# CSE 311: Foundations of Computing

## Lecture 24: NFAs, Regular expressions, and NFA→DFA

# Administrivia

- No lecture Wednesday
  - since some of you may be traveling
- Will have a reading for Wednesday
  - since you still want to learn something
  - posted on the web site

# HW8

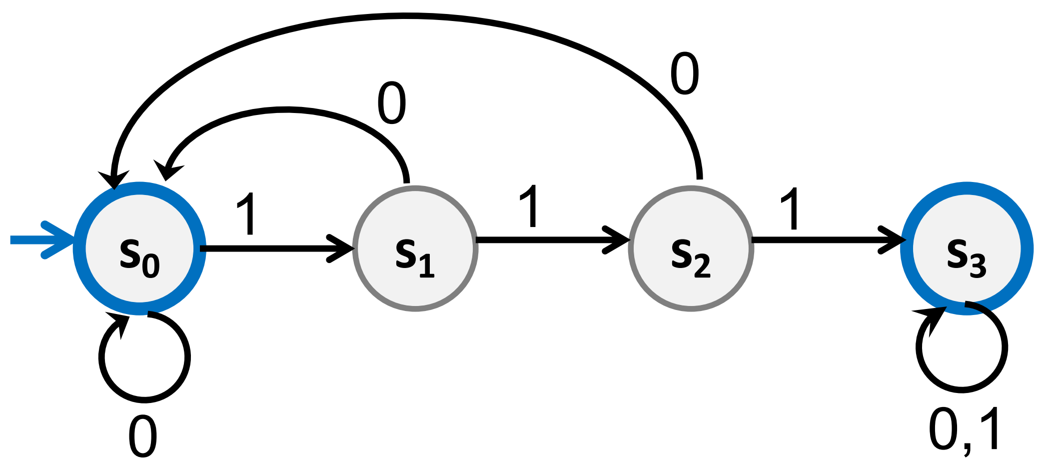
- Due next Wednesday
  - so that we can hand out solutions on Friday

- Please do not wait until next Monday to start

- Material will be heavily featured on the final
  - two examples for each algorithm from last two lectures
  - that plus 1-2 from practice material should be enough

# Recall: Deterministic Finite Automata (DFA)

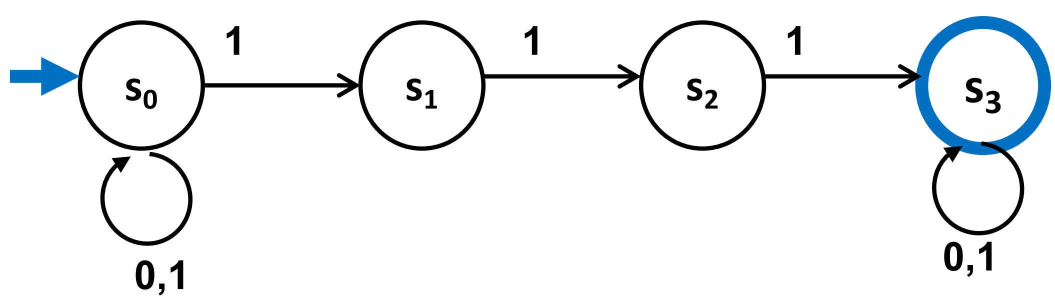
- Def: x is in the language recognized by an DFA if and only if x labels a path from the start state to some final state



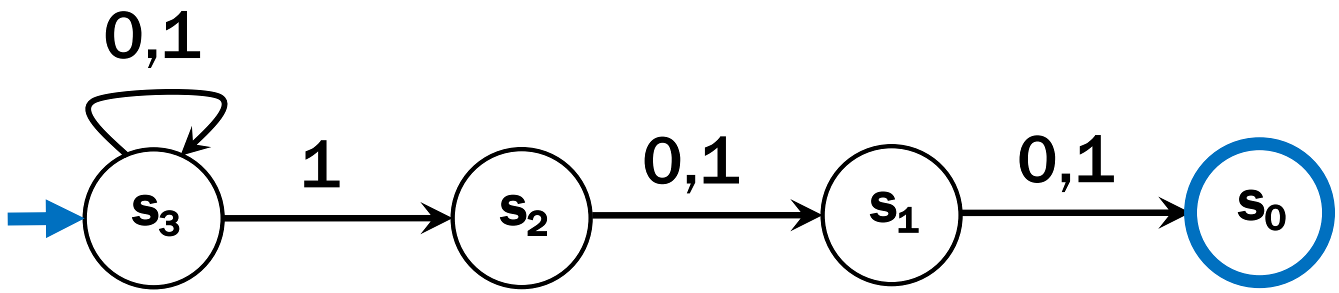

- Path $v_0$, $v_1$, …, $v_n$ with $v_0 = s_0$ and label x describes a correct simulation of the DFA on input x
  - i-th step must match the i-th character of x

# Last time: Nondeterministic Finite Automata (NFA)

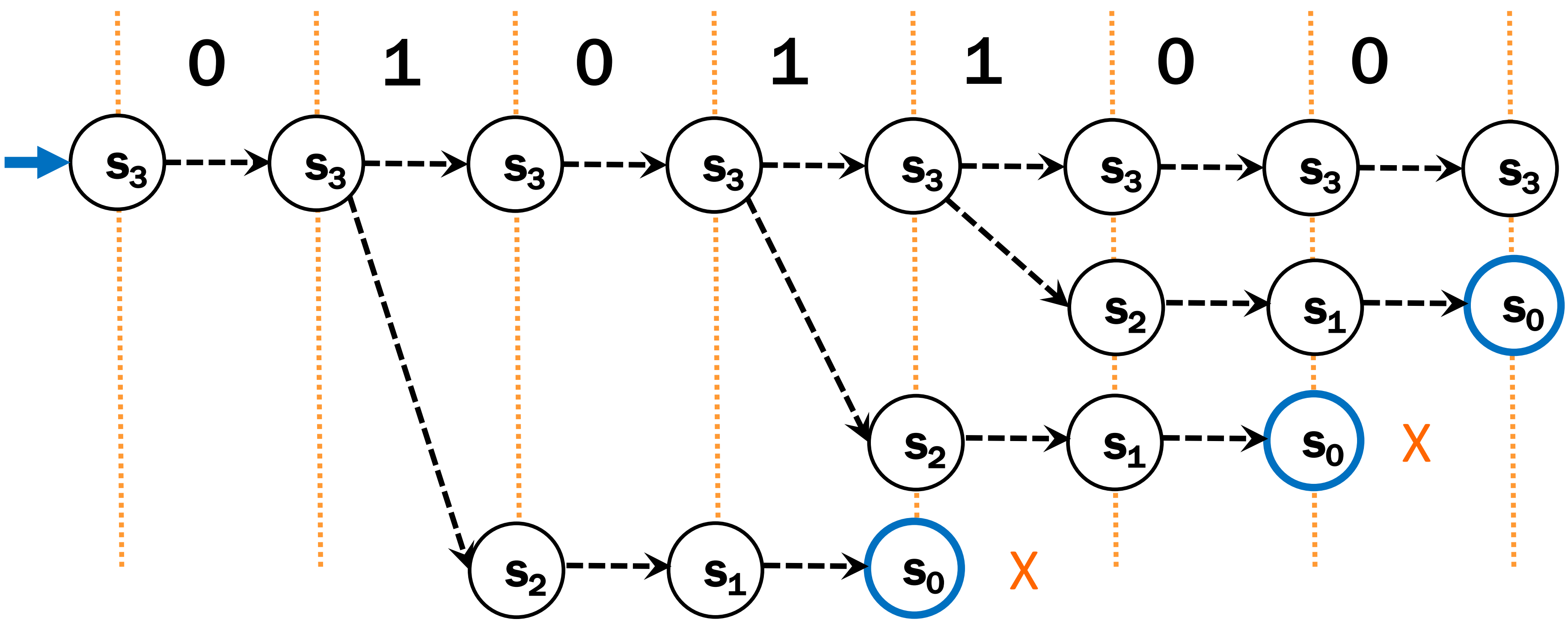
- Graph with start state, final states, edges labeled by symbols (like DFA) but
  - Not required to have exactly 1 edge out of each state labeled by each symbol— can have 0 or >1
  - Also can have edges labeled by empty string ε
- **Def:** x is in the language recognized by an NFA if and only if x labels a path from the start state to some final state



