CSC 6011: Theory of Computation

Lecture 2: Nondeterministic Finite Automata

Lecture 2

Last time:

- (Deterministic) Finite Automata, regular languages
- Regular operations $\cup$, $\circ$, $*$
- Regular expressions (= finite automata)
- Closure under $\cup$

Today:

- Nondeterministic finite automata (NFA)
- NFA = DFA
- Closure under $\circ$ and $*$
- Regular expressions → finite automata

PS: interrupt me if you have questions during the lecture.

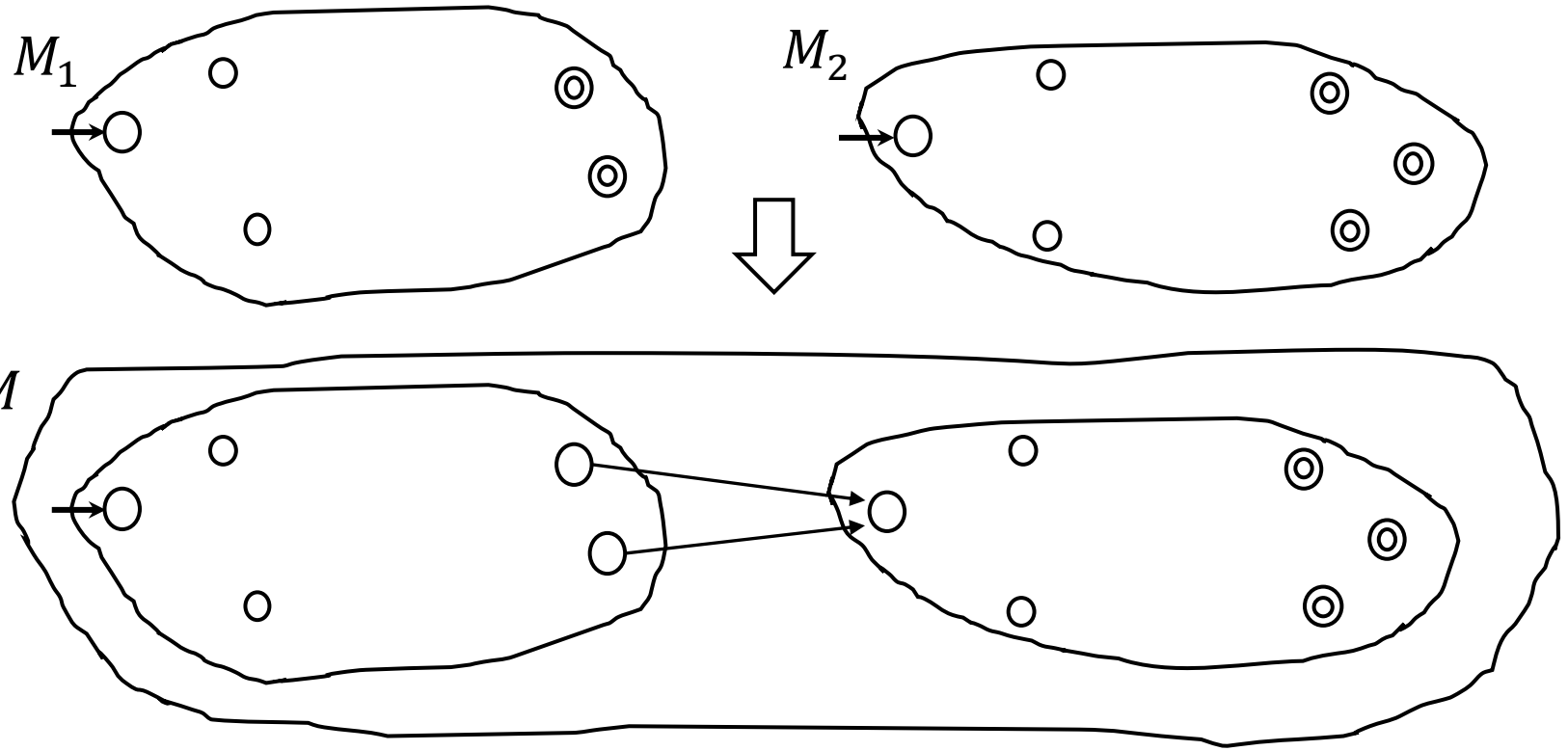
Closure Properties for Regular Languages

