Non-deterministic Finite Automata (NFAs)

Lecture 4

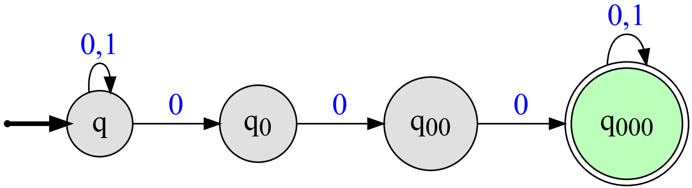
Thursday, September 1, 2022

LaTeXed: October 13, 2022 14:18

4.1
NFA Introduction

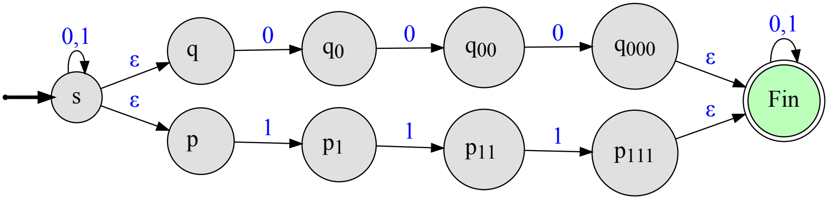
Non-deterministic Finite State Automata by example

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



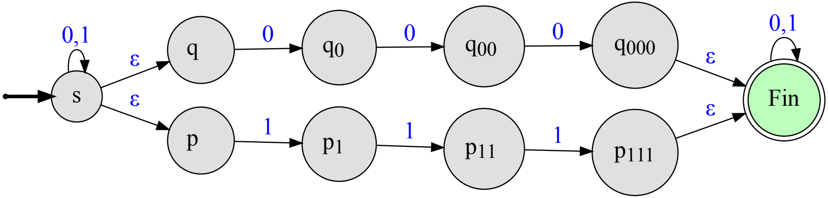
Non-deterministic Finite State Automata by example II

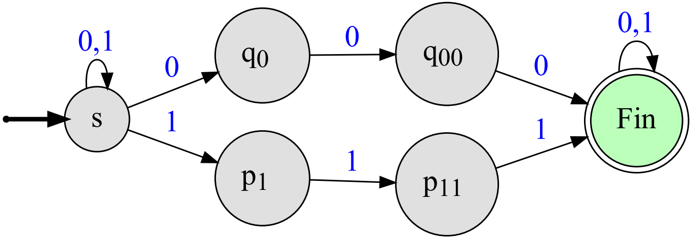
..but only if it is made out of silver.



Non-deterministic Finite State Automata by example II

..but only if it is made out of silver.



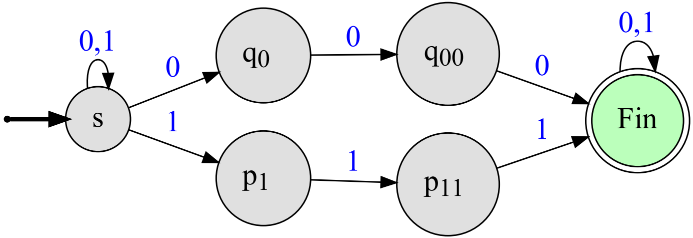


More efficient NFA:

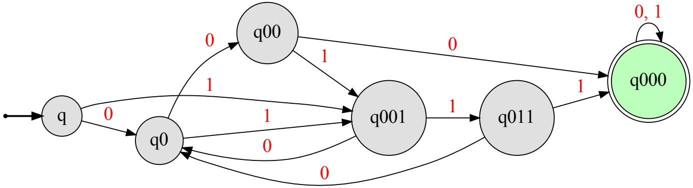
Non-deterministic Finite State Automata by example II

..but only if it is made out of silver.

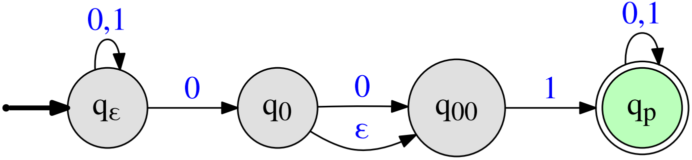
More efficient NFA:



Not the point...
...because DFA can still do it efficiently.



Non-deterministic Finite State Automata (NFAs)



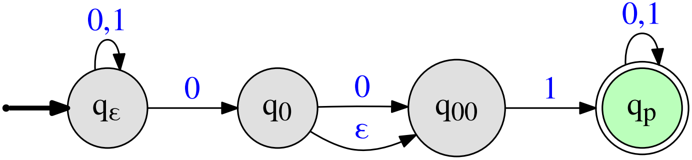
Differences from DFA

- From state **q** on same letter **a** $\in$ **Σ** multiple possible states
- No transitions from **q** on some letters
- ε-transitions!

Questions:

- Is this a "real" machine?
- What does it do?

Non-deterministic Finite State Automata (NFAs)



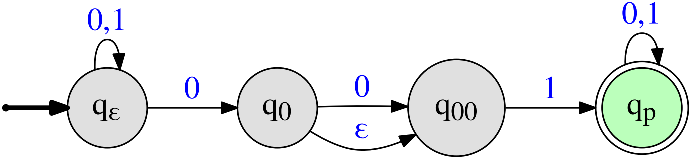
Differences from DFA

- From state **q** on same letter **a** $\in$ **Σ** multiple possible states
- No transitions from **q** on some letters
- ε-transitions!

Questions:

- Is this a "real" machine?
- What does it do?

Non-deterministic Finite State Automata (NFAs)



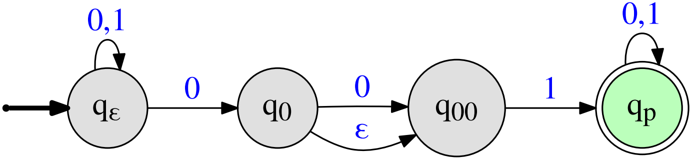
Differences from DFA

- From state **q** on same letter $\mathbf{a} \in \mathbf{\Sigma}$ multiple possible states
- No transitions from **q** on some letters
- ε-transitions!

Questions:

- Is this a "real" machine?
- What does it do?

