CS420
Operations on Regular Languages

Wed Sept 16, 2020

In-class example (from last time)

- Design machine M that recognizes: {w |w has exactly three 1's}

- Where Σ= {0, 1},

- Remember:

DEFINITION 1.5

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. $\delta \colon Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

Proving that a language is regular

- *Prove that this lang is regular*: {w |w has exactly three 1’s}

> A language is called a ***regular language*** if some finite automaton recognizes it.

HWO Recap

HW1

HW1

- Will include core set of pkgs
  - python3
  - default-jdk
  - build-essential
- Any other libraries/tools: manually install in Makefile \`setup\`

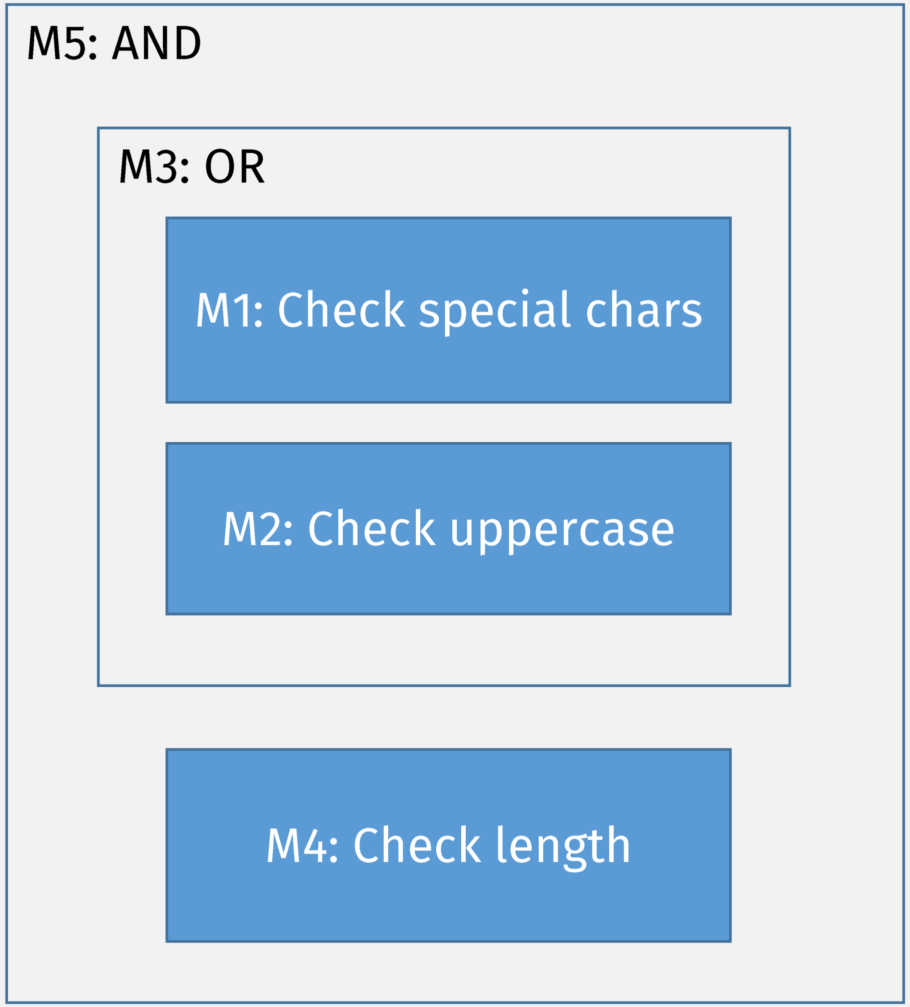
Operations on Regular Languages

From: https://www.umb.edu/it/password

Password Requirements

- Passwords must have a minimum length of ten (10) characters - but more is better!
- Passwords **must include at least 3** different types of characters:
    - upper-case letters (A-Z)
    - lower-case letters (a-z)
    - symbols or special characters (%, &, *, $, etc.)
    - numbers (0-9)
- Passwords cannot contain all or part of your email address
- Passwords cannot be re-used

Password checker



Want to be able to easily combine finite automata machines

To keep combining operations must be **closed**!

