Pre-lecture brain teaser

Find the regular expression for the language containing all binary strings that do not contain the subsequence 111000

CS/ECE-374: Lecture 4 - NFAs

Lecturer: Nickvash Kani
Chat moderator: Samir Khan

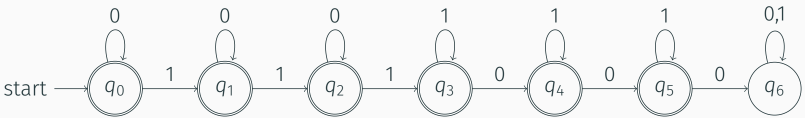
February 04, 2021

University of Illinois at Urbana-Champaign

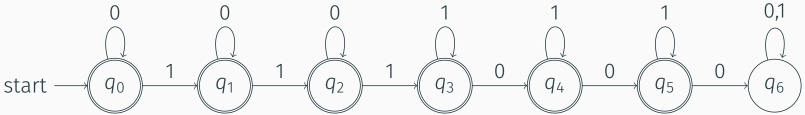
Pre-lecture brain teaser

Find the regular expression for the language containing all binary strings that **do not** contain the subsequence 111000

Simplifying DFAs

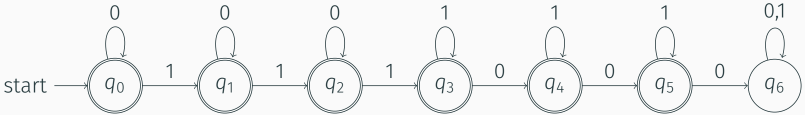


Simplifying DFAs

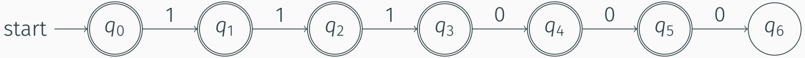


What if we draw the above figure as:

Simplifying DFAs



What if we draw the above figure as:



What does this mean?

Non-deterministic finite automata (NFA) Introduction

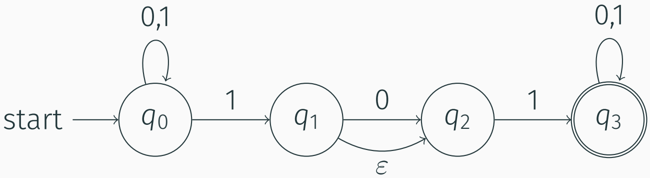
Non-deterministic Finite State Automata by example

When you come to a fork in the road, take it.

Non-deterministic Finite State Automata by example

When you come to a fork in the road, take it.

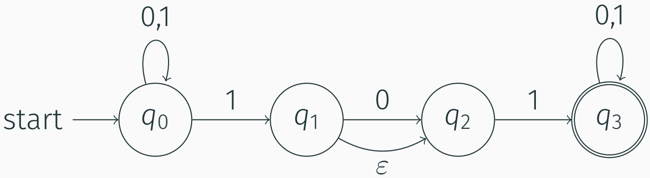
Today we'll talk about automata whose logic **is not** deterministic.



Non-deterministic Finite State Automata by example

When you come to a fork in the road, take it.

Today we'll talk about automata whose logic **is not** deterministic.



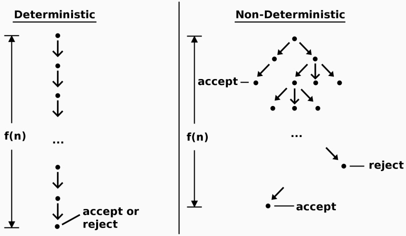
But first.... what the heck is non-determinism?

Non-determinism in computing

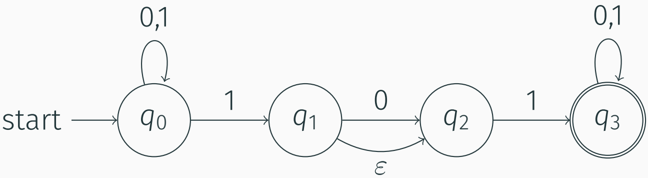
Non-determinism is a special property of algorithms.