# Last time: Parallel Exploration view of an NFA



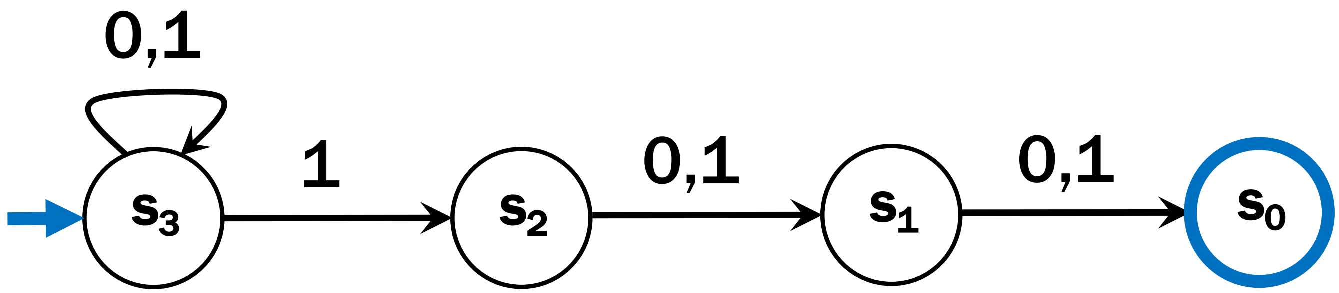

Input string 0101100

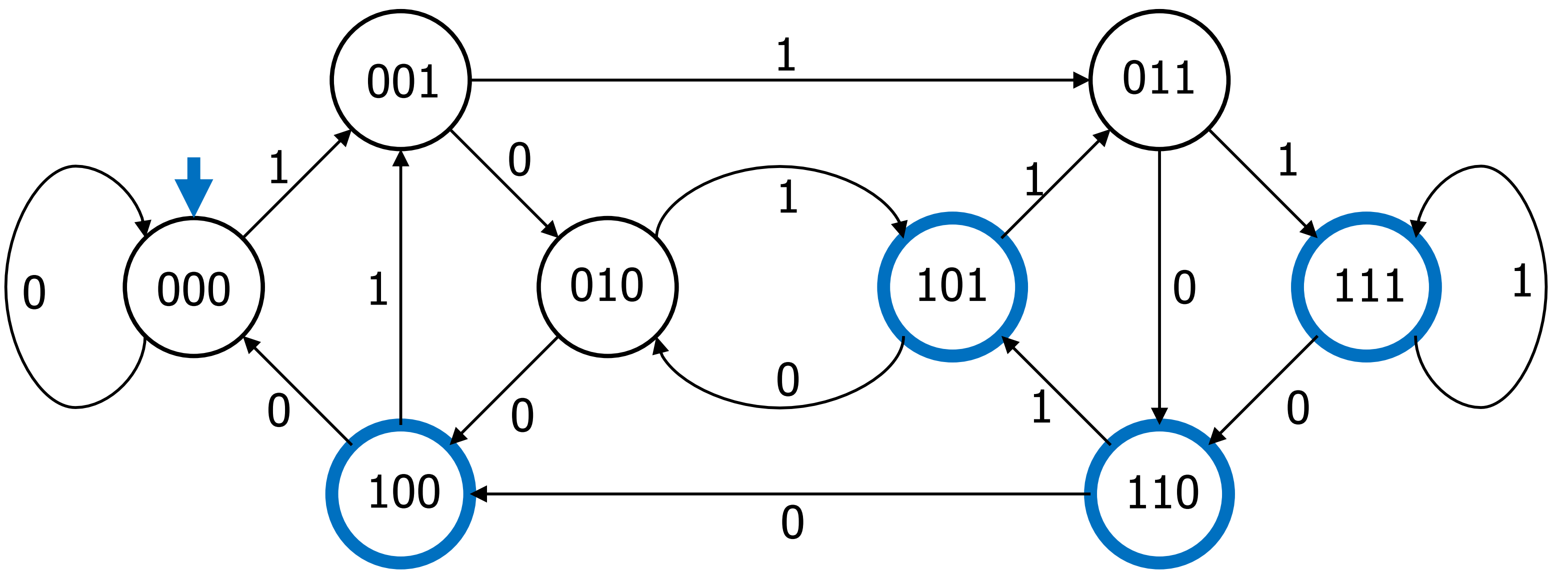


# Last time: Three ways of thinking about NFAs

- Outside observer: Is there a path labeled by x from the start state to some final state?
- Parallel exploration: The NFA computation runs all possible computations on x step-by-step at the same time in parallel
- Perfect guesser: The NFA has input x and whenever there is a choice of what to do it magically guesses a good one (if one exists)

# Last time: Compare with the smallest DFA



# The story so far...

REs

CFGs

DFAs

NFAs

## The story so far...

REs $\subseteq$ CFGs

DFAs $\subseteq$ NFAs

# NFAs and regular expressions

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

Proof idea: Structural induction based on the recursive definition of regular expressions...

## Regular Expressions over $\Sigma$

- **Basis:**
  - $\varepsilon$ is a regular expression
  - ***a*** is a regular expression for any $a \in \Sigma$
- **Recursive step:**
  - If **A** and **B** are regular expressions then so are:

    **A $\cup$ B**

    **AB**

    **A***

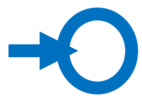
# Base Case

- Case ε:

- Case *a*:

## Base Case

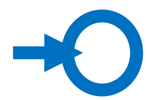
- **Case ε:**



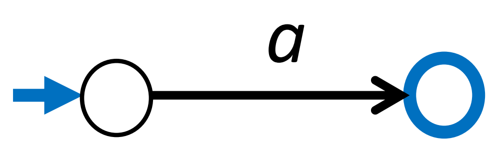
- **Case *a*:**

## Base Case

- **Case ε:**

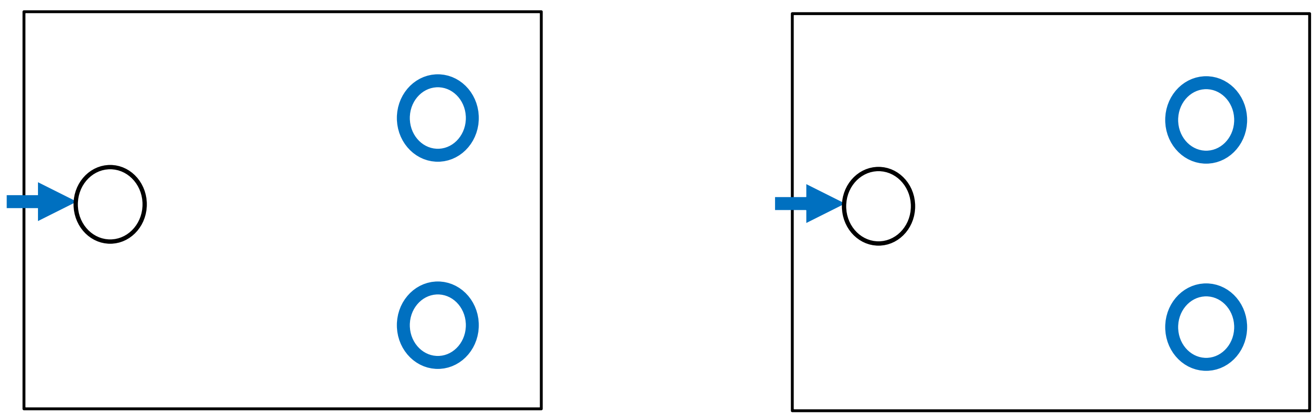


- **Case *a*:**



# Inductive Hypothesis

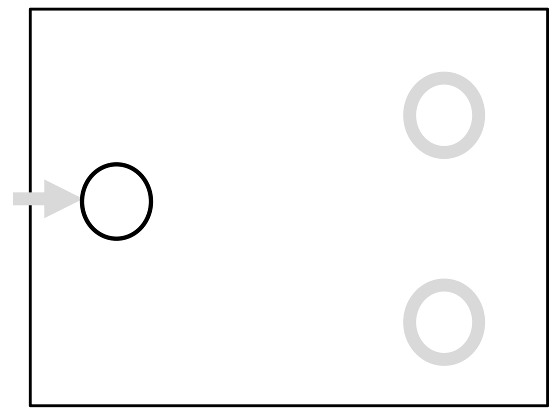
- Suppose that for some regular expressions A and B there exist NFAs $N_A$ and $N_B$ such that $N_A$ recognizes the language given by A and $N_B$ recognizes the language given by B



