Non-deterministic Finite Automata (NFAs)

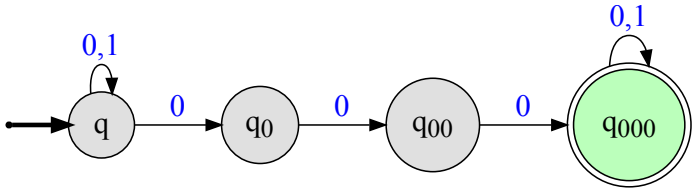
Lecture 4
Thursday, September 3, 2020

LaTeXed: July 22, 2020 21:54

4.1
NFA Introduction

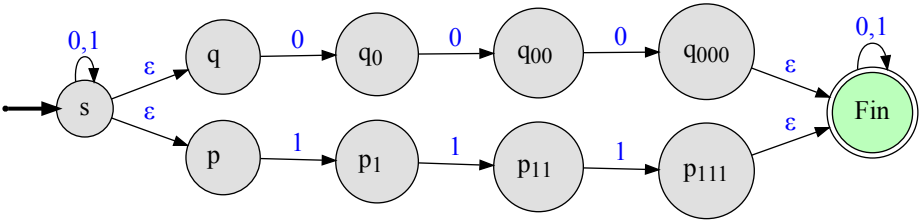
Non-deterministic Finite State Automata by example

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



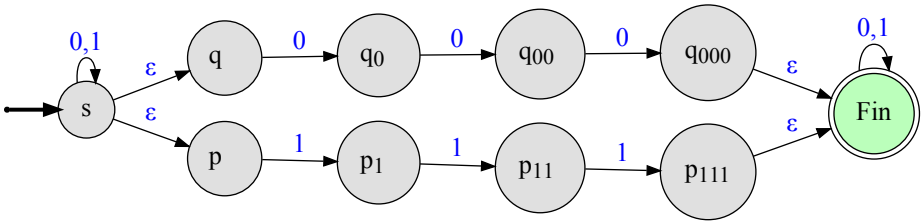
Non-deterministic Finite State Automata by example II

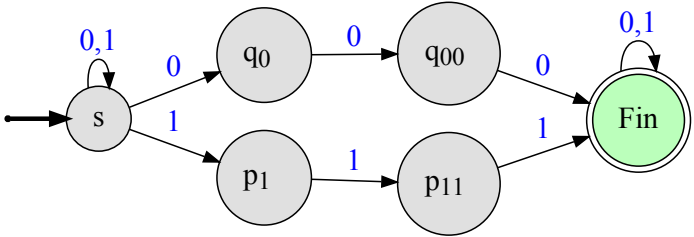
..but only if it is made out of silver.



Non-deterministic Finite State Automata by example II

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

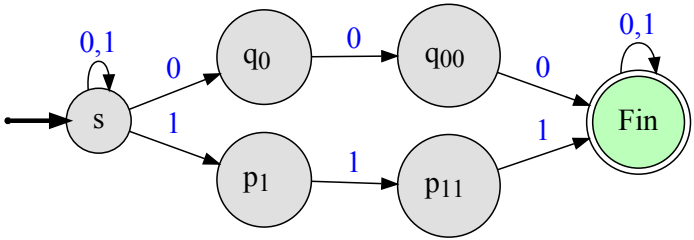




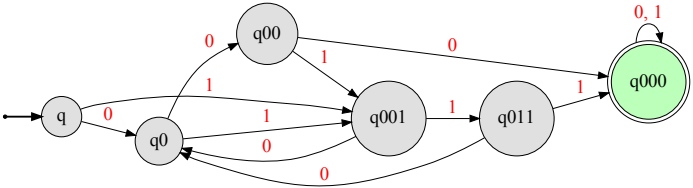
More efficient NFA:

Non-deterministic Finite State Automata by example II

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

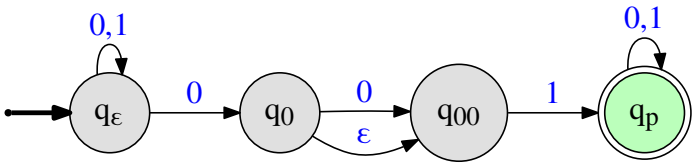


More efficient NFA:



Not the point...
...because DFA can still do it ef-

Non-deterministic Finite State Automata (NFAs)



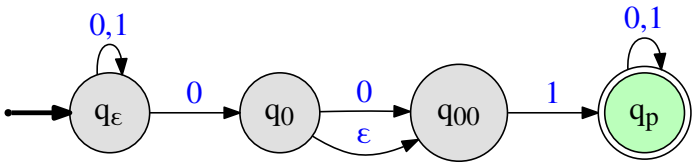
Differences from DFA

- From state q on same letter $a \in \Sigma$ 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)



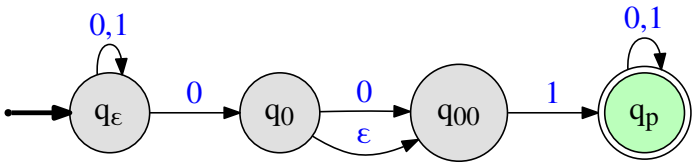
Differences from DFA

- From state q on same letter $a \in \Sigma$ 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)



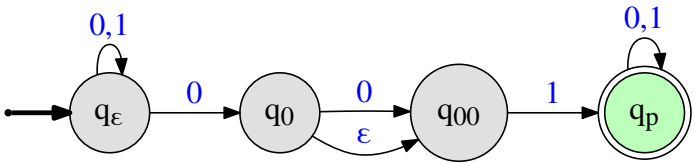
Differences from DFA

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

Questions:

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

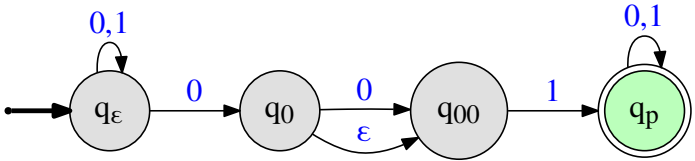
NFA behavior



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

- From q_ε on 1
- From q_ε on 0
- From q_0 on ε
- From q_ε on 01
- From q_{00} on 00

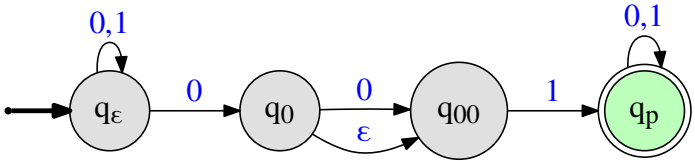
NFA behavior



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

- From q_ε on 1
- From q_ε on 0
- From q_0 on ε
- From q_ε on 01
- From q_{00} on 00

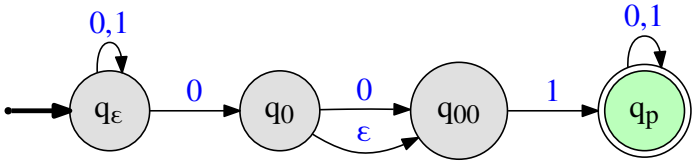
NFA behavior



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

- From q_ε on 1
- From q_ε on 0
- From q_0 on ε
- From q_ε on 01
- From q_{00} on 00

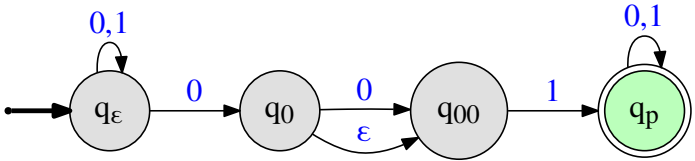
NFA behavior



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

- From q_ε on 1
- From q_ε on 0
- From q_0 on ε
- From q_ε on 01
- From q_{00} on 00

