

# CS21

# Decidability and Tractability

Lecture 2

January 5, 2018

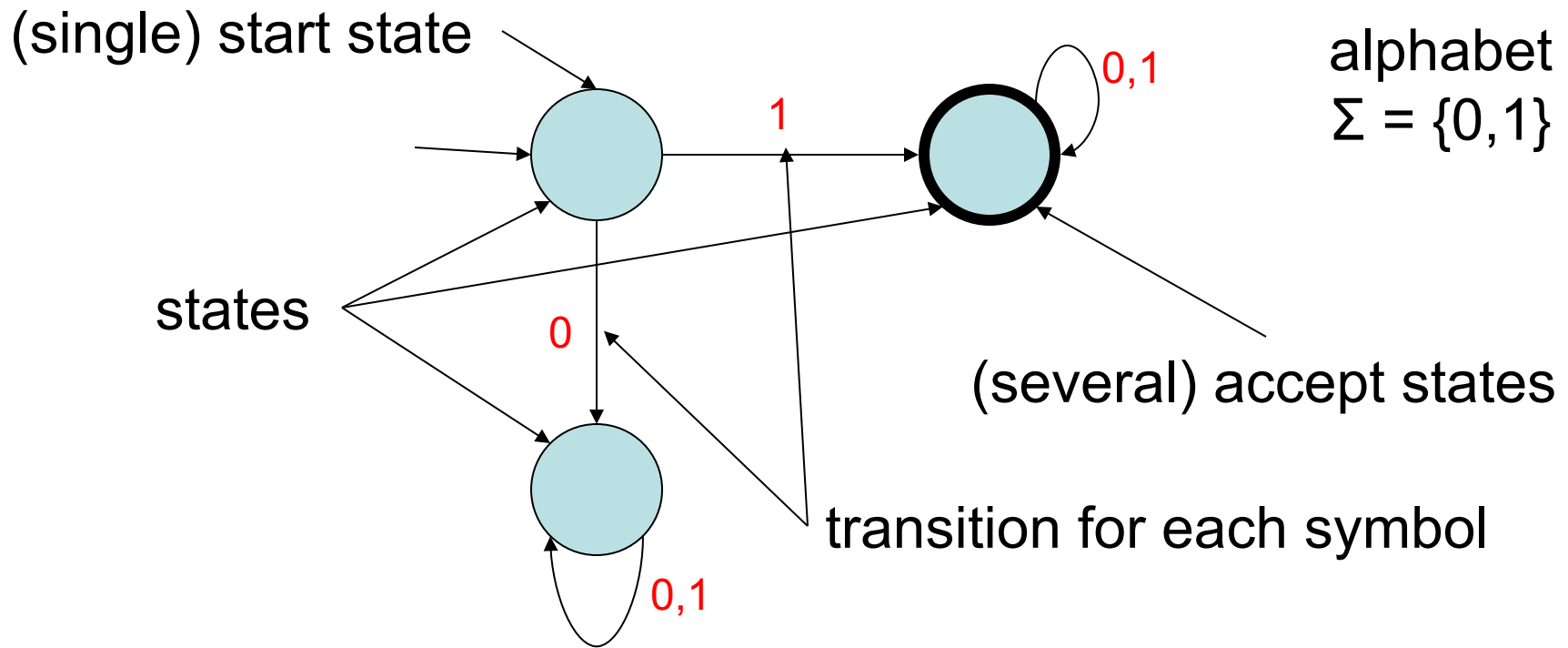
# Outline

- Finite Automata
- Nondeterministic Finite Automata
- Closure under regular operations
- NFA, FA equivalence

# Finite Automata

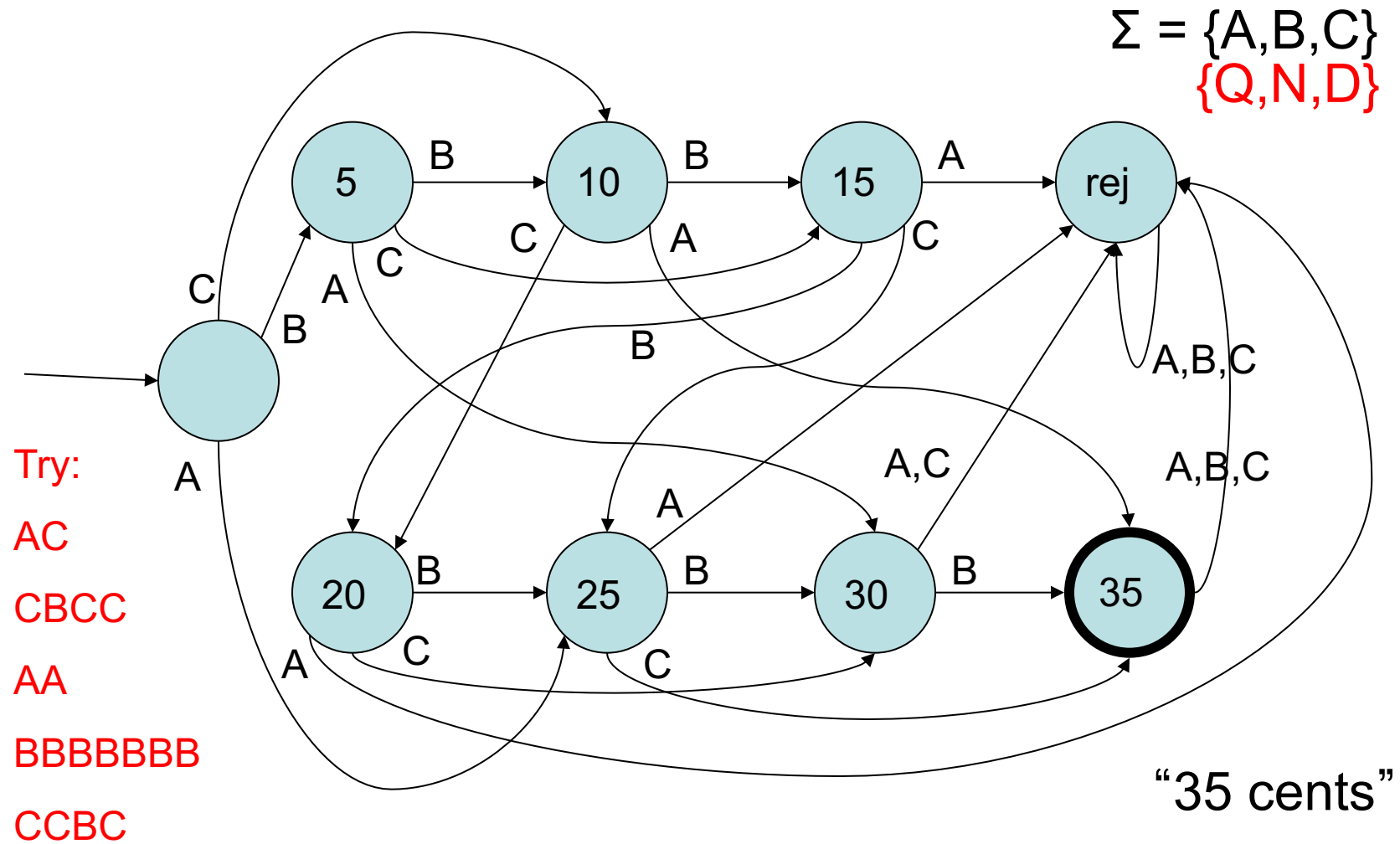
- simple model of computation
- reads input from left to right, one symbol at a time
- maintains **state**: information about what seen so far (“memory”)
  - **finite** automaton has **finite** # of states: cannot remember more things for longer inputs
- 2 ways to describe: by diagram, or formally

