

CS420

NFA \leftrightarrow DFA

Monday, February 26, 2024

UMass Boston CS

A *nondeterministic finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta: Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.



A *finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the *states*,
2. Σ is a finite set called the *alphabet*,
3. $\delta: Q \times \Sigma \rightarrow Q$ is the *transition function*,
4. $q_0 \in Q$ is the *start state*, and
5. $F \subseteq Q$ is the *set of accept states*.

Announcements

- HW 3 out
 - Due Mon 3/4 12pm EST (noon)
- HW 1 grades returned
- Use Gradescope re-grade request for all questions / complaints!

Previously

Concatenation: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$

Is Concatenation Closed?

THEOREM

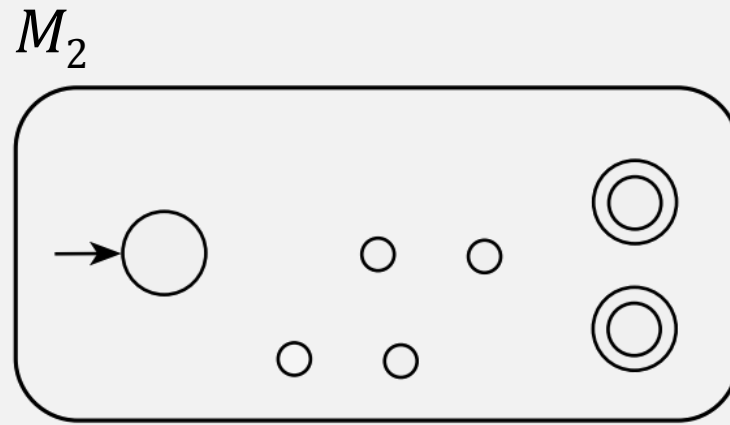
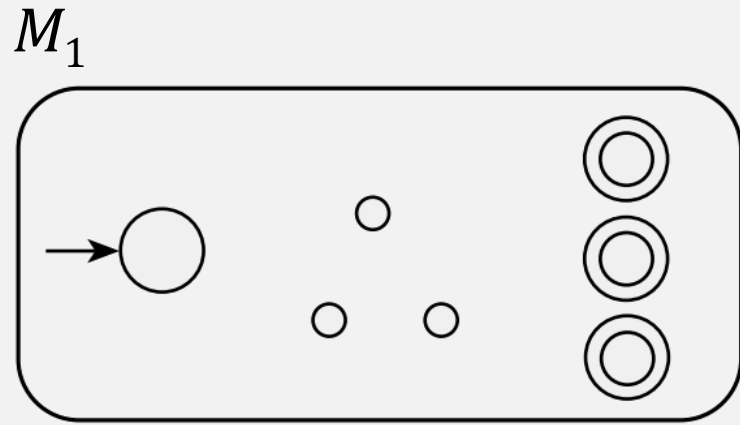
The class of regular languages is closed under the concatenation operation.

In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

Proof requires: Constructing new machine

- How does it know when to switch machines?
 - Can only read input once

Concatenation



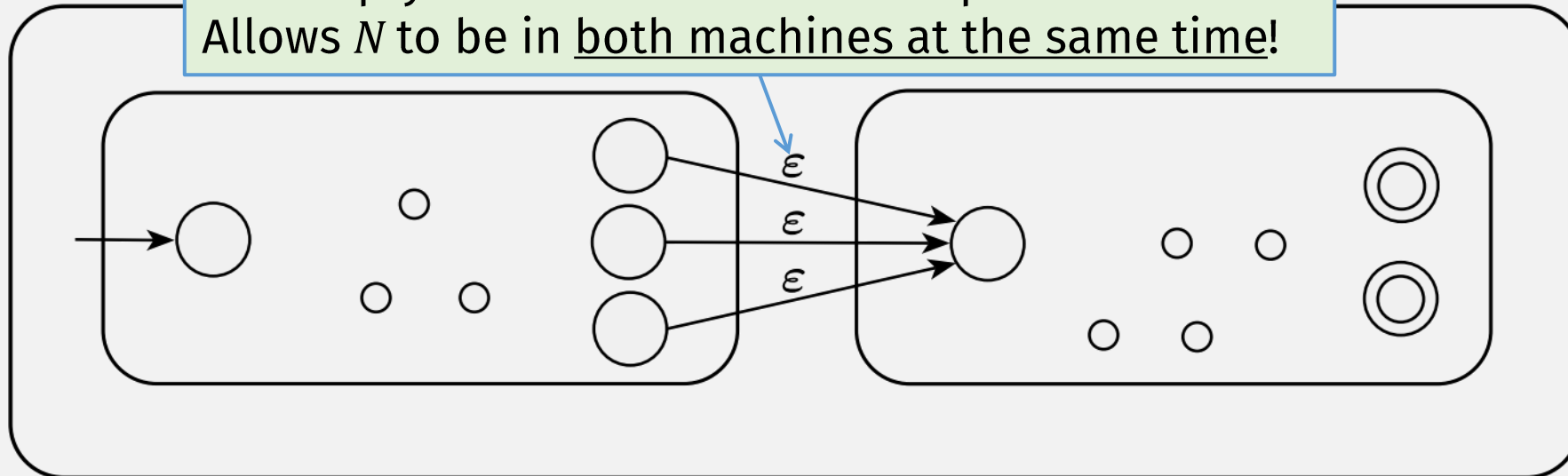
Let M_1 recognize A_1 , and M_2 recognize A_2 .

Want: Construction of N to recognize $A_1 \circ A_2$

- N is an **NFA!** It can:
- Keep checking 1st part with M_1
and
 - Move to M_2 to check 2nd part

N

ϵ = "empty transition" = reads no input
Allows N to be in both machines at the same time!