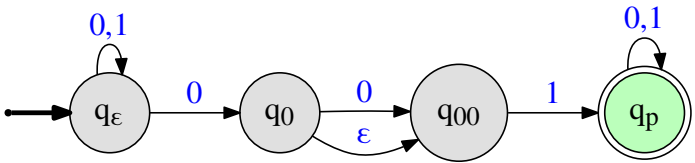
NFA behavior



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

- From q_ε on 1
- From q_ε on 0
- From q_0 on ε
- From q_ε on 01
- From q_{00} on 00

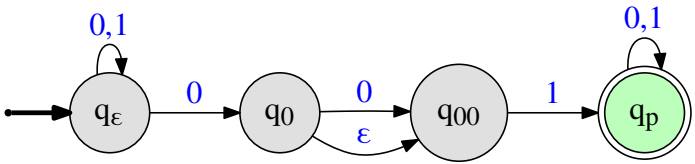
NFA behavior



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

- From q_ε on 1
- From q_ε on 0
- From q_0 on ε
- From q_ε on 01
- From q_{00} on 00

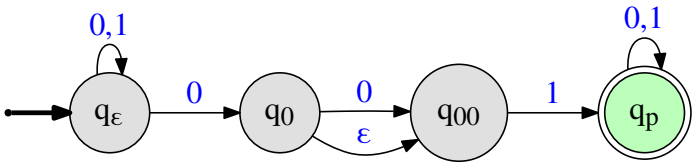
NFA acceptance: informal



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

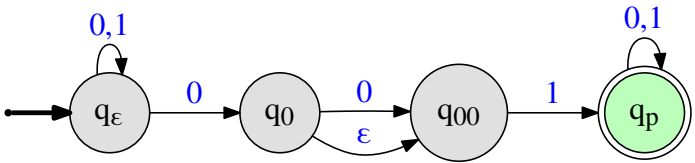
NFA acceptance: informal



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

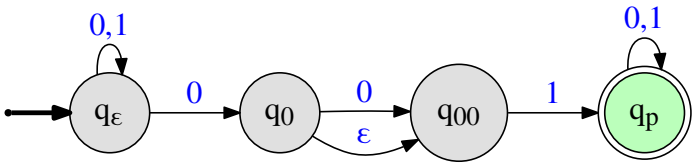
NFA acceptance: example



- Is 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.

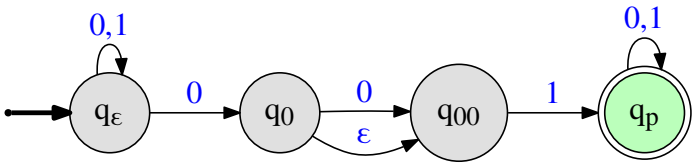
NFA acceptance: example



- 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.

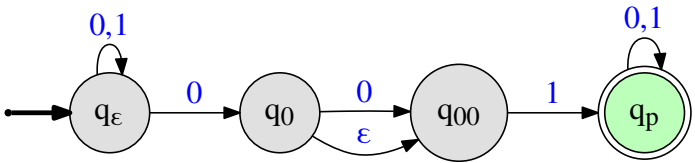
NFA acceptance: example



- 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.

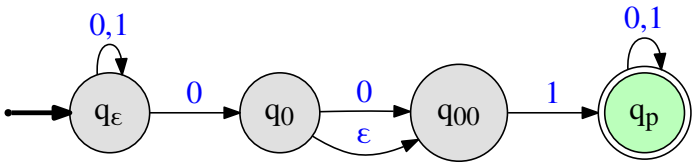
NFA acceptance: example



- 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.

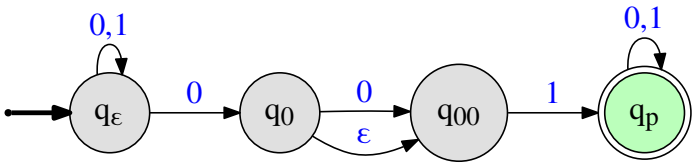
NFA acceptance: example



- 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.