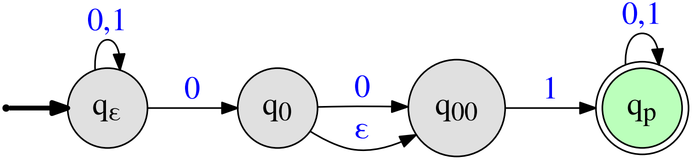
NFA behavior



Machine on input string **w** from state **q** can lead to set of states (could be empty)

- From $\mathbf{q}_\varepsilon$ on **1**
- From $\mathbf{q}_\varepsilon$ on **0**
- From $\mathbf{q}_0$ on ε
- From $\mathbf{q}_\varepsilon$ on **01**
- From $\mathbf{q}_{00}$ on **00**

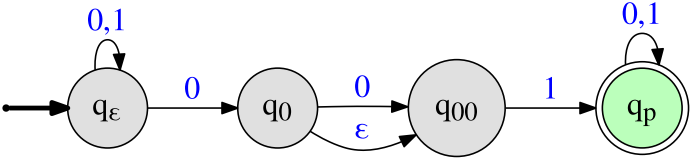
NFA behavior



Machine on input string **w** from state **q** can lead to set of states (could be empty)

- From $\mathbf{q_\varepsilon}$ on **1**
- From $\mathbf{q_\varepsilon}$ on **0**
- From $\mathbf{q_0}$ on ε
- From $\mathbf{q_\varepsilon}$ on **01**
- From $\mathbf{q_{00}}$ on **00**

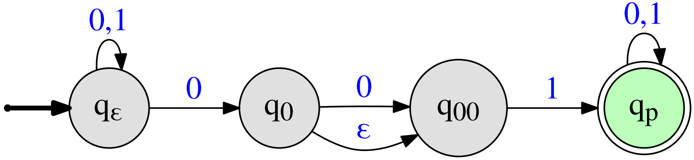
NFA behavior



Machine on input string **w** from state **q** can lead to set of states (could be empty)

- From $\mathbf{q}_\varepsilon$ on **1**
- From $\mathbf{q}_\varepsilon$ on **0**
- From $\mathbf{q}_0$ on ε
- From $\mathbf{q}_\varepsilon$ on **01**
- From $\mathbf{q}_{00}$ on **00**

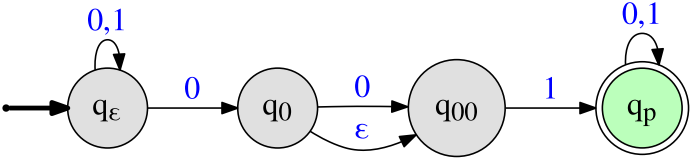
NFA behavior



Machine on input string **w** from state **q** can lead to set of states (could be empty)

- From $\mathbf{q}_\varepsilon$ on **1**
- From $\mathbf{q}_\varepsilon$ on **0**
- From $\mathbf{q}_0$ on ε
- From $\mathbf{q}_\varepsilon$ on **01**
- From $\mathbf{q}_{00}$ on **00**

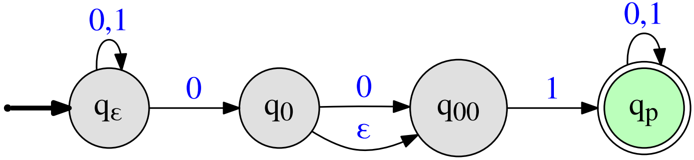
NFA behavior



Machine on input string **w** from state **q** can lead to set of states (could be empty)

- From $\mathbf{q}_\varepsilon$ on **1**
- From $\mathbf{q}_\varepsilon$ on **0**
- From $\mathbf{q}_0$ on ε
- From $\mathbf{q}_\varepsilon$ on **01**
- From $\mathbf{q}_{00}$ on **00**

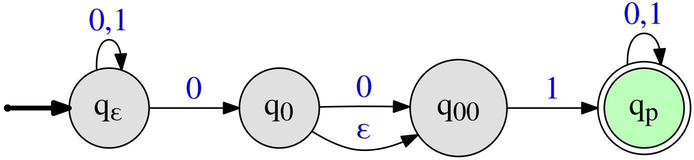
NFA behavior



Machine on input string **w** from state **q** can lead to set of states (could be empty)

- From $\mathbf{q}_\varepsilon$ on **1**
- From $\mathbf{q}_\varepsilon$ on **0**
- From $\mathbf{q}_0$ on ε
- From $\mathbf{q}_\varepsilon$ on **01**
- From $\mathbf{q}_{00}$ on **00**

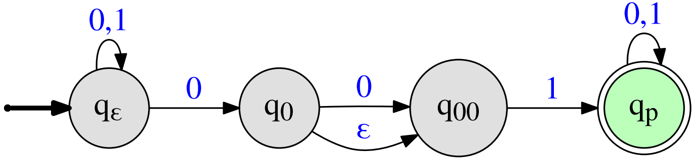
NFA acceptance: informal



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: $\mathbf{L(N)} = \{\mathbf{w} \mid \mathbf{N} \text{ accepts } \mathbf{w}\}$.

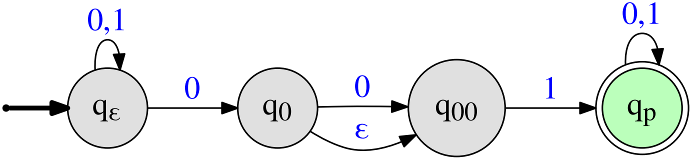
NFA acceptance: informal



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 | N accepts w}.**

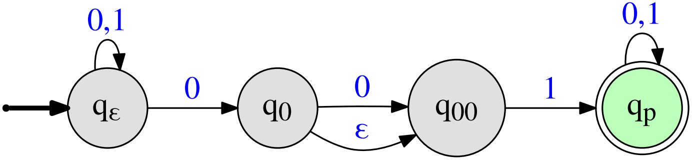
NFA acceptance: example



- Is **01** accepted?
- Is **001** accepted?
- Is **100** accepted?
- Are all strings in **1*01** 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.

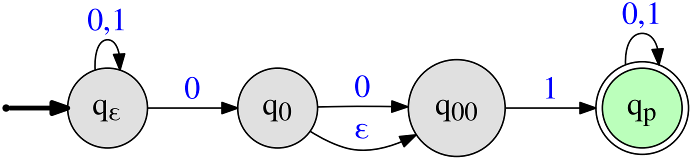
NFA acceptance: example



- Is **01** accepted?
- Is **001** accepted?
- Is **100** accepted?
- Are all strings in **1*01** 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.

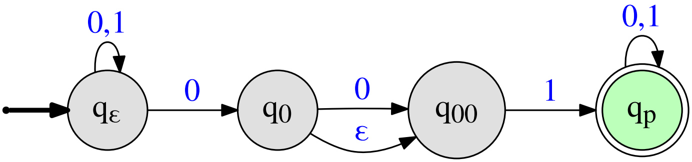
NFA acceptance: example



- Is **01** accepted?
- Is **001** accepted?
- Is **100** accepted?
- Are all strings in **1*01** 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.

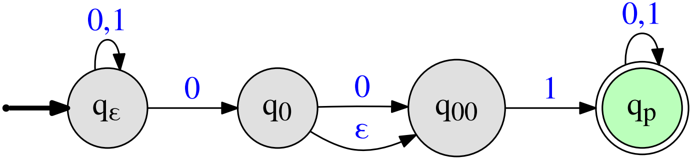
NFA acceptance: example



- Is **01** accepted?
- Is **001** accepted?
- Is **100** accepted?
- Are all strings in **1*01** 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.

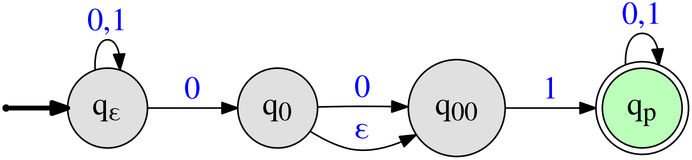
NFA acceptance: example



- Is **01** accepted?
- Is **001** accepted?
- Is **100** accepted?
- Are all strings in **1*01** 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.