# FA diagrams



- read input one symbol at a time; follow arrows; accept if end in accept state

# Example FA



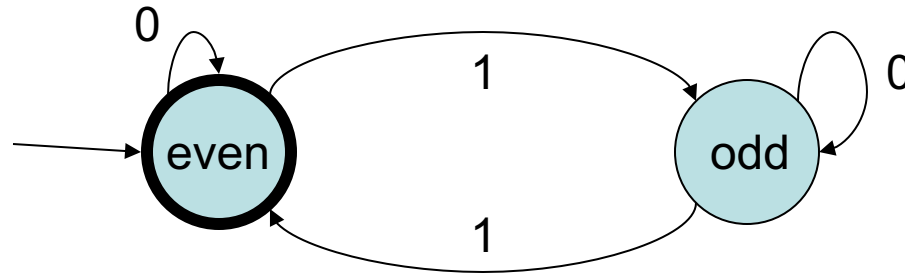
# FA formal definition

A finite automaton is a 5-tuple

$$(Q, \Sigma, \delta, q_0, F)$$

- $Q$  is a finite set called the **states**
- $\Sigma$  is a finite set called the **alphabet**
- $\delta: Q \times \Sigma \rightarrow Q$  is a function called the **transition function**
- $q_0$  is an element of  $Q$  called the **start state**
- $F$  is a subset of  $Q$  called the **accept states**

# FA formal definition



- Specification of this FA in formal terms:

- $Q = \{\text{even}, \text{odd}\}$

- $\Sigma = \{0, 1\}$

- $q_0 = \text{even}$

- $F = \{\text{even}\}$

function  $\delta$ :

$\delta(\text{even}, 0) = \text{even}$

$\delta(\text{even}, 1) = \text{odd}$

$\delta(\text{odd}, 0) = \text{odd}$

$\delta(\text{odd}, 1) = \text{even}$

# Formal description of FA operation

finite automaton

$$M = (Q, \Sigma, \delta, q_0, F)$$

accepts a string

$$w = w_1 w_2 w_3 \dots w_n \in \Sigma^*$$

if  $\exists$  sequence  $r_0, r_1, r_2, \dots, r_n$  of states for which

- $r_0 = q_0$
- $\delta(r_i, w_{i+1}) = r_{i+1}$  for  $i = 0, 1, 2, \dots, n-1$
- $r_n \in F$



# What now?

- We have a **model of computation**  
(Maybe this is it. Maybe everything we can do with real computers we can do with FA...)
- try to **characterize** the languages FAs can recognize
  - investigate closure under certain operations
- show that some languages not of this type

# Characterizing FA languages

- We will show that the set of languages recognized by FA is **closed** under:
  - union “ $C = (A \cup B)$ ”
  - concatenation “ $C = (A \circ B)$ ”
  - star “ $C = A^*$ ”
- Meaning: if A and B are languages recognized by a FA, then C is a language recognized by a FA

# Characterizing FA languages

- union “ $C = (A \cup B)$ ”

$$(A \cup B) = \{x : x \in A \text{ or } x \in B \text{ or both}\}$$

- concatenation “ $C = (A \circ B)$ ”

