Previously

Definition of Regular Language

If a **DFA** recognizes a language L, then L is a **regular language**

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 $\{w | \ M \text{ accepts } w\}$

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 $\hat{\delta}(q_0, w) \cap F \neq \emptyset$

... is not empty set

  - i.e., accept if final states contains <u>at least one</u> accept state

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

<u>Q</u>: 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 ⇔ regular languages

“If and only if” Statements

$X \Leftrightarrow Y$ = “X if and only if Y” = X iff Y = X <=> Y

Represents <u>two</u> statements:

1. $\Rightarrow$ if X, then Y
   - “**forward**” direction

2. $\Leftarrow$ if Y, then X
   - “**reverse**” direction

How to Prove an "iff" Statement

$X \Leftrightarrow Y$ = "X if and only if Y" = X iff Y = X <=> Y

Proof has <u>two</u> (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

⇒ 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 4)

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

Full Statements & Justifications?

"equivalent" = "recognizes the same language"

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

Statements	**Justifications**
1. L is a regular language	1. Assumption
2. A DFA M recognizes L	2. Def of Regular lang (Coro)
3. Construct NFA N = $\text{CONVERT}_{\text{DFA-NFA}}(M)$	3. See hw 4!
4. DFA M is **equivalent** to NFA N	4. See Equiv. table!
5. An NFA N recognizes L	5. ???
6. If L is a regular language, then some NFA N recognizes it	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 $N =$ $\text{CONVERT}_{\text{DFA-NFA}}(M)$

$\hat{\delta}(q_0, w) \in F$ 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: $r_1 \ldots r_n$, where $r_i \in Q$ and

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

Then N accepts?/rejects? w because …

Exercise left for HW
Show that you know how an NFA computes

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 $N = \text{CONVERT}_{\text{DFA-NFA}}(M)$

$\hat{\delta}(q_0, w) \in F$ 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 …

Exercise left for HW

Show that you know how an NFA computes

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:

☑ ⇒ 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 4)

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

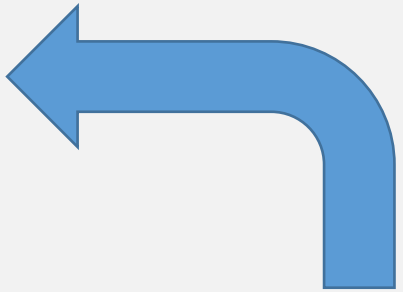
- We know: for L to be **regular**, there must be a **DFA** recognizing it
- Proof Idea for this part: Convert given NFA N → an equivalent DFA

"equivalent" =
"recognizes the same language"

How to convert **NFA→DFA**?

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

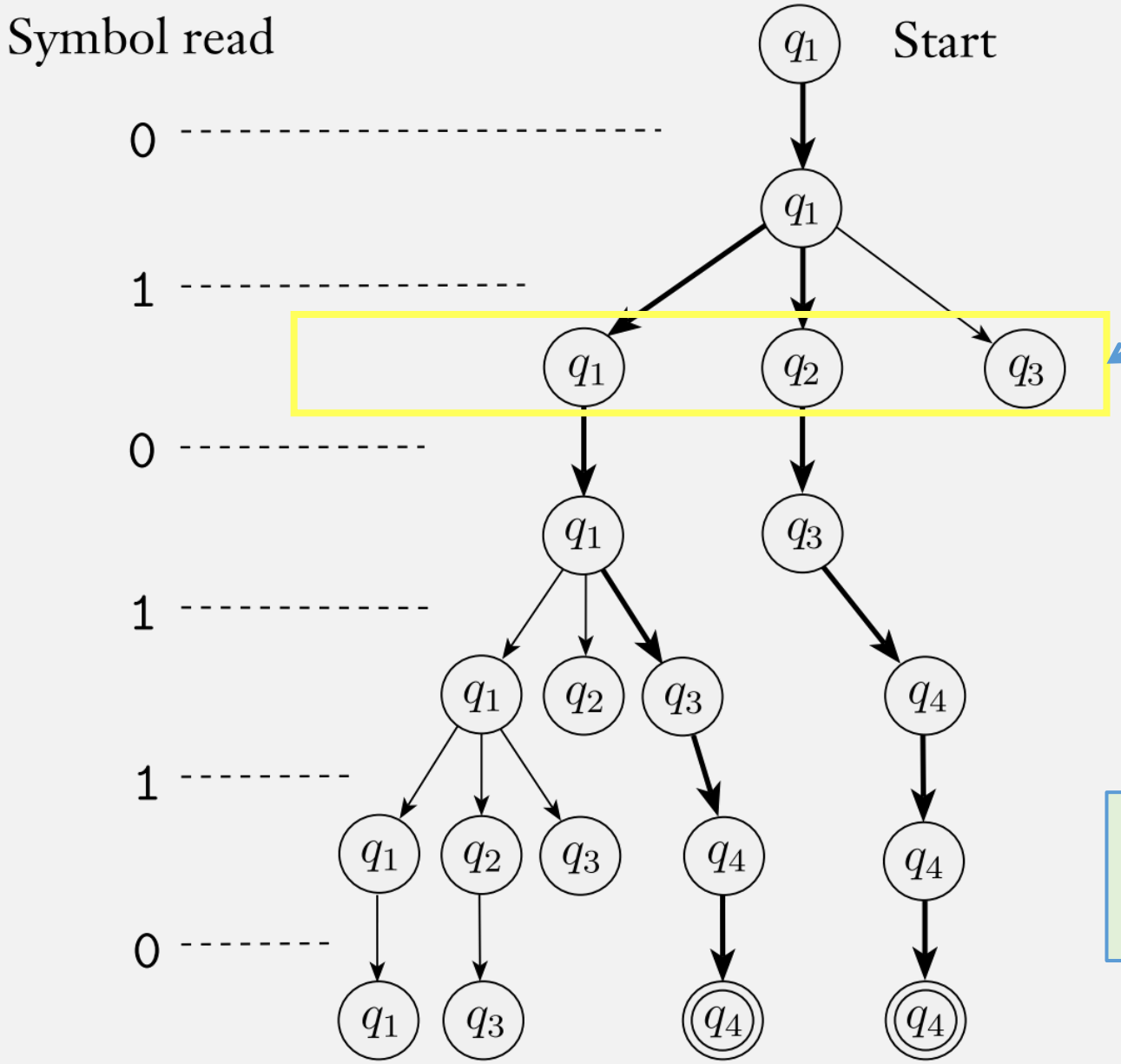
1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.