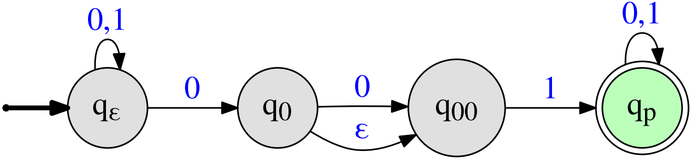
NFA acceptance: example



- Is **01** accepted?
- Is **001** accepted?
- Is **100** accepted?
- Are all strings in **1*01** 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.

NFA acceptance: example

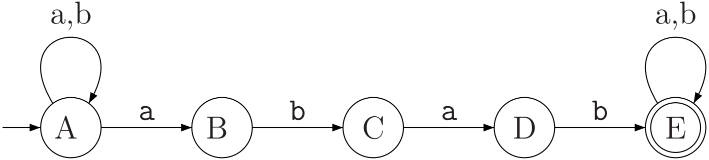


- Is **01** accepted?
- Is **001** accepted?
- Is **100** accepted?
- Are all strings in **1*01** accepted?
- What is the language accepted by **N**?

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

Simulating NFA

Example the first

(N1) 

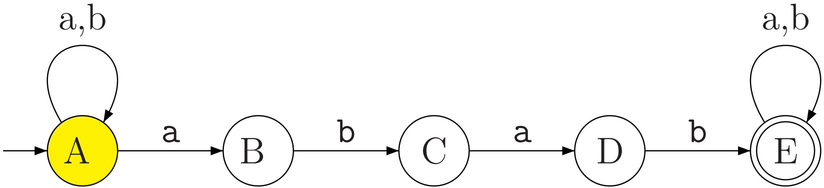
Run it on input **ababa**.

Idea: Keep track of the states where the NFA might be at any given time.

Simulating NFA

Example the first

$t = 0$:

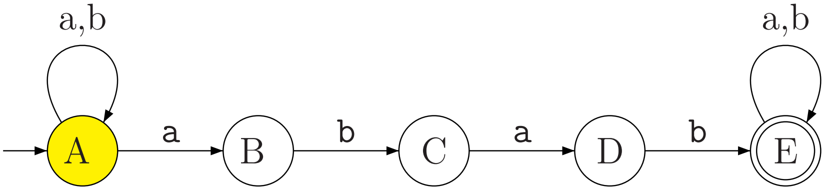


Remaining input: **ababa**.

Simulating NFA

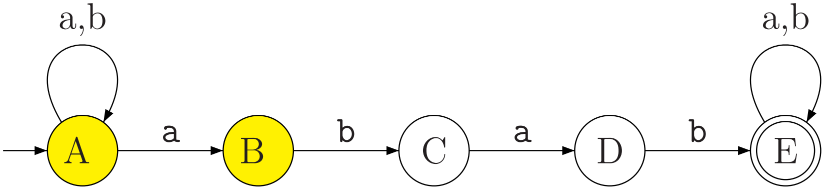
Example the first

$t = 0$:



Remaining input: **ababa**.

$t = 1$:

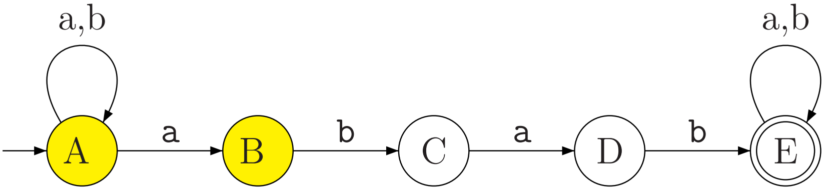


Remaining input: **baba**.

Simulating NFA

Example the first

$t = 1$:

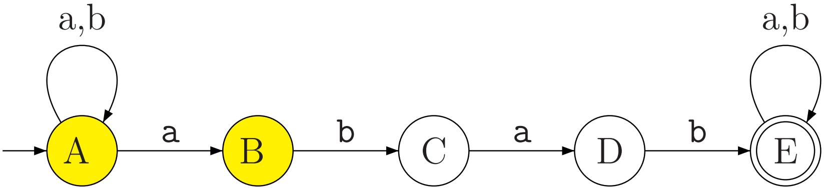


Remaining input: **baba**.

Simulating NFA

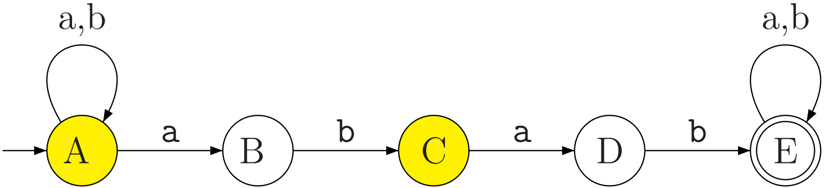
Example the first

$t = 1$:



Remaining input: **baba**.

$t = 2$:

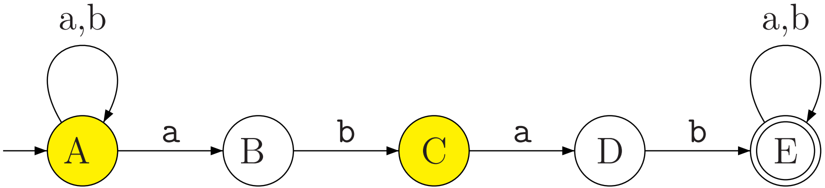


Remaining input: **aba**.

Simulating NFA

Example the first

$t = 2$:

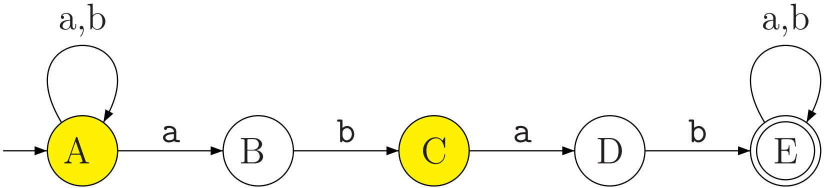


Remaining input: **aba**.

Simulating NFA

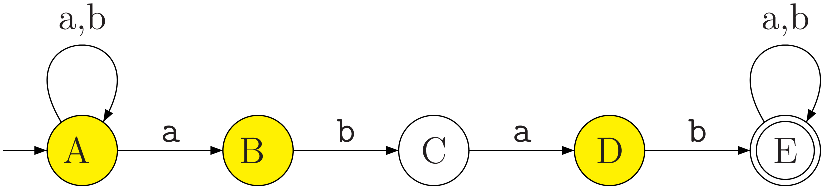
Example the first

$t = 2$:



Remaining input: **aba**.

$t = 3$:

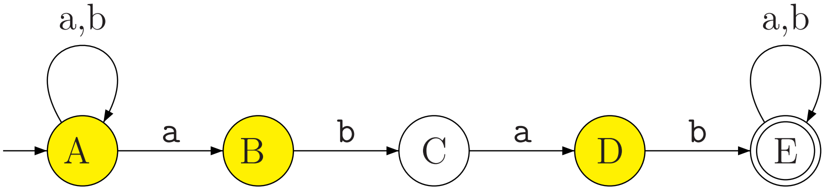


Remaining input: **ba**.

