CSE 311 Lecture 25: Relating NFAs, DFAs, and Regular Expressions

Emina Torlak and Sami Davies

Topics

From regular expressions to NFAs

Theorem, algorithm, and examples.

From NFAs to DFAs

Theorem, algorithm, and examples.

From regular expressions to NFAs

Theorem, algorithm, and examples.

NFAs and regular expressions

Theorem

For any set of strings (language) A described by a regular expression, there is an NFA that recognizes A.

NFAs and regular expressions

Theorem

For any set of strings (language) A described by a regular expression, there is an NFA that recognizes A.

How would you prove this theorem?

NFAs and regular expressions

Theorem

For any set of strings (language) A described by a regular expression, there is an NFA that recognizes A.

How would you prove this theorem?

Structural induction on the recursive definition of regular expressions.

NFAs and regular expressions

Theorem

For any set of strings (language) A described by a regular expression, there is an NFA that recognizes A.

How would you prove this theorem?

Structural induction on the recursive definition of regular expressions.

This proof will also give us an algorithm for converting regular expressions to NFAs!

Recall the definition of regular expressions over Σ

Basis step:

$\emptyset$, ε are regular expressions.

a is a regular expression for any $a \in \Sigma$.

Recursive step:

If A and B are regular expressions, then so are AB, $A \cup B$, and A^*.

Recall the definition of regular expressions over Σ

Basis step:

$\emptyset$, ε are regular expressions.

a is a regular expression for any $a \in \Sigma$.

Recursive step:

If A and B are regular expressions, then so are

AB, $A \cup B$, and A^*.

Base cases

We will first show how construct the NFAs that accept the languages for the regular expressions $\emptyset$, ε, and $a \in \Sigma$, respectively.

Inductive step

Then, assuming we have NFAs N_A and N_B for A and B, we'll use them to construct NFAs for AB, $A \cup B$, and A^*.

Regular expressions to NFAs: base cases $\emptyset$, ε, and $a \in \Sigma$

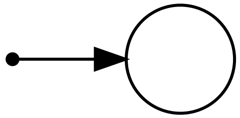
NFA that accepts the language $\emptyset$

NFA that accepts the language $\{\varepsilon\}$

NFA that accepts the language $\{a\}$ for $a \in \Sigma$

Regular expressions to NFAs: base cases $\emptyset$, ε, and $a \in \Sigma$

NFA that accepts the language $\emptyset$

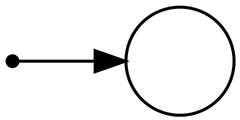


NFA that accepts the language $\{\varepsilon\}$

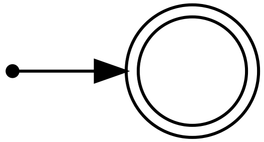
NFA that accepts the language $\{a\}$ for $a \in \Sigma$

Regular expressions to NFAs: base cases $\emptyset$, ε, and $a \in \Sigma$

NFA that accepts the language $\emptyset$



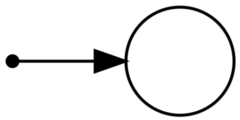
NFA that accepts the language $\{\varepsilon\}$



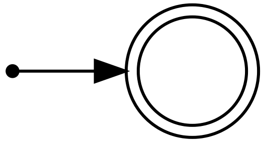
NFA that accepts the language $\{a\}$ for $a \in \Sigma$

Regular expressions to NFAs: base cases $\emptyset$, ε, and $a \in \Sigma$

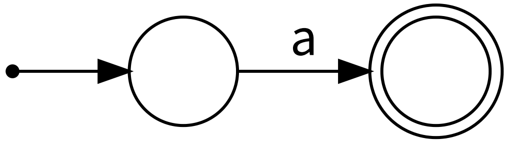
NFA that accepts the language $\emptyset$



NFA that accepts the language $\{\varepsilon\}$



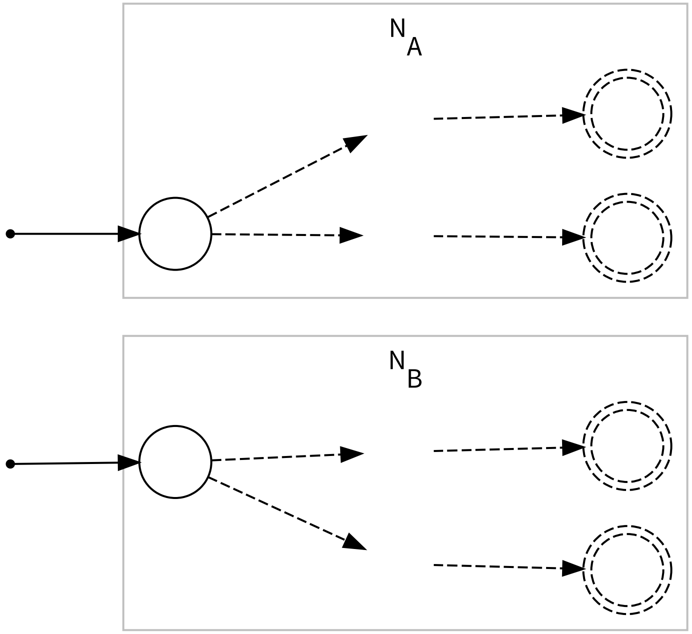
NFA that accepts the language $\{a\}$ for $a \in \Sigma$