“Closed” Operations

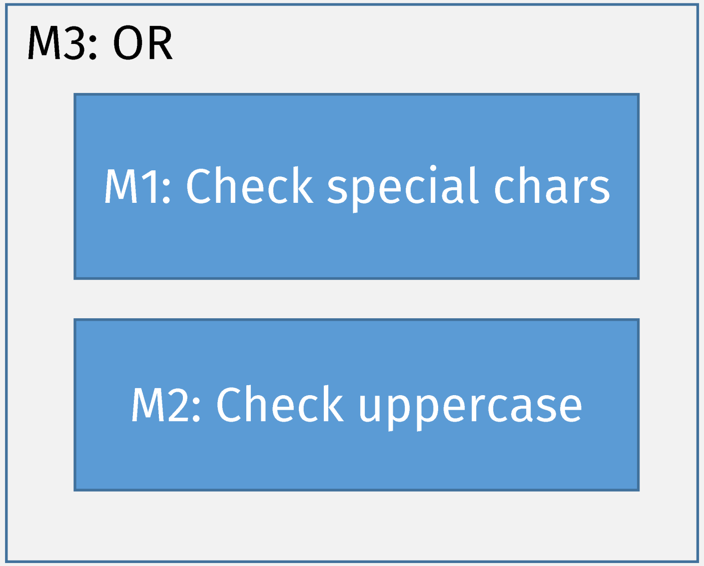
- Natural numbers = {0, 1, 2, …}
  - <u>Closed</u> under addition: if x and y are Natural, then z = x + y is a Nat
  - Closed under multiplication?
    - yes
  - Closed under subtraction?
    - no
- Integers = {…, -2, -1, 0, 1, 2, …}
  - Closed under addition and multiplication
  - Closed under subtraction?
    - yes
  - Closed under division?
    - no
- Rational numbers = {x | x = y/z, y and z are ints}
  - Closed under division?
    - No?
    - Yes if z !=0

Any set is <u>**closed**</u> under some operation if applying that operation to members of the set returns an object still in the set.

Why Closed Operations on RegularLangs?

- Closed operations preserves “regularness”
- I.e., it preserves the same computation model
- So result of combining machines can be combined again

Password checker: “Or” operation



A Closed Operation: Union

THEOREM 1.25

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 $A_1 \cup A_2$.

- How do we prove that a language is regular?
  - Create a FSM recognizing it!
- Create machine combining machines recognizing A_1 and A_2.

Kinds of Mathematical Proof

- Proof by construction
    - Construct the mathematical object in question
- Proof by contradiction
- Proof by induction

A Closed Operation: Union

THEOREM 1.25

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 (implement for hw1)

- Given:
  - machine $\mathbf{M_1}$ (with states Q_1 and transition fn δ_1) recognizing A_1, and
  - machine $\mathbf{M_2}$ (with states Q_2 and transition fn δ_2) recognizing A_2
- Construct a new machine **M**, using $\mathbf{M_1}$ and $\mathbf{M_2}$
- Given an input, **M** runs it on both $\mathbf{M_1}$ and $\mathbf{M_2}$ in parallel
- So a state of **M** is in Q_1 x Q_2 (Cartesian product of $\mathbf{M_1}$ and $\mathbf{M_2}$'s states)
- **M**'s transition fn δ (q_1, q_2) x = $(\delta_1\ q_1\ x, \delta_2\ q_2\ x)$

Another Operation: Concatenation

THEOREM 1.26

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

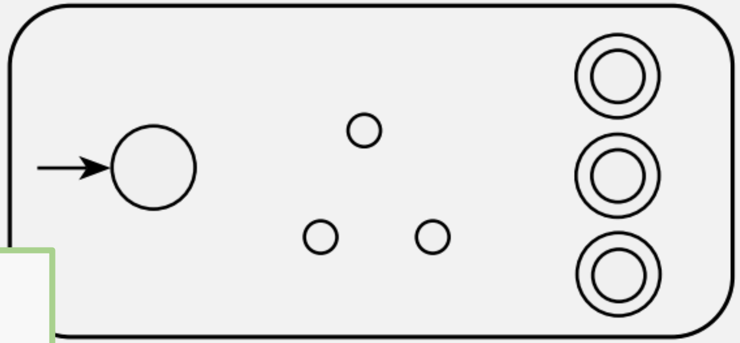
In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

- Can't directly combine A_1 and A_2
  - don't know when to switch from A_1 to A_2 (can only read input once)
- It would create a new kind of machine!
- So is concatenation not closed???