An algorithm that is capable of taking multiple states concurrently. Whenever it reaches a choice, it takes both paths.

If there is a path for the string to be accepted by the machine, then the string is part of the language.

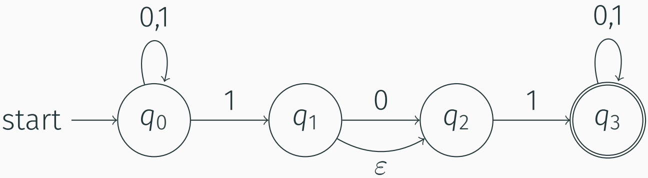


NFA acceptance: Informal



Informal definition: An NFA N accepts a string w iff some accepting state is reached by N from the start state on input w.

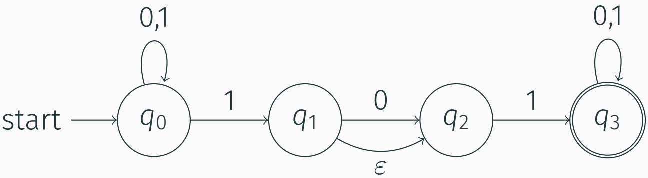
NFA acceptance: Informal



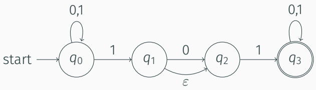
Informal definition: An NFA N accepts a string w iff some accepting state is reached by N from the start state on input w.

The language accepted (or recognized) by a NFA N is denote by $L(N)$ and defined as: $L(N) = \{w \mid N \text{ accepts } w\}$.

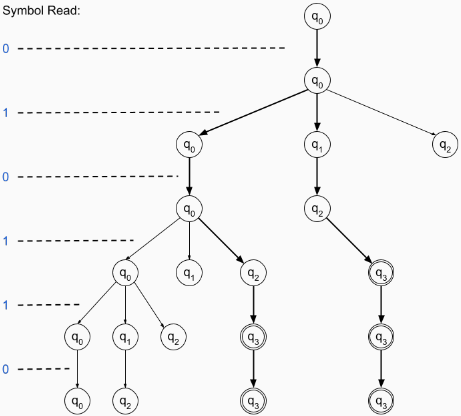
NFA acceptance: Example



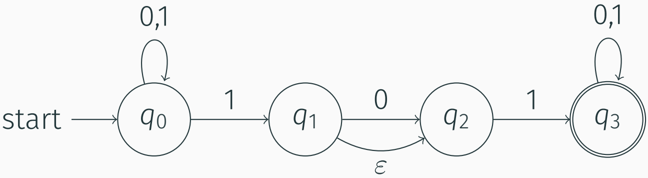
- Is 010110 accepted?



Is 010110 accepted?

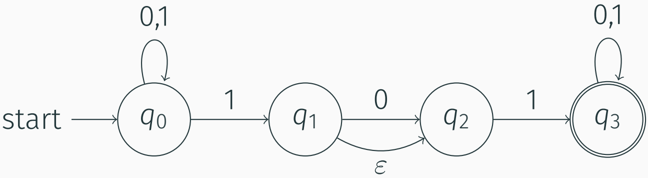


NFA acceptance: Example



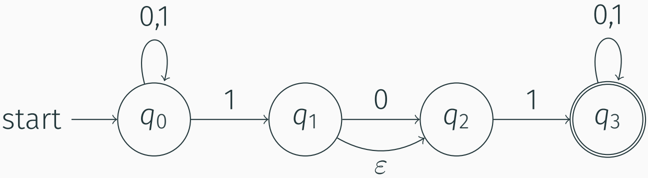
- Is 010110 accepted?

NFA acceptance: Example



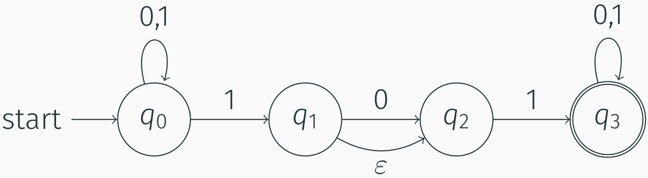
- Is 010110 accepted?
- Is 010 accepted?

