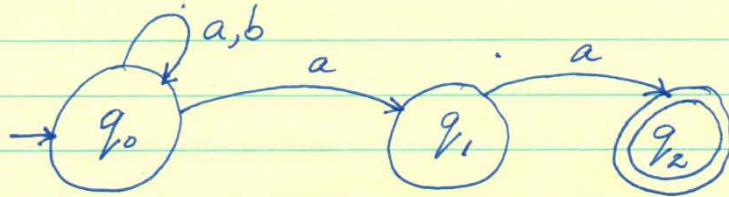


NFAs

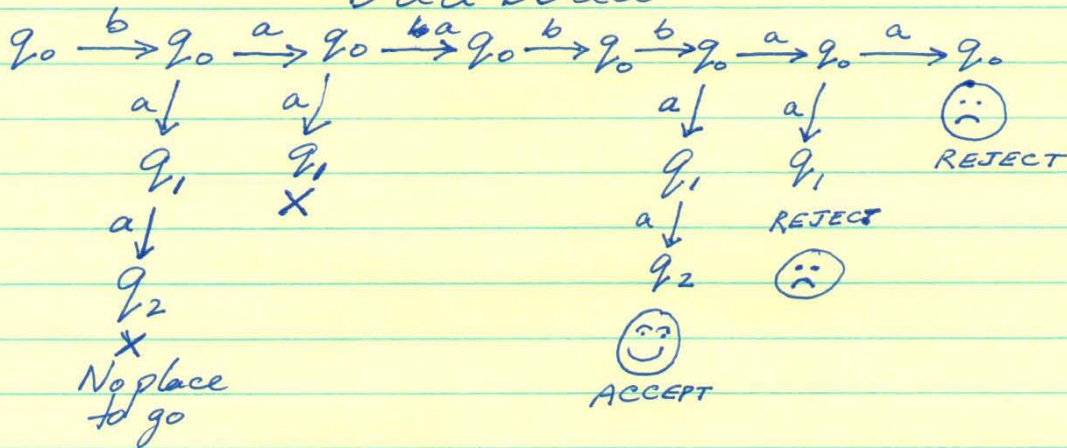
How do we design a machine to accept all words ending in "aa"? Not very hard to make a DFA (do it!) but what if we had a machine that could guess?



Hey! Two transitions labelled "a" coming out of q_0 . Nothing labelled with a "b" coming out of q_1 & nothing at all coming out of q_2 .

New mechanism: nondeterministic guessing + backtracking + reject if there is no move.

baabbbaa



It does not matter that some paths jammed or that some got to a reject state, at least one (more is OK) path must get to an accept state.

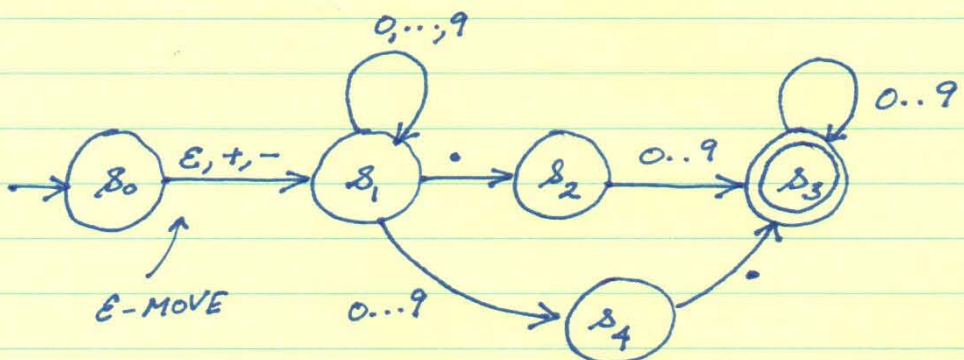
We can always construct a DFA that recognizes the same language as an NFA. In fact this can be done algorithmically and is a very important piece of software. The DFA one gets may be very complicated. Key point: You can use non-determinism as a design aid & then use the algorithm to convert the NFA into an equivalent DFA.

Another extension: NFA + ϵ moves: this can jump without reading a letter.

EXAMPLE A decimal number has (i) an optional + or - sign (ii) a sequence of digits (iii) a decimal point (iv) another sequence of digits.

Either (ii) or (iv) can be empty but not both

$\Sigma = \{ \cdot, +, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
 307.3, -406, +97., 0.0, -3.1, ...



At s_1 it guesses whether you want to put digits after the decimal. If it thinks that there will be no digits after the \cdot it will go to s_4 but it forces at least one digit before the \cdot .

(3)

Formal Def of NFA

Σ : alphabet is fixed

An NFA has 4 things

- (1) Q : a finite set of states
- (2) $Q_0 \subseteq Q$: a set of start states
- (3) $F \subseteq Q$: a set of accept states
- (4) A transition function

$$\Delta: Q \times \Sigma \rightarrow 2^Q$$

(state, letter) \mapsto set of possible next states

We write 2^Q for the set of all subsets of Q .

Note $\emptyset \in 2^Q$ so Δ could say that there is no next state: machine jams.

NFA + ϵ : same (1), (2) & (3) but

$$(4') \Delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$$

It is nice to write $s \xrightarrow{a} s'$ instead of $s' \in \Delta(s, a)$.

For NFA without ϵ :

We define Δ^* by induction $Q \times \Sigma^* \rightarrow 2^Q$

$$\Delta^*(q, \epsilon) = \{q\}$$

$$a \in \Sigma. \quad \Delta^*(q, a) = \Delta(q, a)$$

$$w \in \Sigma^*, a \in \Sigma \quad \Delta^*(q, wa) = \bigcup_{q' \in \Delta^*(q, w)} \Delta(q', a)$$

If N is an NFA

$$L(N) = \{w \in \Sigma^* \mid \exists q \in Q_0, F \cap \Delta^*(q, w) \neq \emptyset\}$$

One of the places where the machine may end up is an accept state.