Simulating NFA

Example the first

$t = 3$:

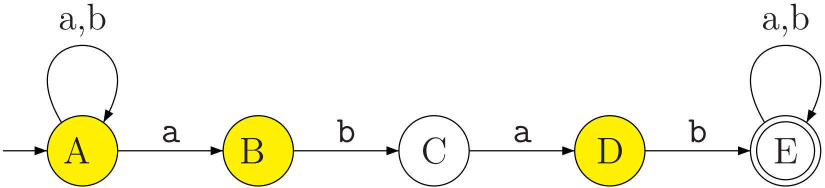


Remaining input: **ba**.

Simulating NFA

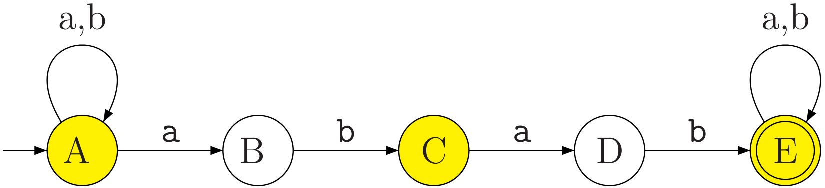
Example the first

$t = 3$:



Remaining input: **ba**.

$t = 4$:

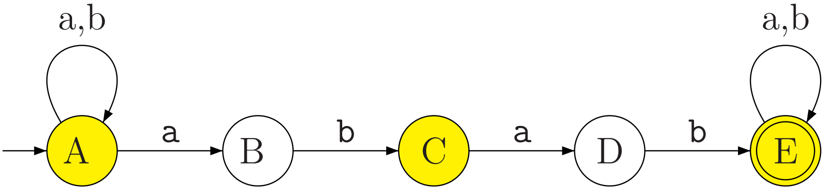


Remaining input: **a**.

Simulating NFA

Example the first

$t = 4$:

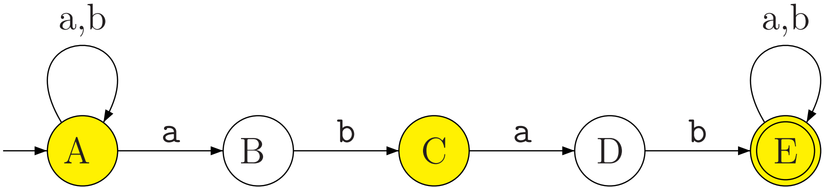


Remaining input: **a**.

Simulating NFA

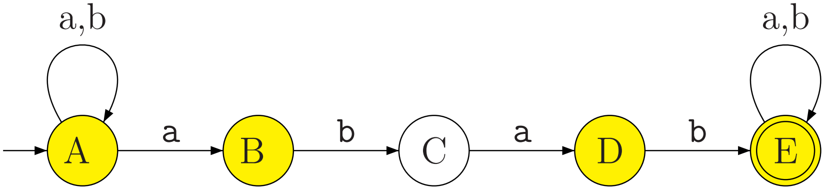
Example the first

$t = 4$:



Remaining input: **a**.

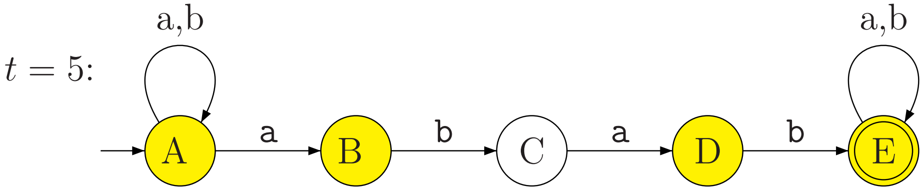
$t = 5$:



Remaining input: ε.

Simulating NFA

Example the first



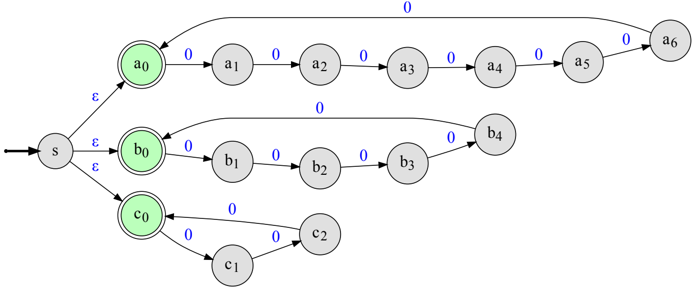
Remaining input: ε.

Accepts: **ababa**.

An exercise

For you to think about...

A. What is the language that the following NFA accepts?



B. What is the minimal number of states in a DFA that recognizes the same language?

4.1.1
Formal definition of NFA

Reminder: Power set

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

Example 4.1.

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

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

Formal Tuple Notation

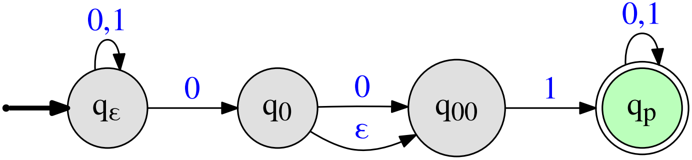
Definition 4.2.

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

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

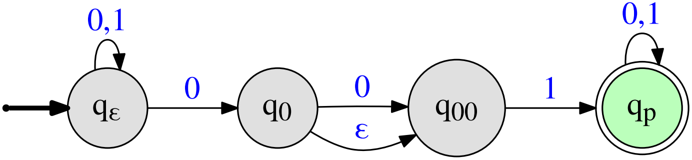
$\delta(\mathbf{q}, \mathbf{a})$ for $\mathbf{a} \in \mathbf{\Sigma} \cup \{\varepsilon\}$ is a subset of $\mathbf{Q}$ — a set of states.

Example



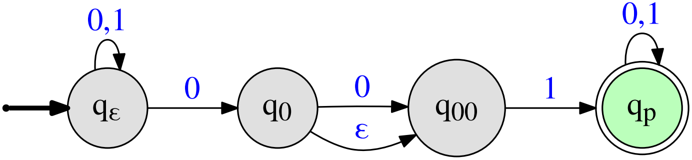
- $Q = \{q_\varepsilon, q_0, q_{00}, q_p\}$
- $\Sigma = \{0, 1\}$
- δ
- $s = q_\varepsilon$
- $A = \{q_p\}$

Example



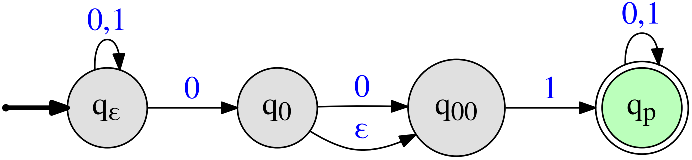
- $\mathbf{Q = \{q_\varepsilon, q_0, q_{00}, q_p\}}$
- $\Sigma = \{0, 1\}$
- δ
- $s = q_\varepsilon$
- $A = \{q_p\}$