Regular expressions to NFAs: inductive step for $A \cup B$

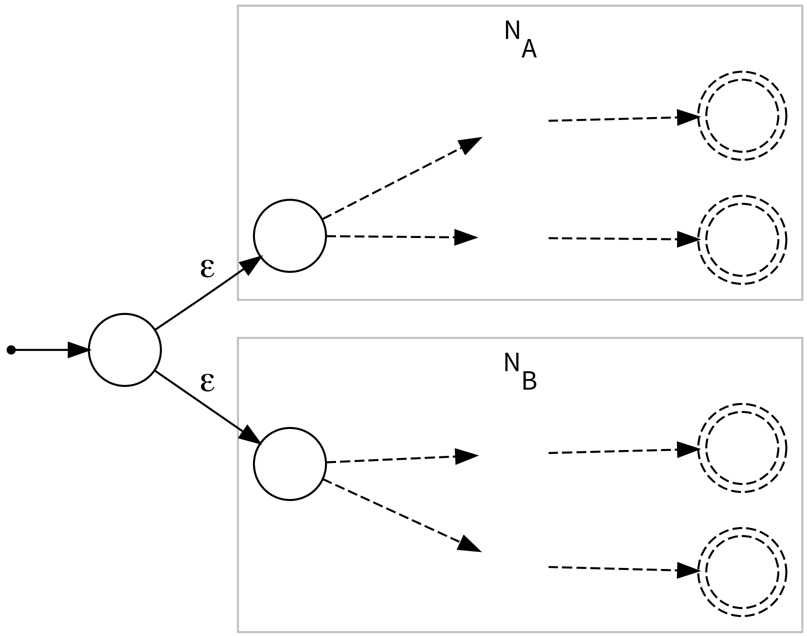
Regular expressions to NFAs: inductive step for $A \cup B$

Suppose N_A and N_B are NFAs for A and B.



Regular expressions to NFAs: inductive step for $A \cup B$

Suppose N_A and N_B are NFAs for A and B.



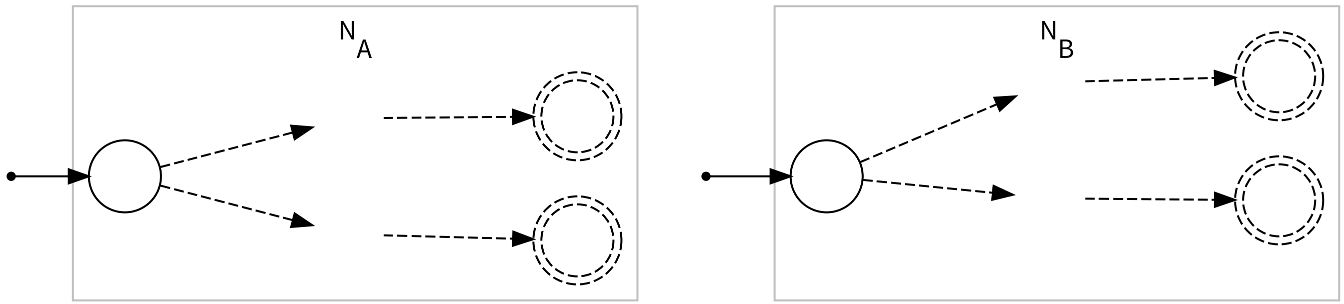
To construct an NFA for $A \cup B$:

Create a new start state.

Add ε edges from the new start state to the old start states of N_A and N_B.

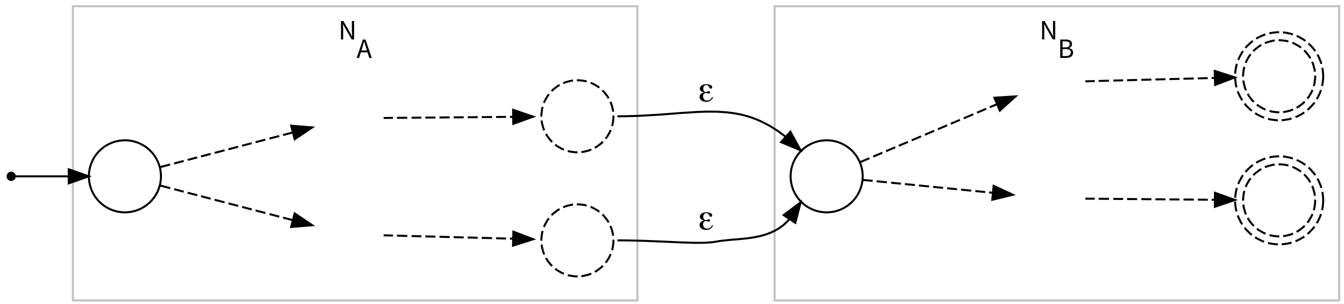
Regular expressions to NFAs: inductive step for AB

Suppose N_A and N_B are NFAs for A and B.



Regular expressions to NFAs: inductive step for AB

Suppose N_A and N_B are NFAs for A and B.



To construct an NFA for AB:

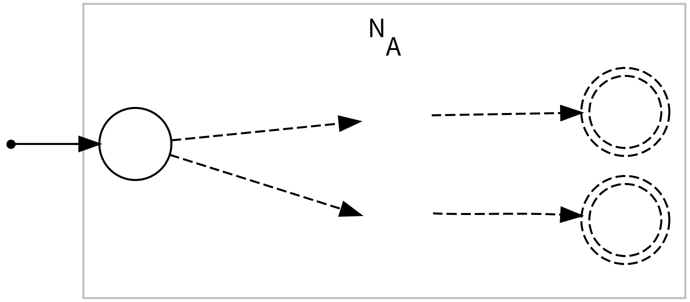
Let the start state of N_A be the start state of the new NFA.

Let the final states of N_B be the final states of the new NFA.

Add an ε edge from every old final state of N_A to the old start state of N_B.

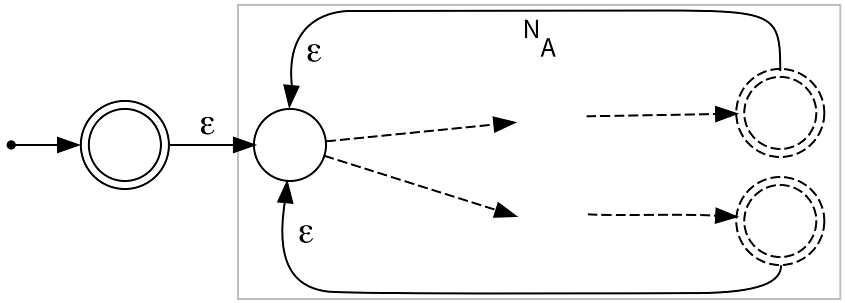
Regular expressions to NFAs: inductive step for A^*

Suppose N_A is an NFA for A.



Regular expressions to NFAs: inductive step for A^*

Suppose N_A is an NFA for A.



To construct an NFA for A^*:

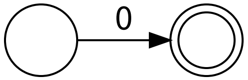
Create a new start state that is a final state.

Add an ε edge from the new start state to the old start state of N_A.