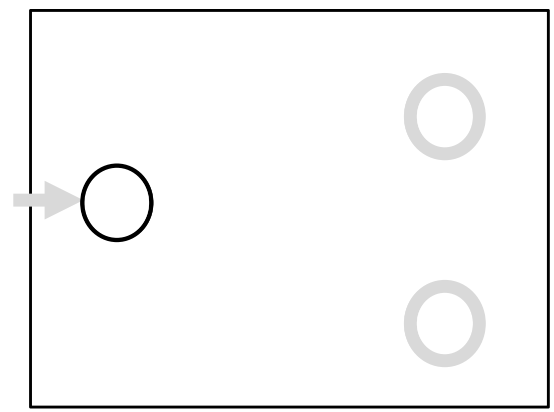
$N_A$ $N_B$

# Inductive Step

**Case A ∪ B:**



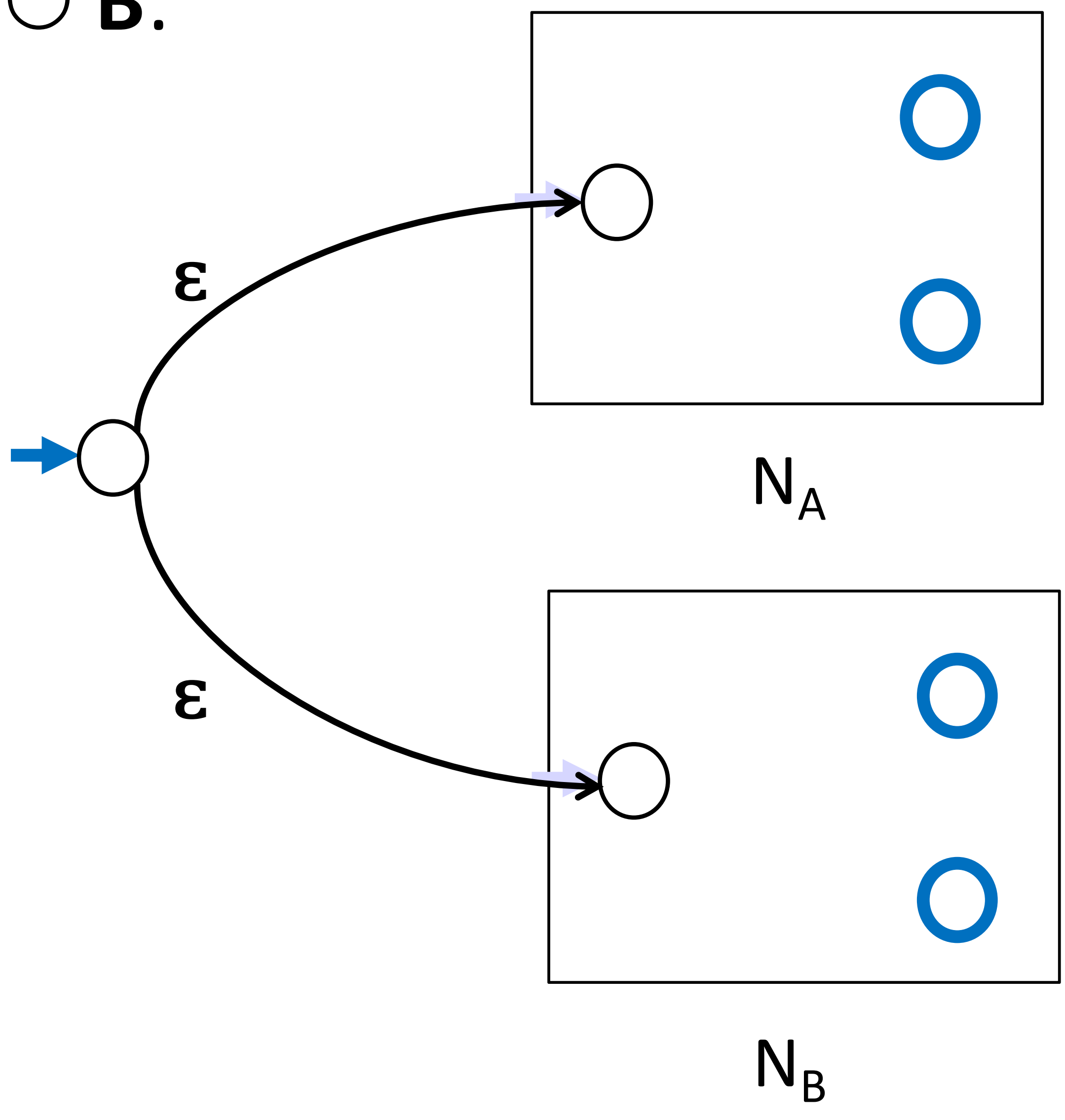
$N_A$



$N_B$

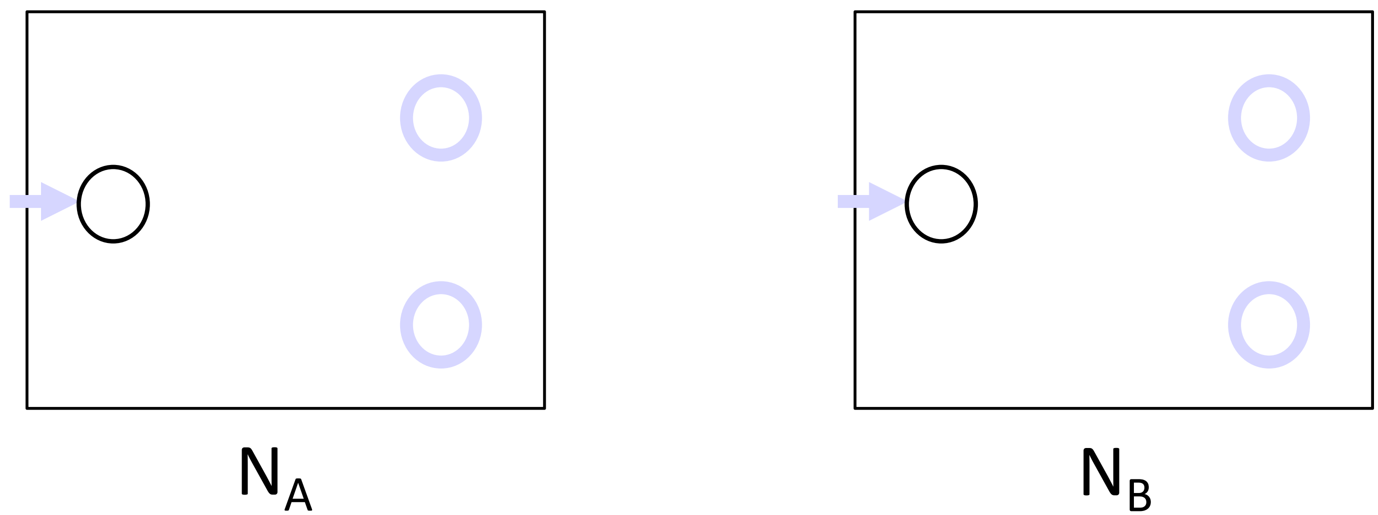
# Inductive Step

**Case A ∪ B:**



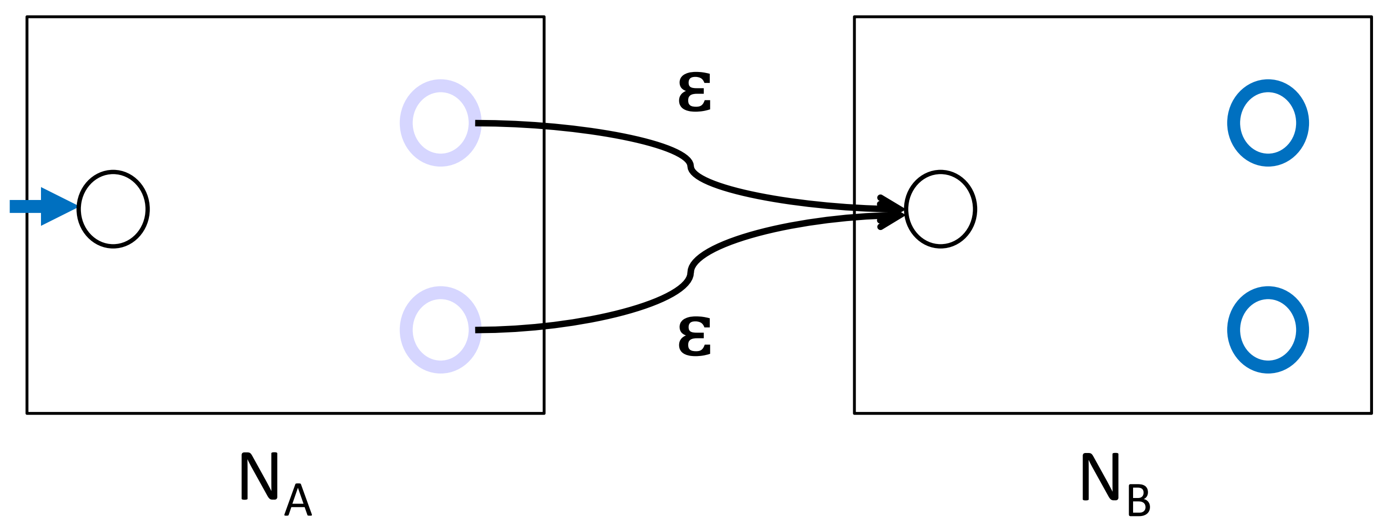
# Inductive Step

## Case AB:



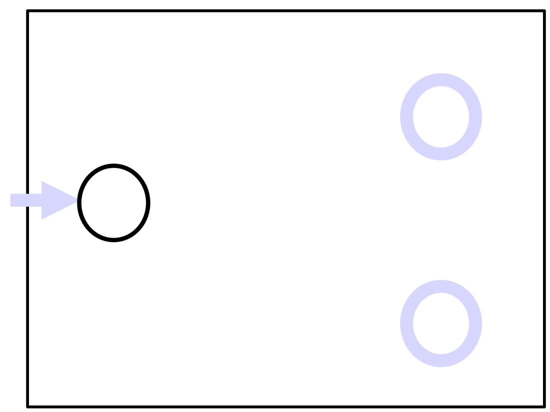
# Inductive Step

## Case AB:



# Inductive Step

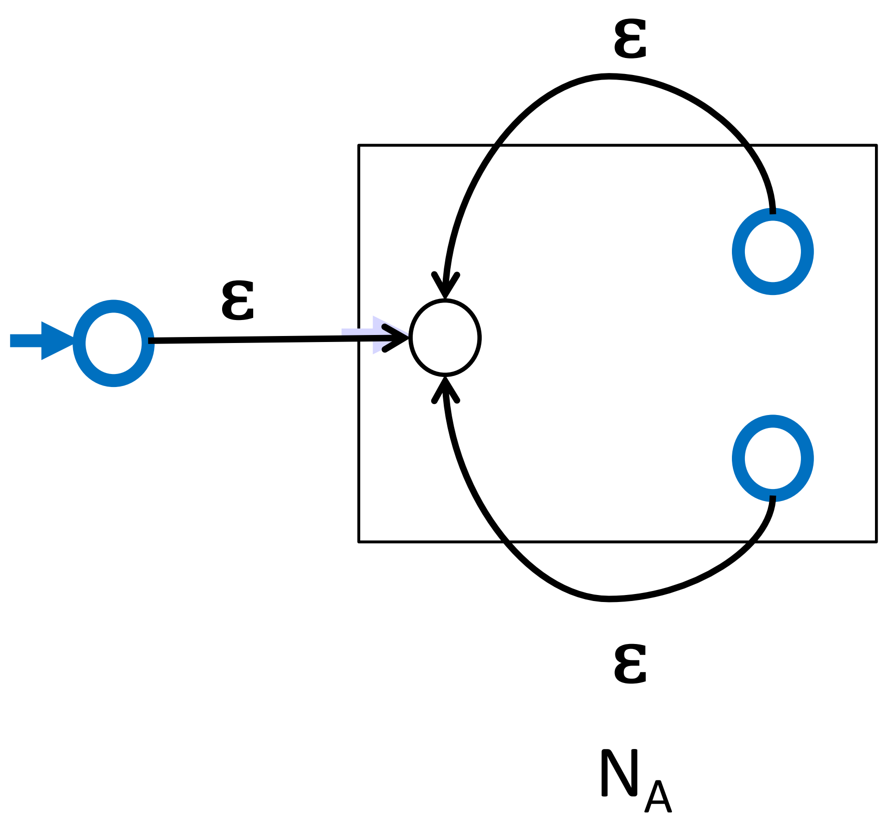
## Case A*



$N_A$

