18.404/6.840 Lecture 2

Last time: (Sipser §1.1)

- Finite automata, regular languages
- Regular operations $\cup, \circ, *$
- Regular expressions
- Closure under $\cup$

Today: (Sipser §1.2 – §1.3)

- Nondeterminism
- Closure under $\circ$ and $*$
- Regular expressions → finite automata

Goal: Show finite automata equivalent to regular expressions

Problem Sets

- 35% of overall grade

- Problems are hard! Leave time to think about them.

- Writeups need to be clear and understandable, handwritten ok.
  Level of detail in proofs comparable to lecture: focus on main ideas.
  Don't need to include minor details.

- Submit via gradescope (see Canvas) by 2:30pm Cambridge time.
  Late submission accepted (on gradescope) until 11:59pm following day:
    1 point (out of 10 points) per late problem penalty.
  After that solutions are posted so not accepted without S3 excuse.

- Optional problems:
    Don't count towards grade except for A+.
    Value to you (besides the challenge):
    Recommendations, employment (future grading, TA, UROP)

- Problem Set 1 is due in one week.

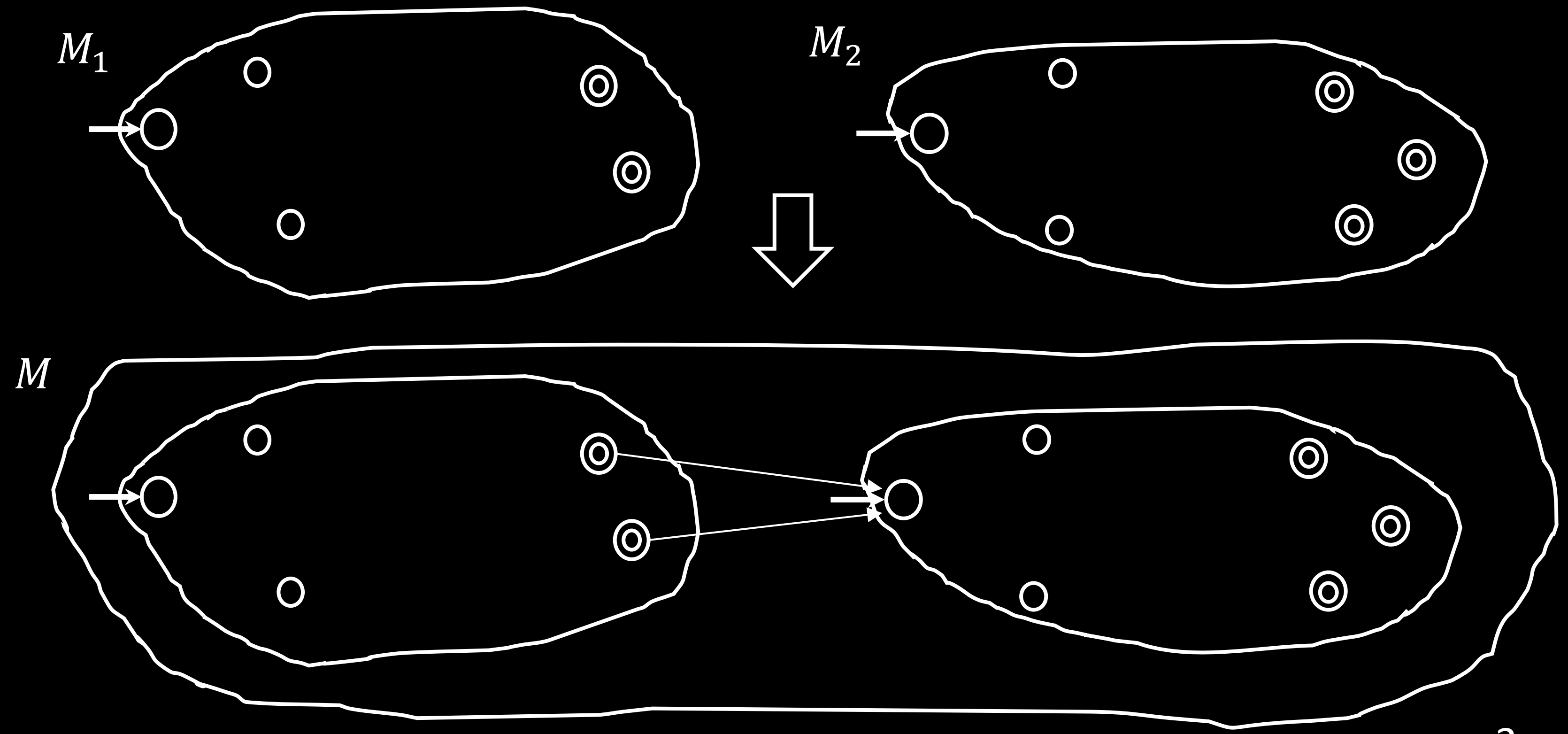
Closure Properties for Regular Languages