NFA acceptance: Example



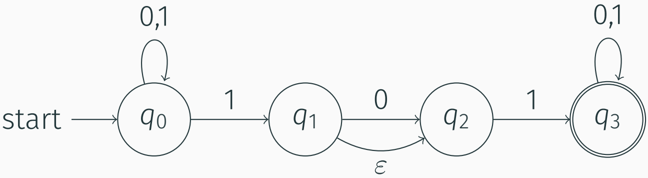
- Is 010110 accepted?
- Is 010 accepted?
- Is 101 accepted?

NFA acceptance: Example



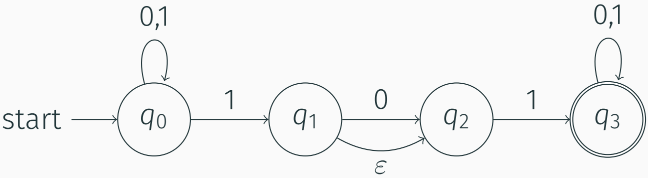
- Is 010110 accepted?
- Is 010 accepted?
- Is 101 accepted?
- Is 10011 accepted?

NFA acceptance: Example



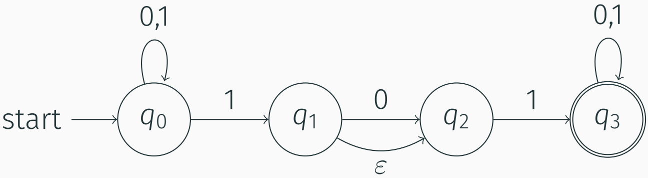
- Is 010110 accepted?
- Is 010 accepted?
- Is 101 accepted?
- Is 10011 accepted?
- What is the language accepted by N?

NFA acceptance: Example



- Is 010110 accepted?
- Is 010 accepted?
- Is 101 accepted?
- Is 10011 accepted?
- What is the language accepted by N?

NFA acceptance: Example



- Is 010110 accepted?
- Is 010 accepted?
- Is 101 accepted?
- Is 10011 accepted?
- What is the language accepted by N?

Comment: Unlike DFAs, it is easier in NFAs to show that a string is accepted than to show that a string is **not** accepted.

Formal definition of NFA

Formal Tuple Notation

Definition
A non-deterministic finite automata (NFA) $N = (Q, \Sigma, \delta, s, A)$ is a five tuple where

Formal Tuple Notation

Definition
A non-deterministic finite automata (NFA) $N = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,

Formal Tuple Notation

Definition
A non-deterministic finite automata (NFA) $N = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,

Formal Tuple Notation

Definition
A non-deterministic finite automata (NFA) $N = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$ is the transition function (here $\mathcal{P}(Q)$ is the power set of Q),

Formal Tuple Notation

Definition
A non-deterministic finite automata (NFA) $N = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$ is the transition function (here $\mathcal{P}(Q)$ is the power set of Q),

$\mathcal{P}(Q)$?

Reminder: Power set

Q: a set. Power set of Q is: $\mathcal{P}(Q) = 2^Q = \{X \mid X \subseteq Q\}$ is set of all subsets of Q.

Example
$Q = \{1, 2, 3, 4\}$

$$\mathcal{P}(Q) = \left\{ \begin{array}{c} \{1,2,3,4\}, \\ \{2,3,4\}, \{1,3,4\}, \{1,2,4\}, \{1,2,3\}, \\ \{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}, \\ \{1\}, \{2\}, \{3\}, \{4\}, \\ \{\} \end{array} \right\}$$

Formal Tuple Notation

Definition
A non-deterministic finite automata (NFA) $N = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$ is the transition function (here $\mathcal{P}(Q)$ is the power set of Q),

Formal Tuple Notation

Definition
A non-deterministic finite automata (NFA) $N = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$ is the transition function (here $\mathcal{P}(Q)$ is the power set of Q),
- $s \in Q$ is the start state,