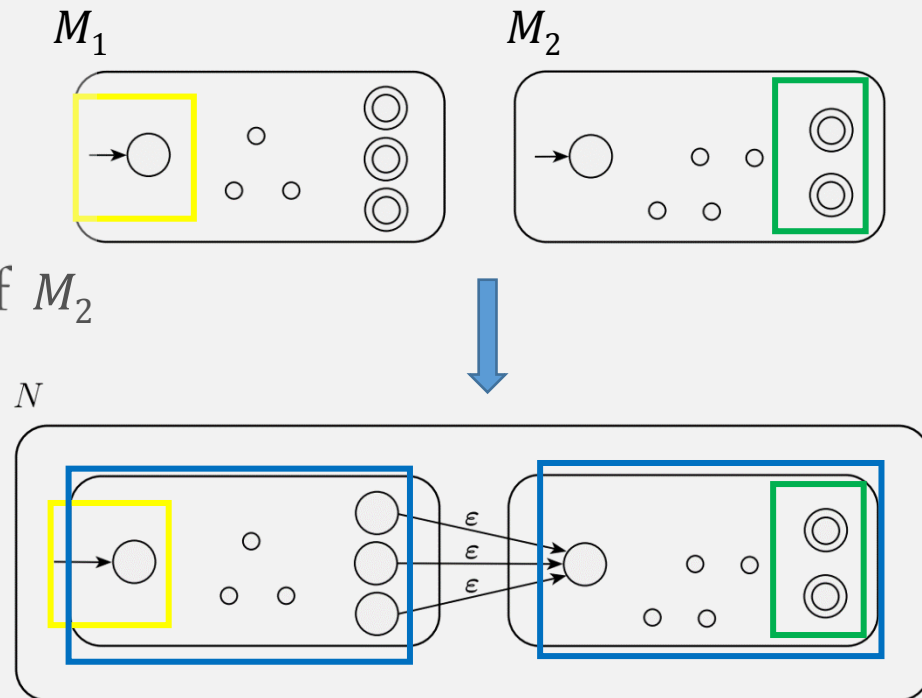
Concatenation is Closed for Regular Languages

PROOF (part of)

Let DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1
DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$
2. The state q_1 is the same as the start state of M_1
3. The accept states F_2 are the same as the accept states of M_2
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,



Concatenation is Closed for Regular Langs

Wait, is this true?

PROOF (part of)

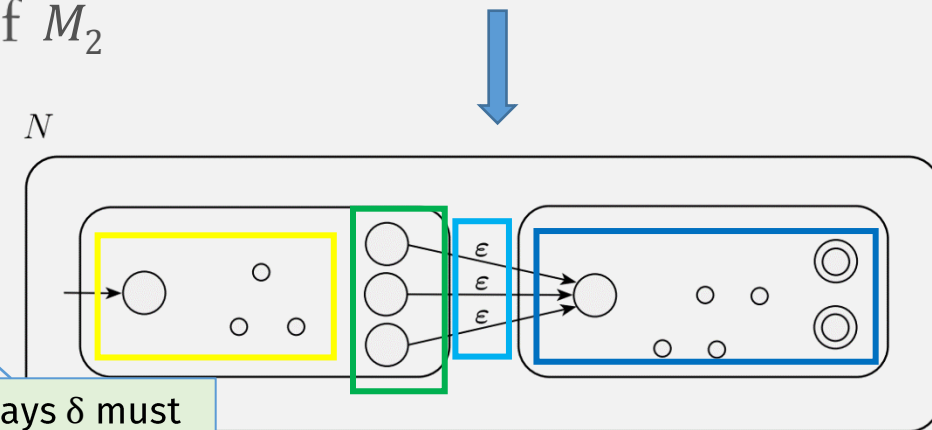
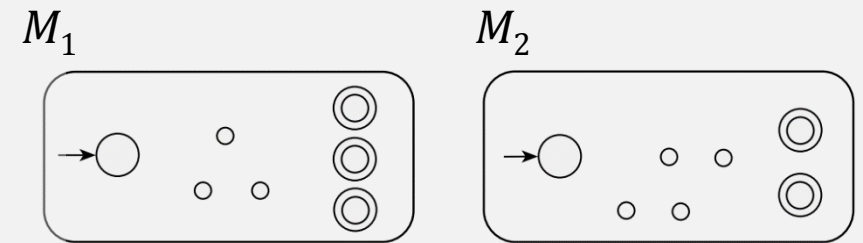
Let DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1
 DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$
2. The state q_1 is the same as the start state of M_1
3. The accept states F_2 are the same as the accept states of M_2
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,

$$\delta(q, a) = \begin{cases} \{\delta_1(q, a)\} & q \in Q_1 \text{ and } q \notin F_1 \\ \{\delta_1(q, a)\} & q \in F_1 \text{ and } a \neq \epsilon \\ ? & \{q_2\} \text{ } q \in F_1 \text{ and } a = \epsilon \\ \{\delta_2(q, a)\} & q \in Q_2. \end{cases}$$

And: $\delta(q, \epsilon) = \emptyset$, for $q \in Q, q \notin F_1$



NFA def says δ must map every state and ϵ to set of states

???

Is Concat Closed For Regular Langs?

Proof?

Statements

1. A_1 and A_2 are regular languages
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2
4. Construct **NFA** $M = (Q, \Sigma, \delta, q_0, F)$ ✓
5. M recognizes $A_1 \cup A_2$ ~~$A_1 \circ A_2$~~
6. ~~$A_1 \cup A_2$~~ $A_1 \circ A_2$ is a regular language
7. The class of regular languages is closed under concatenation operation.

In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

Justifications

1. Assumption
2. Def of Reg Lang (Coro)
3. Def of Reg Lang (Coro)
4. Def of **NFA**
5. See Examples Table
6. **???** Does NFA recognize reg langs?
7. From stmt #1 and #6

Q.E.D.?

Previously

A DFA's Language

- For DFA $M = (Q, \Sigma, \delta, q_0, F)$
- M **accepts** w if $\hat{\delta}(q_0, w) \in F$
- M **recognizes** language $\{w \mid M \text{ accepts } w\}$

Definition: A DFA's language is a **regular language**

An NFA's Language?

- For NFA $N = (Q, \Sigma, \delta, q_0, F)$

Intersection ...

... with accept states ...

- N *accepts* w if $\hat{\delta}(q_0, w) \cap F \neq \emptyset$
 - i.e., accept if final states contains at least one accept state

... is not empty set

- Language of $N = L(N) = \left\{ w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset \right\}$

Q: What kind of languages do NFAs recognize?

Concatenation Closed for Reg Langs?

- Combining DFAs to recognize concatenation of languages ...
 - ... produces an NFA
- So to prove concatenation is closed ...
 - ... we must prove that NFAs also recognize regular languages.