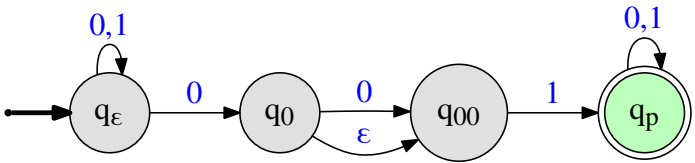
NFA acceptance: example



- 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



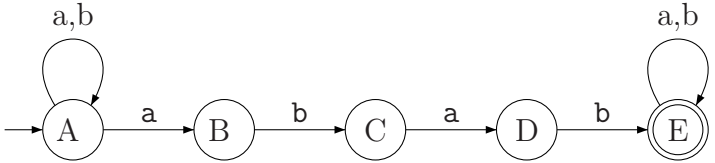
- Is 01 accepted?
- Is 001 accepted?
- Is 100 accepted?
- Are all strings in 1*01 accepted?
- What is the language accepted by ***N***?

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)

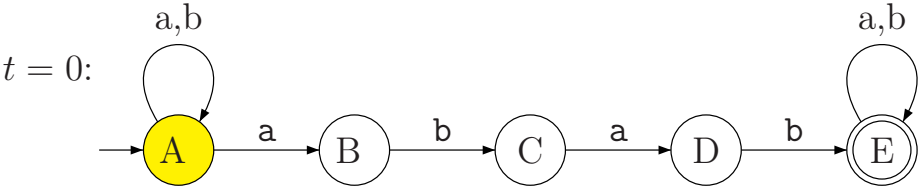


Run it on input ***ababa***.

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

Simulating NFA

Example the first

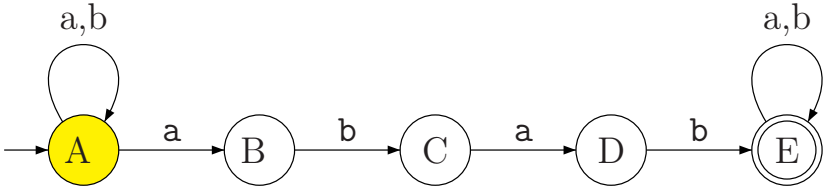


Remaining input: ***ababa***.

Simulating NFA

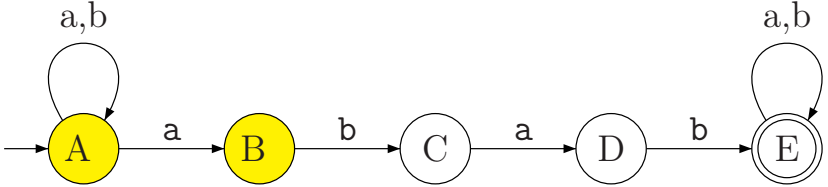
Example the first

$t = 0$:



Remaining input: ***ababa***.

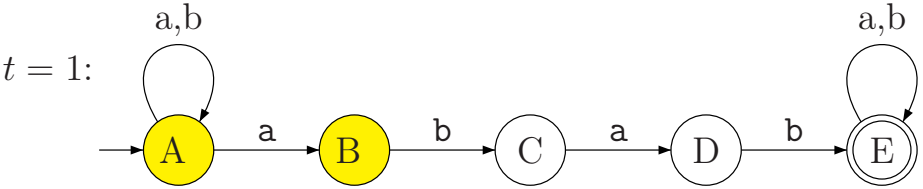
$t = 1$:



Remaining input: ***baba***.