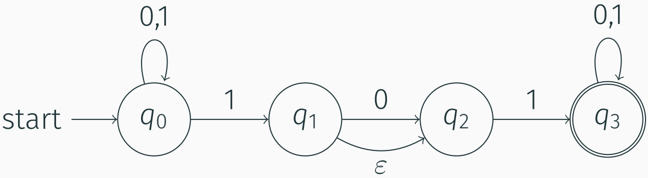
Formal Tuple Notation

Definition
A non-deterministic finite automata (NFA) $N = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$ is the transition function (here $\mathcal{P}(Q)$ is the power set of Q),
- $s \in Q$ is the start state,
- $A \subseteq Q$ is the set of accepting/final states.

$\delta(q, a)$ for $a \in \Sigma \cup \{\varepsilon\}$ is a subset of Q — a set of states.

Example



- $Q =$
- $\Sigma =$
- $\delta =$
- $s =$
- $A =$

Extending the transition function to strings

Extending the transition function to strings

- NFA $N = (Q, \Sigma, \delta, s, A)$

Extending the transition function to strings

- NFA $N = (Q, \Sigma, \delta, s, A)$
- $\delta(q, a)$: set of states that N can go to from q on reading $a \in \Sigma \cup \{\varepsilon\}$.

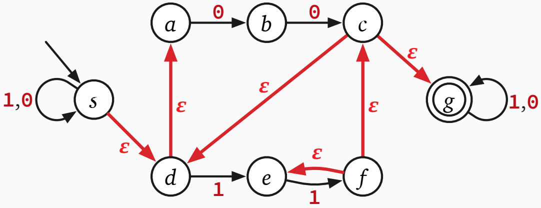
Extending the transition function to strings

- NFA $N = (Q, \Sigma, \delta, s, A)$
- $\delta(q, a)$: set of states that N can go to from q on reading $a \in \Sigma \cup \{\varepsilon\}$.
- Want transition function $\delta^* : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$

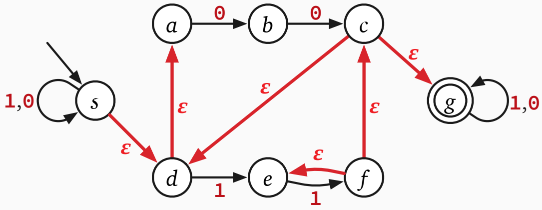
Extending the transition function to strings

- NFA $N = (Q, \Sigma, \delta, s, A)$
- $\delta(q, a)$: set of states that N can go to from q on reading $a \in \Sigma \cup \{\varepsilon\}$.
- Want transition function $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$
- $\delta^*(q, w)$: set of states reachable on input w starting in state q.

Definition
For NFA $N = (Q, \Sigma, \delta, s, A)$ and $q \in Q$ the $\epsilon\text{reach}(q)$ is the set of all states that q can reach using only ε-transitions.



Definition
For NFA $N = (Q, \Sigma, \delta, s, A)$ and $q \in Q$ the $\epsilon\text{reach}(q)$ is the set of all states that q can reach using only ε-transitions.



Definition
For $X \subseteq Q$: $\epsilon\text{reach}(X) = \bigcup_{x \in X} \epsilon\text{reach}(x)$.

Extending the transition function to strings

$\epsilon\text{reach}(q)$: set of all states that q can reach using only ε-transitions.

Definition
Inductive definition of $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$:

- if $w = \varepsilon$, $\delta^*(q, w) = \epsilon\text{reach}(q)$

Extending the transition function to strings

$\epsilon\text{reach}(q)$: set of all states that q can reach using only ε-transitions.

Definition
Inductive definition of $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$:

- if $w = \varepsilon$, $\delta^*(q, w) = \epsilon\text{reach}(q)$
- if $w = a$ where $a \in \Sigma$:

  $$\delta^*(q, a) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \delta(p, a)\right)$$

Extending the transition function to strings

$\epsilon\text{reach}(q)$: set of all states that q can reach using only ε-transitions.