A ***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. $\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Proof idea:
Let each "state" of the DFA
= set of states in the NFA



NFA computation can be in <u>multiple</u> states

DFA computation can only be in <u>one</u> state

So **encode:**
a <u>set of NFA states</u>
as <u>one DFA state</u>

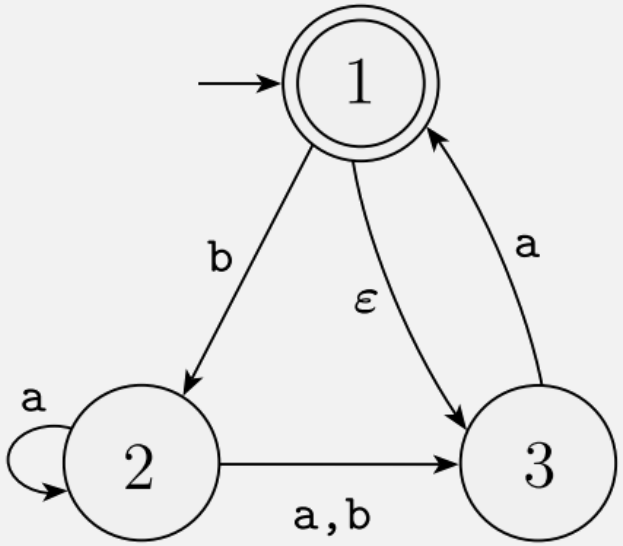
This is similar to the proof strategy from **"Closure of union"** where:
a state = a pair of states

Convert **NFA→DFA**, Formally

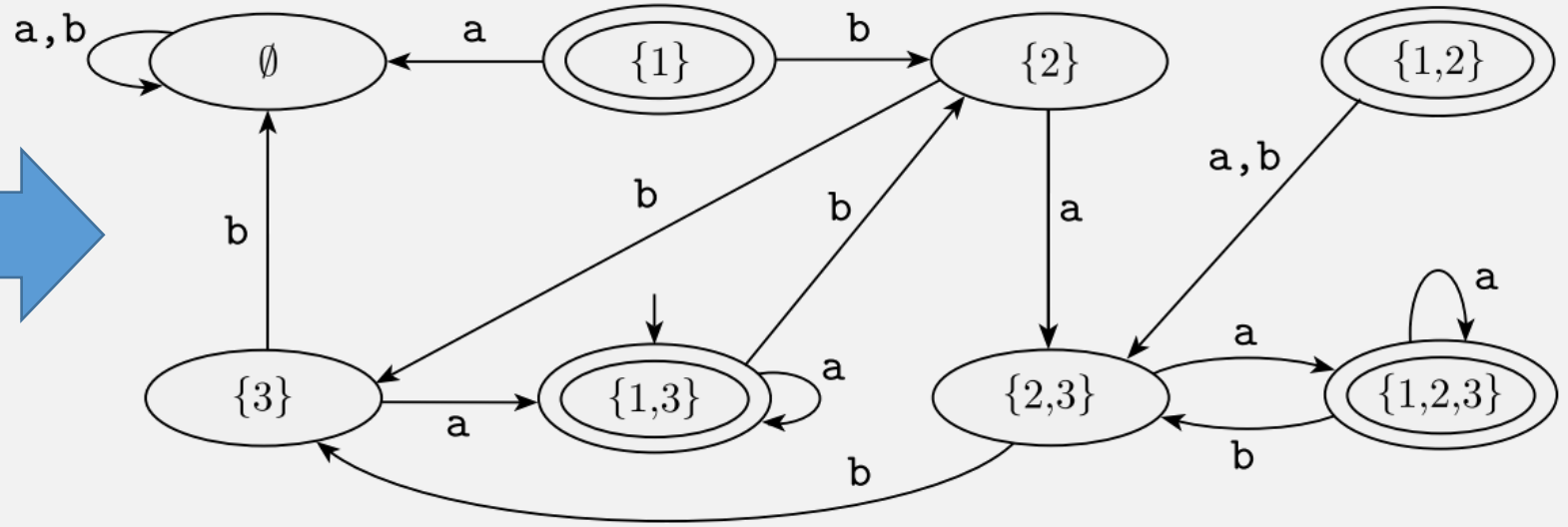
- Let **NFA** N = $(Q, \Sigma, \delta, q_0, F)$
- **An equivalent DFA** M **has states** Q' = $\mathcal{P}(Q)$ (power set of Q)

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

No empty transitions

NFA→DFA

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

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

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

qs = DFA state = set of NFA states

2. For $qs \in Q_{DFA}$ and $a \in \Sigma$
   - $\delta_{DFA}(qs, a) = \bigcup_{q \in qs} \delta_{NFA}(q, a)$ A DFA step = an NFA step for all states in the set

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

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