# Inductive Step

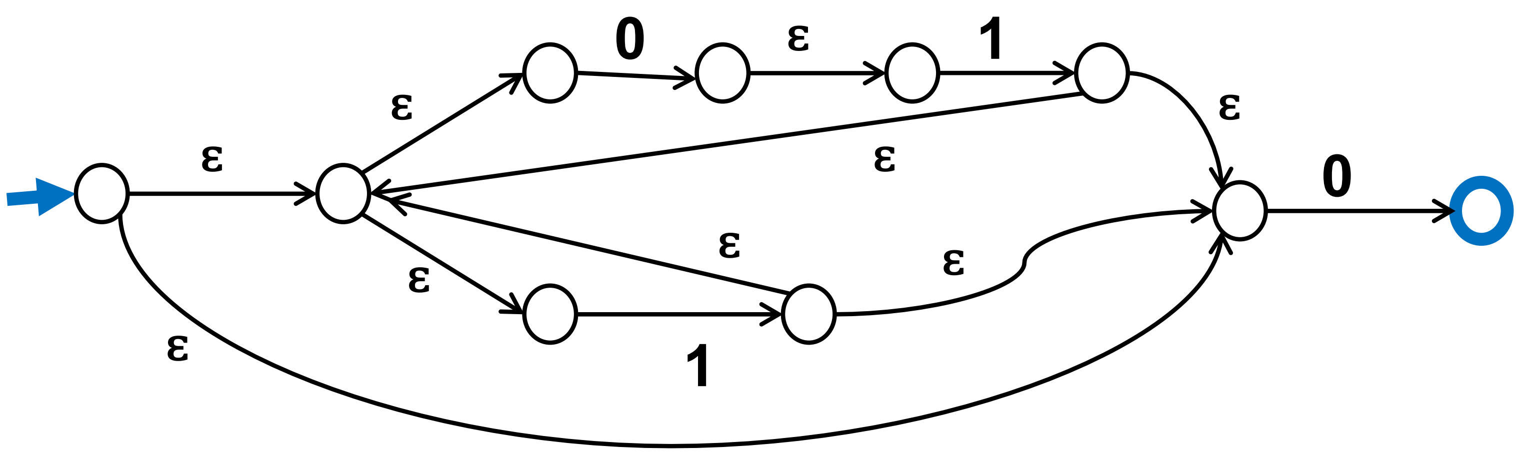
## Case A*



# Build an NFA for $(01 \cup 1)^*0$

# Solution

**(01 ∪1)*0**



# The story so far...

$$\begin{array}{ccc} \boxed{\text{REs}} & \subseteq & \boxed{\text{CFGs}} \\ & \subseteq & \\ \boxed{\text{DFAs}} & \subseteq & \boxed{\text{NFAs}} \end{array}$$

# NFAs and DFAs

Every DFA **is** an NFA

- DFAs have requirements that NFAs don't have

Can NFAs recognize more languages?