Definition
Inductive definition of $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$:

- if $w = \varepsilon$, $\delta^*(q, w) = \epsilon\text{reach}(q)$
- if $w = a$ where $a \in \Sigma$:
$$\delta^*(q, a) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \delta(p, a)\right)$$
- if $w = ax$:
$$\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \left(\bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)\right)$$

Transition for strings: $w = ax$

$$\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \left(\bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)\right)$$

- $R = \epsilon\text{reach}(q) \implies$
  $$\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in R} \bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)$$
- $N = \bigcup_{p \in R} \delta^*(p, a)$: All the states reachable from q with the letter a.
- $\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{r \in N} \delta^*(r, x)\right)$

Formal definition of language accepted by N

Definition
A string w is accepted by NFA N if $\delta_N^*(s, w) \cap A \neq \emptyset$.

Definition
The language $L(N)$ accepted by a NFA $N = (Q, \Sigma, \delta, s, A)$ is

$$\{w \in \Sigma^* \mid \delta^*(s, w) \cap A \neq \emptyset\}.$$

Formal definition of language accepted by N

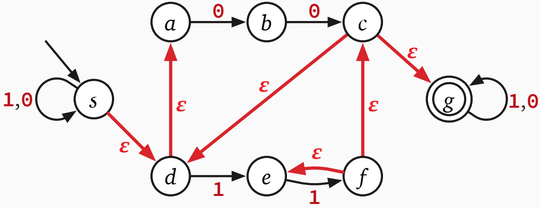
Definition
A string w is accepted by NFA N if $\delta_N^*(s, w) \cap A \neq \emptyset$.

Definition
The language $L(N)$ accepted by a NFA $N = (Q, \Sigma, \delta, s, A)$ is

$$\{w \in \Sigma^* \mid \delta^*(s, w) \cap A \neq \emptyset\}.$$

Important: Formal definition of the language of NFA above uses δ^* and not δ. As such, one does not need to include ε-transitions closure when specifying δ, since δ^* takes care of that.

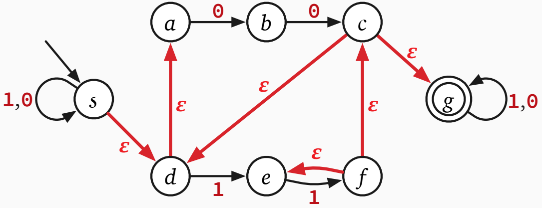
Example



What is:

- $\delta^*(s, \epsilon) =$

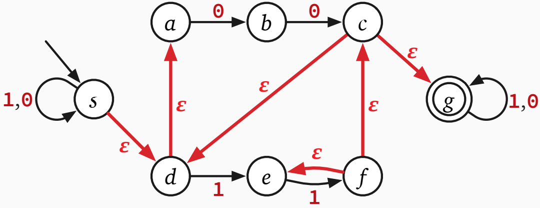
Example



What is:

- $\delta^*(s, \epsilon)$ =
- $\delta^*(s, 0)$ =

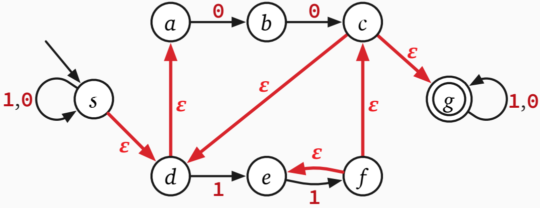
Example



What is:

- $\delta^*(s, \epsilon) =$
- $\delta^*(s, 0) =$
- $\delta^*(b, 0) =$

Example



What is:

- $\delta^*(s, \epsilon) =$
- $\delta^*(s, 0) =$
- $\delta^*(b, 0) =$
- $\delta^*(b, 00) =$

Why non-determinism?

- Non-determinism adds power to the model; richer programming language and hence (much) easier to “design” programs
- Fundamental in **theory** to prove many theorems
- Very important in **practice** directly and indirectly
- Many deep connections to various fields in Computer Science and Mathematics

Many interpretations of non-determinism. Hard to understand at the outset. Get used to it and then you will appreciate it slowly.