Simulating NFA

Example the first

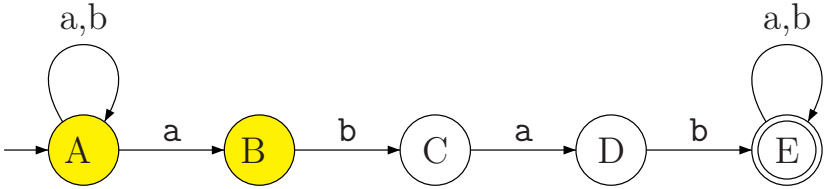


Remaining input: ***baba***.

Simulating NFA

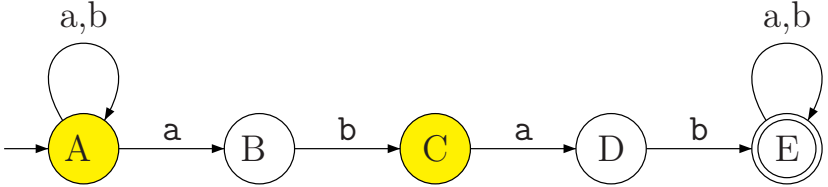
Example the first

$t = 1$:



Remaining input: ***baba***.

$t = 2$:

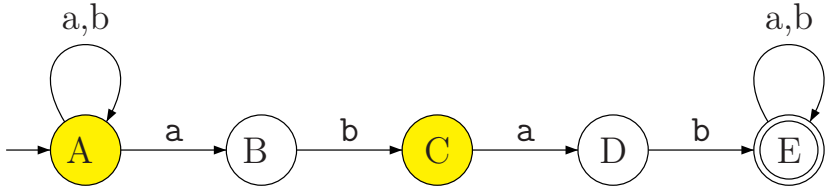


Remaining input: ***aba***.

Simulating NFA

Example the first

$t = 2$:

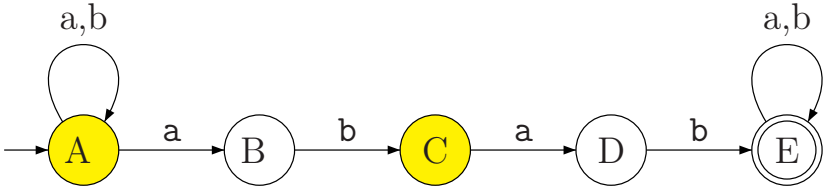


Remaining input: ***aba***.

Simulating NFA

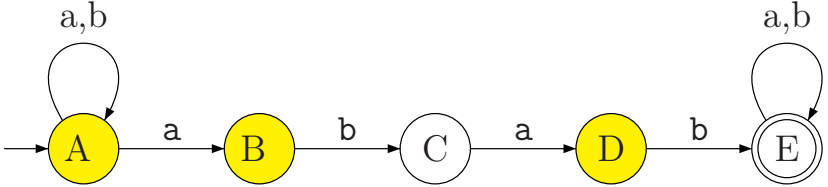
Example the first

$t = 2$:



Remaining input: ***aba***.

$t = 3$:

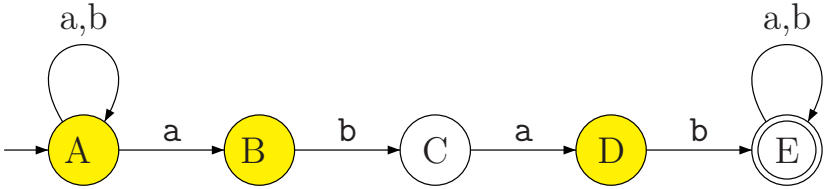


Remaining input: ***ba***.

Simulating NFA

Example the first

$t = 3$:

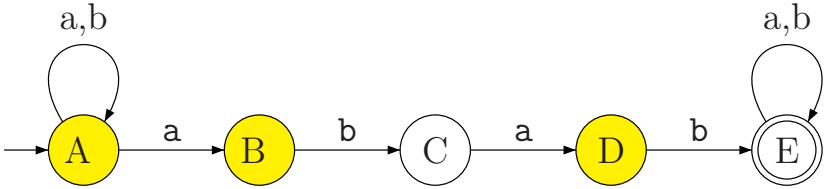


Remaining input: ***ba***.

