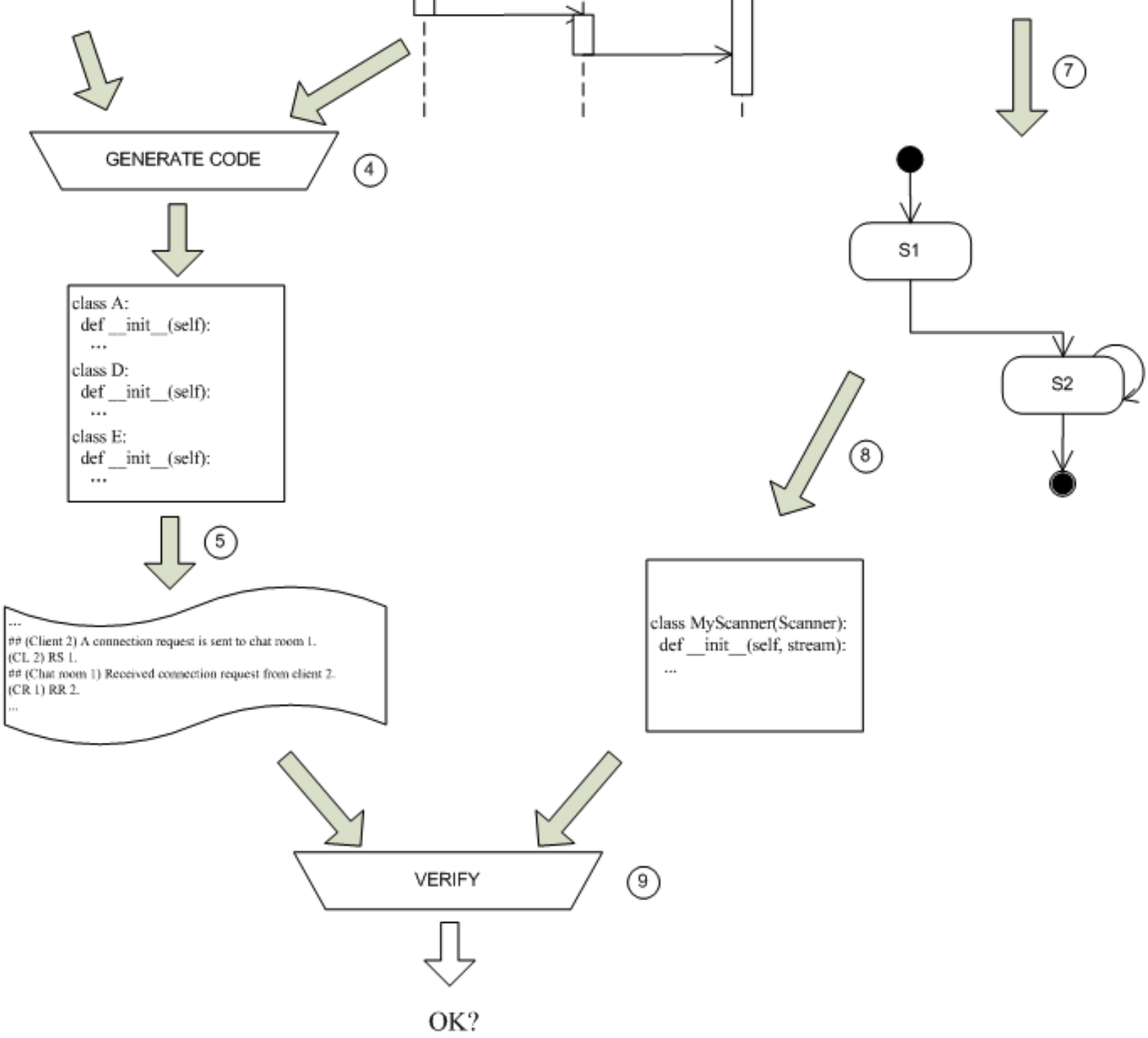
a+b(c|df*)(eh|gi)

The Bigger Picture



From Requirement





To Verification