Theorem: If A_1, A_2 are regular languages, so is $A_1 A_2$ (closure under $\circ$)

Recall 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 $A_1 A_2$

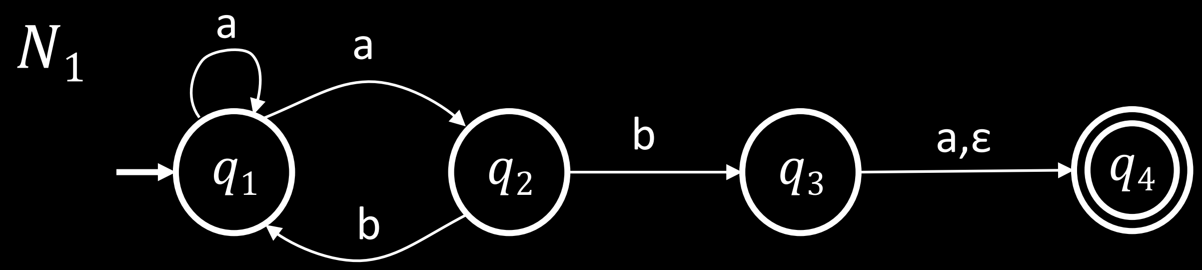


M should accept input w
if $w = xy$ where
M_1 accepts x and M_2 accepts y.

Doesn't work: Where to split w?

Hold off. Need new concept.

Nondeterministic Finite Automata



Check-in 2.1

What does N_1 do on input aab ?

(a) Accept

(b) Reject

(c) Both Accept and Reject

Acceptance overrules rejection

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 ◎ accept

Accept the input if some path leads to an accept. Check-in 2.1

Example inputs:

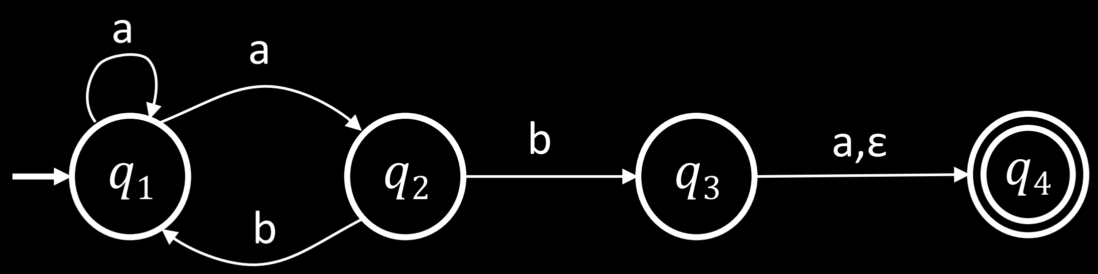
- ab accept
- aa reject
- aba accept
- abb reject

Nondeterminism doesn't correspond to a physical machine we can build. However, it is useful mathematically.

Once the machine had all possibilities have died off, there is no way for them to come back to life on any extensions

NFA – Formal Definition

N_1



Defn: A <u>nondeterministic finite automaton (NFA)</u>

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
- $\delta : Q \times \underbrace{\Sigma_\varepsilon}_{\Sigma \cup \{\varepsilon\}} \to \mathcal{P}(Q) = \{R \mid R \subseteq Q\}$

power set   Subsets of Q

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

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

Ways to think about nondeterminism:

<u>Computational:</u> Fork new parallel thread and accept if any thread leads to an accept state.

<u>Mathematical:</u> Tree with branches. Accept if any branch leads to an accept state.

<u>Magical:</u> Guess at each nondeterministic step which way to go. Machine always makes the right guess that leads to accepting, if possible.

Converting NFAs to DFAs

Theorem: If an NFA recognizes A then A is regular

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 <u>subset of possible states</u> in NFA M.

