Non-determinism

A Non-deterministic Finite Automaton NFA is a language recognizer similar to a DFA.

- At any point in the computation the NFA may take on many next states

when an NFA receives an input symbol it may ~~take on~~ make a transition to 0, 1 or more states.

An NFA N is a 5-tuple

$$N = (Q, \Sigma, \delta, q_0, F)$$

Q is a finite set of states

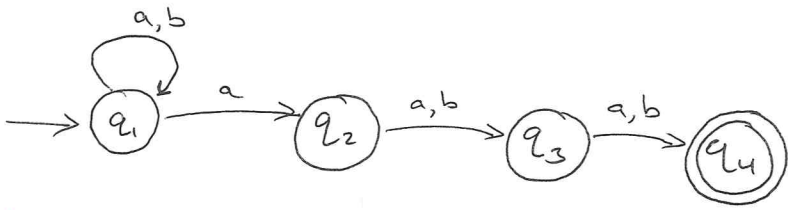
Σ is a finite alphabet

$q_0 \in Q$

$\delta : Q \times \Sigma \to \mathcal{P}(Q)$

$F \subseteq Q$

NFA example

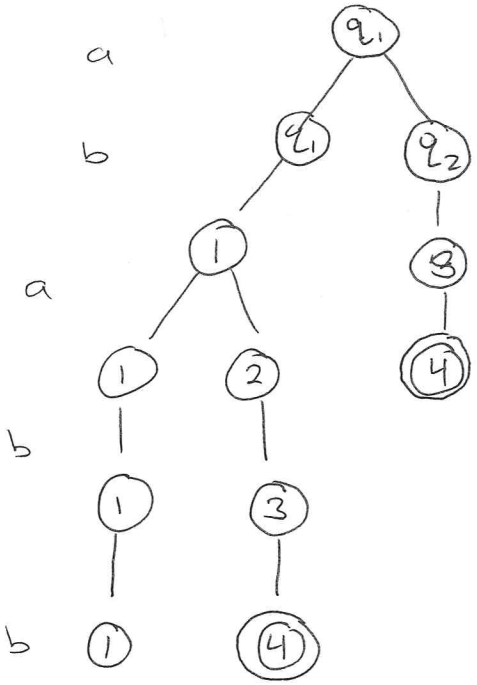


Notice that

- q_1 has 2 transitions on a
- q_4 has no transitions

How does computation work with an NFA

S = a b a b b



do we accept the string s? yes

Formal Definition of Computation for an NFA

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w be a string over the alphabet Σ.

$w = w_1 w_2 \dots w_n$

N accepts w if there exists a sequence of states $r_0, r_1, \dots, r_n$ in Q such that:

1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, w_{i+1})$ for $i = 0 \dots m-1$
3. $r_m \in F$

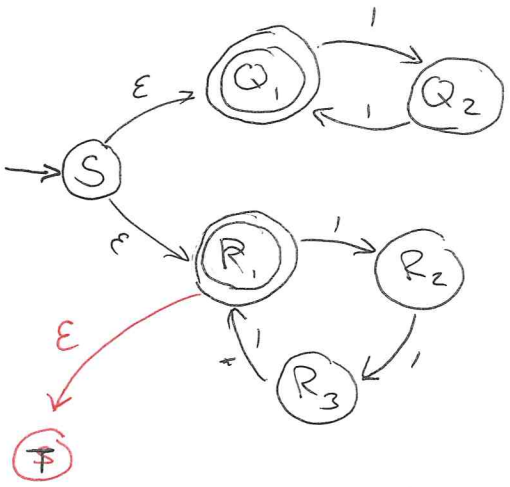
NFAs with epsilon transitions

is allowed to change its state without reading an input symbol.

this means the formal definition shown earlier isn't quite right.

$$\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$$

where $\Sigma_\varepsilon = \Sigma \cup \varepsilon$



what is the alphabet? $\Sigma = \{1\}$

What language does this recognize?

$\{1^k \mid k$ is even or a multiple of 3 $\}$

what happens if I add ~~δ~~ ? does it affect what language is recognized?

No

ε-closures

the set containing the current state and all states that can be reached with only ε transitions

ε-closure(S) = $\{S, Q_1, R_1, T\}$

ε-closure(R_1) = $\{R_1, T\}$

ε-closure(x) = $\{x\}$ for all other states

Do ε's allow us to do anything that can't be done without them? No

they can be completely eliminated.

Theorem

ε doesn't make NFAs more powerful.

Proof by construction.

Let N be an NFA

construct NFA N' with no ε-transitions as follows

For each $p \in Q$

1. make p an accepting state of N' iff the ε-closure of p contains an accepting state of N
2. add a transition from $p \to q$ labeled a iff there is a transition labeled a in N from some state in ε-closure of p to q

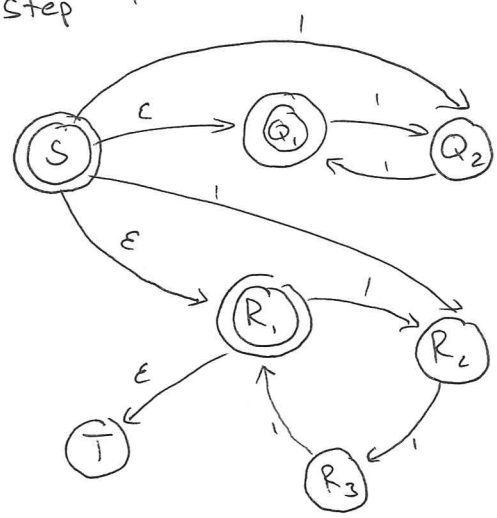
&.