Simulating NFA

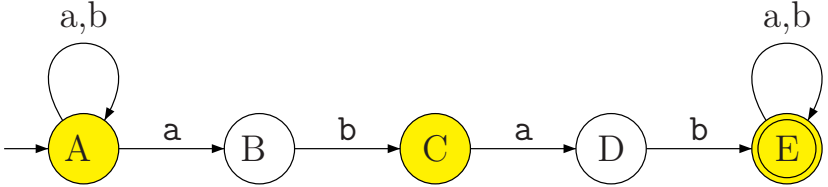
Example the first

$t = 3$:



Remaining input: ***ba***.

$t = 4$:

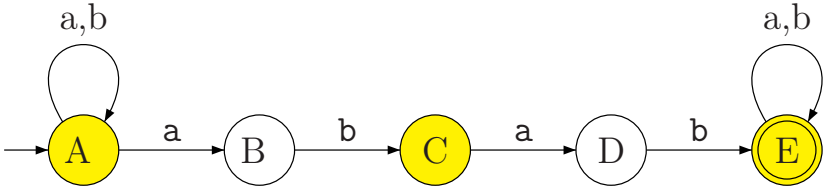


Remaining input: ***a***.

Simulating NFA

Example the first

$t = 4$:

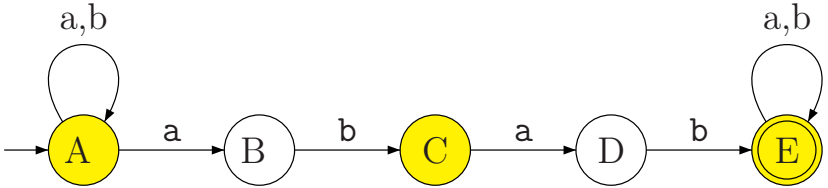


Remaining input: ***a***.

Simulating NFA

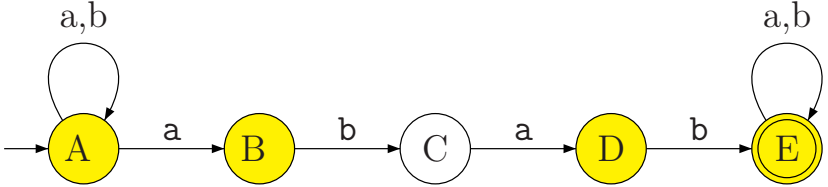
Example the first

$t = 4$:



Remaining input: ***a***.

$t = 5$:

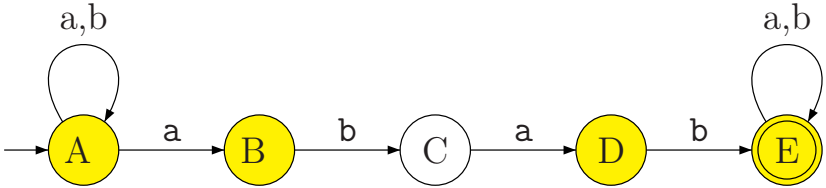


Remaining input: ε.

Simulating NFA

Example the first

$t = 5$:



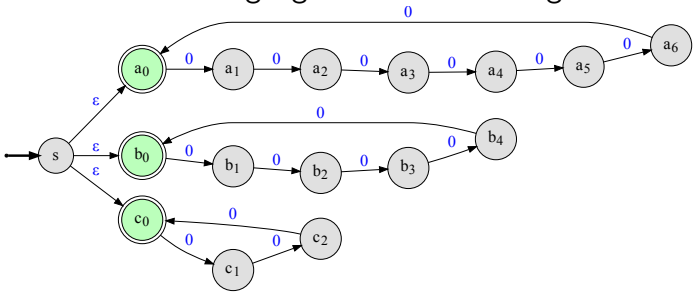
Remaining input: ε.

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?

THE END

...

(for now)