Deterministic Finite Automata (DFAs)

Takes as input a string - says "Yes" or "No"

Finite set of states - no other memory
- when it reads a character - it will move from current state to another state based on some rule. When it has read the whole string, it will be in some state - either this state will be a "Yes" state or a "No" state.
"Deterministic" means that, from each state, there is one arrow going out for each letter.

E.g., Make a DFA that accepts the strings w/ an even # of a's and an odd number of b's.

Does this accept "abaababb"
E.g. Our automaton accepting even #a's and odd #b's strings

\[ Q = \{ EE, EO, OE, 00 \} \]
\[ \Gamma = \{ a, b \} \]

\( S \) is the function with

\[ S(EE, a) = OE \]
\[ S(EE, b) = EO \]
\[ S(OE, a) = EE \]
\[ S(OE, b) = 00 \]
\[ S(EO, a) = 00 \]
\[ S(EO, b) = EE \]
\[ S(00, a) = EO \]
\[ S(00, b) = OE \]

\( q_0 = EE \)
\( F = \{ EE03 \} \).
E.g. Make a DFA that accepts the strings where a's and b's alternate. (empty string should be accepted) (E.g. ab ab a, b, ab, ba should all be accepted) (If you have bb and aa anywhere, you should reject)

"trap state" or "jail state"

Or:
alphabet: \{ a, b \}

If the string is "abaabaa"

This automaton says "Yes" to any string that has both an a and a b (and "No" to every other string)

Conventions:
tells you where we start
We follow the arrows when we read a letter.
"Yes" states have an extra circle.
Mathematical formalization of DFAs

"Data structure of a DFA":
The data we have:
1) An alphabet (finite)
2) Set of states (finite), out of which
   a) one state is the initial state (or start state)
   b) some of the states are accept states (or final states)
3) A rule telling us, given a state and a letter, what the next state is.

So: A DFA is a 5-tuple
\[ M = (Q, \Gamma, \delta, q_0, F), \]
where
- \( Q \) is a finite set (of states)
- \( \Gamma \) is a finite set (the alphabet)
- \( \delta \) is a function \( Q \times \Gamma \rightarrow Q \)
- \( q_0 \in Q \) is the initial state
- \( F \subseteq Q \) is the set of final states.
HW for Friday: Design a DFA that accepts all the strings on \( \{a, b\} \) that have at most 2 a's after the first b.

Notes will show up at

webpages.uidaho.edu/AlexanderWoo/385