Theorem (closure under ∘)**:** If A_1, A_2 are regular languages, so is A_1A_2

Proof Attempt: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2

Construct $M = (Q, \Sigma, \delta, q_0, F)$ recognizing A_1A_2



M should accept input w

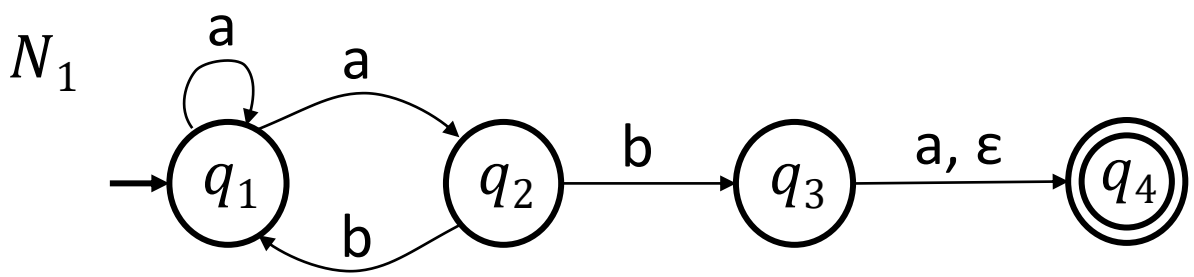
if $w = xy$ where

M_1 accepts x and M_2 accepts y.

Doesn't work: Where to split w?

Need new concept. 🤔

Nondeterministic Finite Automata (NFA)



Nondeterminism doesn't correspond to a physical machine we can build.

However, it is useful mathematically.

New features of nondeterminism:

- multiple paths possible (0, 1 or many at each step)
- ε-transition is a "free" move without reading input
- Accept input if some path leads to ◎

Example inputs:

- ab Possible end states: $\{q_1, q_3, q_4\}$ ==> accept
- aa Possible end states: $\{q_1, q_2\}$ ==> reject
- aba Possible end states: $\{q_1, q_2, q_4\}$ ==> accept
- abb No possible path ==> reject

Mini-Quiz 2.1

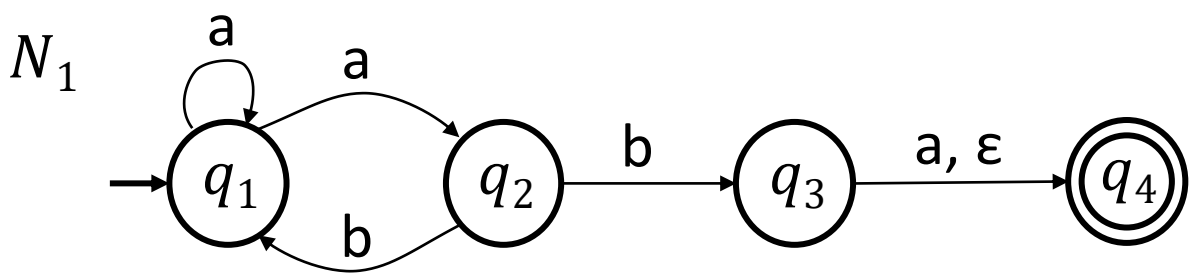
What does N_1 do on input aab ?

(a) Accept

(b) Reject

(c) Both Accept and Reject

Formal Definition of NFA



Defn: A nondeterministic finite automaton (NFA)

N is a 5-tuple $(Q,\ \Sigma,\ \delta,\ q_0,\ F)$

states, alphabet, transition function, start state, accept states

- all same as before except δ
- $\delta: Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q) = \{R | R \subseteq Q\}$

  $\Sigma \cup \{\varepsilon\}$ "power set" (set of all subsets)

- In the N_1 example: $\delta(q_1, \text{a}) = \{q_1, q_2\}$

  $\delta(q_1, \text{b}) = \emptyset$

3 ways to think about NFA:

Computational: Fork many parallel threads and accept if any thread leads to an accept state.

Mathematical: Tree with branches. Accept if any branch leads to an accept state.

Magical: At each nondeterministic step, "guess" which way to go. Machine always makes the right guess that leads to accepting, if possible.