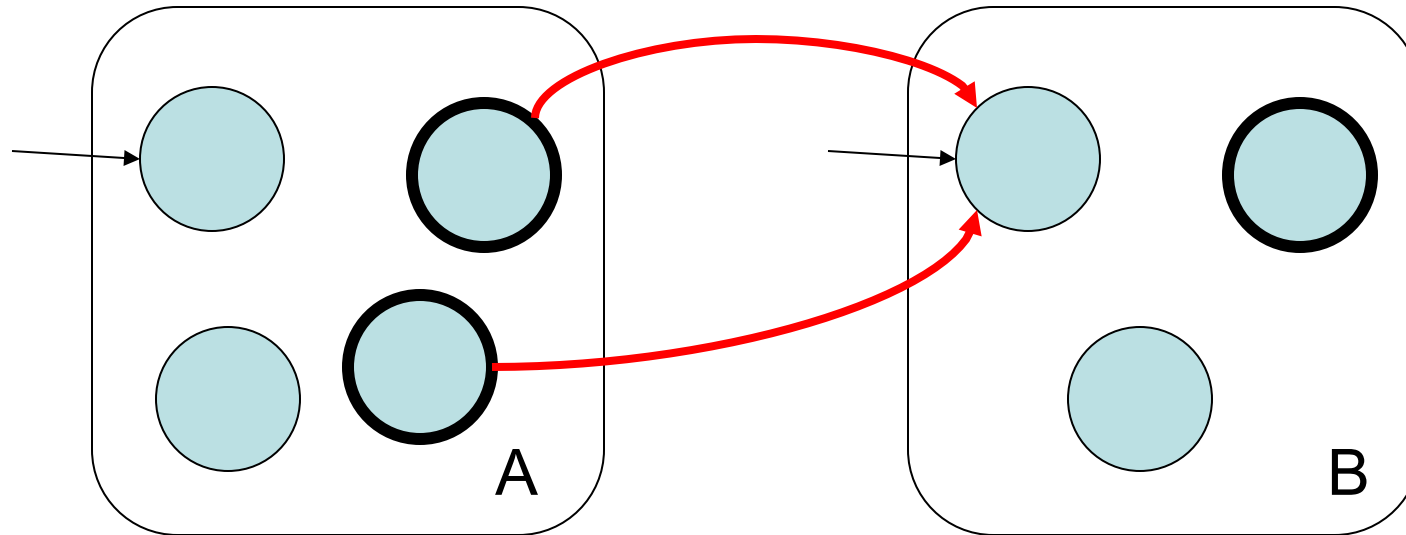
$$(A \circ B) = \{xy : x \in A \text{ and } y \in B\}$$

- star “ $C = A^*$ ” (note:  $\varepsilon$  always in  $A^*$ )

$$A^* = \{x_1x_2x_3\dots x_k : k \geq 0 \text{ and each } x_i \in A\}$$

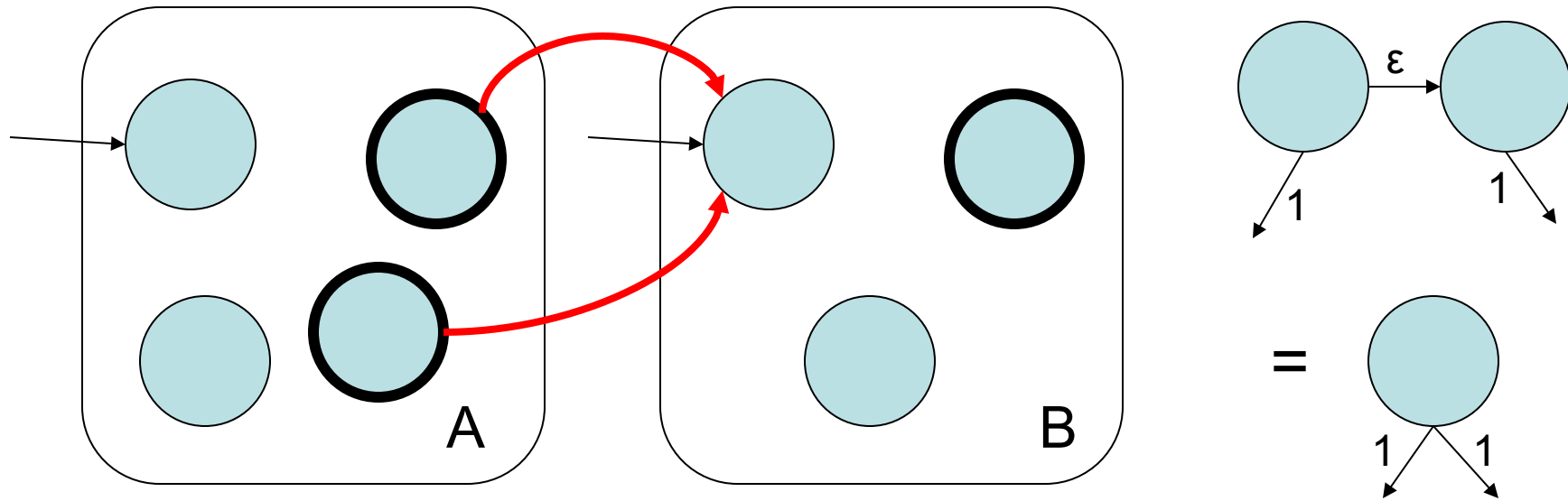
# Concatenation attempt

$$(A \circ B) = \{xy : x \in A \text{ and } y \in B\}$$



What label do we put on the new transitions?

