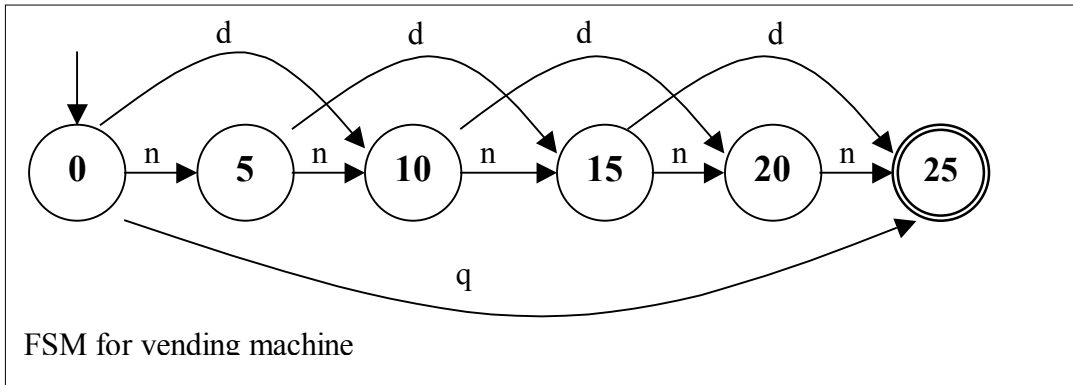


Main topics of the week:

- **Finite State Machine Diagrams**
- **Formal Definition of Deterministic Finite Automaton**
- **Formal Definition of computation of a DFA**
- **Language recognized by a DFA**
- **Constructing DFAs for a language**
- **Definition of regular language and regular operations**
- **Closure of regular languages under union**

Examples from class not in Sipser:

Example of a DFA: Suppose we have a vending machine that dispenses a can of soda when we deposit 25 cents. The coins we may insert are nickels, dimes and quarters, and the machine will count to see when we reach 25 cents. In our simple model of this machine, no change will be given, and the machine will keep any extra money. The state diagram of the machine looks like this:



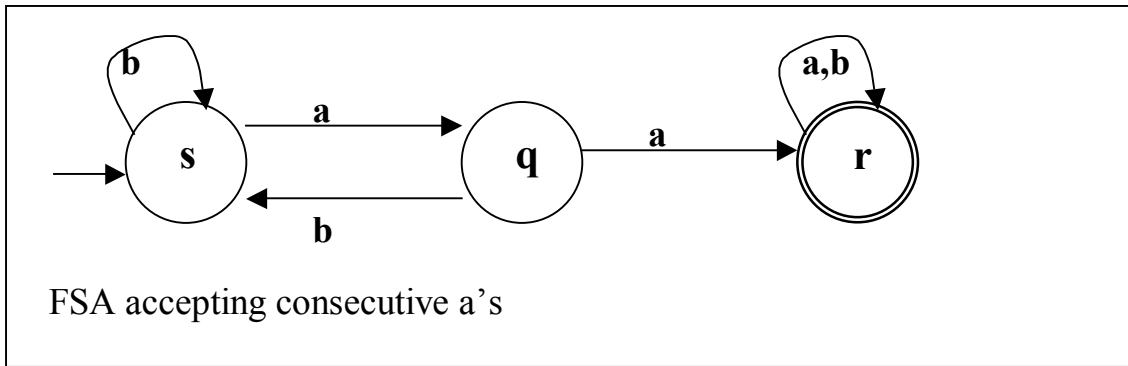
We have omitted the arrows for coin combinations that total more than 25 cents, but they could be added for each state, with all of them terminating in the accept state. Also, we could add arrows for inputs other than nickels, dimes, and quarters (call them all slugs). The arrows for such input in each state would loop back to the same state.