Specifically, we must prove:
NFAs \Leftrightarrow regular languages

“If and only if” Statements

$$X \Leftrightarrow Y = \text{“}X \text{ if and only if } Y\text{”} = X \text{ iff } Y = X \Leftrightarrow Y$$

Represents two statements:

1. \Rightarrow if X , then Y
 - “forward” direction
2. \Leftarrow if Y , then X
 - “reverse” direction

How to Prove an “iff” Statement

$$X \Leftrightarrow Y = \text{“}X \text{ if and only if } Y\text{”} = X \text{ iff } Y = X \Leftrightarrow Y$$

Proof has two (If-Then proof) parts:

1. \Rightarrow if X , then Y
 - “forward” direction
 - assume X , then use it to prove Y
2. \Leftarrow if Y , then X
 - “reverse” direction
 - assume Y , then use it to prove X

Proving NFAs Recognize Regular Langs

Theorem:

A language L is regular **if and only if** some NFA N recognizes L .

Proof: 2 parts

\Rightarrow If L is regular, then some NFA N recognizes it.

(Easier)

- We know: if L is **regular**, then a **DFA** exists that recognizes it.
- So to prove this part: Convert that DFA \rightarrow an equivalent NFA! (see HW 3)

\Leftarrow If an NFA N recognizes L , then L is regular.

Full Statements
&
Justifications?

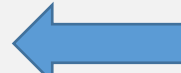
“equivalent” =
“recognizes the same language”

\Rightarrow If L is regular, then some NFA N recognizes it

Statements

1. L is a regular language
2. A DFA M recognizes L
3. Construct NFA $N = \text{convert}(M)$
4. DFA M is **equivalent** to NFA N
5. An NFA N recognizes L
6. If L is a regular language,
then some NFA N recognizes it

Justifications

1. Assumption
2. Def of Regular lang (Coro)
3. See hw \neq 3!
4. See Equiv. table! 
5. ???
6. By Stmts #1 and # 5

Assume the
"if" part ...

... use it to prove
"then" part

“Proving” Machine Equivalence (Table)

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

NFA $N = \text{convert}(M)$

$\hat{\delta}(q_0, w) \in F$ for some string w

Note:
extra column

String	M accepts?	N accepts?	N accepts? Justification
w	Yes	???	See justification #1
w'	No	???	See justification #2?
...			

If M accepts w ...

Then we know ...

There is some sequence of states: $r_1 \dots r_n$, where $r_i \in Q$ and

$$r_1 = q_0 \text{ and } r_n \in F$$

Then N accepts?/rejects? w because ...

Justification #1?

There is an accepting sequence of set of states in N ... for string w

“Proving” Machine Equivalence (Table)

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

NFA $N = \text{convert}(M)$

$\hat{\delta}(q_0, w) \in F$ for some string w

$\hat{\delta}(q_0, w') \notin F$ for some string w'

String	M accepts?	N accepts?	N accepts? Justification
w	Yes	???	See justification #1
w'	No	???	See justification #2?
...			

If M rejects w' ...

Then we know ...

Then N accepts?/rejects? w' because ...

Justification #2?

Proving NFAs Recognize Regular Langs

Theorem:

A language L is regular **if and only if** some NFA N recognizes L .

Proof:

☑ \Rightarrow If L is regular, then some NFA N recognizes it.

(Easier)

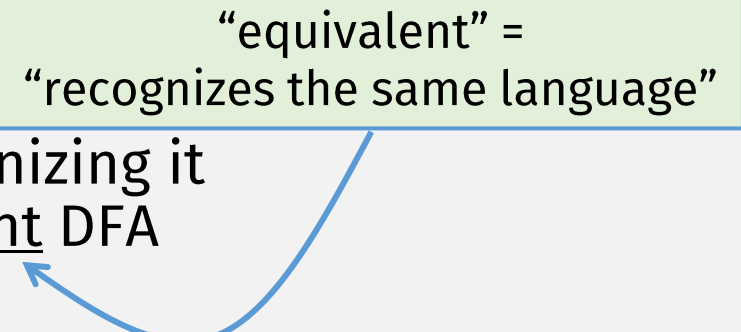
- We know: if L is **regular**, then a **DFA** exists that recognizes it.
- So to prove this part: Convert that DFA \rightarrow an equivalent NFA! (see HW 3)

\Leftarrow If an NFA N recognizes L , then L is regular.

(Harder)

- We know: for L to be **regular**, there must be a **DFA** recognizing it
- Proof Idea for this part: Convert given NFA $N \rightarrow$ an equivalent DFA

“equivalent” =
“recognizes the same language”



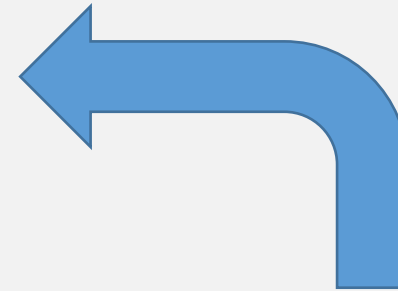
How to convert NFA→DFA?

A *finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the *states*,
2. Σ is a finite set called the *alphabet*,
3. $\delta: Q \times \Sigma \rightarrow Q$ is the *transition function*,
4. $q_0 \in Q$ is the *start state*, and
5. $F \subseteq Q$ is the *set of accept states*.

Proof idea:

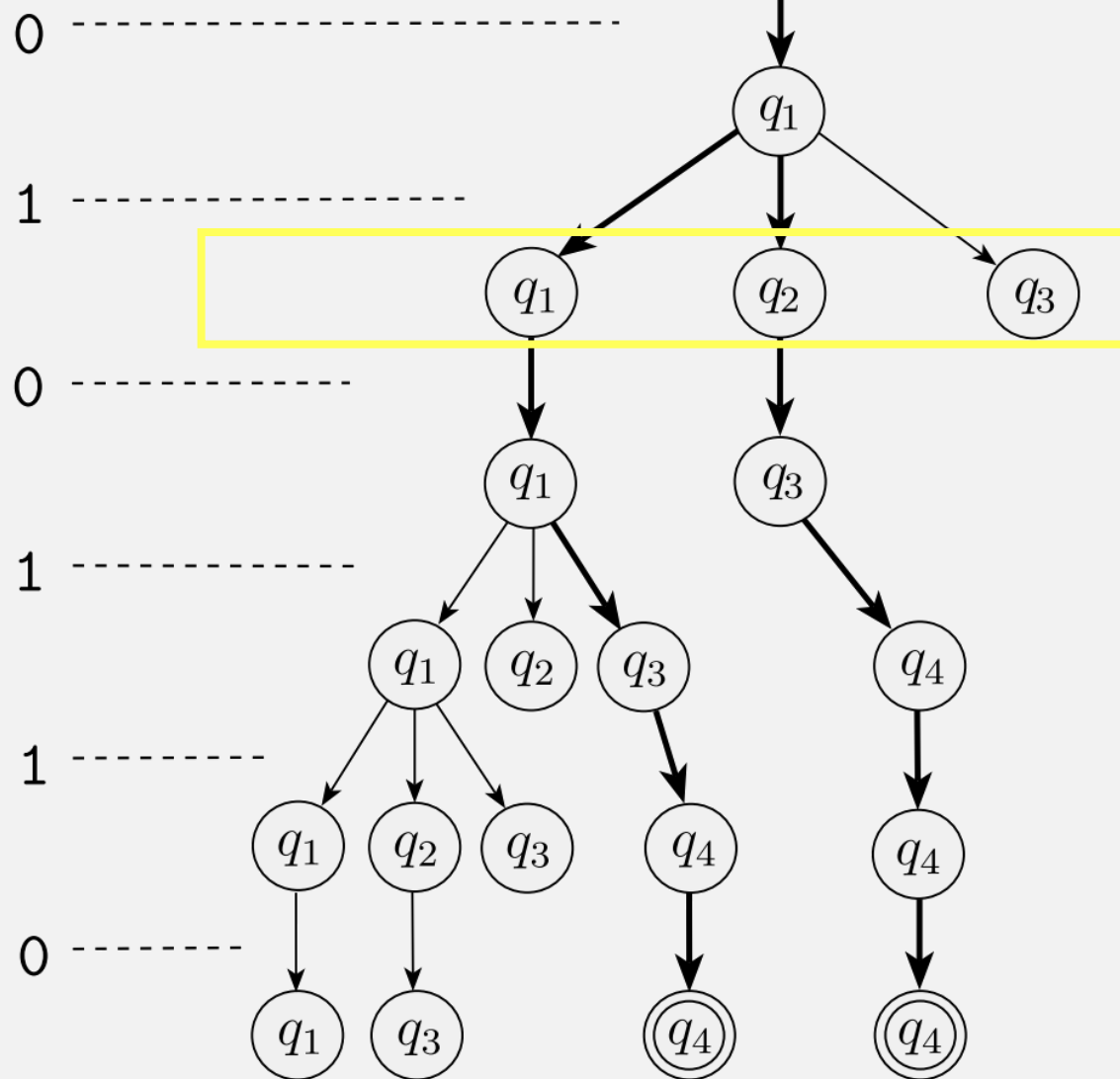
Let each “state” of the DFA
= set of states in the NFA



A *nondeterministic finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta: Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Symbol read



NFA computation can be in multiple states

DFA computation can only be in one state

So encode:
a set of NFA states
as one DFA state

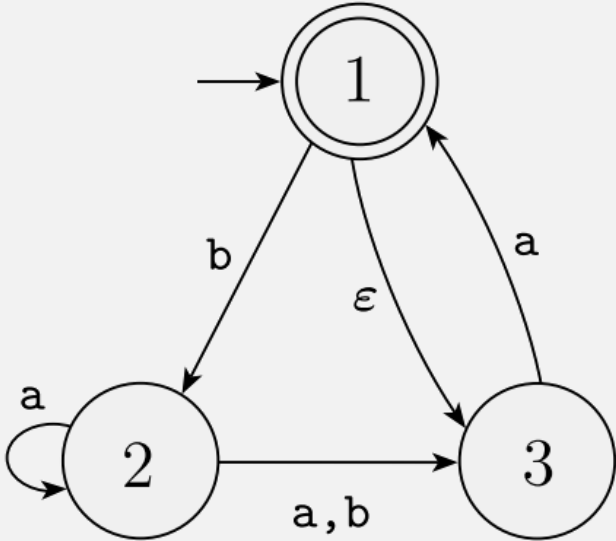
This is similar to the proof strategy from
"Closure of union" where:
a state = a pair of states

Convert NFA→DFA, Formally

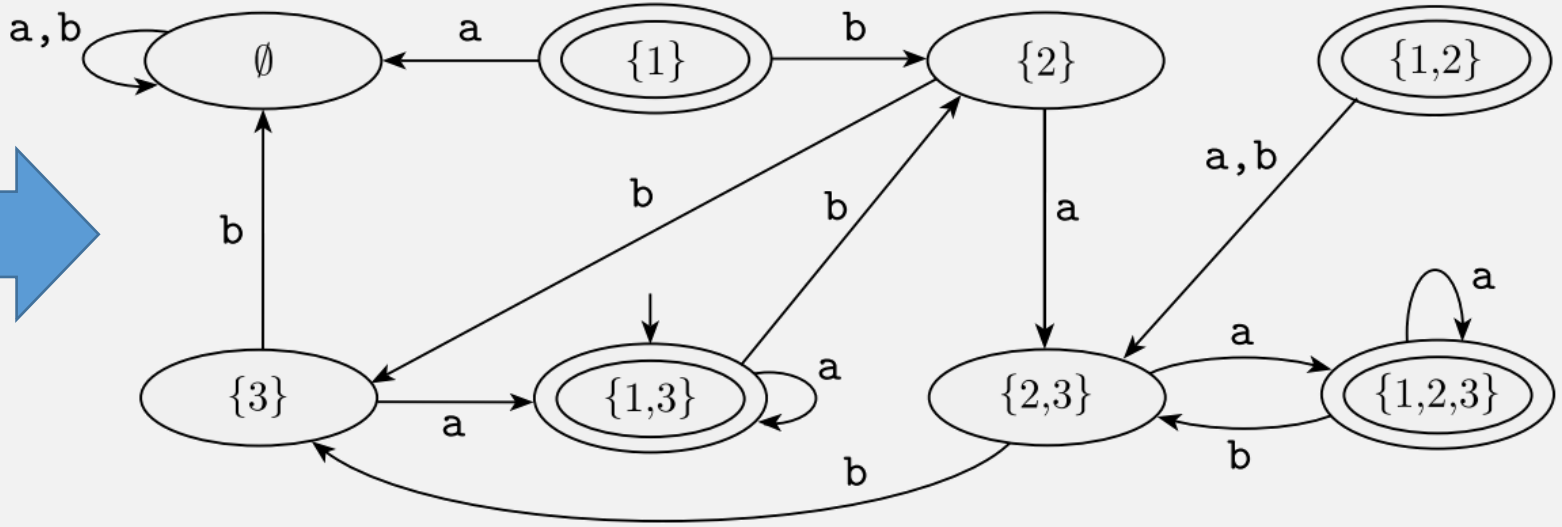
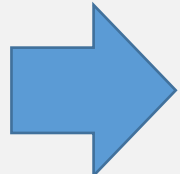
- Let NFA $N = (Q, \Sigma, \delta, q_0, F)$
- An equivalent DFA M has states $Q' = \mathcal{P}(Q)$ (power set of Q)