# NFAs and DFAs

Every DFA **is** an NFA

– DFAs have requirements that NFAs don't have

Can NFAs recognize more languages? No!

**Theorem:** For every NFA there is a DFA that recognizes exactly the same language

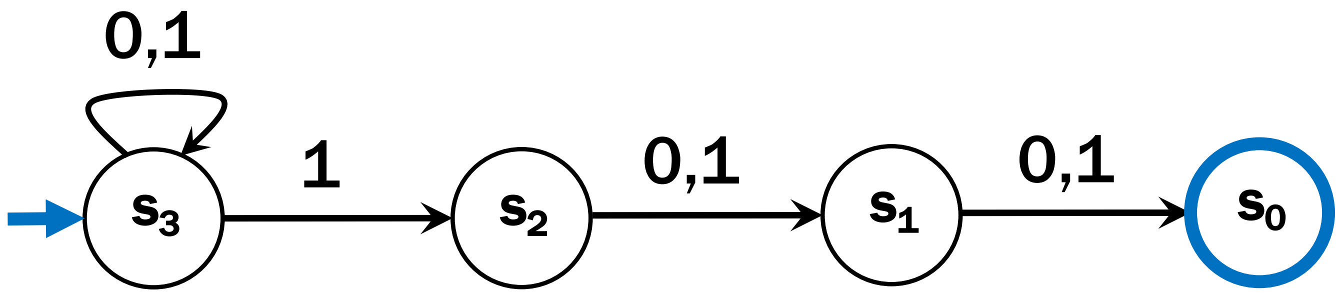
# Three ways of thinking about NFAs

- Outside observer: Is there a path labeled by x from the start state to some final state?

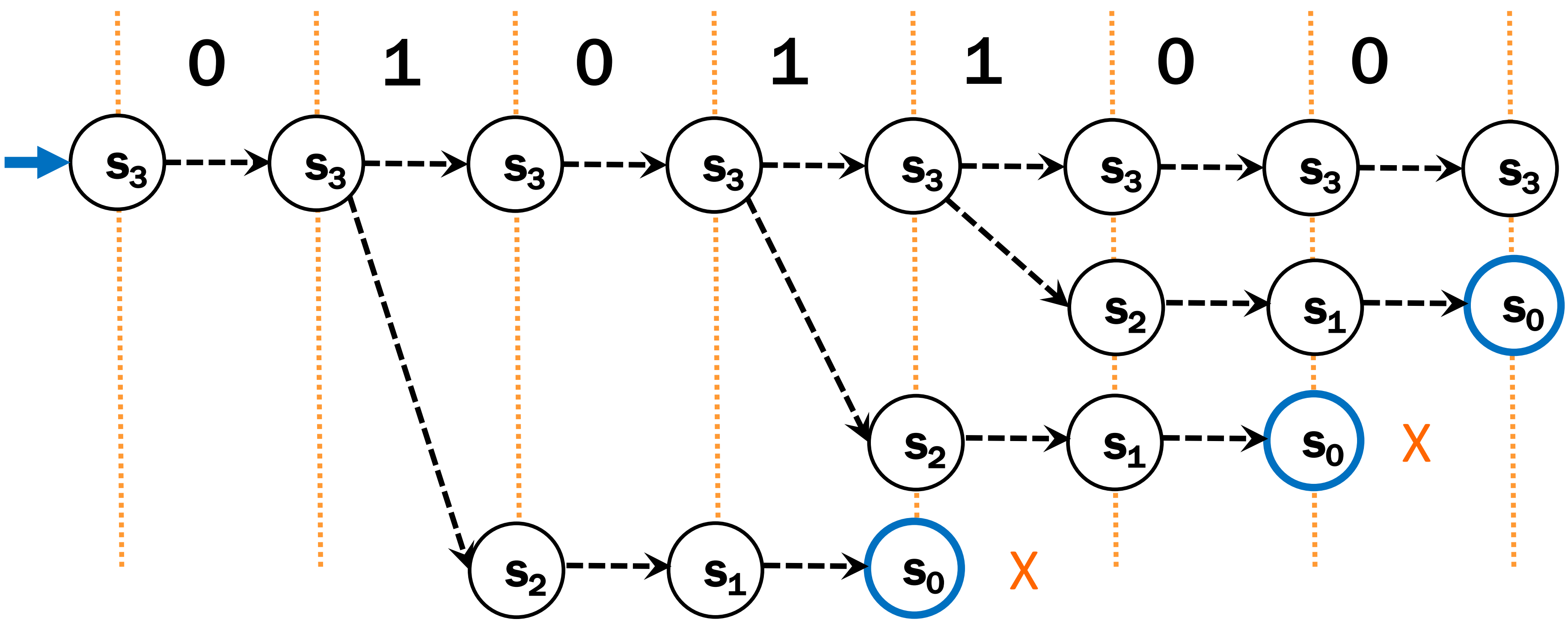
- Perfect guesser: The NFA has input x and whenever there is a choice of what to do it magically guesses a good one (if one exists)

- Parallel exploration: The NFA computation runs all possible computations on x step-by-step at the same time in parallel

# Parallel Exploration view of an NFA




Input string 0101100



# Conversion of NFAs to a DFAs

- Construction Idea:
  - The DFA keeps track of ALL states reachable in the NFA along a path labeled by the input so far

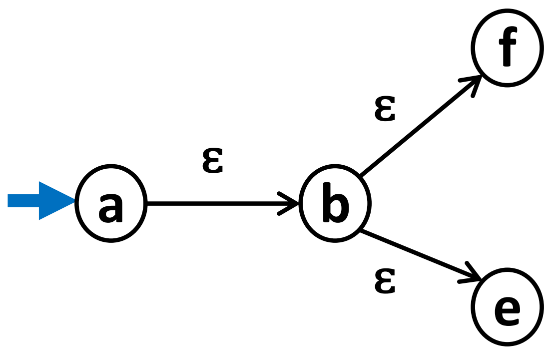
    (Note: not all *paths*; all *last states* on those paths.)

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

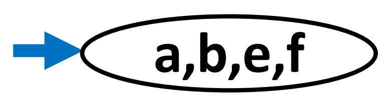
# Conversion of NFAs to a DFAs

## New start state for DFA

– The set of all states reachable from the start state of the NFA using only edges labeled ε




NFA

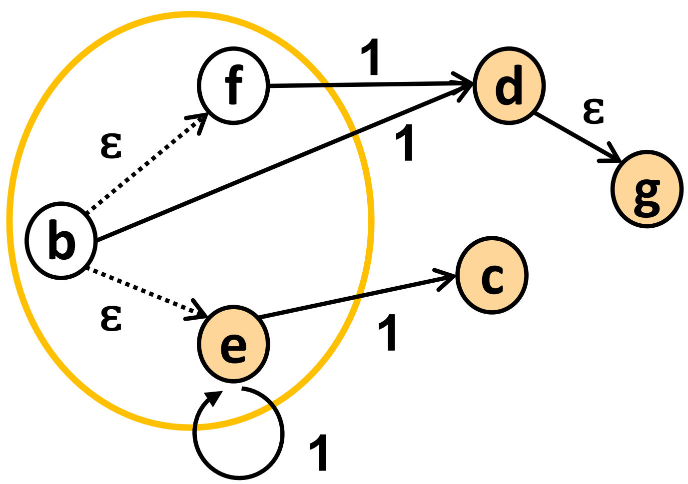


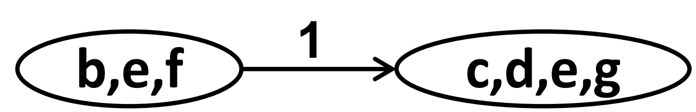

DFA

# Conversion of NFAs to a DFAs

**For each state of the DFA corresponding to a set S of states of the NFA and each symbol s**

- **Add an edge labeled s to state corresponding to T, the set of states of the NFA reached by**
  - starting from some state in S, then
  - following one edge labeled by s, and
    then following some number of edges labeled by **ε**
- T **will be ∅ if no edges from** S **labeled** s **exist**