Add an ε edge from every final state of N_A to the old start state.

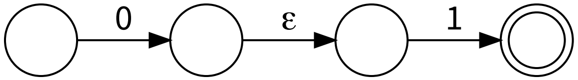
Example: build an NFA for $(01 \cup 1)^*0$

Example: build an NFA for $(01 \cup 1)^*0$

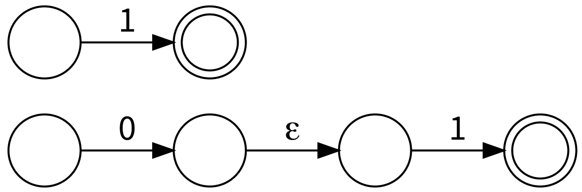


Example: build an NFA for $(01 \cup 1)^*0$

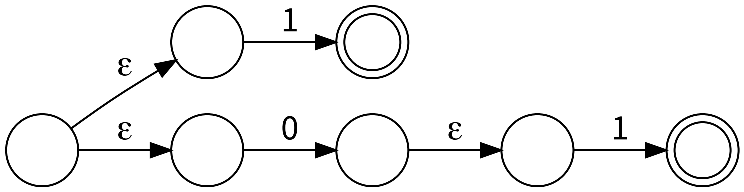
Example: build an NFA for $(01 \cup 1)^*0$



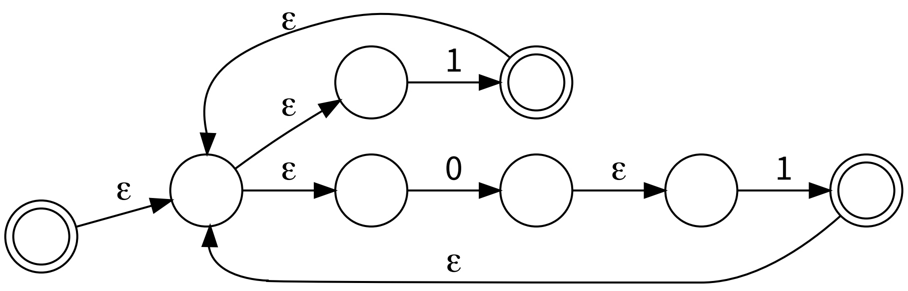
Example: build an NFA for $(01 \cup 1)^*0$



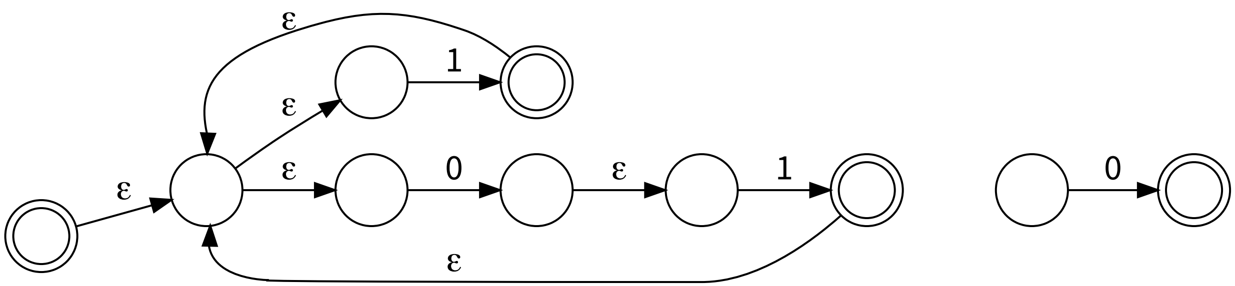
Example: build an NFA for $(01 \cup 1)^*0$



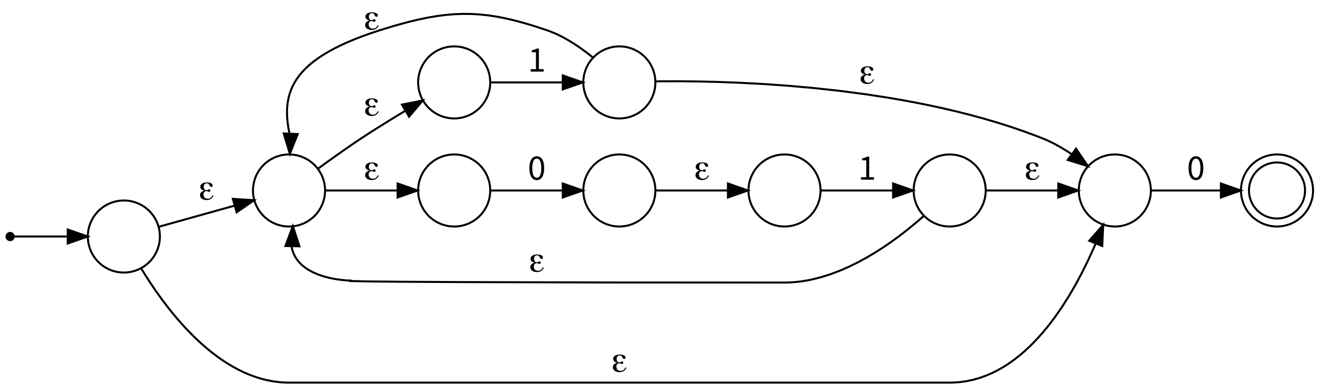
Example: build an NFA for $(01 \cup 1)^*0$



Example: build an NFA for $(01 \cup 1)^*0$



Example: build an NFA for $(01 \cup 1)^*0$



From NFAs to DFAs

Theorem, algorithm, and examples.

NFAs and DFAs

Every DFA is an NFA.

A DFA is an NFA that satisfies more constraints.

NFAs and DFAs

Every DFA is an NFA.

A DFA is an NFA that satisfies more constraints.

Theorem

For every NFA there is a DFA that recognizes exactly the same language.

NFAs and DFAs

Every DFA is an NFA.

A DFA is an NFA that satisfies more constraints.

Theorem

For every NFA there is a DFA that recognizes exactly the same language.

Proof (and algorithm) idea:

The DFA constructed for an NFA keeps track of *all* the states that a prefix of an input string can reach in the NFA.

NFAs and DFAs

Every DFA is an NFA.

A DFA is an NFA that satisfies more constraints.