Example:

- Let NFA $N_4 = (Q, \Sigma, \delta, q_0, F)$
- An equivalent DFA D has states $= \mathcal{P}(Q)$ (power set of Q)



The NFA N_4



A DFA D that is equivalent to the NFA N_4

NFA → DFA

Have: NFA $N = (Q_{\text{NFA}}, \Sigma, \delta_{\text{NFA}}, q_{0\text{NFA}}, F_{\text{NFA}})$

Want: DFA $D = (Q_{\text{DFA}}, \Sigma, \delta_{\text{DFA}}, q_{0\text{DFA}}, F_{\text{DFA}})$

1. $Q_{\text{DFA}} = \mathcal{P}(Q_{\text{NFA}})$ A DFA state = a set of NFA states

$qs = \text{DFA state} = \text{set of NFA states}$

2. For $qs \in Q_{\text{DFA}}$ and $a \in \Sigma$

• $\delta_{\text{DFA}}(qs, a) = \bigcup_{q \in qs} \delta_{\text{NFA}}(q, a)$ A DFA step = an NFA step for all states in the set

3. $q_{0\text{DFA}} = \{q_{0\text{NFA}}\}$

4. $F_{\text{DFA}} = \{qs \in Q_{\text{DFA}} \mid qs \text{ contains accept state of } N\}$

Flashback: Adding Empty Transitions

- Define the set $\varepsilon\text{-REACHABLE}(q)$
 - ... to be all states reachable from q via zero or more empty transitions

(Defined recursively)

- Base case: $q \in \varepsilon\text{-REACHABLE}(q)$

- Recursive case:

A state is in the reachable set if ...

$$\varepsilon\text{-REACHABLE}(q) = \{r \mid p \in \varepsilon\text{-REACHABLE}(q) \text{ and } r \in \delta(p, \varepsilon)\}$$

... there is an empty transition to it from another state in the reachable set

NFA → DFA

Have: NFA $N = (Q_{NFA}, \Sigma, \delta_{NFA}, q_{0NFA}, F_{NFA})$

Want: DFA $D = (Q_{DFA}, \Sigma, \delta_{DFA}, q_{0DFA}, F_{DFA})$

Almost the same, except ...

1. $Q_{DFA} = \mathcal{P}(Q_{NFA})$

2. For $qs \in Q_{DFA}$ and $a \in \Sigma$
• $\delta_{DFA}(qs, a) = \bigcup_{q \in qs} \delta_{NFA}(q, a)$

$\bigcup_{s \in S} \epsilon\text{-REACHABLE}(s)$

3. $q_{0DFA} = \{q_{0NFA}\} \epsilon\text{-REACHABLE}(q_{0NFA})$

4. $F_{DFA} = \{ qs \in Q_{DFA} \mid qs \text{ contains accept state of } N \}$

Proving NFAs Recognize Regular Langs

Theorem:

A language L is regular **if and only if** some NFA N recognizes L .

Proof:

⇒ If L is regular, then some NFA N recognizes it.

(Easier)

- We know: if L is **regular**, then a **DFA** exists that recognizes it.
- So to prove this part: Convert that DFA → an equivalent NFA! (see HW 3)

⇐ If an NFA N recognizes L , then L is regular.

(Harder)

- We know: for L to be **regular**, there must be a **DFA** recognizing it
- Proof Idea for this part: Convert given NFA N → an equivalent DFA ...
... using our NFA to DFA algorithm! ■

Statements
&
Justifications?

Examples table?



Concatenation is Closed for Regular Langs

PROOF

Let DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1
 DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2

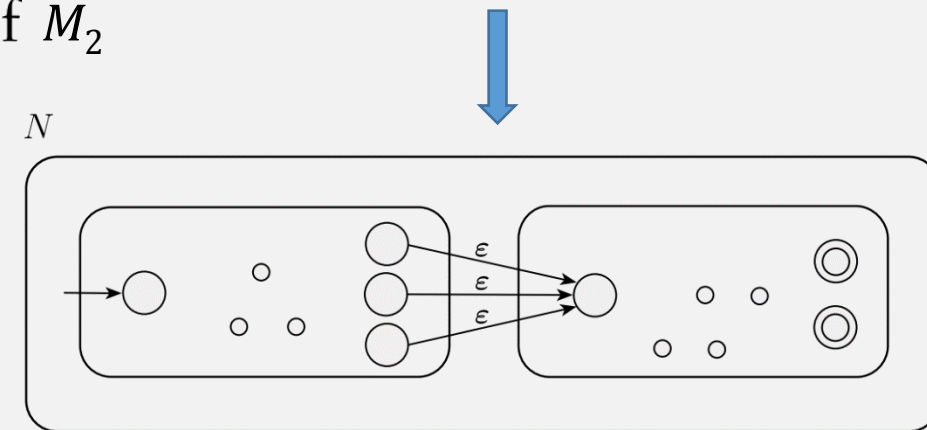
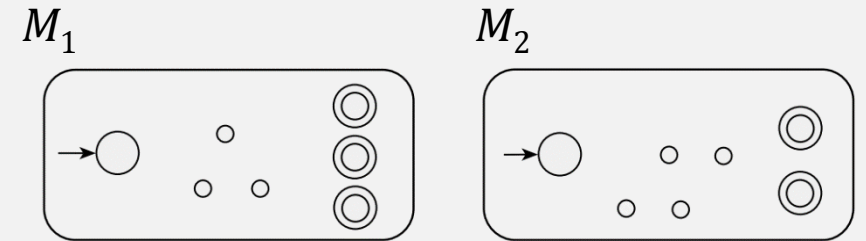
Wait, is this true?

If a language has an NFA recognizing it, then it is a **regular** language

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$
2. The state q_1 is the same as the start state of M_1
3. The accept states F_2 are the same as the accept states of M_2
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,

$$\delta(q, a) = \begin{cases} \{\delta_1(q, a)\} & q \in Q_1 \text{ and } q \notin F_1 \\ \{\delta_1(q, a)\} & q \in F_1 \text{ and } a \neq \epsilon \\ \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\ \{\delta_2(q, a)\} & q \in Q_2. \end{cases}$$



And: $\delta(q, \epsilon) = \emptyset$, for $q \in Q, q \notin F_1$ ~~???~~

New possible proof strategy!