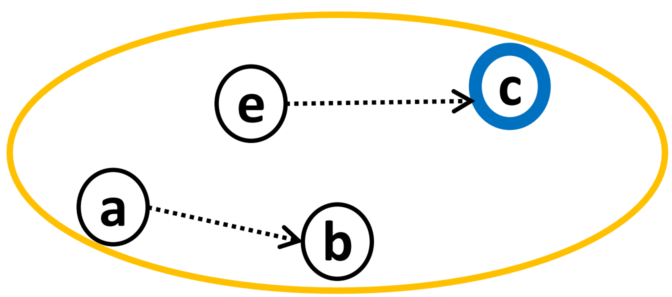


# Conversion of NFAs to a DFAs

## Final states for the DFA

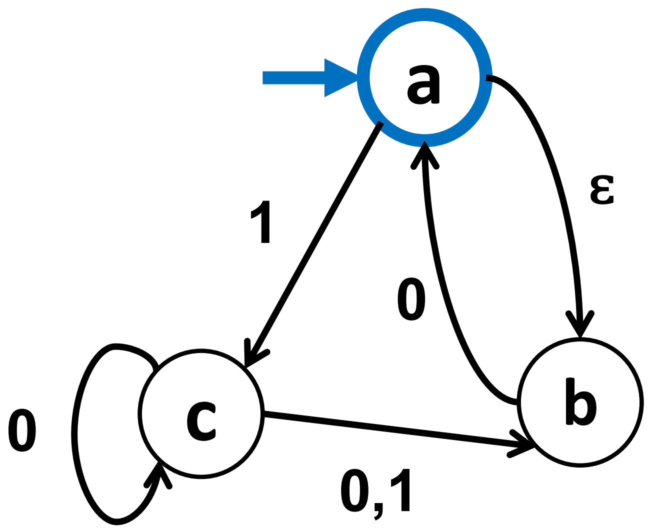
– All states whose set contain some final state of the NFA