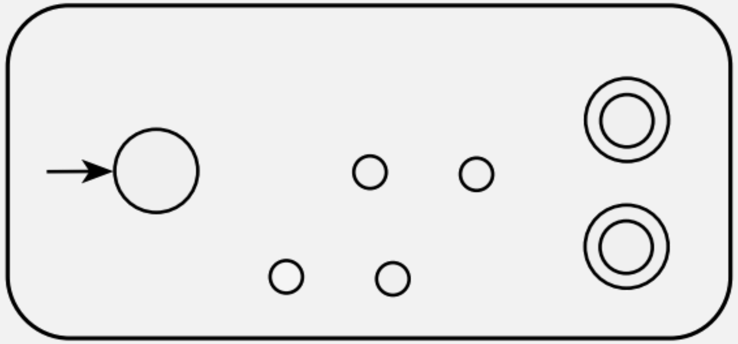
Non-determinism

Concatentation

N_1



N_2



N is a new kind of machine, an NFA!

Let N_1 recognize A_1, and N_2 recognize A_2.

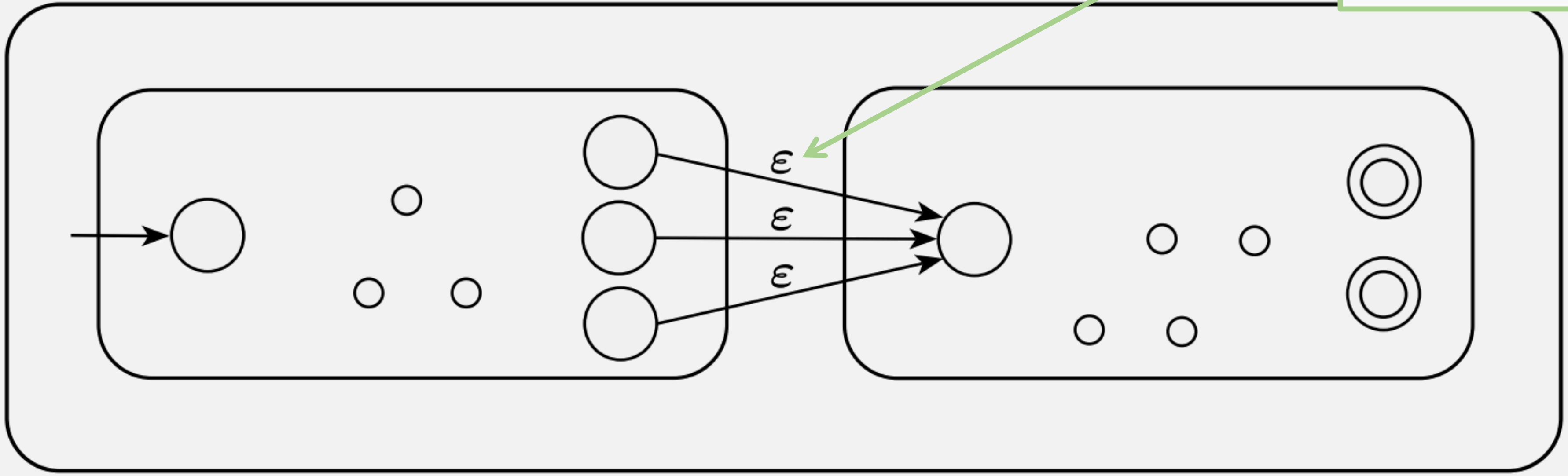
Want: Construction of N to recognize $A_1 \circ A_2$