# Concatenation attempt



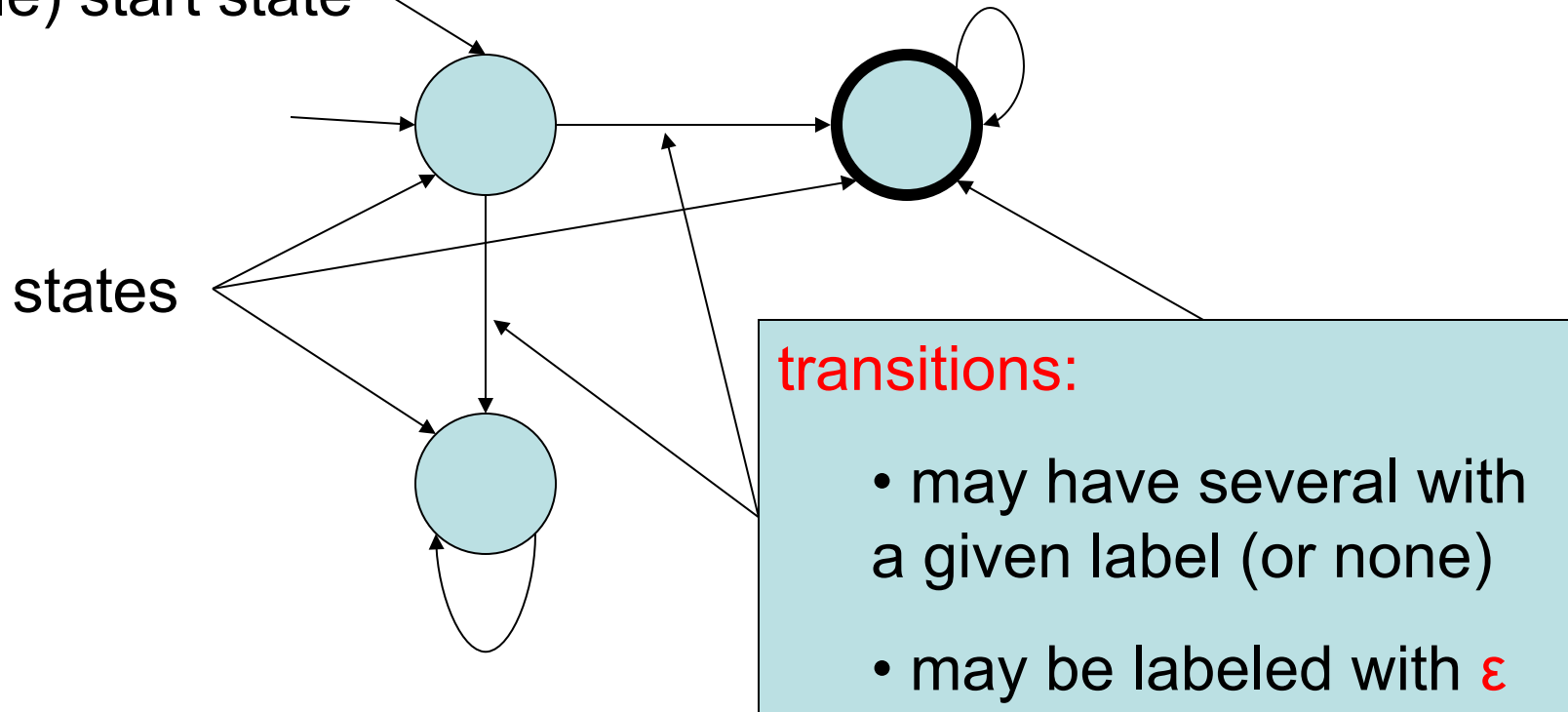
- Need it to happen “for free”: label with  $\epsilon$  (?)
- allows construct with multiple transitions with the same label (!?)

# Nondeterministic FA

- We will make life easier by describing an additional feature (**nondeterminism**) that helps us to “program” FAs
- We will **prove** that FAs with this new feature can be **simulated** by ordinary FA
  - same spirit as programming constructs like procedures
- The concept of **nondeterminism** has a significant role in TCS and this course.

# NFA diagrams

(single) start state

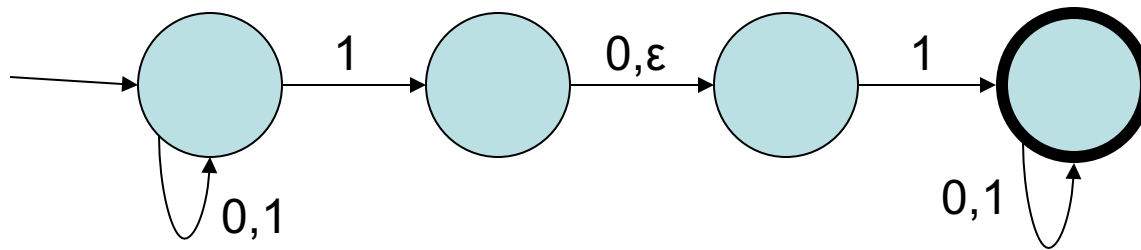


- At each step, **several** choices for next state

# NFA operation

- Example of NFA operation:

alphabet  
 $\Sigma = \{0,1\}$



input: 0 1 0

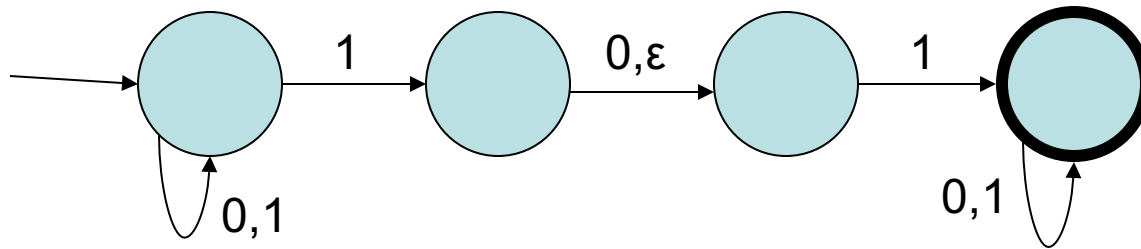
not accepted



# NFA operation

- Example of NFA operation:

alphabet  
 $\Sigma = \{0,1\}$



input: 1 1 0

accepted

# NFA operation

- One way to think of NFA operation:
- string  $x = x_1x_2x_3\dots x_n$  accepted if and only if
  - there exists a way of inserting  $\epsilon$ 's into  $x$

$$x_1 \epsilon \epsilon x_2 x_3 \dots \epsilon x_n$$

- so that there exists a path of transitions from the start state to an accept state