Example



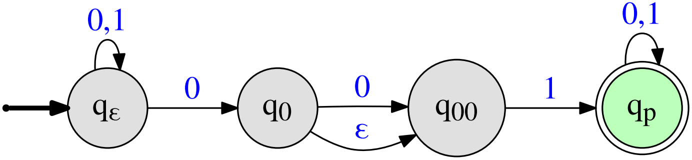
- $\mathbf{Q = \{q_\varepsilon, q_0, q_{00}, q_p\}}$
- $\mathbf{\Sigma} = \{0, 1\}$
- δ
- $s = q_\varepsilon$
- $A = \{q_p\}$

Example



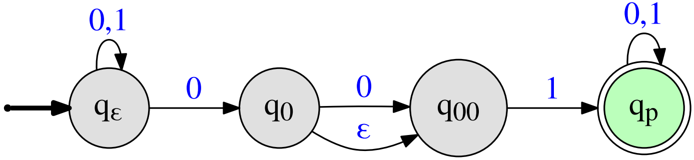
- $Q = \{q_\varepsilon, q_0, q_{00}, q_p\}$
- $\Sigma = \{0, 1\}$
- δ
- $s = q_\varepsilon$
- $A = \{q_p\}$

Example



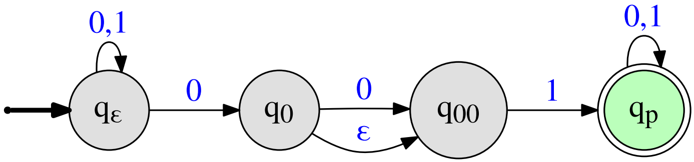
- $Q = \{q_\varepsilon, q_0, q_{00}, q_p\}$
- $\Sigma = \{0, 1\}$
- δ
- $s = q_\varepsilon$
- $A = \{q_p\}$

Example



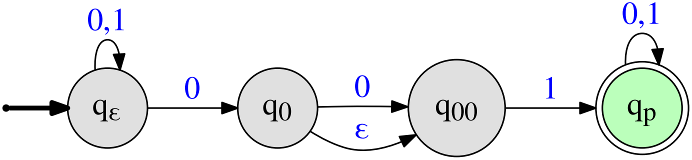
- $\mathbf{Q = \{q_\varepsilon, q_0, q_{00}, q_p\}}$
- $\mathbf{\Sigma = \{0, 1\}}$
- δ
- $\mathbf{s} = q_\varepsilon$
- $\mathbf{A} = \{q_p\}$

Example



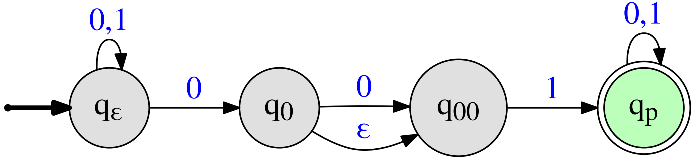
- $Q = \{q_\varepsilon, q_0, q_{00}, q_p\}$
- $\Sigma = \{0, 1\}$
- δ
- $s = q_\varepsilon$
- $A = \{q_p\}$

Example



- $\mathbf{Q = \{q_\varepsilon, q_0, q_{00}, q_p\}}$
- $\mathbf{\Sigma = \{0, 1\}}$
- δ
- $\mathbf{s = q_\varepsilon}$
- $\mathbf{A = \{q_p\}}$

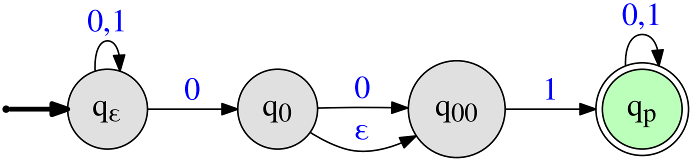
Example



- $\mathbf{Q} = \{q_\varepsilon, q_0, q_{00}, q_p\}$
- $\boldsymbol{\Sigma} = \{0, 1\}$
- δ
- $\mathbf{s} = q_\varepsilon$
- $\mathbf{A} = \{q_p\}$

Example

Transition function in detail...



$$\delta(q_\varepsilon, \varepsilon) = \{q_\varepsilon\}$$
$$\delta(q_\varepsilon, 0) = \{q_\varepsilon, q_0\}$$
$$\delta(q_\varepsilon, 1) = \{q_\varepsilon\}$$

$$\delta(q_0, \varepsilon) = \{q_0, q_{00}\}$$
$$\delta(q_0, 0) = \{q_{00}\}$$
$$\delta(q_0, 1) = \{\}$$

$$\delta(q_{00}, \varepsilon) = \{q_{00}\}$$
$$\delta(q_{00}, 0) = \{\}$$
$$\delta(q_{00}, 1) = \{q_p\}$$

$$\delta(q_p, \varepsilon) = \{q_p\}$$
$$\delta(q_p, 0) = \{q_p\}$$
$$\delta(q_p, 1) = \{q_p\}$$

4.1.2
Extending the transition function to strings

Extending the transition function to strings

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

Extending the transition function to strings

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

Extending the transition function to strings

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

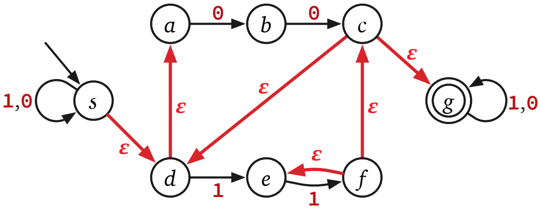
Extending the transition function to strings

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

Extending the transition function to strings

Definition 4.3.

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



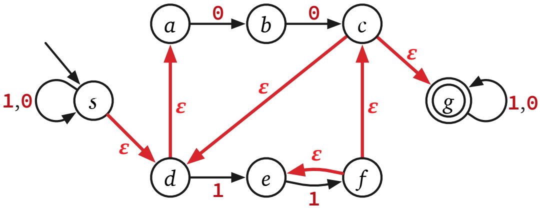
Definition 4.4.

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

Extending the transition function to strings

Definition 4.3.

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



Definition 4.4.

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

Extending the transition function to strings

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

Definition 4.5.

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

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

Extending the transition function to strings

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

Definition 4.5.

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

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

Extending the transition function to strings

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

Definition 4.5.

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

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

Transition for strings: $\mathbf{w = ax}$

Translation...

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

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

Transition for strings: $\mathbf{w = ax}$

Translation...

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

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

Transition for strings: $\mathbf{w} = \mathbf{ax}$

Translation...

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

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

Transition for strings: $\mathbf{w = ax}$

Translation...

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

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

Formal definition of language accepted by $\mathbf{N}$

Definition 4.6.

A string $\mathbf{w}$ is accepted by NFA $\mathbf{N}$ if $\delta_{\mathbf{N}}^{*}(\mathbf{s}, \mathbf{w}) \cap \mathbf{A} \neq \emptyset$.

Definition 4.7.

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

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

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.

Formal definition of language accepted by $\mathbf{N}$

Definition 4.6.