Converting NFAs to DFAs (so NFAs = DFAs)

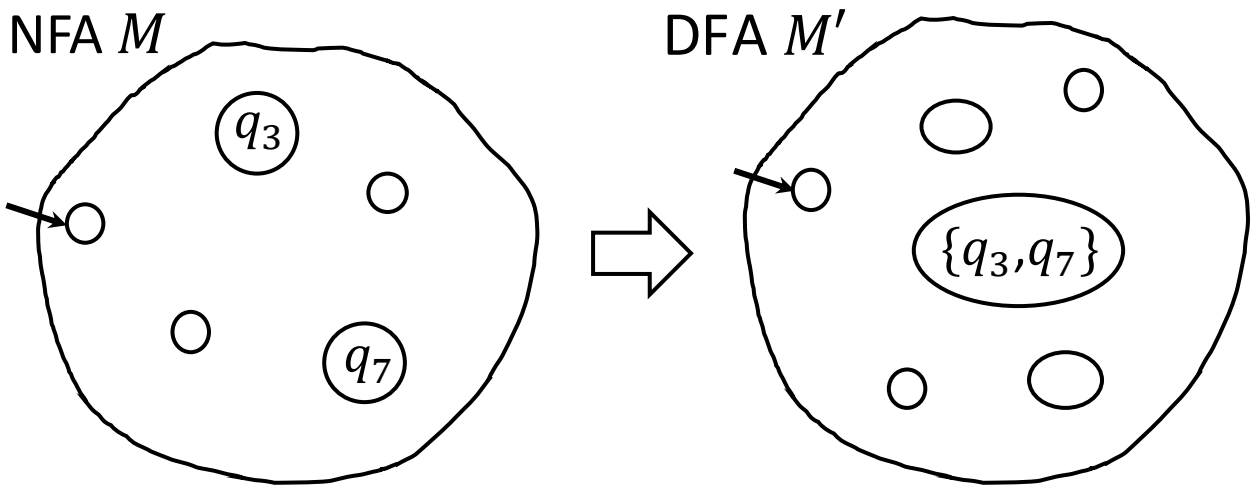
Theorem: If an NFA recognizes A then A is regular (i.e., recognized by a DFA)

Proof: Let NFA $M = (Q, \Sigma, \delta, q_0, F)$ recognize A

Construct DFA $M' = (Q', \Sigma, \delta', q_0', F')$ recognizing A

(Ignore the ε-transitions, can easily modify to handle them)

Idea: DFA M' keeps track of the subset of possible states in NFA M.



Construction of M':

$Q' = \boldsymbol{\mathcal{P}}(Q)$

$\delta'(R, a) = \{q \mid q \in \delta(r, a)$ for some $r \in R\}$

$R \in Q'$ (i.e., $R \subseteq Q$)

$q_0' = \{q_0\}$

$F' = \{R \in Q' \mid R$ intersects $F\}$

Converting NFAs to DFAs (so NFAs = DFAs)