ε = empty string = no input

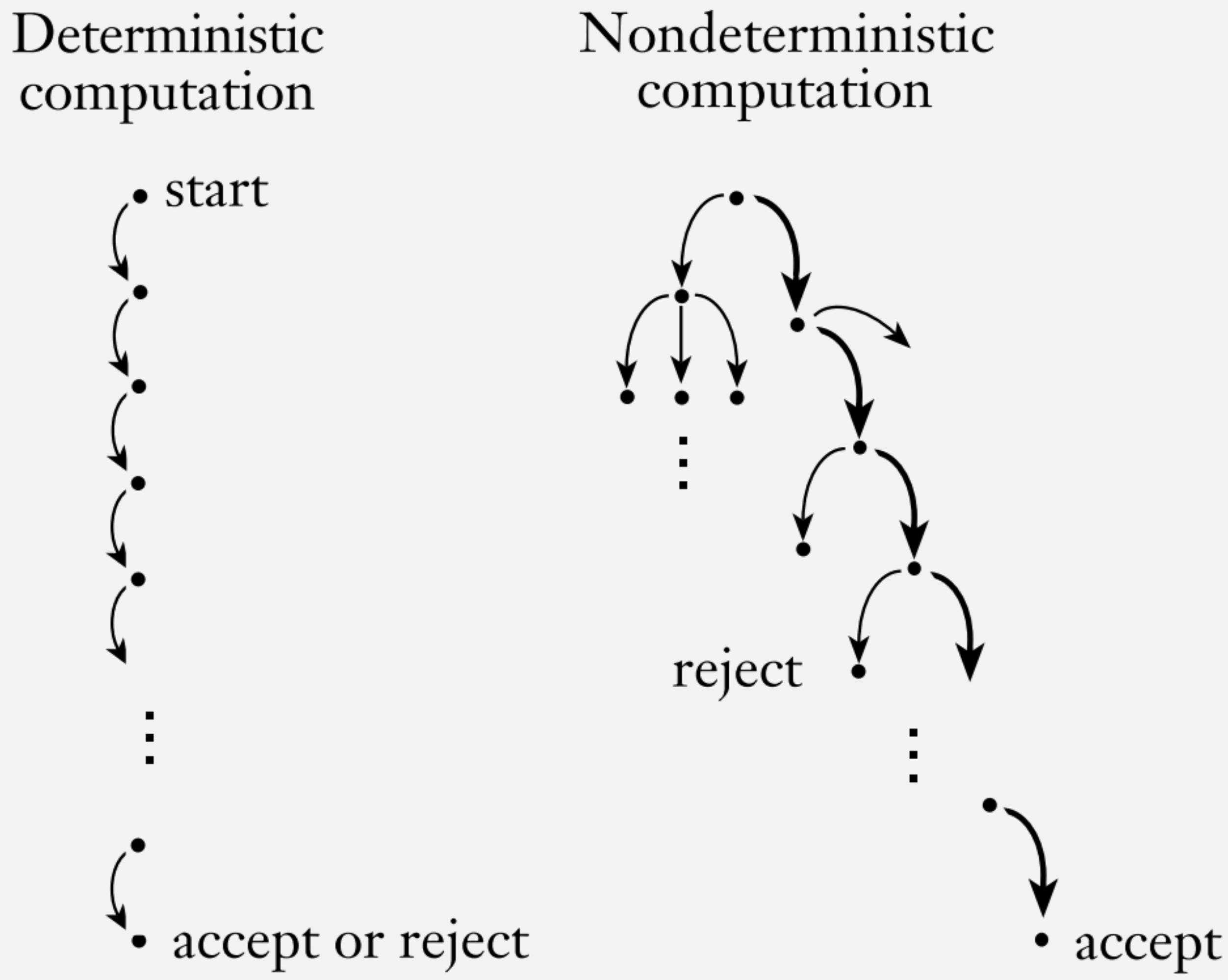
So N can:

- stay in current state **and**
- move to next state

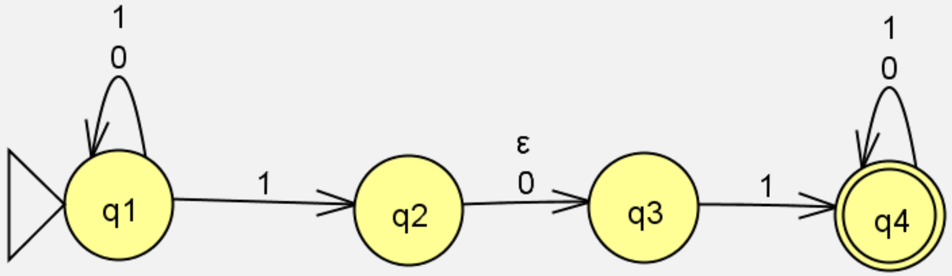
N

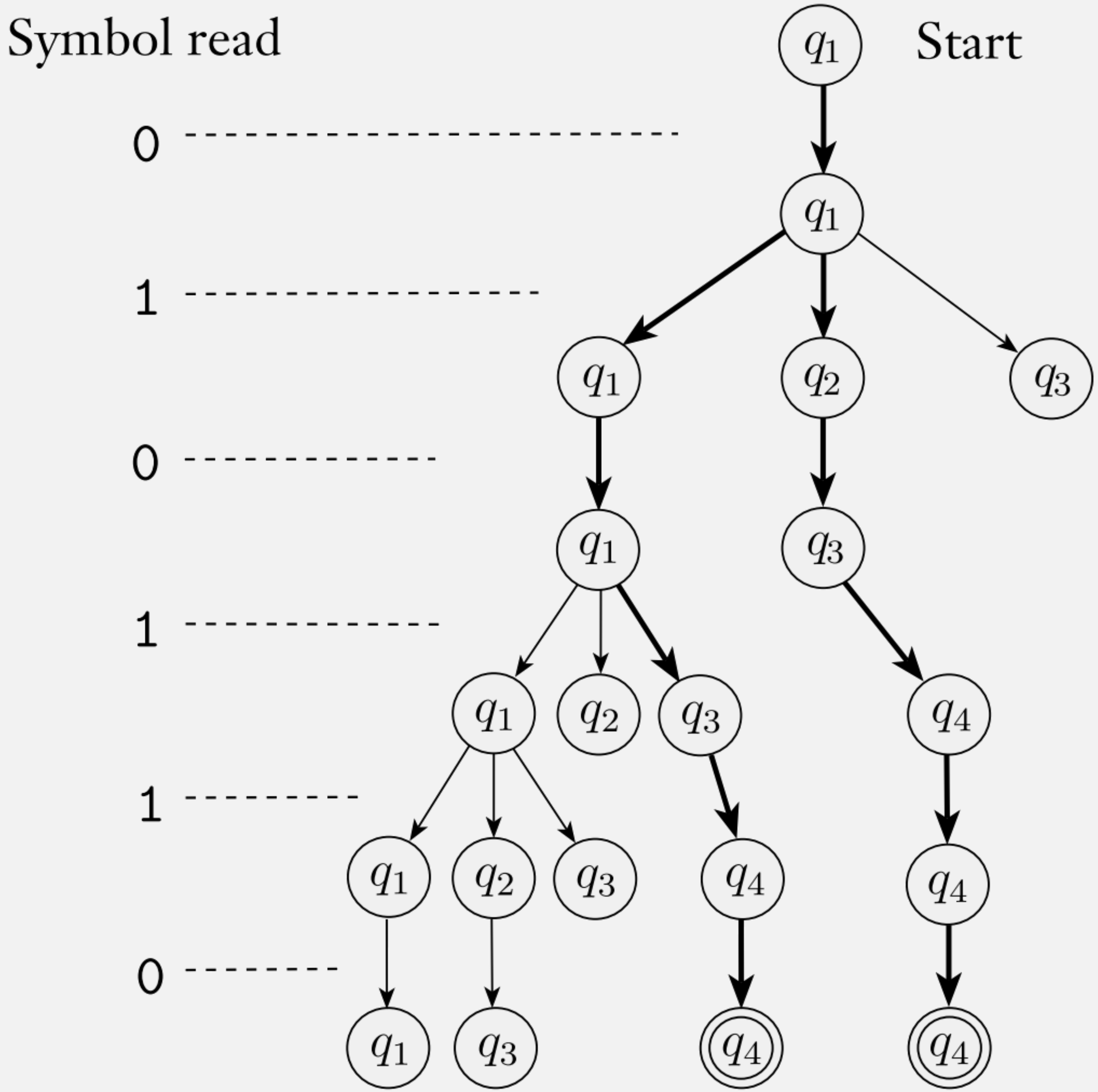


NFA = Non-deterministic Finite Automata



Example fig1.27 (JFLAP demo): 010110




Symbol read
Start
0
1
0
1
1
0
q_1
q_1
q_1 q_2 q_3
q_1 q_3
q_1 q_2 q_3 q_4
q_1 q_2 q_3 q_4 q_4
q_1 q_3 q_4 q_4