A string $\mathbf{w}$ is accepted by NFA $\mathbf{N}$ if $\delta^*_{\mathbf{N}}(\mathbf{s}, \mathbf{w}) \cap \mathbf{A} \neq \emptyset$.

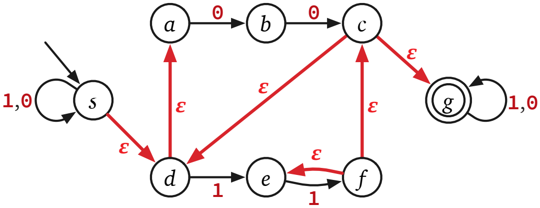
Definition 4.7.

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

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

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.

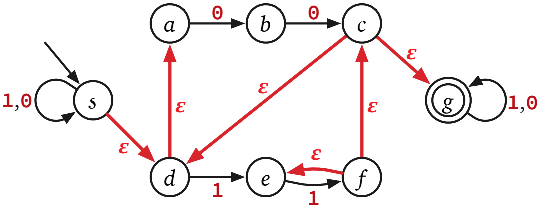
Example



What is:

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

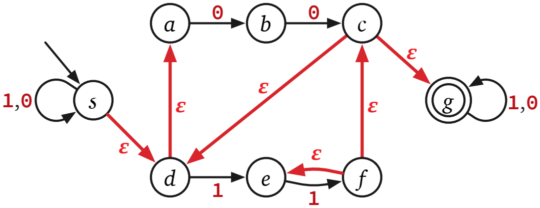
Example



What is:

- $\delta^*(\mathbf{s}, \epsilon)$
- $\delta^*(\mathbf{s}, 0)$
- $\delta^*(\mathbf{c}, 0)$
- $\delta^*(\mathbf{b}, 00)$

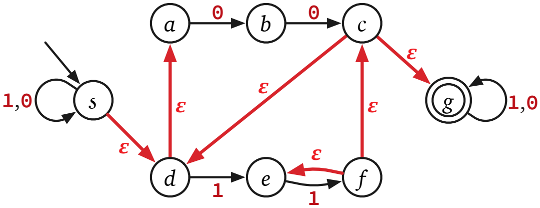
Example



What is:

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

Example



What is:

- $\delta^*(\mathbf{s}, \epsilon)$
- $\delta^*(\mathbf{s}, \mathbf{0})$
- $\delta^*(\mathbf{c}, \mathbf{0})$
- $\delta^*(\mathbf{b}, \mathbf{00})$

Another definition of computation

Definition 4.8.

$q \xrightarrow{w}_N p$: State p of NFA N is <u>**reachable**</u> from q on w $\iff$ there exists a sequence of states $r_0, r_1, \ldots, r_k$ and a sequence $x_1, x_2, \ldots, x_k$ where $x_i \in \Sigma \cup \{\varepsilon\}$, for each i, such that:

- $r_0 = q$,
- for each i, $r_{i+1} \in \delta^*(r_i, x_{i+1})$,
- $r_k = p$, and
- $w = x_1 x_2 x_3 \cdots x_k$.

Definition 4.9.

$$\delta_N^*(q, w) = \left\{ p \in Q \;\middle|\; q \xrightarrow{w}_N p \right\}.$$

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.

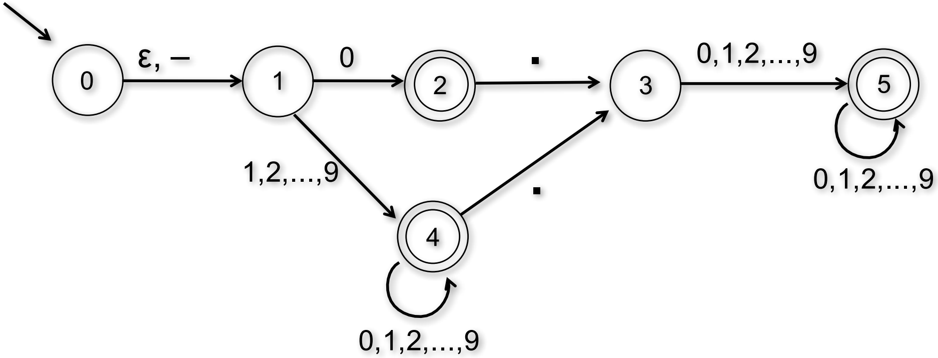
4.2
Constructing NFAs

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.



Example

- {strings that contain CS374 as a substring}
- {strings that contain CS374 or CS473 as a substring}
- {strings that contain CS374 and CS473 as substrings}

Example

- {strings that contain CS374 as a substring}
- {strings that contain CS374 or CS473 as a substring}
- {strings that contain CS374 and CS473 as substrings}

Example

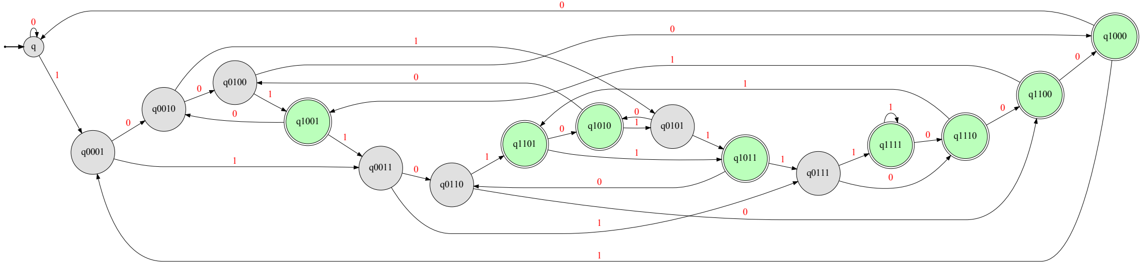
- {strings that contain CS374 as a substring}
- {strings that contain CS374 or CS473 as a substring}
- {strings that contain CS374 and CS473 as substrings}

Example

$\mathbf{L_k} = \{$bitstrings that have a 1 $\mathbf{k}$ positions from the end$\}$

DFA for same task is much bigger...

$\mathbf{L_4}$ = {bitstrings that have a **1** in fourth position from the end}



A simple transformation

Theorem 4.1.

For every NFA $\mathbf{N}$ *there is another* NFA $\mathbf{N'}$ *such that* $\mathbf{L(N) = L(N')}$ *and such that* $\mathbf{N'}$ *has the following two properties:*

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

4.3
Closure Properties of NFAs

Closure properties of NFAs

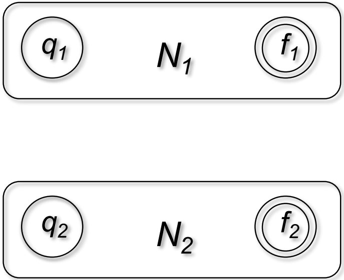
Are the class of languages accepted by NFAs closed under the following operations?

- union
- intersection
- concatenation
- Kleene star
- complement

Closure under union

Theorem 4.1.