Concat Closed for Reg Langs: Use **NFAs** Only

PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 , and

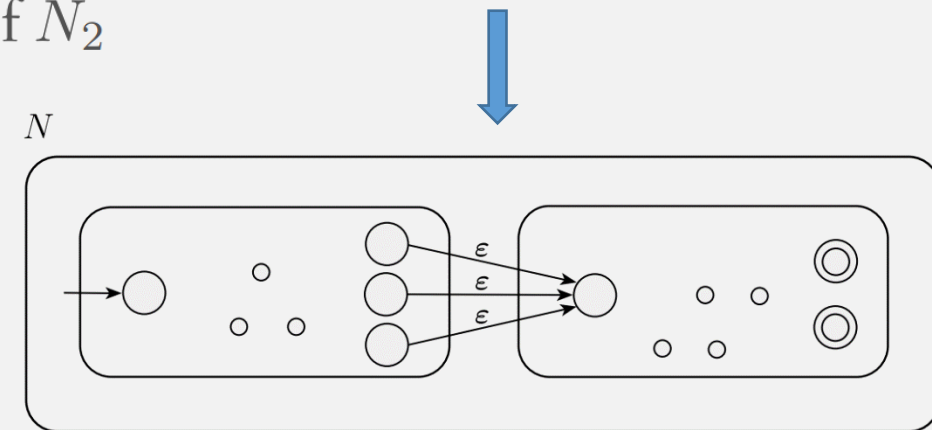
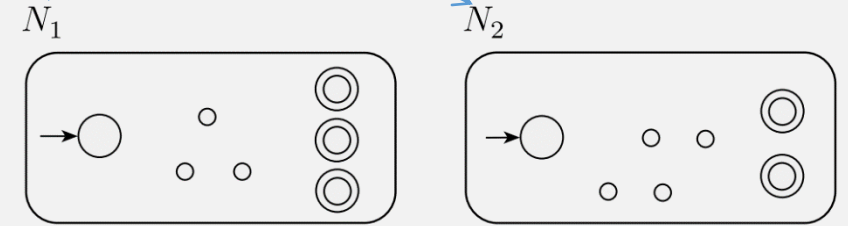
NFAs $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2 .

If language is regular,
then it has an **NFA** recognizing it ...

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$
2. The state q_1 is the same as the start state of N_1
3. The accept states F_2 are the same as the accept states of N_2
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a) & q \in F_1 \text{ and } a \neq \epsilon \\ ? \quad \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q, a) & q \in Q_2. \end{cases}$$



Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

Flashback: Union is Closed For Regular Langs

THEOREM

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof:

- How do we prove that a language is regular?
 - Create a DFA or **NFA** recognizing it!
- Combine the machines recognizing A_1 and A_2
 - Should we create a DFA **or** **NFA**?

Flashback: Union is Closed For Regular Langs

Proof

- Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1 ,
 $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2 ,

- Construct: a new machine $M = (Q, \Sigma, \delta, q_0, F)$ using M_1 and M_2

- states of M : $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$
 This set is the *Cartesian product* of sets Q_1 and Q_2

State in $M =$
 M_1 state +
 M_2 state

- M transition fn: $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

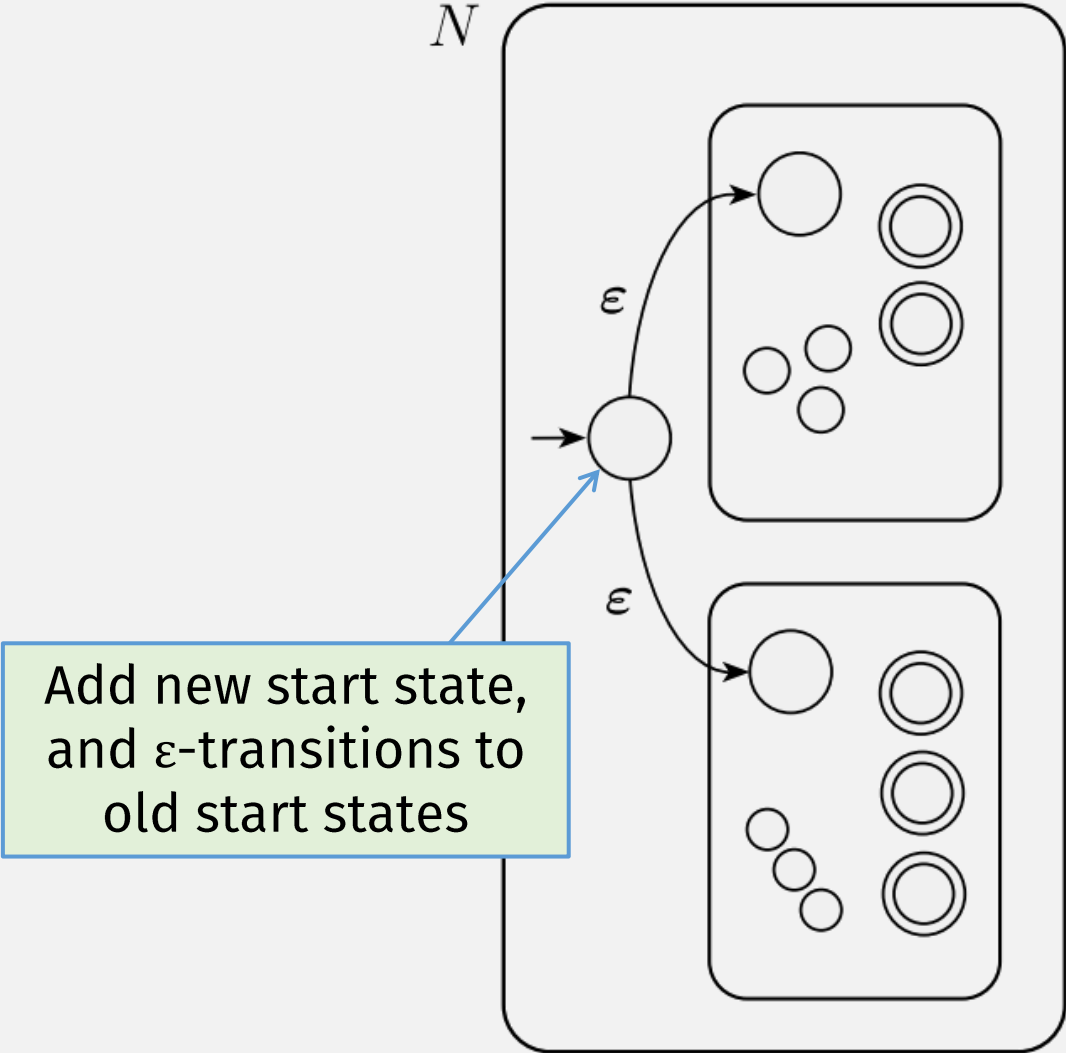
M step =
 a step in M_1 + a step in M_2

- M start state: (q_1, q_2)

Accept if either M_1 or M_2 accept

- M accept states: $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$.