Theorem

For every NFA there is a DFA that recognizes exactly the same language.

Proof (and algorithm) idea:

The DFA constructed for an NFA keeps track of *all* the states that a prefix of an input string can reach in the NFA.

So there will be one state in the DFA for each *subset* of the states of the NFA that can be reached by some string.

NFAs and DFAs

Every DFA is an NFA.

A DFA is an NFA that satisfies more constraints.

Theorem

For every NFA there is a DFA that recognizes exactly the same language.

Proof (and algorithm) idea:

The DFA constructed for an NFA keeps track of *all* the states that a prefix of an input string can reach in the NFA.

So there will be one state in the DFA for each *subset* of the states of the NFA that can be reached by some string.

We'll see how to construct the start state, remaining states and transitions, and the final states of the DFA.

NFAs to DFAs: the start state

The start state of the DFA represents the following set of states in the NFA:

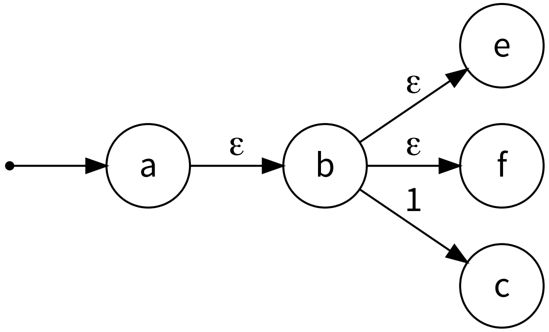
All states reachable from the start state of the NFA using only ε edges.

NFAs to DFAs: the start state

The start state of the DFA represents the following set of states in the NFA:

All states reachable from the start state of the NFA using only ε edges.

NFA

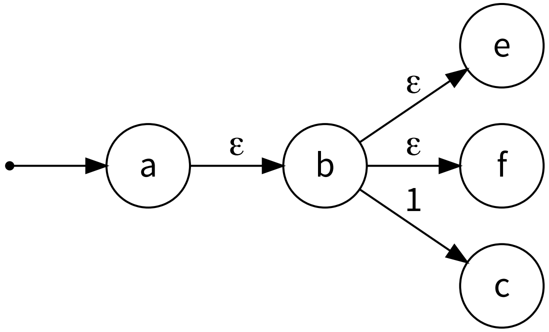


NFAs to DFAs: the start state

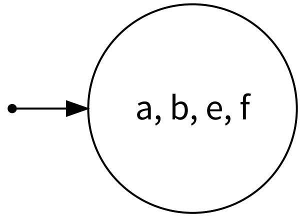
The start state of the DFA represents the following set of states in the NFA:

All states reachable from the start state of the NFA using only ε edges.

NFA



DFA



NFAs to DFAs: states and transitions

Repeat until fixed point:

NFAs to DFAs: states and transitions

Repeat until fixed point:

Let D_Q be a state of the DFA corresponding to a set Q of the NFA states.

NFAs to DFAs: states and transitions

Repeat until fixed point:

Let D_Q be a state of the DFA corresponding to a set Q of the NFA states.

Let $a \in \Sigma$ be a symbol for which D_Q has no outgoing edge.

NFAs to DFAs: states and transitions

Repeat until fixed point:

Let D_Q be a state of the DFA corresponding to a set Q of the NFA states.
Let $a \in \Sigma$ be a symbol for which D_Q has no outgoing edge.
Let T be the (possibly empty) set of NFA states reachable from some state in Q by following one a edge and zero or more ε edges.

NFAs to DFAs: states and transitions

Repeat until fixed point:

Let D_Q be a state of the DFA corresponding to a set Q of the NFA states.
Let $a \in \Sigma$ be a symbol for which D_Q has no outgoing edge.
Let T be the (possibly empty) set of NFA states reachable from some state in Q by following one a edge and zero or more ε edges.
Add a state D_T to the DFA, if not included, that represents the set T.

NFAs to DFAs: states and transitions

Repeat until fixed point:

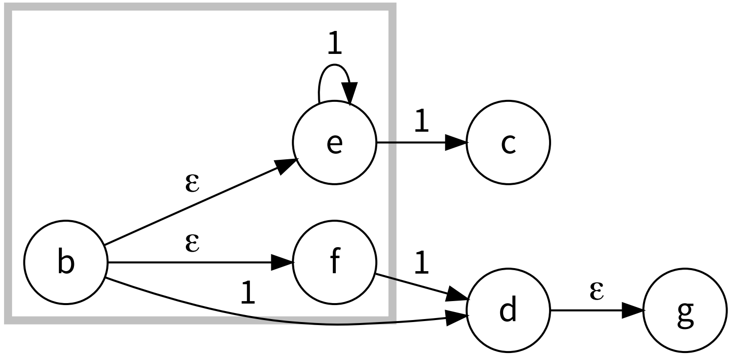
Let D_Q be a state of the DFA corresponding to a set Q of the NFA states.
Let $a \in \Sigma$ be a symbol for which D_Q has no outgoing edge.
Let T be the (possibly empty) set of NFA states reachable from some state in Q by following one a edge and zero or more ε edges.
Add a state D_T to the DFA, if not included, that represents the set T.
Add an edge labeled a from D_Q to D_T.

NFAs to DFAs: states and transitions

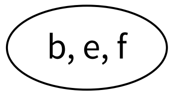
Repeat until fixed point:

Let D_Q be a state of the DFA corresponding to a set Q of the NFA states.
Let $a \in \Sigma$ be a symbol for which D_Q has no outgoing edge.
Let T be the (possibly empty) set of NFA states reachable from some state in Q by following one a edge and zero or more ε edges.
Add a state D_T to the DFA, if not included, that represents the set T.
Add an edge labeled a from D_Q to D_T.