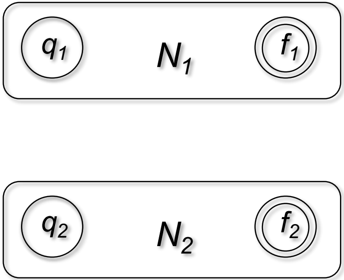
For any two NFAs $\mathbf{N_1}$ *and* $\mathbf{N_2}$ *there is a* NFA $\mathbf{N}$ *such that* $\mathbf{L(N) = L(N_1) \cup L(N_2)}$.



Closure under union

Theorem 4.1.

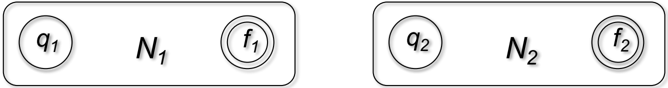
For any two NFAs $\mathbf{N_1}$ *and* $\mathbf{N_2}$ *there is a NFA* $\mathbf{N}$ *such that* $\mathbf{L(N) = L(N_1) \cup L(N_2)}$.



Closure under concatenation

Theorem 4.2.

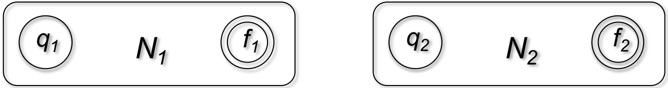
For any two NFAs $\mathbf{N_1}$ *and* $\mathbf{N_2}$ *there is a* NFA $\mathbf{N}$ *such that* $\mathbf{L(N) = L(N_1) \bullet L(N_2)}$.



Closure under concatenation

Theorem 4.2.

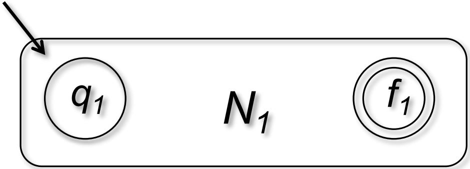
For any two NFA*s* $\mathbf{N_1}$ *and* $\mathbf{N_2}$ *there is a* NFA $\mathbf{N}$ *such that* $\mathbf{L(N) = L(N_1) \bullet L(N_2)}$.



Closure under Kleene star

Theorem 4.3.

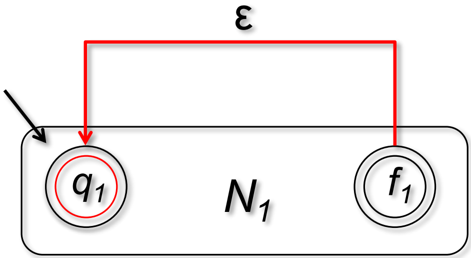
For any NFA $\mathbf{N_1}$ *there is a* NFA $\mathbf{N}$ *such that* $\mathbf{L(N) = (L(N_1))^*}$.



Closure under Kleene star

Theorem 4.4.

For any NFA $\mathbf{N_1}$ *there is a* NFA $\mathbf{N}$ *such that* $\mathbf{L(N) = (L(N_1))^*}$.

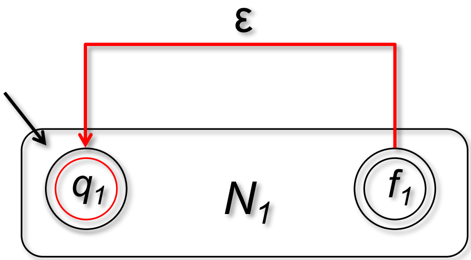


Does not work! Why?

Closure under Kleene star

Theorem 4.4.

For any NFA $\mathbf{N_1}$ *there is a* NFA $\mathbf{N}$ *such that* $\mathbf{L(N) = (L(N_1))^*}$.

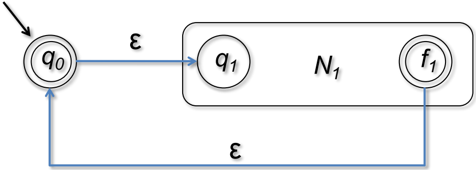


Does not work! Why?

Closure under Kleene star

Theorem 4.5.

For any NFA $\mathbf{N_1}$ *there is a* NFA $\mathbf{N}$ *such that* $\mathbf{L(N) = (L(N_1))}^*$.



4.4
NFAs capture Regular Languages

Regular Languages Recap

Regular Languages

$\emptyset$ regular
$\{\epsilon\}$ regular
$\{\mathbf{a}\}$ regular for $\mathbf{a} \in \mathbf{\Sigma}$
$\mathbf{R_1} \cup \mathbf{R_2}$ regular if both are
$\mathbf{R_1 R_2}$ regular if both are
$\mathbf{R}^*$ is regular if $\mathbf{R}$ is

Regular Expressions

$\emptyset$ denotes $\emptyset$
ϵ denotes $\{\epsilon\}$
$\mathbf{a}$ denote $\{\mathbf{a}\}$
$\mathbf{r_1} + \mathbf{r_2}$ denotes $\mathbf{R_1} \cup \mathbf{R_2}$
$\mathbf{r_1 r_2}$ denotes $\mathbf{R_1 R_2}$
$\mathbf{r}^*$ denote $\mathbf{R}^*$

Regular expressions denote regular languages — they explicitly show the operations that were used to form the language

NFAs and Regular Language

Theorem 4.1.

For every regular language $\mathbf{L}$ *there is an* NFA $\mathbf{N}$ *such that* $\mathbf{L} = \mathbf{L(N)}$.

Proof strategy:

- For every regular expression $\mathbf{r}$ show that there is a NFA $\mathbf{N}$ such that $\mathbf{L(r)} = \mathbf{L(N)}$
- Induction on length of $\mathbf{r}$

NFAs and Regular Language

- For every regular expression $\mathbf{r}$ show that there is a NFA $\mathbf{N}$ such that $\mathbf{L(r) = L(N)}$
- Induction on length of $\mathbf{r}$

Base cases: $\emptyset$, $\{\varepsilon\}$, $\{\mathbf{a}\}$ for $\mathbf{a} \in \mathbf{\Sigma}$.

NFAs and Regular Language

- For every regular expression $\mathbf{r}$ show that there is a NFA $\mathbf{N}$ such that $\mathbf{L(r) = L(N)}$
- Induction on length of $\mathbf{r}$

Inductive cases:

- $\mathbf{r_1, r_2}$ regular expressions and $\mathbf{r = r_1 + r_2}$.
  By induction there are NFAs $\mathbf{N_1, N_2}$ s.t $\mathbf{L(N_1) = L(r_1)}$ and $\mathbf{L(N_2) = L(r_2)}$. We have already seen that there is NFA $\mathbf{N}$ s.t $\mathbf{L(N) = L(N_1) \cup L(N_2)}$, hence $\mathbf{L(N) = L(r)}$
- $\mathbf{r = r_1 \bullet r_2}$. Use closure of NFA languages under concatenation
- $\mathbf{r = (r_1)^*}$. Use closure of NFA languages under Kleene star

NFAs and Regular Language

- For every regular expression $\mathbf{r}$ show that there is a NFA $\mathbf{N}$ such that $\mathbf{L(r) = L(N)}$
- Induction on length of $\mathbf{r}$

Inductive cases:

- $\mathbf{r_1, r_2}$ regular expressions and $\mathbf{r = r_1 + r_2}$.
  By induction there are NFAs $\mathbf{N_1, N_2}$ s.t
  $\mathbf{L(N_1) = L(r_1)}$ and $\mathbf{L(N_2) = L(r_2)}$. We have already seen that there is NFA $\mathbf{N}$ s.t $\mathbf{L(N) = L(N_1) \cup L(N_2)}$, hence $\mathbf{L(N) = L(r)}$
- $\mathbf{r = r_1 \bullet r_2}$. Use closure of NFA languages under concatenation
- $\mathbf{r = (r_1)^*}$. Use closure of NFA languages under Kleene star

NFAs and Regular Language

- For every regular expression $\mathbf{r}$ show that there is a NFA $\mathbf{N}$ such that $\mathbf{L(r) = L(N)}$
- Induction on length of $\mathbf{r}$

Inductive cases:

- $\mathbf{r_1, r_2}$ regular expressions and $\mathbf{r = r_1 + r_2}$.
  By induction there are NFAs $\mathbf{N_1, N_2}$ s.t
  $\mathbf{L(N_1) = L(r_1)}$ and $\mathbf{L(N_2) = L(r_2)}$. We have already seen that there is NFA $\mathbf{N}$
  s.t $\mathbf{L(N) = L(N_1) \cup L(N_2)}$, hence $\mathbf{L(N) = L(r)}$
- $\mathbf{r = r_1 \bullet r_2}$. Use closure of NFA languages under concatenation
- $\mathbf{r = (r_1)^*}$. Use closure of NFA languages under Kleene star

NFAs and Regular Language

- For every regular expression $\mathbf{r}$ show that there is a NFA $\mathbf{N}$ such that $\mathbf{L(r) = L(N)}$
- Induction on length of $\mathbf{r}$

Inductive cases:

- $\mathbf{r_1, r_2}$ regular expressions and $\mathbf{r = r_1 + r_2}$.
  By induction there are NFAs $\mathbf{N_1, N_2}$ s.t
  $\mathbf{L(N_1) = L(r_1)}$ and $\mathbf{L(N_2) = L(r_2)}$. We have already seen that there is NFA $\mathbf{N}$ s.t $\mathbf{L(N) = L(N_1) \cup L(N_2)}$, hence $\mathbf{L(N) = L(r)}$
- $\mathbf{r = r_1 \bullet r_2}$. Use closure of NFA languages under concatenation
- $\mathbf{r = (r_1)^*}$. Use closure of NFA languages under Kleene star

NFAs and Regular Language

- For every regular expression $\mathbf{r}$ show that there is a NFA $\mathbf{N}$ such that $\mathbf{L(r) = L(N)}$
- Induction on length of $\mathbf{r}$

Inductive cases:

- $\mathbf{r_1, r_2}$ regular expressions and $\mathbf{r = r_1 + r_2}$.
  By induction there are NFAs $\mathbf{N_1, N_2}$ s.t
  $\mathbf{L(N_1) = L(r_1)}$ and $\mathbf{L(N_2) = L(r_2)}$. We have already seen that there is NFA $\mathbf{N}$ s.t $\mathbf{L(N) = L(N_1) \cup L(N_2)}$, hence $\mathbf{L(N) = L(r)}$
- $\mathbf{r = r_1 \bullet r_2}$. Use closure of NFA languages under concatenation
- $\mathbf{r = (r_1)^*}$. Use closure of NFA languages under Kleene star

NFAs and Regular Language

- For every regular expression $\mathbf{r}$ show that there is a NFA $\mathbf{N}$ such that $\mathbf{L(r) = L(N)}$
- Induction on length of $\mathbf{r}$

Inductive cases:

- $\mathbf{r_1, r_2}$ regular expressions and $\mathbf{r = r_1 + r_2}$.
  By induction there are NFAs $\mathbf{N_1, N_2}$ s.t
  $\mathbf{L(N_1) = L(r_1)}$ and $\mathbf{L(N_2) = L(r_2)}$. We have already seen that there is NFA $\mathbf{N}$ s.t $\mathbf{L(N) = L(N_1) \cup L(N_2)}$, hence $\mathbf{L(N) = L(r)}$
- $\mathbf{r = r_1 \bullet r_2}$. Use closure of NFA languages under concatenation
- $\mathbf{r = (r_1)^*}$. Use closure of NFA languages under Kleene star

NFAs and Regular Language

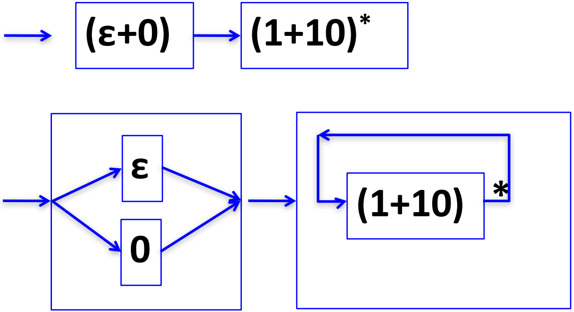
- For every regular expression $\mathbf{r}$ show that there is a NFA $\mathbf{N}$ such that $\mathbf{L(r) = L(N)}$
- Induction on length of $\mathbf{r}$

Inductive cases:

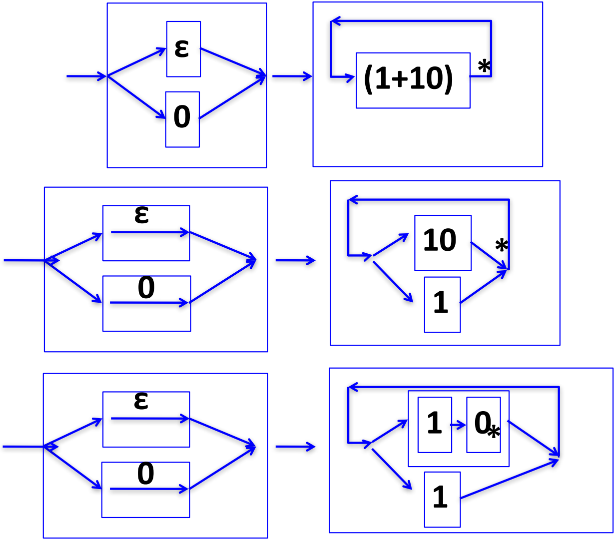
- $\mathbf{r_1, r_2}$ regular expressions and $\mathbf{r = r_1 + r_2}$.
  By induction there are NFAs $\mathbf{N_1, N_2}$ s.t
  $\mathbf{L(N_1) = L(r_1)}$ and $\mathbf{L(N_2) = L(r_2)}$. We have already seen that there is NFA $\mathbf{N}$ s.t $\mathbf{L(N) = L(N_1) \cup L(N_2)}$, hence $\mathbf{L(N) = L(r)}$
- $\mathbf{r = r_1 \bullet r_2}$. Use closure of NFA languages under concatenation
- $\mathbf{r = (r_1)^*}$. Use closure of NFA languages under Kleene star

Example

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



Example



Example

Final NFA simplified slightly to reduce states