Constructing NFAs

DFAs and NFAs

- Every DFA is a NFA so NFAs are at least as powerful as DFAs.
- NFAs prove ability to “guess and verify” which simplifies design and reduces number of states
- Easy proofs of some closure properties

Example

Strings that represent decimal numbers.

Examples: 154, 345.75332, 534677567.1

Example

Strings that represent valid C comments.

Examples:
* Comment 1 *\
\\Comment 2

Example

$L_3 = \{$bitstrings that have a 1 three positions from the end$\}$

A simple transformation

Theorem
For every NFA N there is another NFA N' such that $L(N) = L(N')$ and such that N' has the following two properties:

- *N' has single final state f that has no outgoing transitions*
- *The start state s of N is different from f*

A simple transformation

Theorem
For every NFA N there is another NFA N' such that $L(N) = L(N')$ and such that N' has the following two properties:

- *N' has single final state f that has no outgoing transitions*
- *The start state s of N is different from f*

Why couldn't we say this for DFA's?

A simple transformation

Hint: Consider the L = $01 + 10$.

Closure Properties of NFAs

Closure properties of NFAs

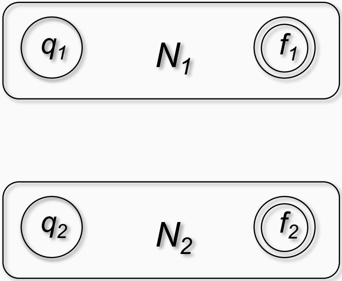
Are the class of languages accepted by NFAs closed under the following operations?

- union
- intersection
- concatenation
- Kleene star
- complement

Theorem
For any two NFAs N_1 and N_2 there is a NFA N such that $L(N) = L(N_1) \cup L(N_2)$.

Closure under union

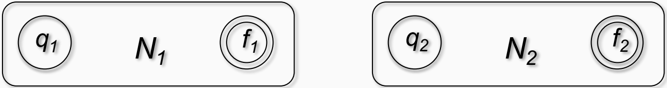
Theorem
For any two NFAs N_1 and N_2 there is a NFA N such that $L(N) = L(N_1) \cup L(N_2)$.



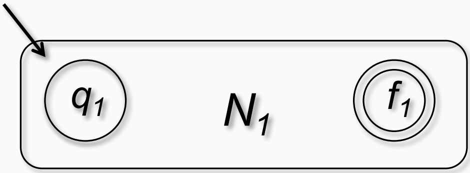
Closure under concatenation

Theorem
For any two NFAs N_1 and N_2 there is a NFA N such that $L(N) = L(N_1) \bullet L(N_2)$.

Theorem
For any two NFAs N_1 and N_2 there is a NFA N such that $L(N) = L(N_1) \bullet L(N_2)$.

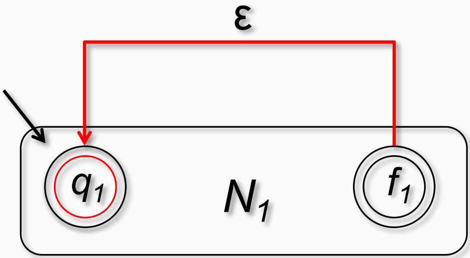


Theorem
For any NFA N_1 there is a NFA N such that $L(N) = (L(N_1))^*$.

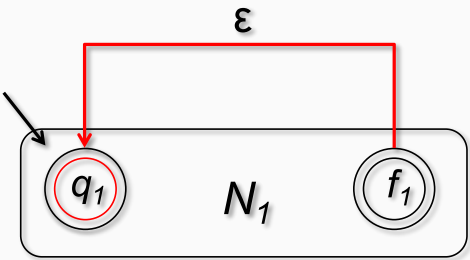


Closure under Kleene star

Theorem
For any NFA N_1 there is a NFA N such that $L(N) = (L(N_1))^*$.



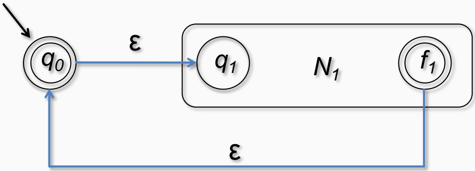
Theorem
For any NFA N_1 there is a NFA N such that $L(N) = (L(N_1))^*$.