NFA



DFA

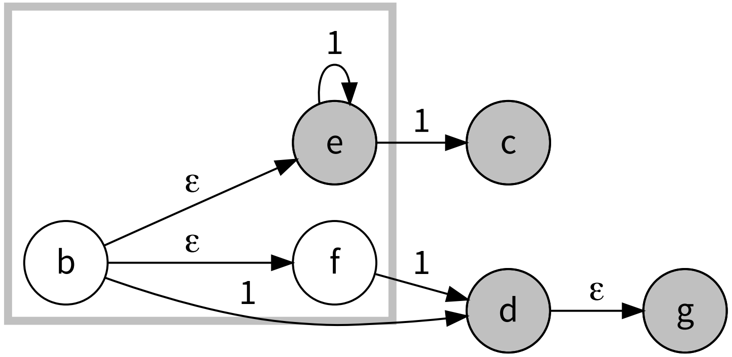


NFAs to DFAs: states and transitions

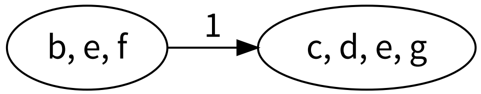
Repeat until fixed point:

Let D_Q be a state of the DFA corresponding to a set Q of the NFA states.
Let $a \in \Sigma$ be a symbol for which D_Q has no outgoing edge.
Let T be the (possibly empty) set of NFA states reachable from some state in Q by following one a edge and zero or more ε edges.
Add a state D_T to the DFA, if not included, that represents the set T.
Add an edge labeled a from D_Q to D_T.

NFA



DFA

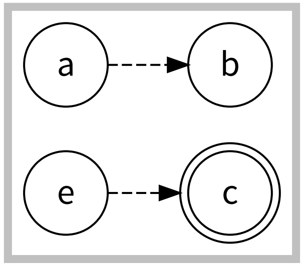


NFAs to DFAs: final states

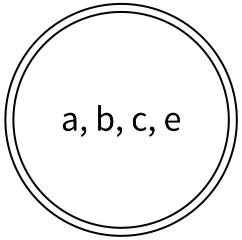
The final states of the DFA:

Every DFA state that represents a set of NFA states containing a final state.