Delete all transitions labeled ε

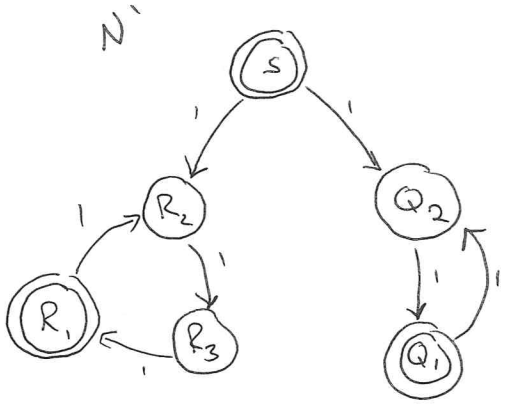
N' now recognizes the same language as N but has no ε-transitions.

Therefore ε-transitions don't make NFAs more powerful.

step 1 and 2



Remove ε-transitions



N and N' are not the same but they ~~represent~~ recognize the same language.

So we say that N and N' are equivalent.

Equivalence of DFAs and NFAs

Theorem: Every NFA has an equivalent DFA

Proof by construction

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA recognizing some language A.

We construct DFA $M = (Q', \Sigma, \delta', q_0', F')$ recognizing A.

- $Q' = \mathcal{P}(Q)$ each state in M is a set of states in N
- For $R \in Q'$ and $a \in \Sigma$

  let $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R\}$

  $R \in Q'$ means that R is a set of states in N

  $$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

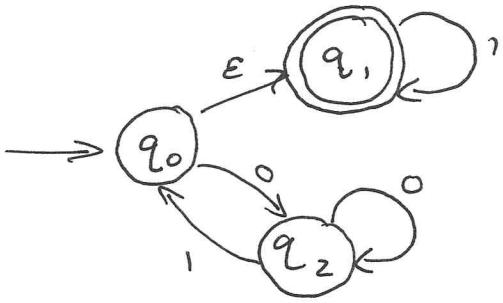
- $q_0' = \{q_0\}$
- $F' = \{R \in Q' \mid R \text{ contains an accepting state of } N\}$

  if any of the states N might be in are accepting states then M accepts

How many states does this machine have? 2^k

assume $|Q| = k$

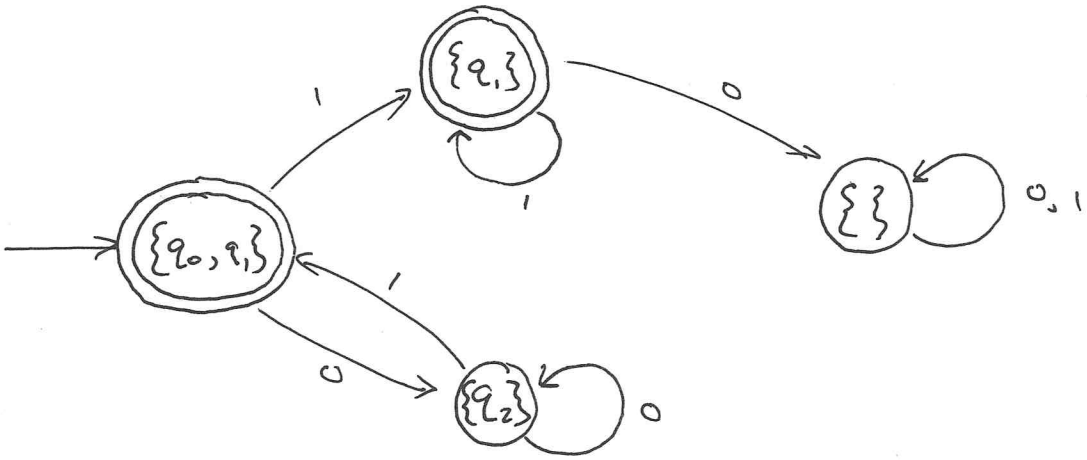
NFA → DFA example



	0	1
$\{q_0, q_1, q_2\}$ ★	$\{q_2\}$	$\{q_0, q_1\}$
→ $\{q_0, q_1\}$ f	$\{q_2\}$	$\{q_1\}$
$\{q_0, q_2\}$ ★	$\{q_2\}$	$\{q_0\}$
$\{q_1, q_2\}$ ★	$\{q_2\}$	$\{q_0, q_1\}$
$\{q_0\}$ ★	$\{q_2\}$	$\{\}$
$\{q_1\}$ f	$\{\}$	$\{q_1\}$
$\{q_2\}$	$\{q_2\}$	$\{q_0, q_1\}$
$\{\}$	$\{\}$	$\{\}$