The formal definition of this machine would be given as the 5-tuple $(Q, \Sigma, \delta, 0, \{25\})$ where

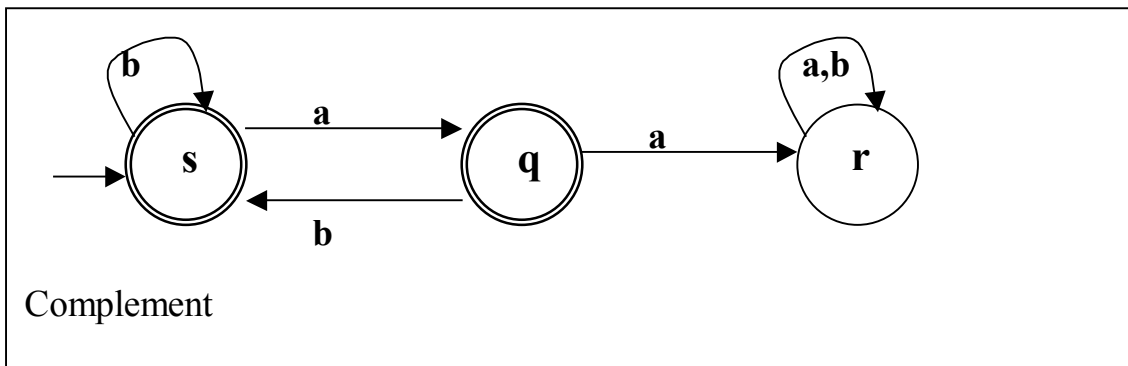
- 1) $Q = \{0,5,10,15,20,25\}$
- 2) $\Sigma = \{n,d,q,s\}$
- 3) δ is given by

	n	d	q	s
0	5	10	25	0
5	10	15	25	5
10	15	20	25	10
15	20	25	25	15
20	25	25	25	20
25	25	25	25	25

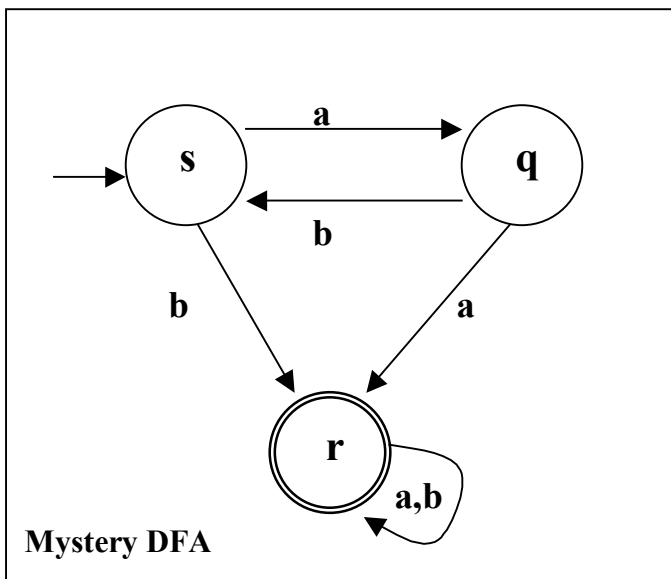
Example from class of an automaton to recognize two consecutive a's:



And its complement:

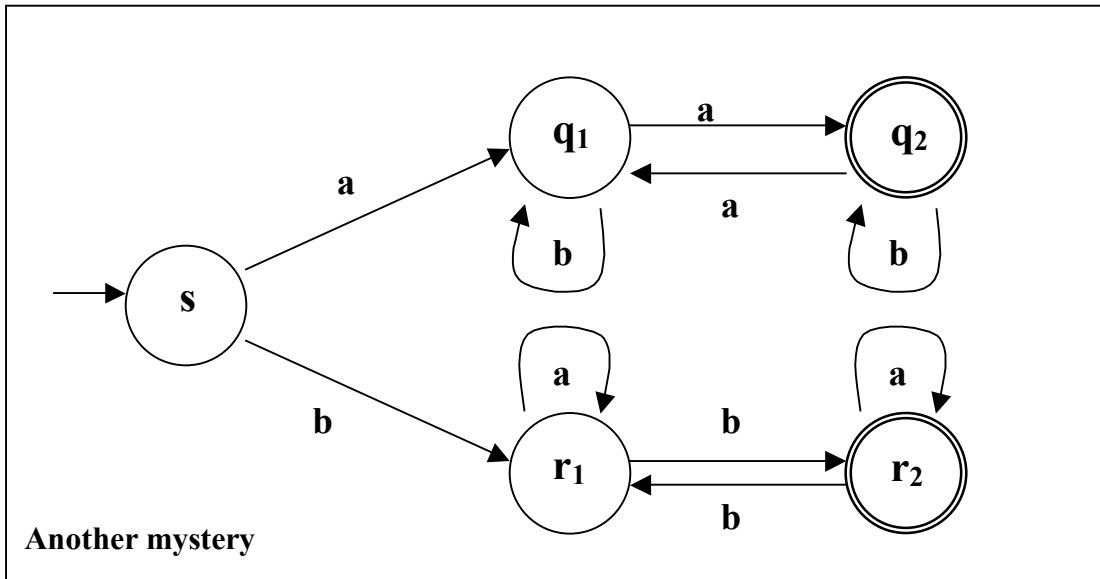


Another example from class. What strings does this FSD recognize?



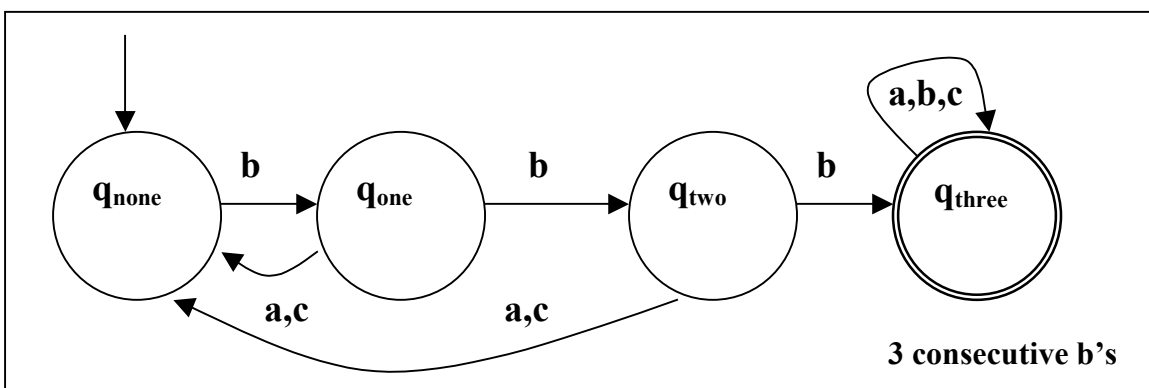
Suppose we think about its complement. That would be all strings that begin with an a and have alternating a's and b's. So this machine recognizes all the other strings, i.e., strings that begin with a b or have two consecutive a's or b's.

Another example:



In this example, we go separate paths from the start state depending on whether we start with an a or b. In the top case, we get to the accept state after another a, but return to q_1 if we see another a. We are basically ignoring all b's. If you think about what is happening, you'll realize that we end up in the accept state along the top path only when we see an even number of a's. Likewise, in the bottom path, we accept only an even number of b's. So we could describe this machine as accepting strings that have an even number of occurrences of the first character. Note this is different from the machine that would accept any string with an even number of a's or an even number of b's. What would that FSD look like?

Constructing DFAs: Design an automata that accepts the language over $\{a, b, c\}$ that has three consecutive b's. As we think about this, we realize we need to count the number of b's that we see in a row. We could have none, one, two, or three. In any of these cases, seeing a b takes us to the next state (but not beyond three). Seeing either a or c causes us to reset to the none state. Seeing anything once we have seen three b's doesn't change anything since we only care about finding at least three consecutive b's. So we have a machine with four states, and have described all transitions. The last state is the one and only accept state, and the first state is the start state. In a diagram, we have:



Here's an example of the construction used in the proof that the union of regular languages is regular for two simple DFAs. Let A be the language over the alphabet $\{a,b\}$ with even occurrences of a and B be the language with even occurrences of b . The union is pretty clearly the strings with an even number of a 's or b 's. The diagram shows the construction of the DFA to recognize this language. Note that we have labeled the states in the constructed DFA as the pair of states, one from each of the DFAs for A and B .