# NFA formal definition

A nondeterministic FA

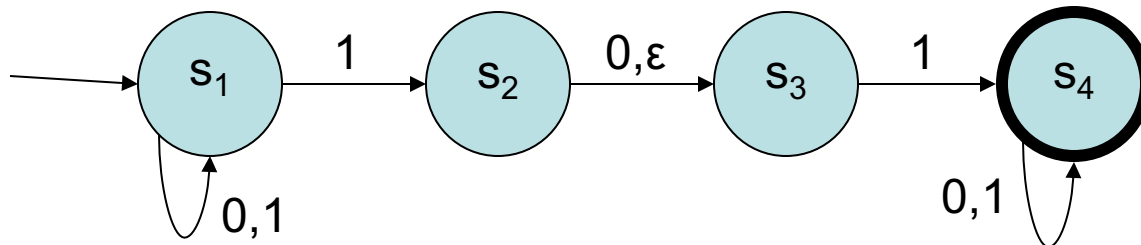
$(Q, \Sigma, \delta$

- $Q$  is a finite set called the **states**
- $\Sigma$  is a finite set called the **alphabet**
- $\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \wp(Q)$  is a function called the **transition function**
- $q_0$  is an element of  $Q$  called the **start state**
- $F$  is a subset of  $Q$  called the **accept states**

transition  
labeled  
alphabet  
symbols or  $\epsilon$

“powerset of  $Q$ ”:  
the set of all  
subsets of  $Q$

# NFA formal definition



- Specification of this NFA in formal terms:

–  $Q = \{s_1, s_2, s_3, s_4\}$

–  $\Sigma = \{0, 1\}$

–  $q_0 = s_1$

–  $F = \{s_4\}$

$\delta(s_1, 0) = \{s_1\}$

$\delta(s_1, 1) = \{s_1, s_2\}$

$\delta(s_1, \epsilon) = \{\}$

$\delta(s_2, 0) = \{s_3\}$

$\delta(s_2, 1) = \{\}$

$\delta(s_2, \epsilon) = \{s_3\}$

$\delta(s_3, 0) = \{\}$

$\delta(s_3, 1) = \{s_4\}$

$\delta(s_3, \epsilon) = \{\}$

$\delta(s_4, 0) = \{s_4\}$

$\delta(s_4, 1) = \{s_4\}$

$\delta(s_4, \epsilon) = \{\}$

# Formal description of NFA operation

NFA  $M = (Q, \Sigma, \delta, q_0, F)$

accepts a string  $w = w_1w_2w_3\dots w_n \in \Sigma^*$

if  $w$  can be written (by inserting  $\varepsilon$ 's) as:

$$y = y_1y_2y_3\dots y_m \in (\Sigma \cup \{\varepsilon\})^*$$

and  $\exists$  sequence  $r_0, r_1, \dots, r_m$  of states for which

$$- r_0 = q_0$$

$$- r_{i+1} \in \delta(r_i, y_{i+1}) \quad \text{for } i = 0, 1, 2, \dots, m-1$$

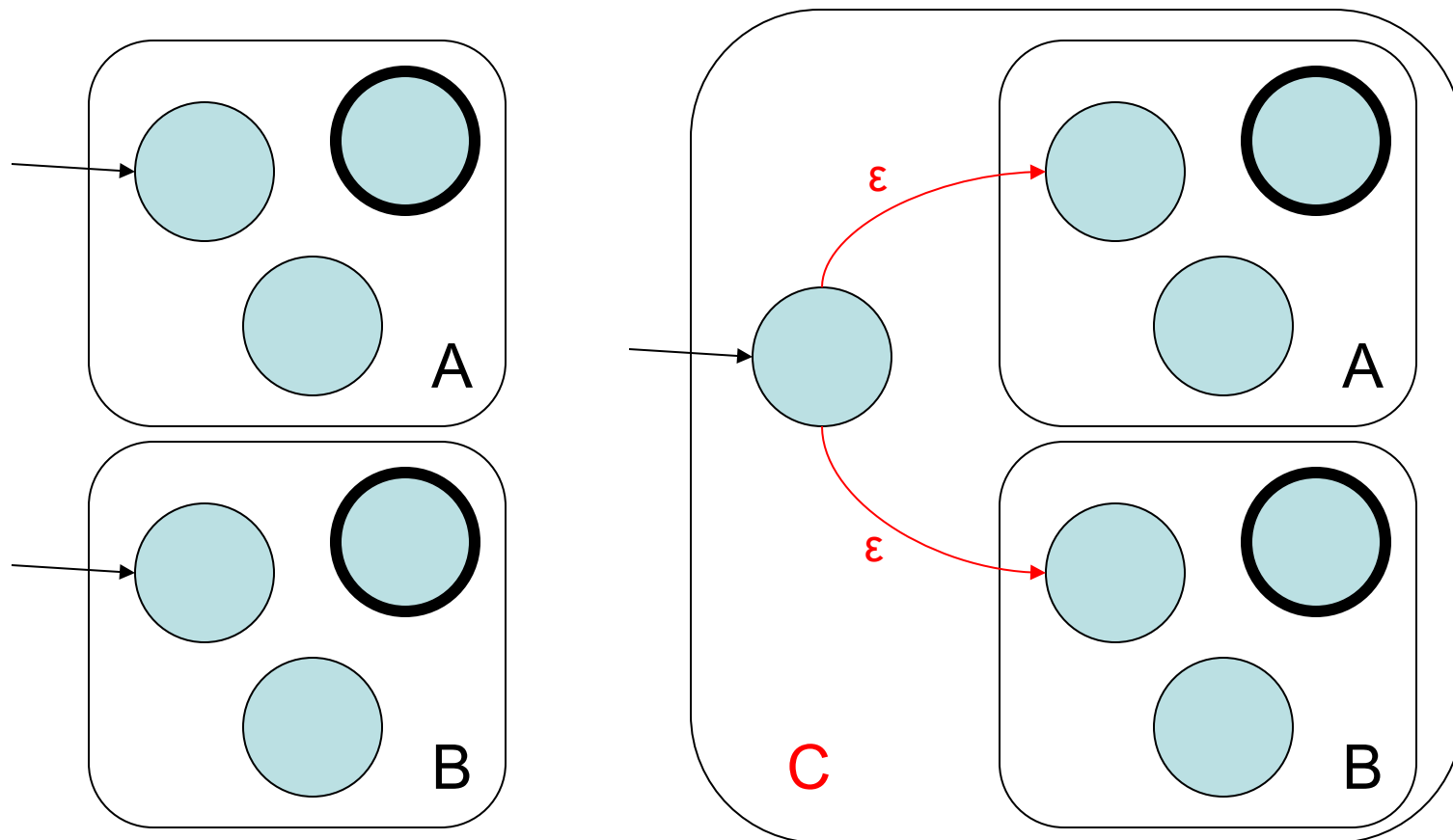
$$- r_m \in F$$

# Closures

- Recall: want to show the set of languages recognized by **NFA** is **closed** under:
  - union “ $C = (A \cup B)$ ”
  - concatenation “ $C = (A \circ B)$ ”
  - star “ $C = A^*$ ”