Theorem: If an NFA recognizes A then A is regular (i.e., reco

Proof: Let NFA $M = (Q, \Sigma, \delta, q_0, F)$ recognize A

Construct DFA $M' = (Q', \Sigma, \delta', q_0', F')$ recognizing A

(Ignore the ε-transitions, can easily modify to handle then

Idea: DFA M' keeps track of the subset of possible states in

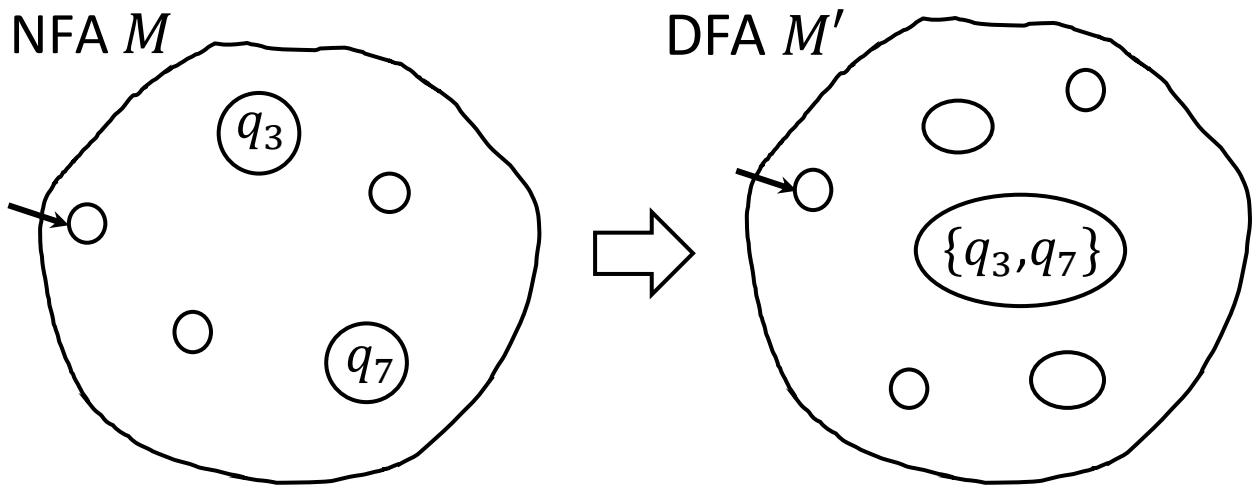
Mini-Quiz 2.2

If NFA M has n states, how many states does DFA M' have by this construction?

(a) $2n$

(b) n^2

(c) 2^n



Construction of M':

$Q' = \boldsymbol{\mathcal{P}}(Q)$

$\delta'(R, a) = \{q \mid q \in \delta(r, a)$ for some $r \in R\}$

$R \in Q'$ (i.e., $R \subseteq Q$)

$q_0' = \{q_0\}$

$F' = \{R \in Q' \mid R$ intersects $F\}$

Return to Closure Properties

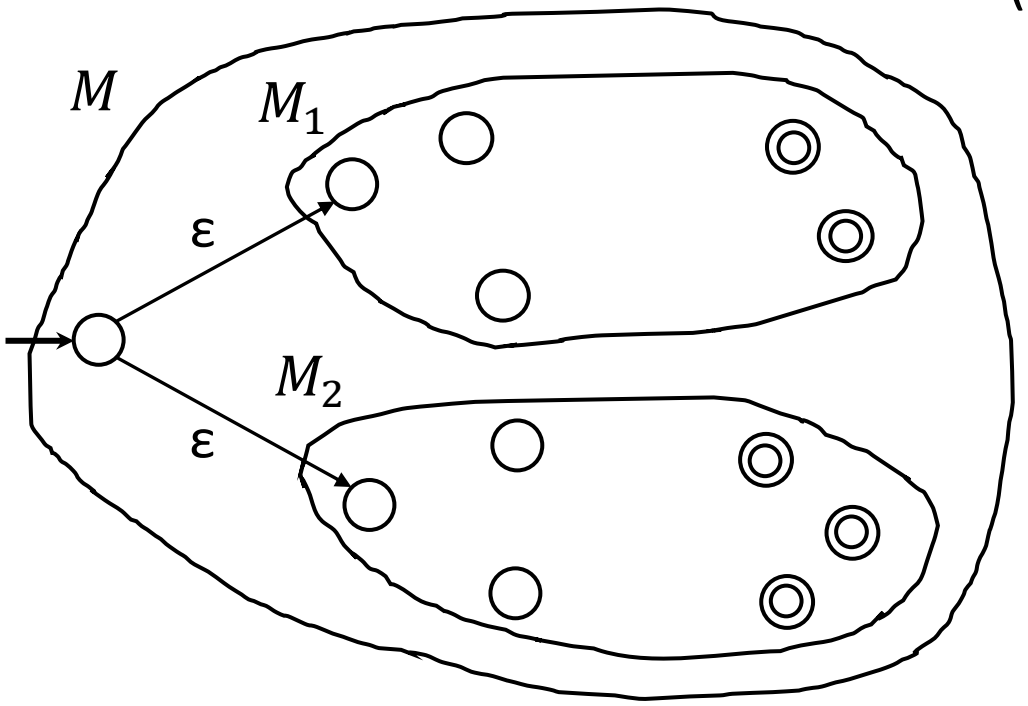
Recall Theorem: If A_1, A_2 are regular languages, so is $A_1 \cup A_2$

(The class of regular languages is closed under union)

New Proof (sketch): Given DFAs M_1 and M_2 recognizing A_1 and A_2

Construct NFA M recognizing $A_1 \cup A_2$

(implies the existence of such a DFA as well)