NFA



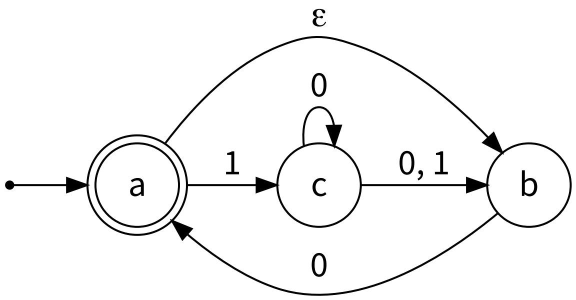
DFA



Example: NFA to DFA conversion

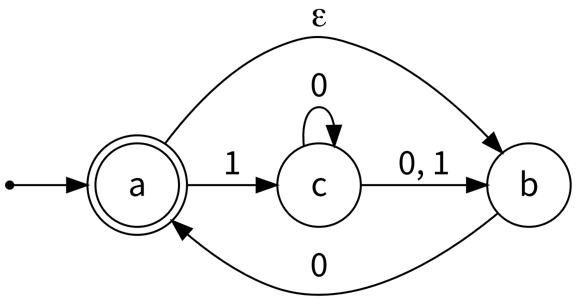
DFA

NFA

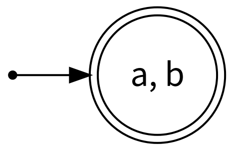


Example: NFA to DFA conversion

NFA



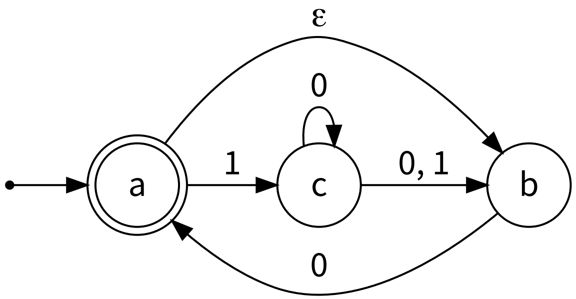
DFA

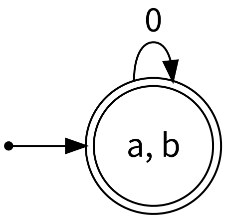


Example: NFA to DFA conversion

DFA

NFA

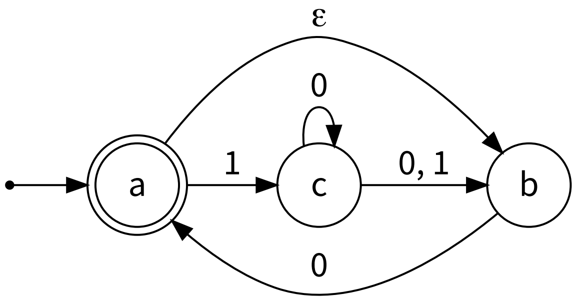


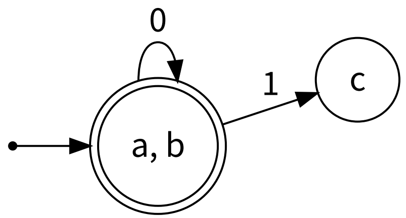


Example: NFA to DFA conversion

DFA

NFA

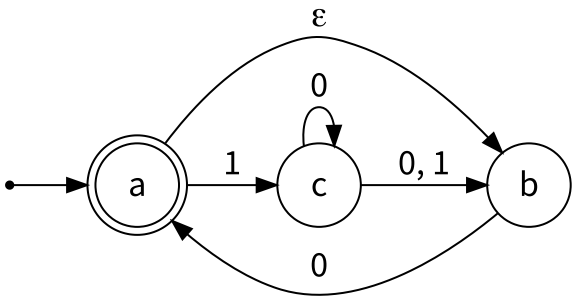


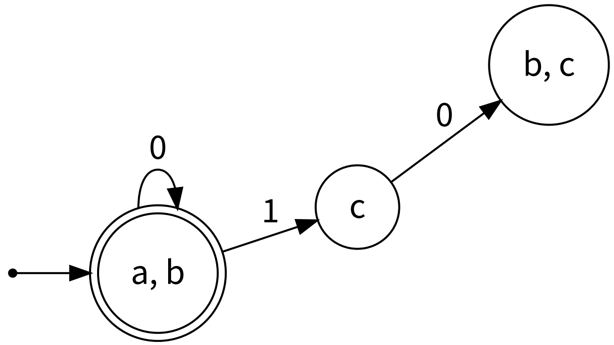


Example: NFA to DFA conversion

DFA

NFA

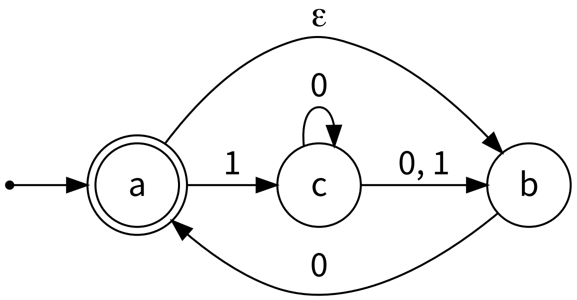


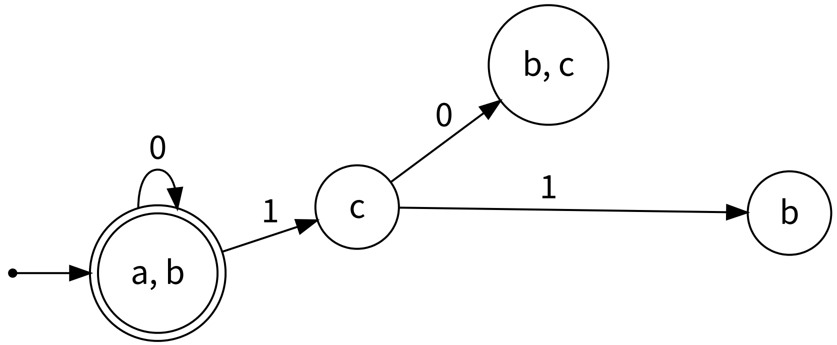


Example: NFA to DFA conversion

DFA

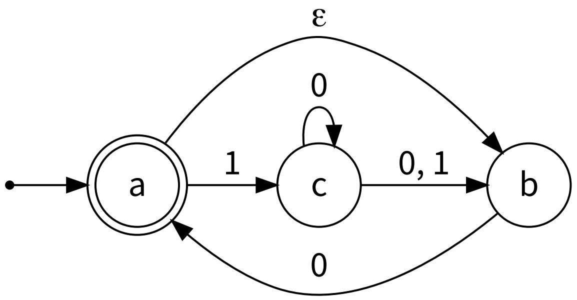
NFA



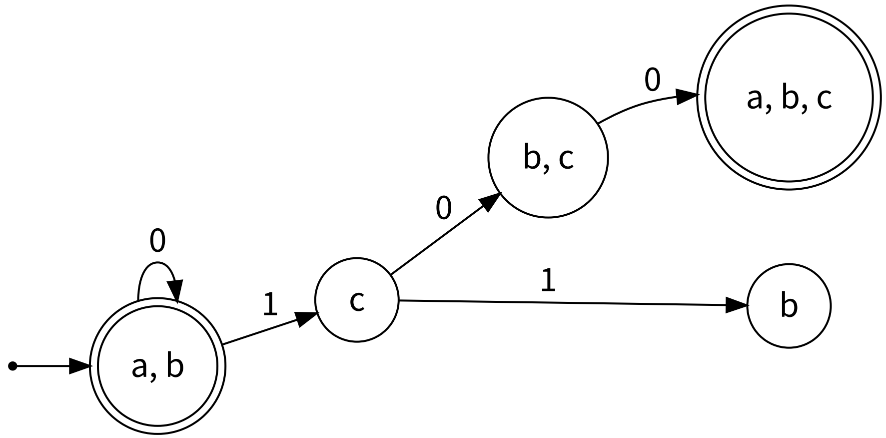


Example: NFA to DFA conversion

NFA



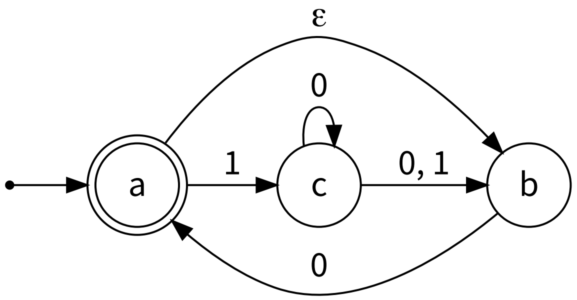
DFA

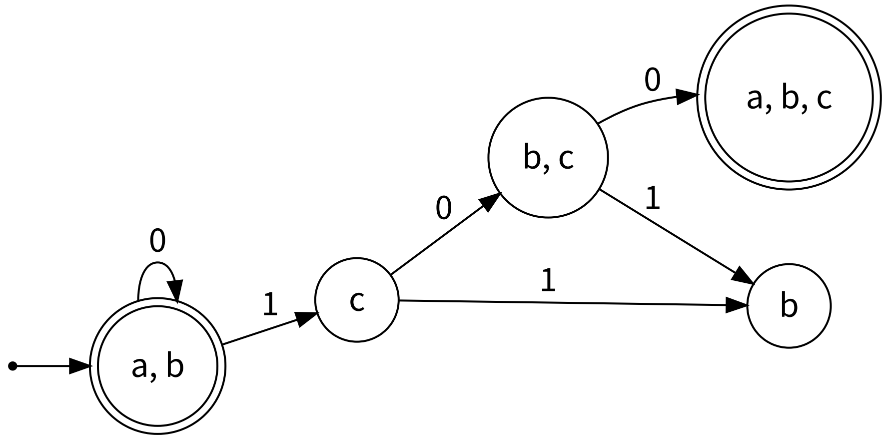


Example: NFA to DFA conversion

DFA

NFA

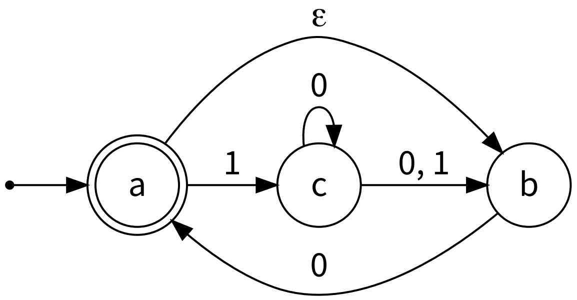


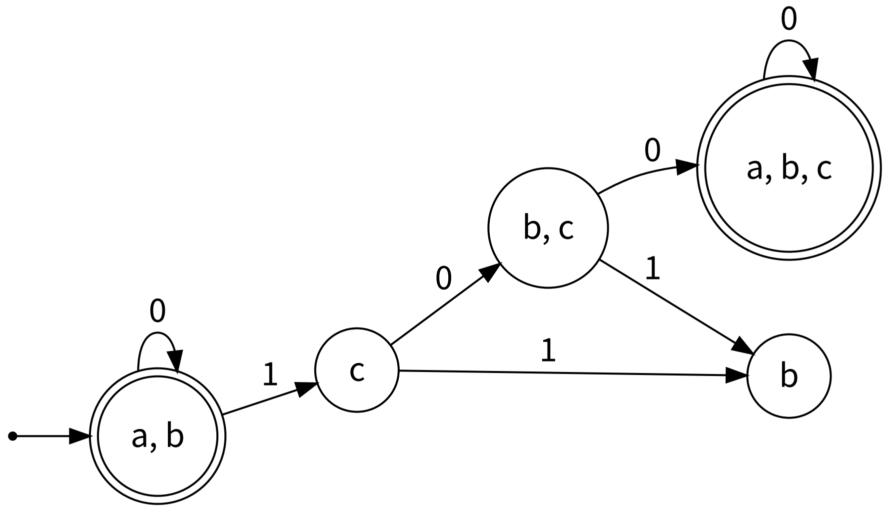


Example: NFA to DFA conversion

DFA

NFA

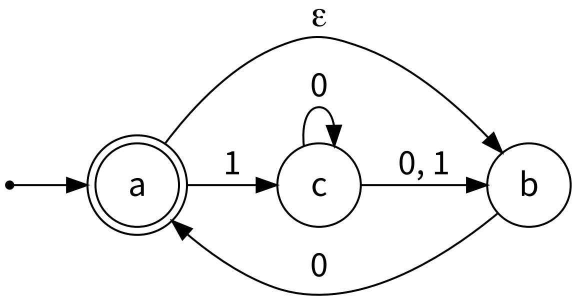