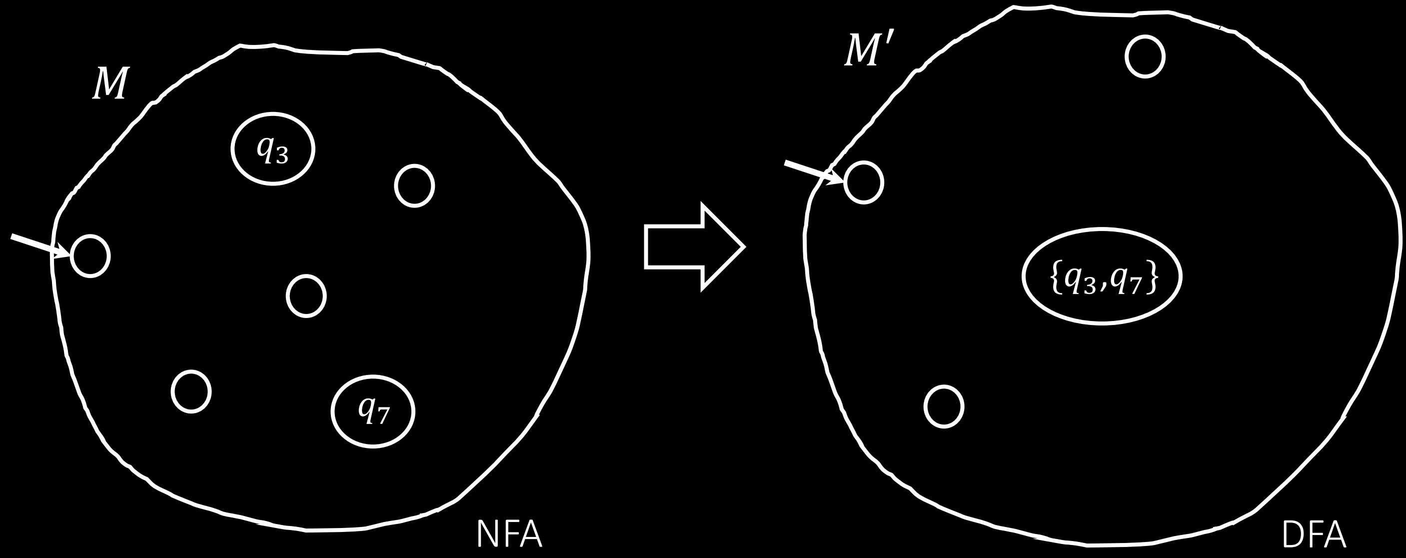
Check-in 2.2

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

(a) $2n$

(b) n^2

(c) 2^n



Construction of $\boldsymbol{M'}$:

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

$\delta'(\underbrace{R}_{R \in Q'}, a) =$

$q_0' = \{q_0\}$

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

Check-in 2.2

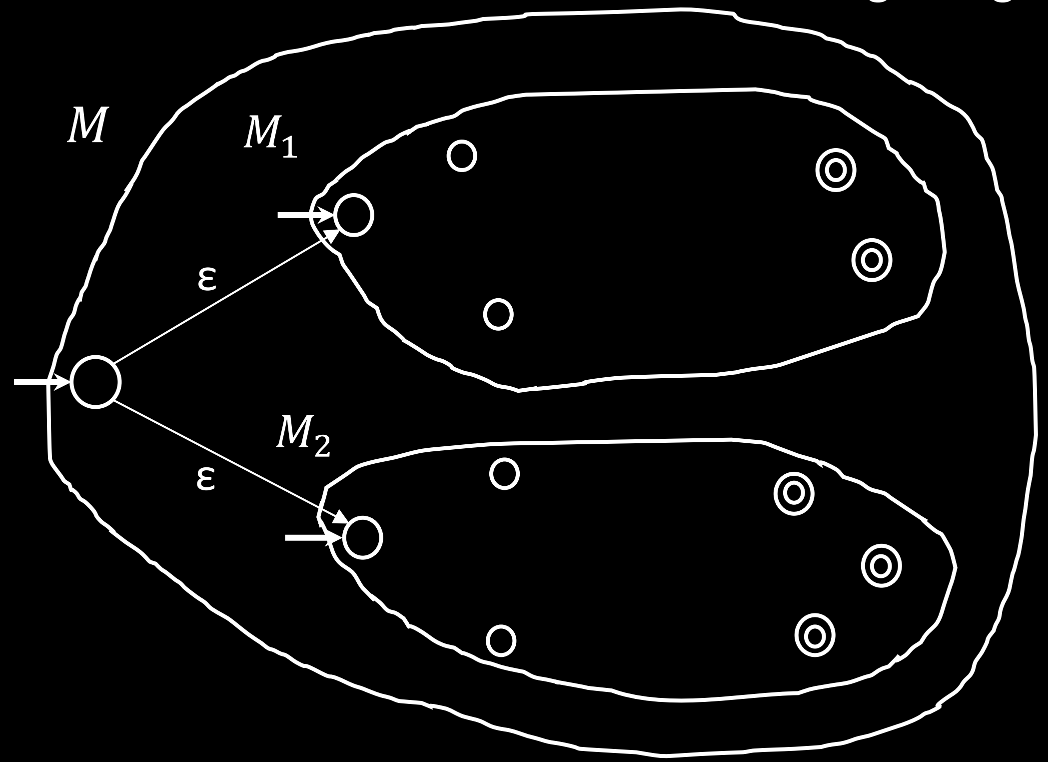
Return to Closure Properties

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 $A_1 \cup A_2$



Nondeterminism

parallelism

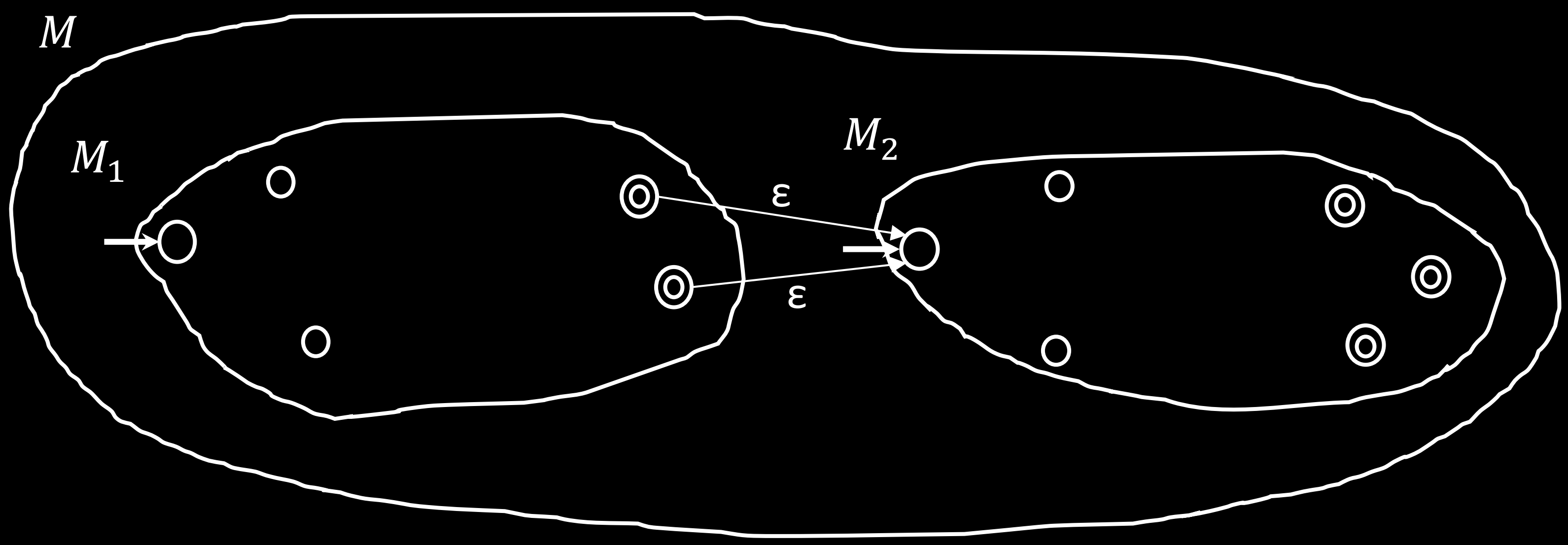
vs

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.