Union is Closed for Regular Languages



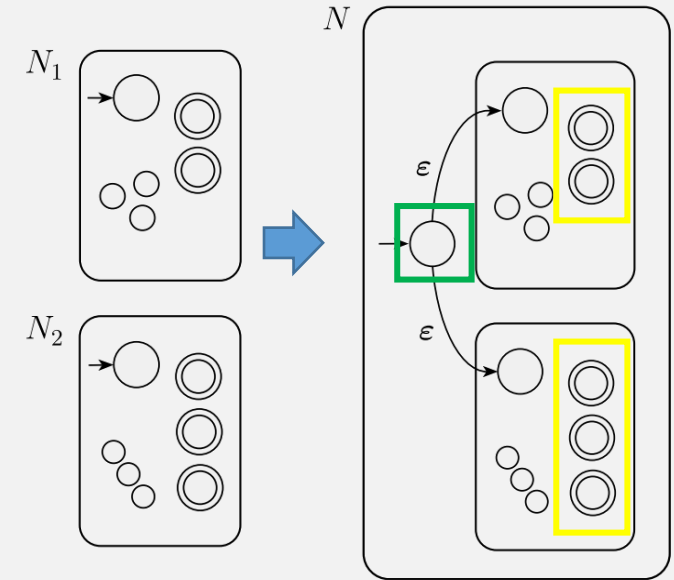
Union is Closed for Regular Languages

PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 , and
 $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2 .

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
2. The state q_0 is the start state of N .
3. The set of accept states $F = F_1 \cup F_2$.



Union is Closed for Regular Languages

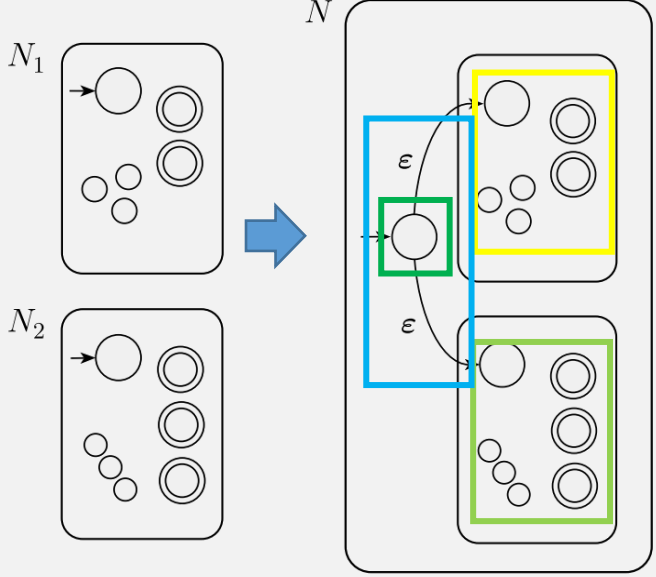
PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 , and
 $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2 .

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
2. The state q_0 is the start state of N .
3. The set of accept states $F = F_1 \cup F_2$.
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$



Don't forget Statements and Justifications!

List of Closed Ops for Reg Langs (so far)

• Union

• Concatentation

• Kleene Star (repetition) ?

Star: $A^* = \{x_1x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

Kleene Star Example

Let the alphabet Σ be the standard 26 letters $\{a, b, \dots, z\}$.

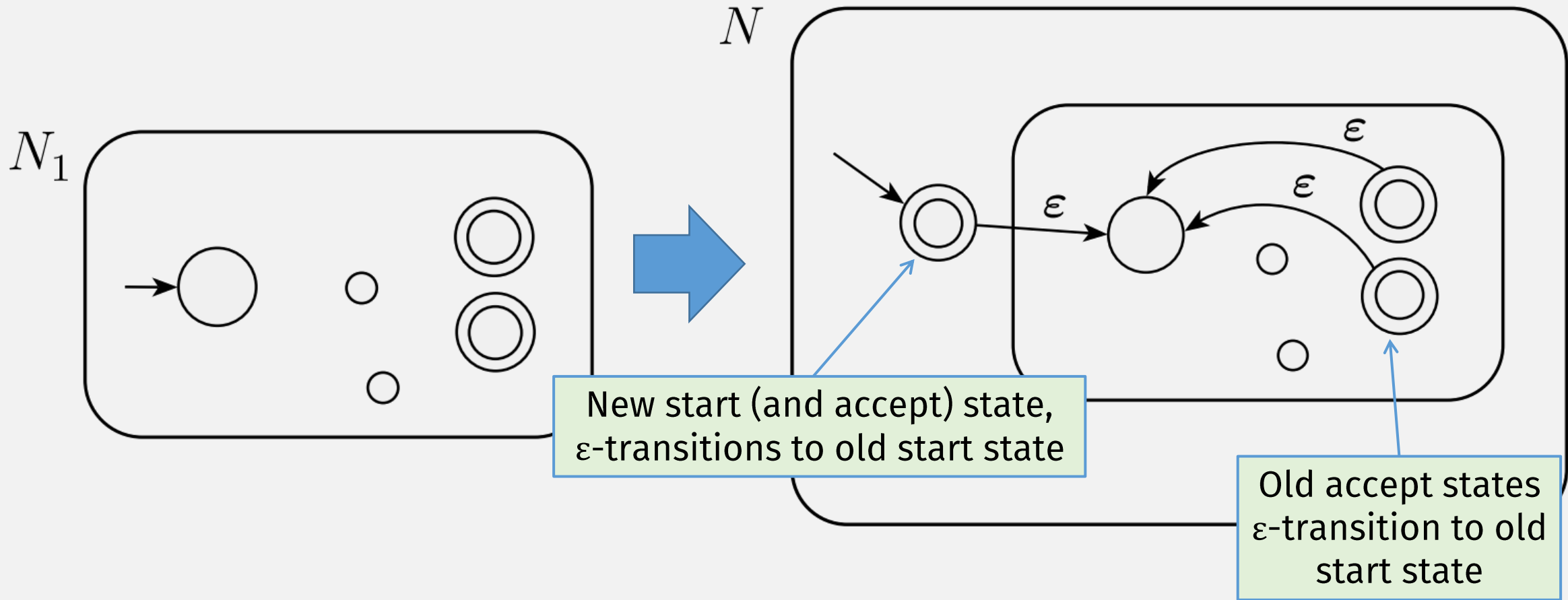
If $A = \{\text{good}, \text{bad}\}$

$A^* = \{\epsilon, \text{good}, \text{bad}, \text{goodgood}, \text{goodbad}, \text{badgood}, \text{badbad},$
 $\text{goodgoodgood}, \text{goodgoodbad}, \text{goodbadgood}, \text{goodbadbad}, \dots\}$