Does not work! Why?

Closure under Kleene star

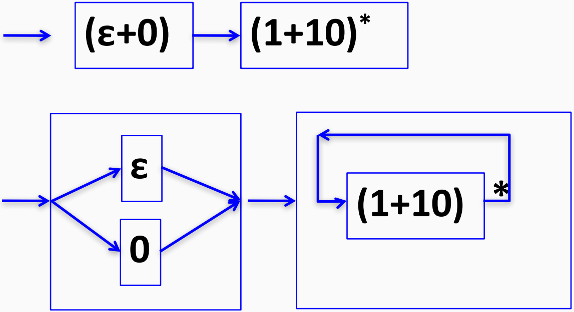
Theorem
For any NFA N_1 there is a NFA N such that $L(N) = (L(N_1))^*$.



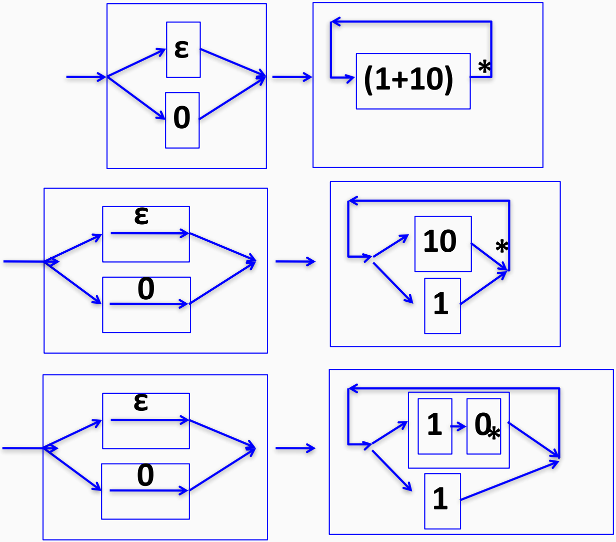
NFAs capture Regular Languages

Example

$(\varepsilon+0)(1+10)^*$

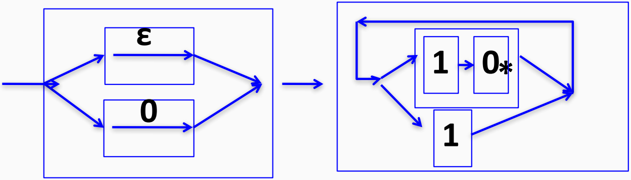


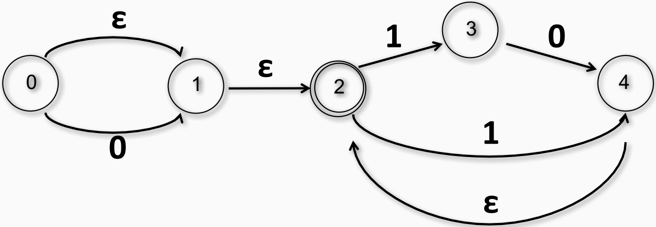
Example



Example

Final NFA simplified slightly to reduce states

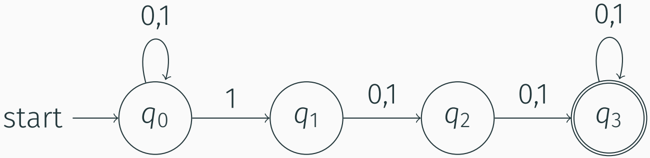




Last thought

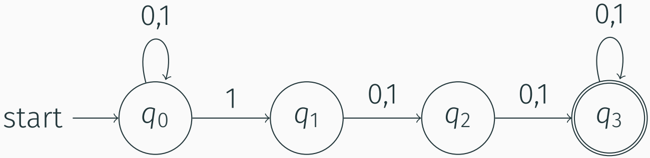
Equivalence

Do all NFAs have a corresponding DFA?



Equivalence

Do all NFAs have a corresponding DFA?



Yes but it likely won't be pretty.