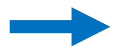

NFA


DFA

# Example: NFA to DFA

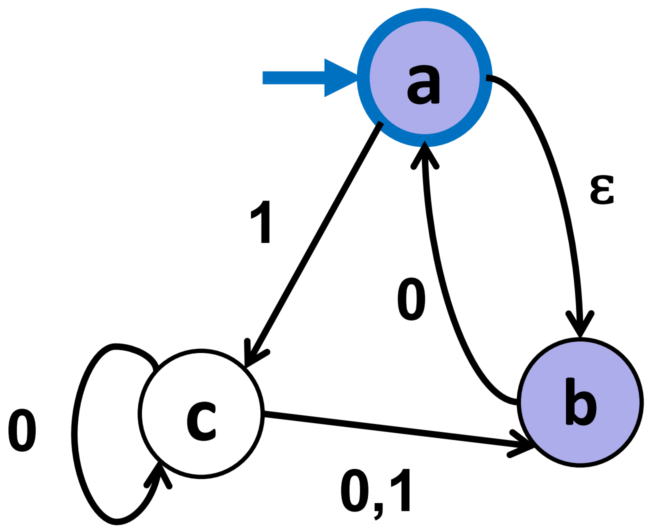



NFA

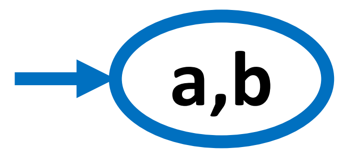


DFA

# Example: NFA to DFA

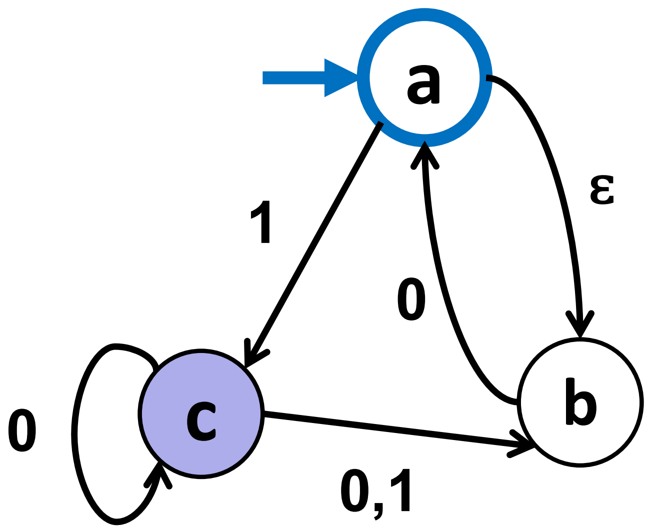



NFA

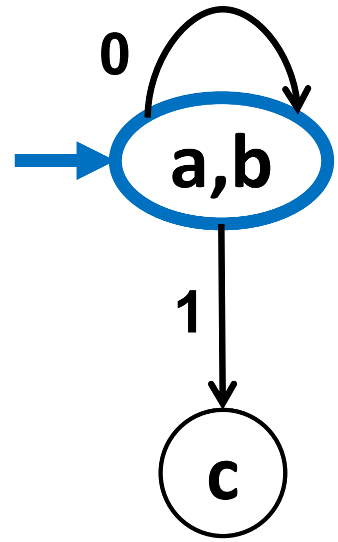



DFA

# Example: NFA to DFA

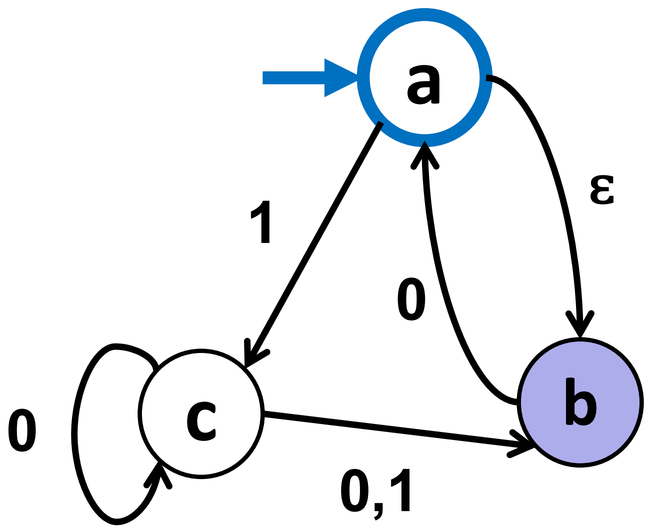



NFA

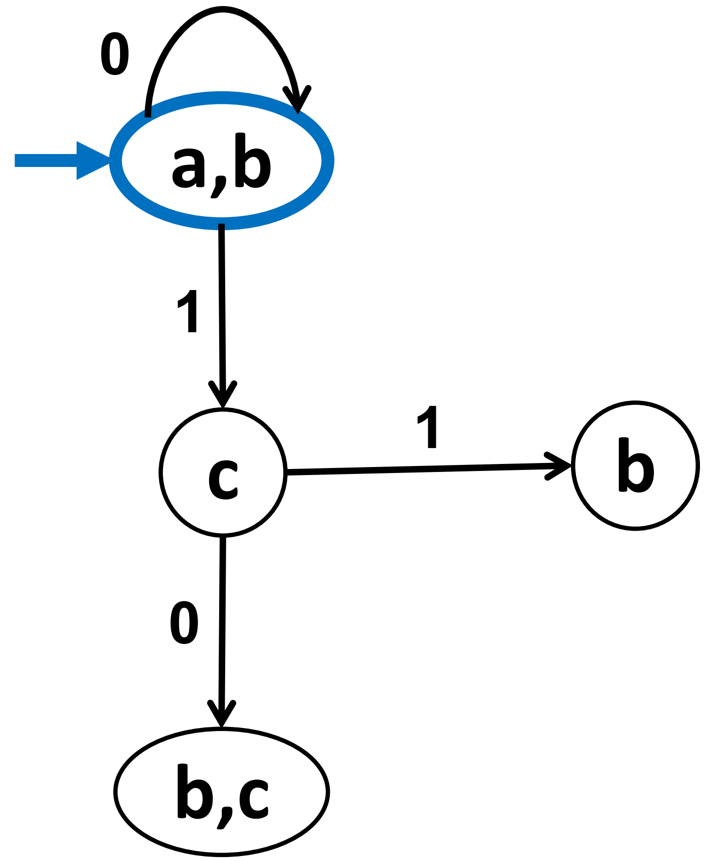



DFA

# Example: NFA to DFA

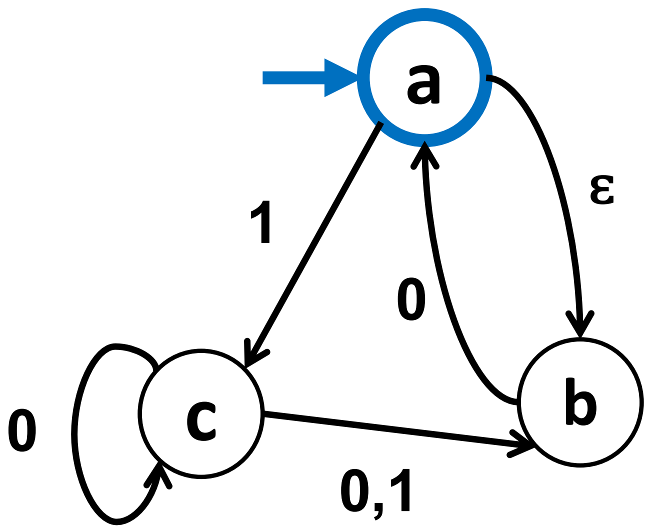



NFA

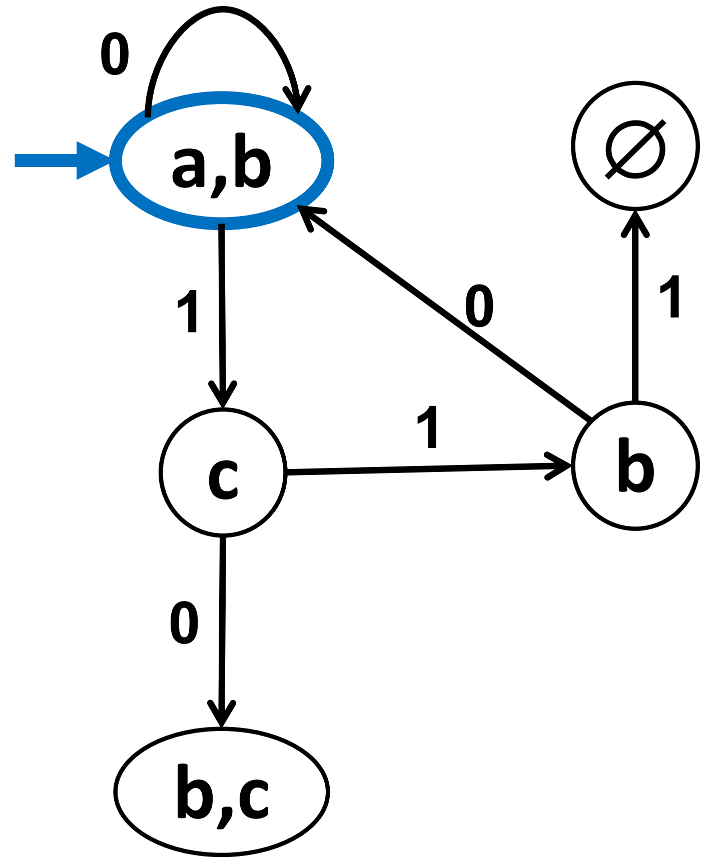



DFA

# Example: NFA to DFA

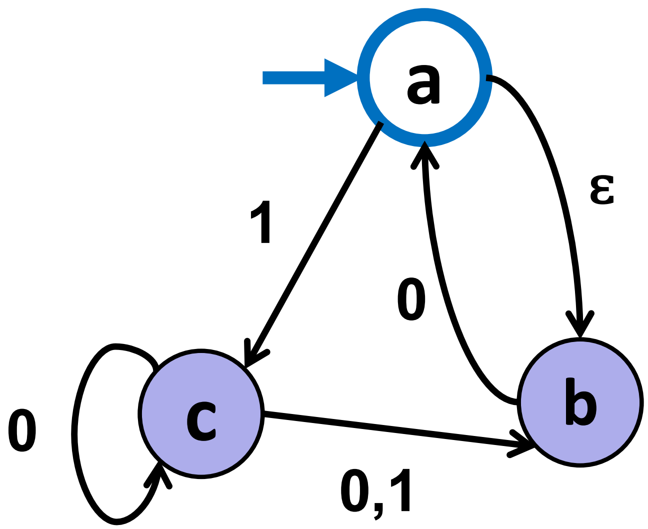



NFA

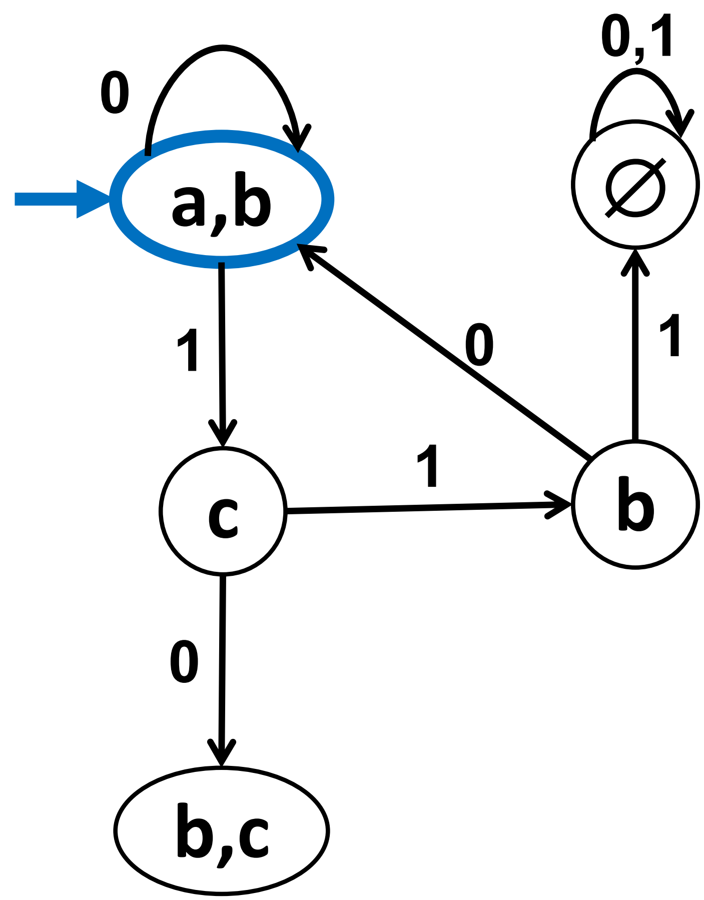



DFA

# Example: NFA to DFA

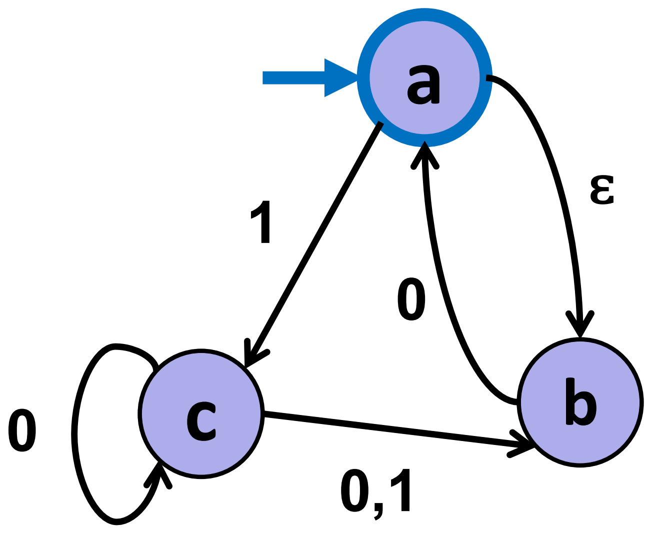



NFA

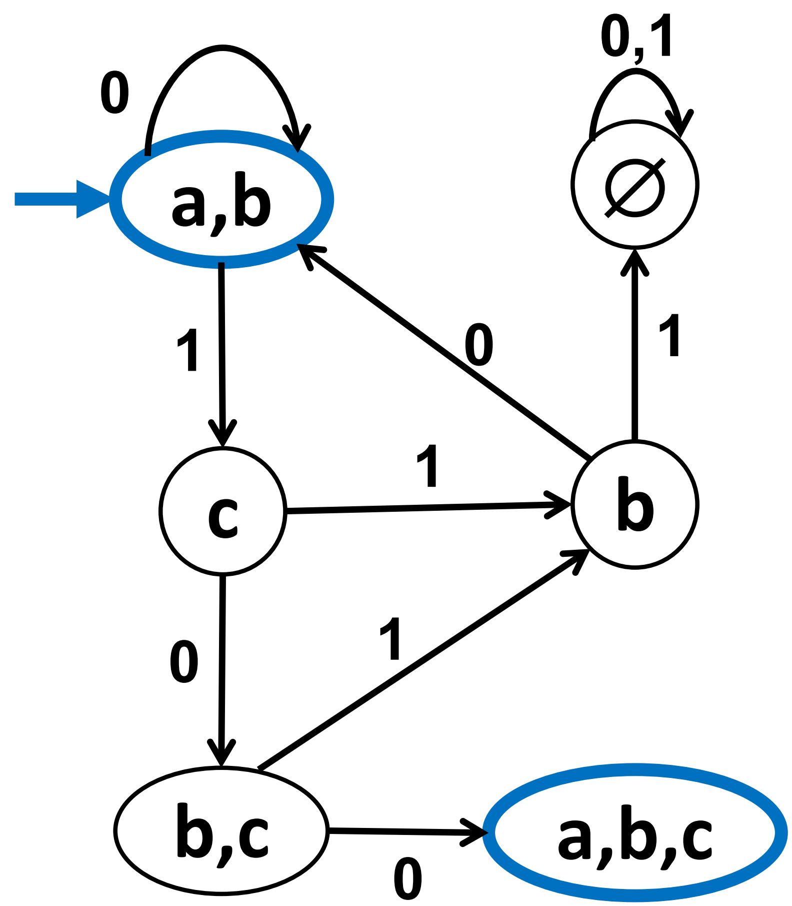



DFA

# Example: NFA to DFA

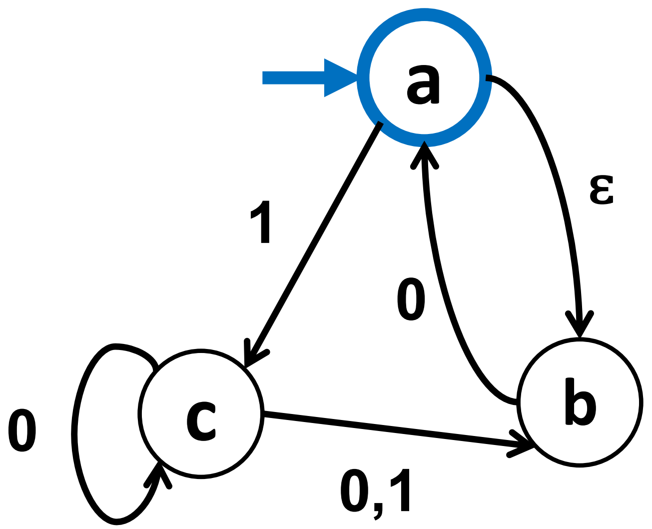



NFA