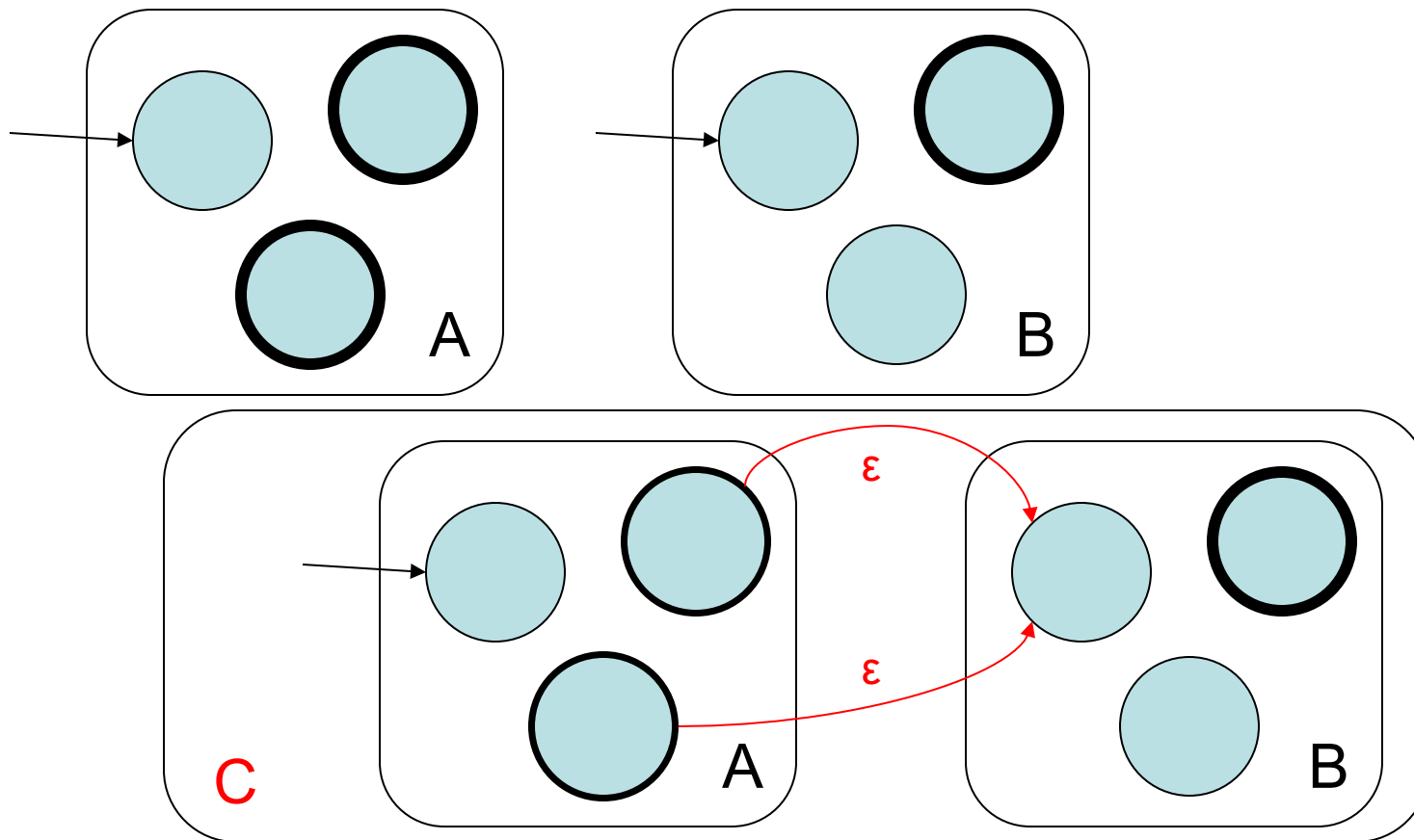
# Closure under union

$$C = (A \cup B) = \{x : x \in A \text{ or } x \in B\}$$



# Closure under concatenation

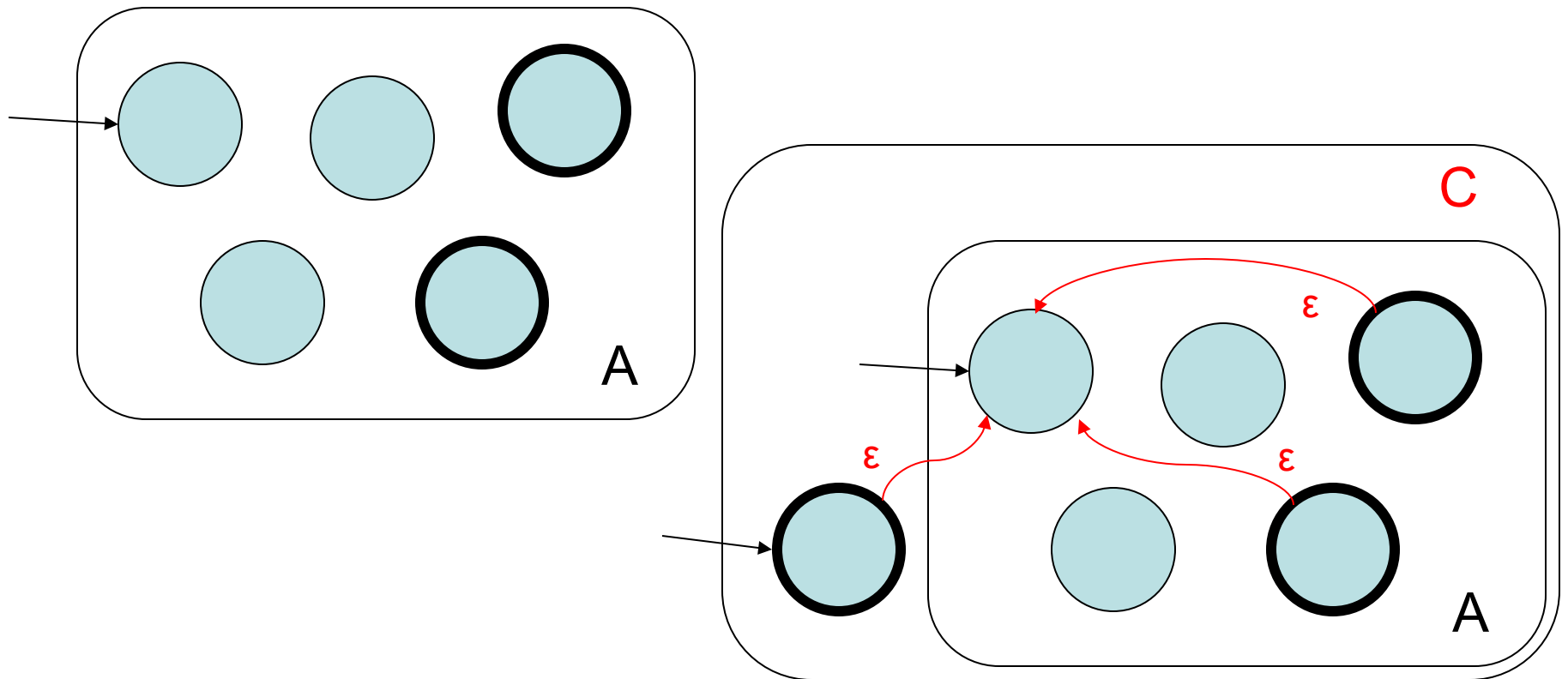
$$C = (A \circ B) = \{xy : x \in A \text{ and } y \in B\}$$





# Closure under star

$$C = A^* = \{x_1x_2x_3\dots x_k : k \geq 0 \text{ and each } x_i \in A\}$$



# NFA, FA equivalence

**Theorem**: a language  $L$  is recognized by a FA **if and only if**  $L$  is recognized by a NFA.

Must prove *two* directions:

$(\Rightarrow)$   $L$  is recognized by a FA **implies**  $L$  is recognized by a NFA.

$(\Leftarrow)$   $L$  is recognized by a NFA **implies**  $L$  is recognized by a FA.

(usually one is easy, the other more difficult)

# NFA, FA equivalence

( $\Rightarrow$ ) L is recognized by a FA **implies** L is recognized by a NFA

**Proof:** a finite automaton **is** a nondeterministic finite automaton that happens to have no  $\epsilon$ -transitions, and for which each state has exactly one outgoing transition for each symbol.

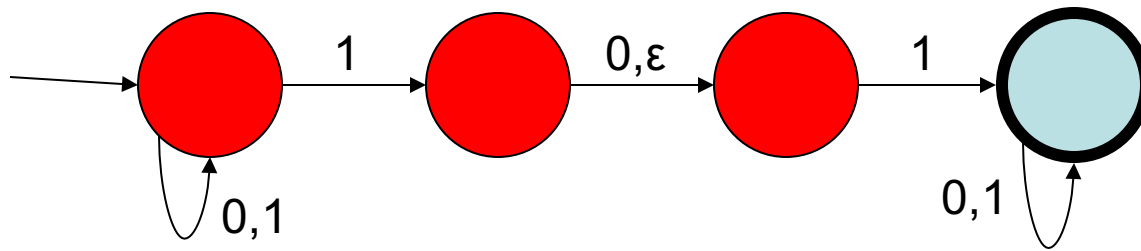
# NFA, FA equivalence

( $\Leftarrow$ ) L is recognized by a NFA **implies** L is recognized by a FA.