<u>Nondeterminism</u>

parallelism

vs

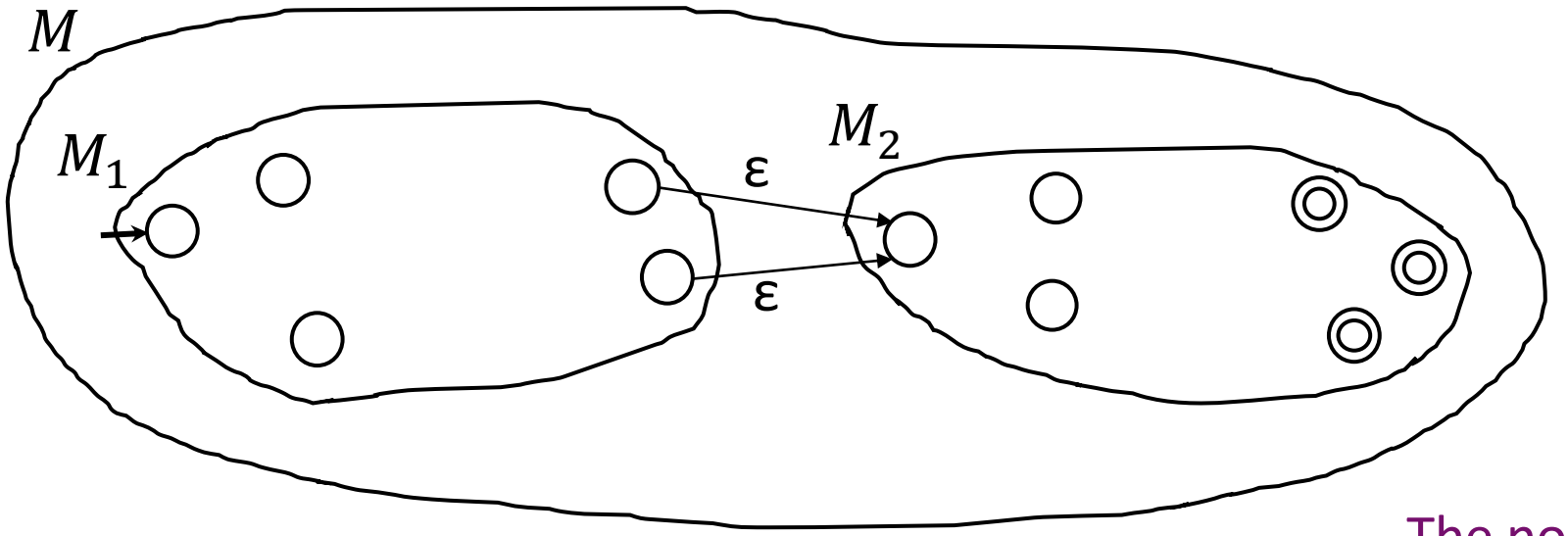
guessing

Closure under ∘ (concatenation)

Theorem: If A_1, A_2 are regular languages, so is $A_1 A_2$

Proof sketch: Given DFAs M_1 and M_1 recognizing A_1 and A_2

Construct NFA M recognizing $A_1 A_2$



M should accept input w

if $w = xy$ where

M_1 accepts x and M_2 accepts y.

$w =$ x | y

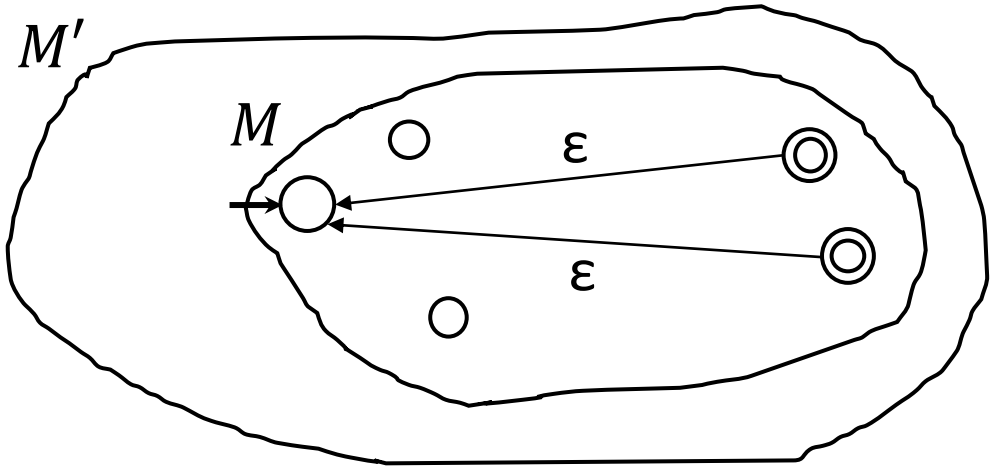
The nondeterministic M' can guess whether to jump to M_2 when M_1 accepts.