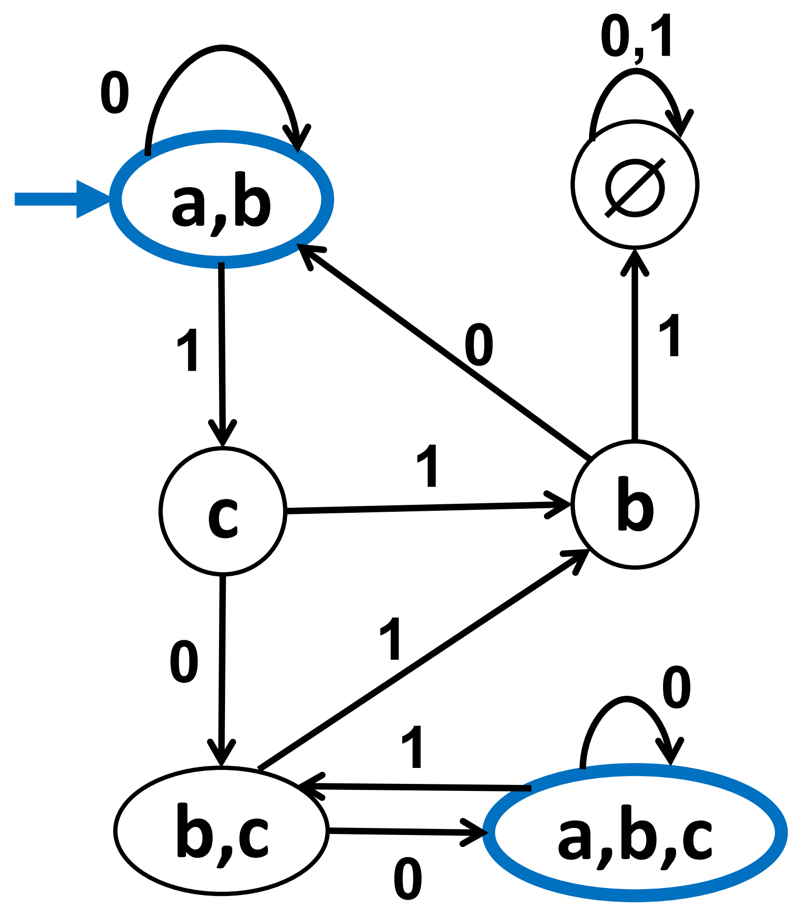



DFA

# Example: NFA to DFA




NFA




DFA

# The story so far...

$$
\begin{tikzcd}
\text{REs} \arrow[r, phantom, "\subseteq"] \arrow[dr, phantom, sloped, "\subseteq"] & \text{CFGs} \\
\text{DFAs} \arrow[r, phantom, "="] & \text{NFAs}
\end{tikzcd}
$$

# Regular expressions ⊆ NFAs ≡ DFAs

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

– Build NFA
– Convert NFA to DFA using subset construction
– Minimize resulting DFA

# Regular expressions ≡ NFAs ≡ DFAs

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

- Build NFA
- Convert NFA to DFA using subset construction
- Minimize 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 (though you will not be tested on the details). Algorithm described in Wed reading.

# The story so far...

| | | |
|---|---|---|
| REs | $\subseteq$ | CFGs |
| $\parallel$ | | |
| DFAs | $=$ | NFAs |