★ have no inputs

F final states



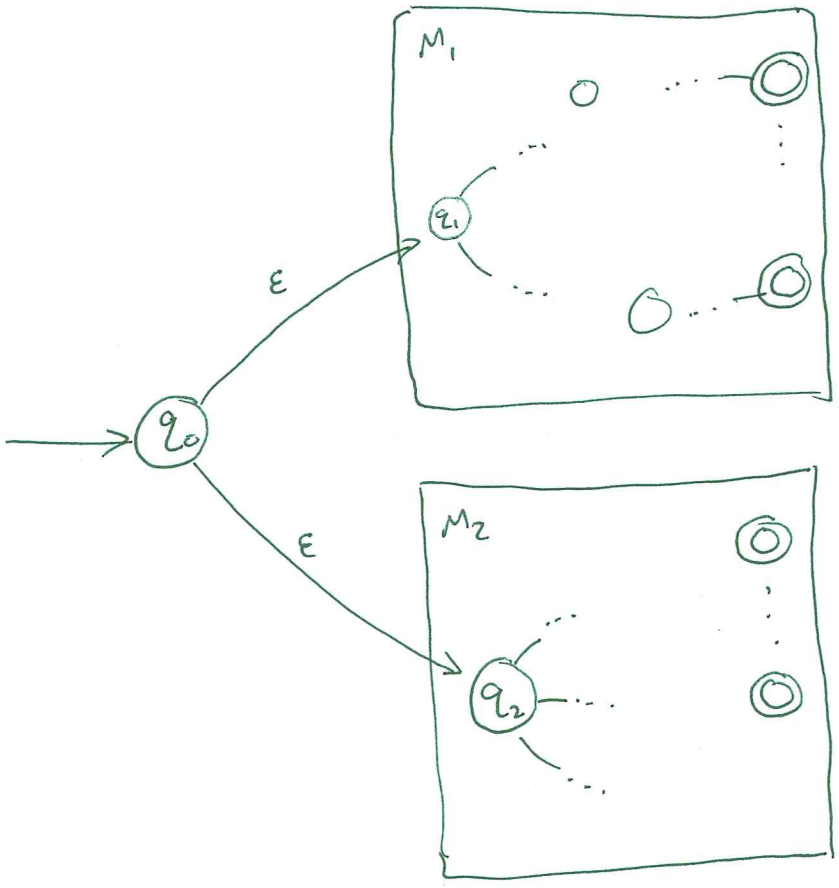
Regular languages are closed under Union (NFA version)

Let A, B be regular languages, and M_1, M_2 be DFAs

$L(M_1) = A$

$L(M_2) = B$

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ $\quad$ $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$



Regular Operations on Languages

Let A and B be languages

- Union: $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$
- Concatenation: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
- Kleene Star: $A^* = \{x_1 x_2 \ldots x_n \mid n \geq 0 \text{ and each } x_i \in A\}$

Examples

$A = \{aa, aba, abba\}$

$B = \{aa, baa, bbaa\}$

$A \cup B = \{aa, aba, abba, baa, bbaa\}$

$A \circ B = \{aaaa, aabaa, aabbaa, abaaa, ababaa \ldots\}$

$A^* = \{\varepsilon, aa, aaaa, aaaba, abbaaaaa \ldots\}$

Regular languages are closed under concatenation

Theorem

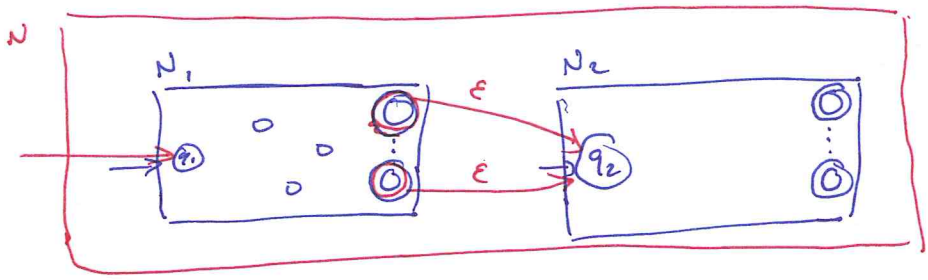
If $A, B \in$ Regular languages $\quad A \circ B \in$ Regular languages

Proof by Construction

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A

Let $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize B

Construct $N = (Q, \Sigma, \delta, q_0, F)$ that recognize $A \circ B$



don't forget to remove the Accept from final states of N_1

$Q = Q_1 \cup Q_2$

$q_0 = q_1$

$F = F_2$

Define δ as

haven't reached accepting state in N_1

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \wedge q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \wedge a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \wedge a = \varepsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

Theorem

The class of Regular languages is closed under Kleene star.

Proof by Construction

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be an NFA that recognizes A.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A^*

$Q = Q_1 \cup \{q_0\}$ where q_0 is some newly added start state.

$F = \{q_0\} \cup F_1$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \wedge q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \wedge a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \wedge a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \wedge a \neq \varepsilon \end{cases}$$