Closure under $*$ (star)

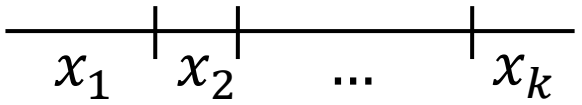
Theorem: If A is a regular language, so is $A^* = \{x_1 x_2 \ldots x_k \mid \text{each } x_i \in A,\ k \geq 0\}$

Proof sketch: Given DFA M recognizing A

Construct NFA M' recognizing A^*



M' should accept input w

if $w =$ 

where $k \geq 0$ and M accepts each x_i

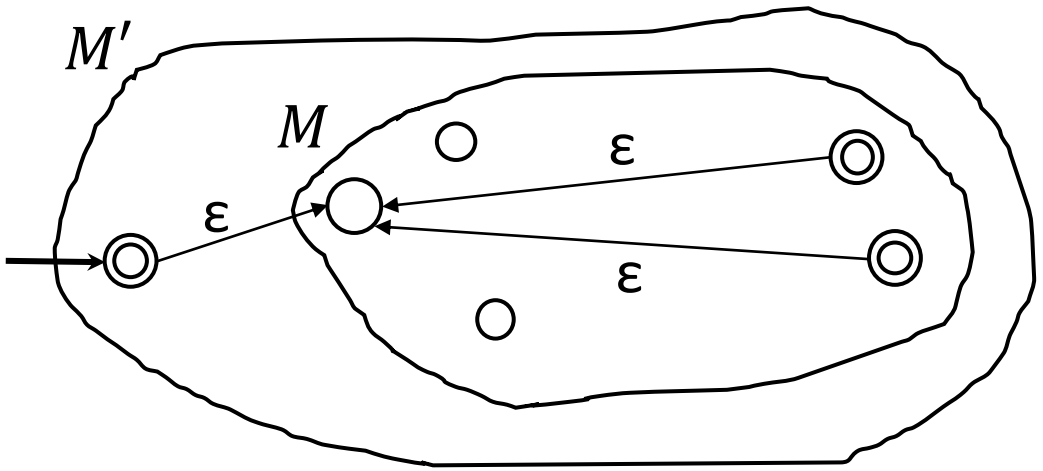
Question: Have we done yet?

Closure under $*$ (star)

Theorem: If A is a regular language, so is $A^* = \{x_1 x_2 \ldots x_k \mid \text{each } x_i \in A,\ k \geq 0\}$

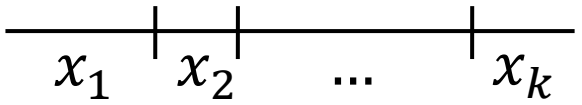
Proof sketch: Given DFA M recognizing A

Construct NFA M' recognizing A^*



Make sure M' accepts ε

M' should accept input w

if $w =$ 

where $k \geq 0$ and M accepts each x_i

Recall: Regular Expressions

Regular Expressions:

- Σ, members in Σ, $\emptyset$, ε [Atomic]
- By using $\cup$, $\circ$, $*$ [Composite]

Examples:

- $(0 \cup 1)^* = \Sigma^*$ gives all strings over Σ
- $\Sigma^* 1$ gives all strings that end with 1
- $\Sigma^* 11 \Sigma^*$ = all strings that contain 11 $= L(M_1)$

Kleene's Theorem:

Finite Automata and Regular Expressions are equivalent

(recognize the same class of languages, i.e., regular languages)

We now prove one direction: Regular Expressions can be converted to Finite Automata

Regular Expressions → NFA (= DFA)

Theorem: If R is a regular expression and $A = L(R)$, then A is regular

Proof: Convert R to equivalent NFA M:

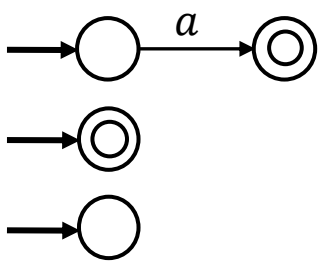
If R is <u>atomic:</u>

$R = a$ for $a \in \Sigma$

$R = \varepsilon$

$R = \emptyset$

Equivalent M is:



If R is <u>composite:</u>

$R = R_1 \cup R_2$

$R = R_1 \circ R_2$

$R = R_1^*$

} Use closure constructions

Example:

Convert $(\text{a} \cup \text{ab})^*$ to an equivalent NFA

a:

b:

ab:

a ∪ ab:

$(\text{a} \cup \text{ab})^*$:

Exercise!