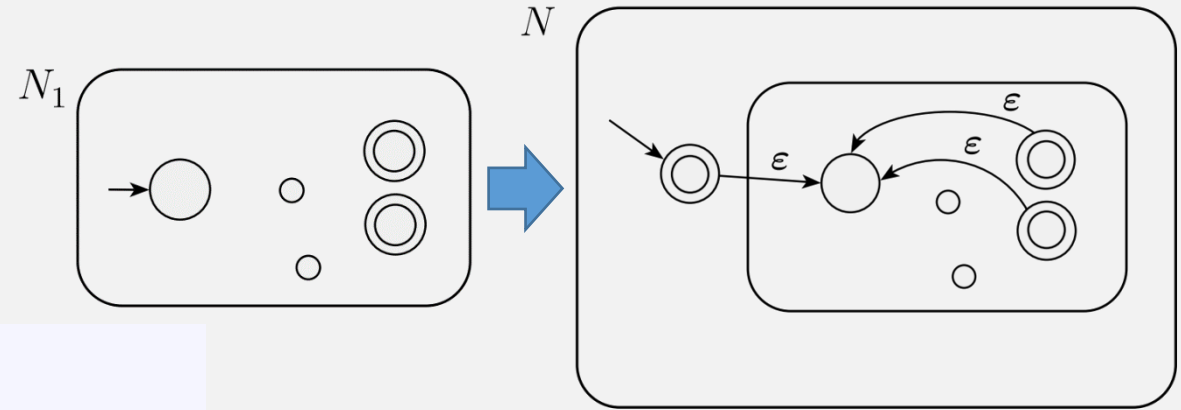
Note: repeat zero or more times

(this is an infinite language!)

Kleene Star



Kleene Star is Closed for Regular Languages



THEOREM

The class of regular languages is closed under the star operation.

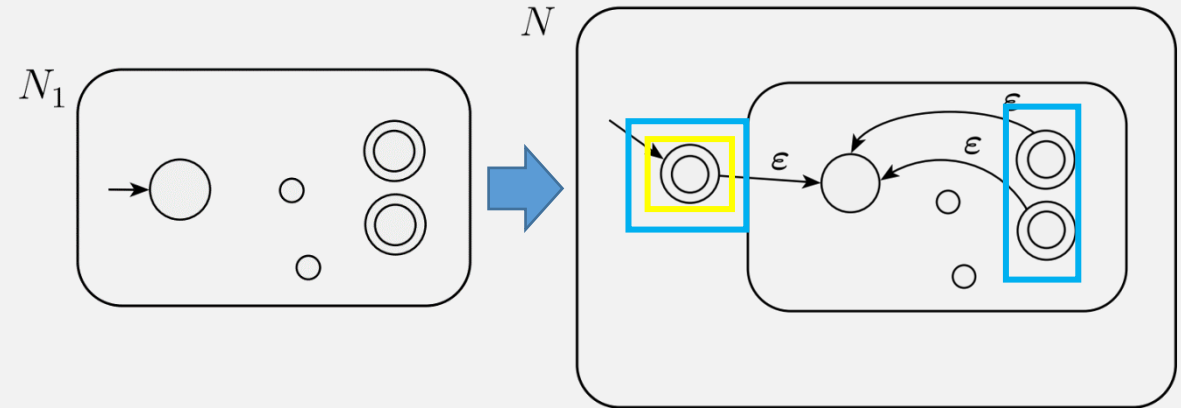
Kleene Star is Closed for Regular Languages

(part of)

PROOF Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 .
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^* .

1. $Q = \{q_0\} \cup Q_1$
2. The state q_0 is the new start state.
3. $F = \{q_0\} \cup F_1$

Kleene star of a language must accept the empty string!



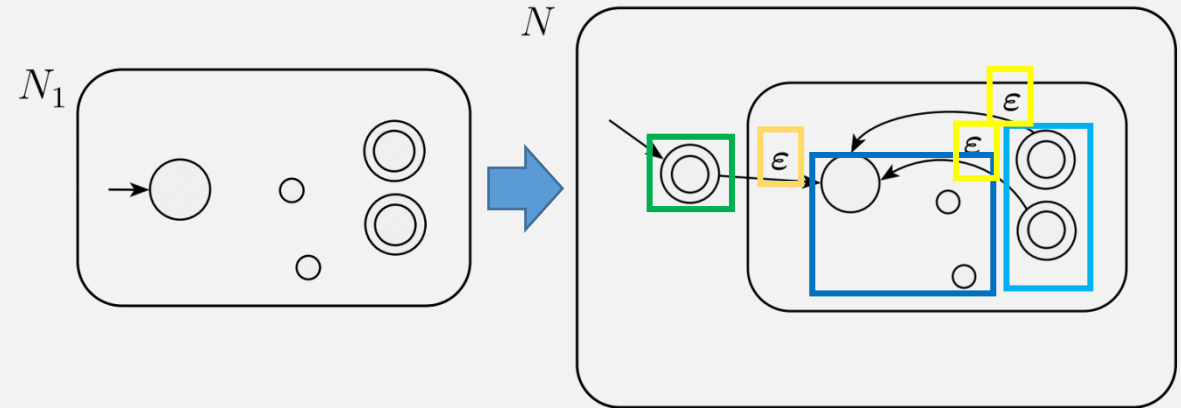
Kleene Star is Closed for Regular Langs

(part of)

PROOF Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 .
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^* .

1. $Q = \{q_0\} \cup Q_1$
2. The state q_0 is the new start state.
3. $F = \{q_0\} \cup F_1$
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_1\} & q \in F_1 \text{ and } a = \epsilon \\ \{q_1\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon. \end{cases}$$



Next Time: Why These Closed Operations?

- Union
- Concat
- Kleene star

All regular languages can be constructed from:

- single-char strings, and
- these three combining operations!

Submit in-class work 2/26

On gradescope