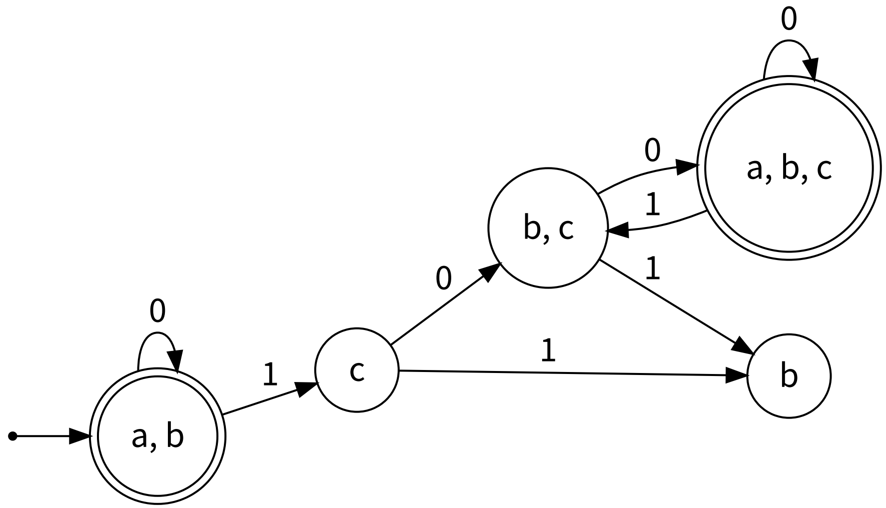


Example: NFA to DFA conversion

DFA

NFA

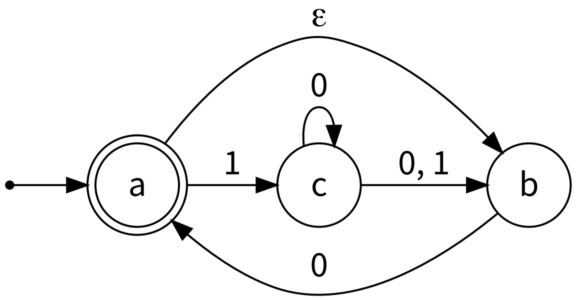


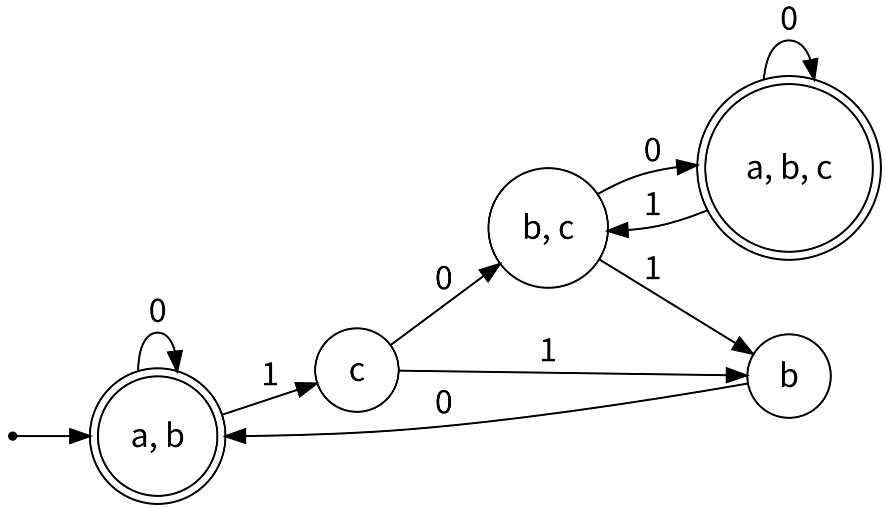


Example: NFA to DFA conversion

DFA

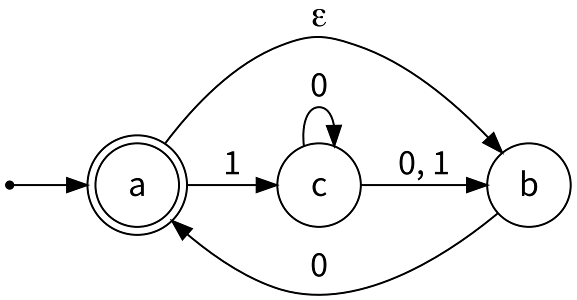
NFA



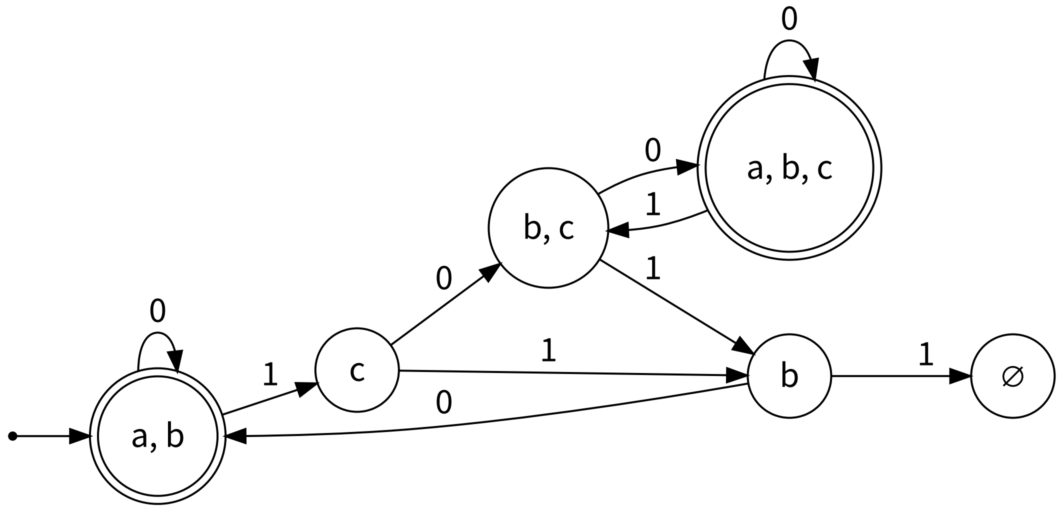


Example: NFA to DFA conversion

NFA



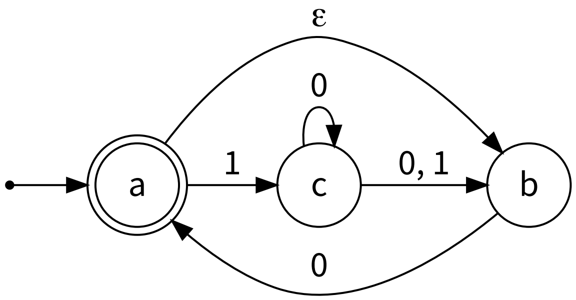
DFA

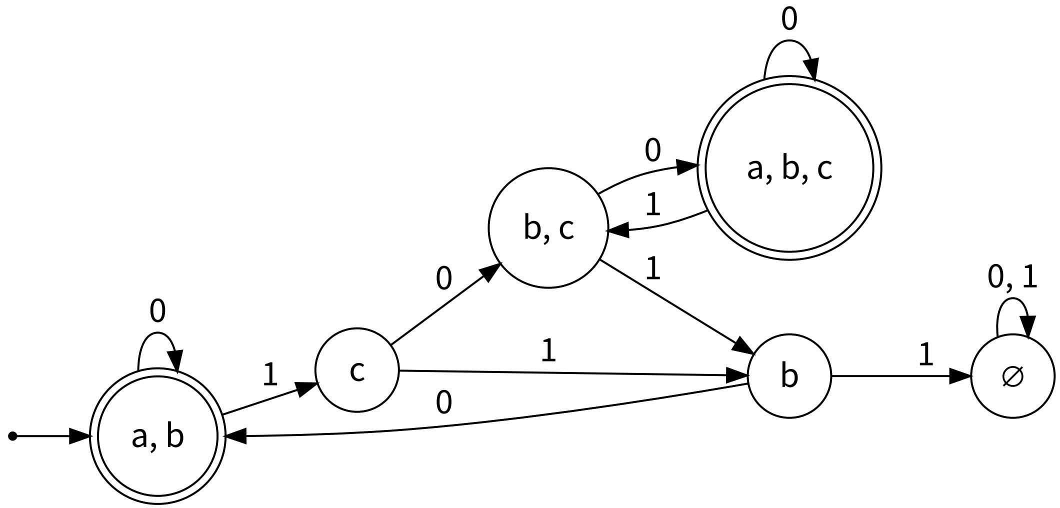


Example: NFA to DFA conversion

DFA

NFA





Exponential blow-up in simulating nondeterminism

In general the DFA might need a state for every subset of states of the NFA.

- Power set of the set of states of the NFA.
- n -state NFA yields DFA with up to 2^n states.
- We saw an example of this worst case outcome.

Exponential blow-up in simulating nondeterminism

In general the DFA might need a state for every subset of states of the NFA.

Power set of the set of states of the NFA.

n -state NFA yields DFA with up to 2^n states.

We saw an example of this worst case outcome.

The famous “P=NP?” question asks whether a similar blow-up is always necessary to get rid of nondeterminism for polynomial-time algorithms.

DFAs $\equiv$ NFAs $\equiv$ regular expressions

Equivalence of DFAs, NFAs, and regular expressions

We have shown how to build an optimal DFA for every regular expression.

Build an NFA.

Convert the NFA to a DFA using the subset construction.

Minimize the resulting DFA.

Equivalence of DFAs, NFAs, and regular expressions

We have shown how to build an optimal DFA for every regular expression.

Build an NFA.

Convert the NFA to a DFA using the subset construction.

Minimize the resulting DFA.

Theorem

A language is recognized by a DFA (or NFA) if and only if it has a regular expression.

You need to know this fact but we won't ask you anything about the "only if" direction from DFAs/NFAs to regular expressions.

Summary

Every regular expression has a corresponding NFA.

Constructed using the algorithm shown in this lecture.

Every NFA has a corresponding DFA.

Constructed using the algorithm shown in this lecture.

Worst case outcome: exponential blowup in the number of states!

DFAs $\equiv$ NFAs $\equiv$ regular expressions.

We've shown how to go from a regular expression to an NFA to a DFA.

But we won't show how to go from an NFA/DFA to a regular expression.