Regular Expressions → NFA (= DFA)

Theorem: If R is a regular expression and $A = L(R)$, then A is regular

Proof: Convert R to equivalent NFA M:

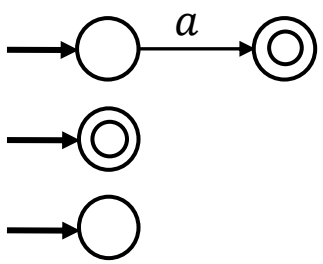
If R is <u>atomic:</u>

$R = a$ for $a \in \Sigma$

$R = \varepsilon$

$R = \emptyset$

Equivalent M is:



If R is <u>composite:</u>

$R = R_1 \cup R_2$

$R = R_1 \circ R_2$

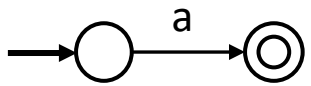
$R = R_1^*$

} Use closure constructions

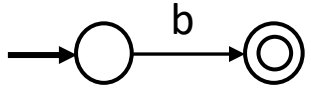
Example:

Convert $(\text{a} \cup \text{ab})^*$ to an equivalent NFA

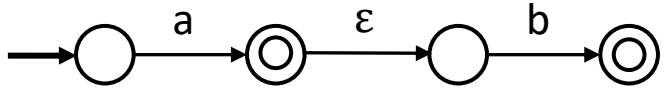
a:



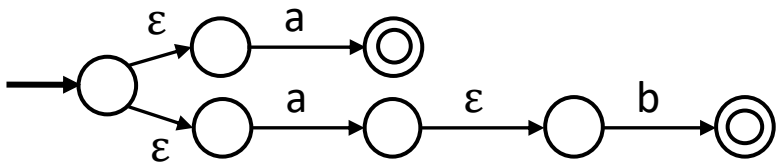
b:



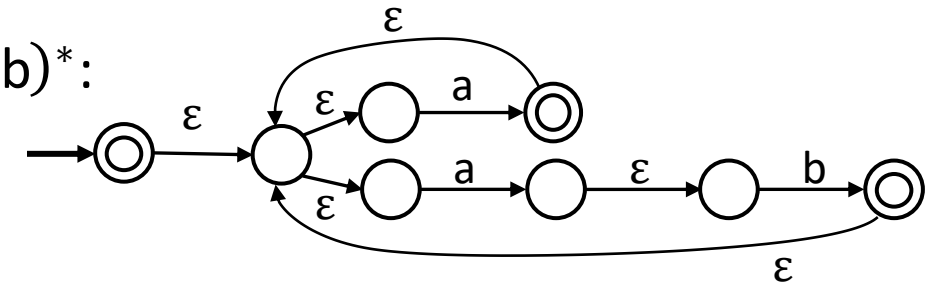
ab:



a ∪ ab:



$(\text{a} \cup \text{ab})^*$:



Quick review of today

1. Nondeterministic finite automata (NFA)
2. Proved: NFA and DFA are equivalent (both recognize regular languages)
3. Proved: The class of regular languages is closed under $\circ, *$
4. Convert regular expressions to NFA (DFA)

Next week: Convert DFA to regular expressions

Survey:

Which date is NOT a good time for you for Midterm Exam (4 pm – 5:20 pm)?

(a) Oct 19 (Mon) (b) Oct 21 (Wed)

(c) Oct 26 (Mon) (d) Oct 28 (Wed) EMNLP 2026

(e) Nov 2 (Mon) (f) Nov 4 (Wed) INFORMS 2026

(g) Nov 9 (Mon) (h) Nov 11 (Wed)

If you plan to take this course and one of the dates above is not good, let us know.

These slides are adapted from Michael Sipser, *18.404J Theory of Computation*, Fall 2020. Massachusetts Institute of Technology:

- Source: MIT OpenCourseWare https://ocw.mit.edu