**Proof:** we will build a FA that *simulates* the NFA (and thus recognizes the same language).

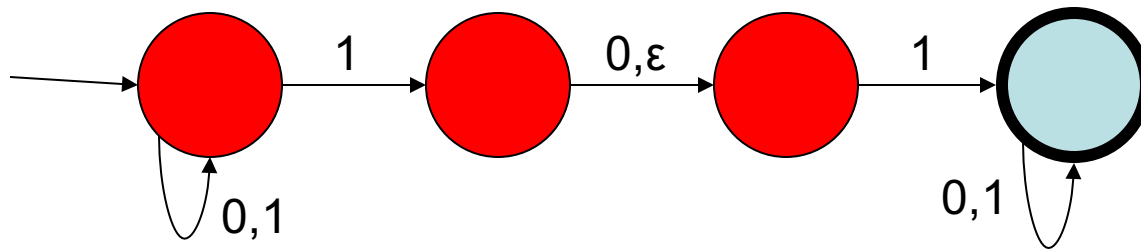
- alphabet will be the same
- what are the states of the FA?

# NFA, FA equivalence



- given NFA  $M = (Q, \Sigma, \delta, q_0, F)$
- construct FA  $M' = (Q', \Sigma', \delta', q_0', F')$
- same alphabet:  $\Sigma' = \Sigma$
- states are **subsets** of  $M$ 's states:  $Q' = \wp(Q)$
- if we are in state  $R \in Q'$  and we read symbol  $a \in \Sigma'$ , what is the new state?

# NFA, FA equivalence

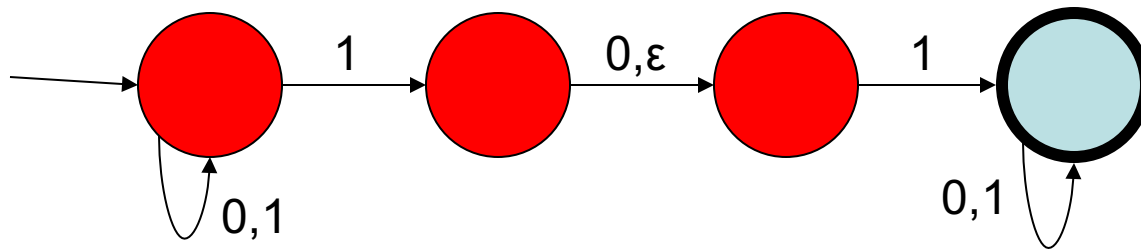


- given NFA  $M = (Q, \Sigma, \delta, q_0, F)$
- construct FA  $M' = (Q', \Sigma', \delta', q_0', F')$

**Helpful def'n:**  $E(S) = \{q \in Q : q \text{ reachable from } S \text{ by traveling along 0 or more } \varepsilon\text{-transitions}\}$

- new transition fn:  $\delta'(R, a) = \cup_{r \in R} E(\delta(r, a))$   
= “all nodes reachable from R by following an a-transition, and then 0 or more  $\varepsilon$ -transitions”

# NFA, FA equivalence



– given NFA

$$M = (Q, \Sigma, \delta, q_0, F)$$

– construct FA

$$M' = (Q', \Sigma', \delta', q_0', F')$$

– new start state:  $q_0' = E(\{q_0\})$

– new accept states:

$$F' = \{R \in Q' : R \text{ contains an accept state of } M\}$$

# NFA, FA equivalence

- We have proved ( $\Leftrightarrow$ ) by construction.

Formally we should also prove that the construction works, by induction on the number of steps of the computation.

- at each step, the state of the FA  $M'$  is exactly the set of **reachable** states of the NFA  $M$ ...



# So far...

**Theorem**: the set of languages recognized by NFA is closed under union, concatenation, and star.

**Theorem**: a language  $L$  is recognized by a FA if and only if  $L$  is recognized by a NFA.

**Theorem**: the set of languages recognized by FA is closed under union, concatenation, and star.

# Next...

- Describe the set of languages that can be built up from:
  - unions
  - concatenations
  - star operations
- Called “patterns” or **regular expressions**
- **Theorem**: a language  $L$  is recognized by a FA **if and only if**  $L$  is described by a regular expression.

# Regular expressions

- R is a regular expression if R is
  - $a$ , for some  $a \in \Sigma$
  - $\varepsilon$ , the empty string
  - $\emptyset$ , the empty set
  - $(R_1 \cup R_2)$ , where  $R_1$  and  $R_2$  are reg. exprs.
  - $(R_1 \circ R_2)$ , where  $R_1$  and  $R_2$  are reg. exprs.
  - $(R_1^*)$ , where  $R_1$  is a regular expression

A reg. expression R describes the **language**  $L(R)$ .

# Regular expressions

- example:  $R = (0 \cup 1)$ 
  - if  $\Sigma = \{0, 1\}$  then use “ $\Sigma$ ” as shorthand for  $R$
- example:  $R = 0 \circ \Sigma^*$ 
  - shorthand: omit “ $\circ$ ”       $R = 0\Sigma^*$
  - precedence:  $*$ , then  $\circ$ , then  $\cup$ , unless override by parentheses
  - in example  $R = 0(\Sigma^*)$ , not  $R = (0\Sigma)^*$

# Some examples

alphabet  
 $\Sigma = \{0,1\}$

- $\{w : w \text{ has at least one } 1\}$   
 $= \Sigma^*1\Sigma^*$
- $\{w : w \text{ starts and ends with same symbol}\}$   
 $= 0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$
- $\{w : |w| \leq 5\}$   
 $= (\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)$
- $\{w : \text{every } 3^{\text{rd}} \text{ position of } w \text{ is } 1\}$   
 $= (1\Sigma\Sigma)^*(\varepsilon \cup 1 \cup 1\Sigma)$