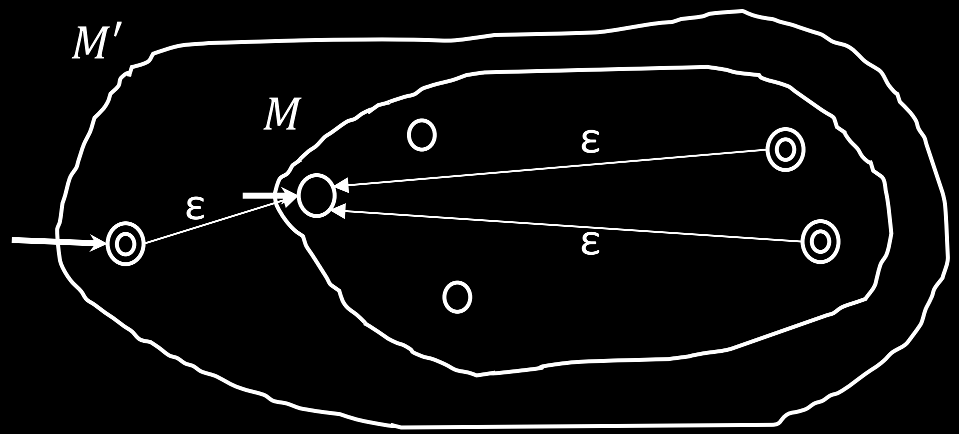
Nondeterministic M' has the option
to jump to M_2 when M_1 accepts.

Putting in an empty(epsilon) transitions going
from the accept state of M1 to the start of M2

Closure under $*$ (star)

Theorem: If A is a regular language, so is A^*

Proof sketch: Given DFA M recognizing A
Construct NFA M' recognizing A^*



Make sure M' accepts ε

Check-in 2.3

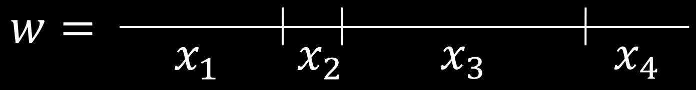
If M has n states, how many states does M' have by this construction?

(a) n
(b) $n+1$
(c) $2n$

M' should accept input w

if $w = x_1 x_2 \ldots x_k$

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



Check-in 2.3

Regular Expressions → NFA

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

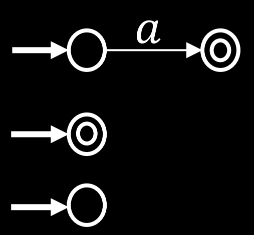
Proof: Convert R to equivalent NFA M:

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

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

$R = \varepsilon$

$R = \emptyset$



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

$R = R_1 \cup R_2$

$R = R_1 \circ R_2$

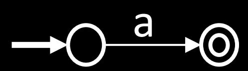
$R = R_1^*$

Use closure constructions

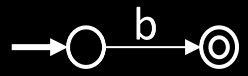
Example:

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

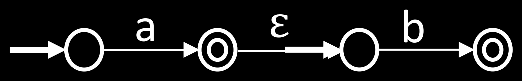
a:



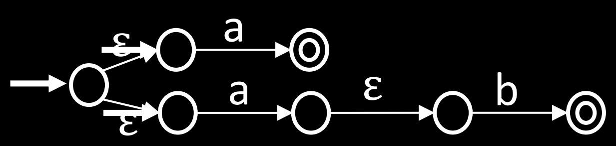
b:



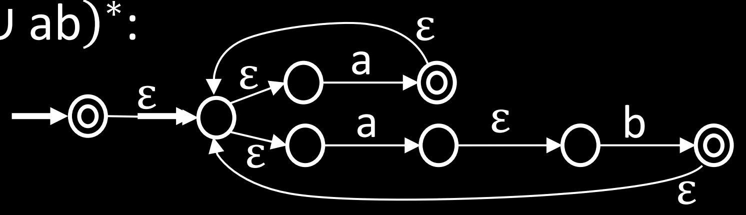
ab:



a ∪ ab:



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



Quick review of today

1. Nondeterministic finite automata (NFA)
2. Proved: NFA and DFA are equivalent in power
3. Proved: Class of regular languages is closed under $\circ, *$
4. Conversion of regular expressions to NFA

Check-in 2.4

Recitations start tomorrow online (same link as for lectures).
They are optional, unless you need more help.
You may attend any recitation(s).
Which do you think you'll attend? (you may check several)

(a) 10:00 (b) 11:00 (c) 12:00
(d) 1:00 (e) 2:00 (f) I prefer a different time (please post on piazza, but no promises)

Check-in 2.4