Nondeterministic machine can be in multiple states at once

DEFINITION 1.37

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

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. $\delta \colon Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

Power Sets

- A power set is the set of all subsets of a set
- <u>Example</u>: `S = {a,b,c}`
- Power set of S =
  - `{{},{a},{b},{c},{a,b},{a,c},{b,c},{a,b,c}}`

Formal Definition of "Computation"

- DFA:

M ***accepts*** w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

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

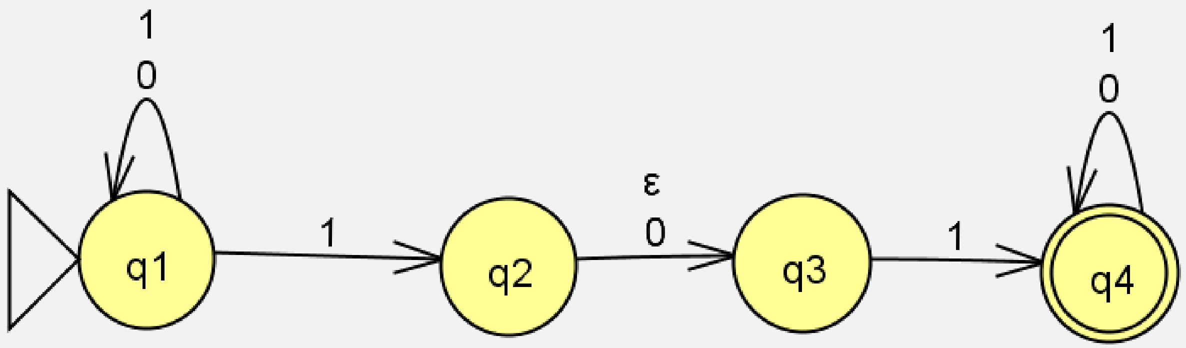
- NFA:

N ***accepts*** w if a sequence of states $r_0, r_1, \ldots, r_m$ exists in Q with three conditions:

1. $r_0 = q_0$,
2. $\boxed{r_{i+1} \in \delta(r_i, y_{i+1})}$, for $i = 0, \ldots, m-1$, and
3. $r_m \in F$.

In-class exercise

- Come up with a formal description of the following NFA:

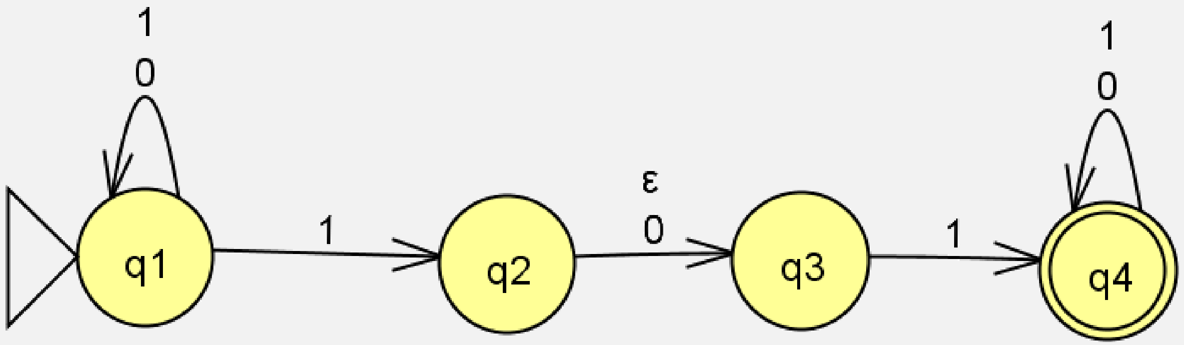


The formal description of N_1 is $(Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3, q_4\}$,
2. $\Sigma = \{0,1\}$,
3. δ is given as

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$,

4. q_1 is the start state, and
5. $F = \{q_4\}$.



So is concat not closed for regular langs?

- It is closed!

- Because NFAs <u>also</u> recognize regular languages!
  - Prove it!
  - (How do we prove that a language is regular?)

A language is called a ***regular language*** if some finite automaton recognizes it.

Need a way to convert NFA -> DFA

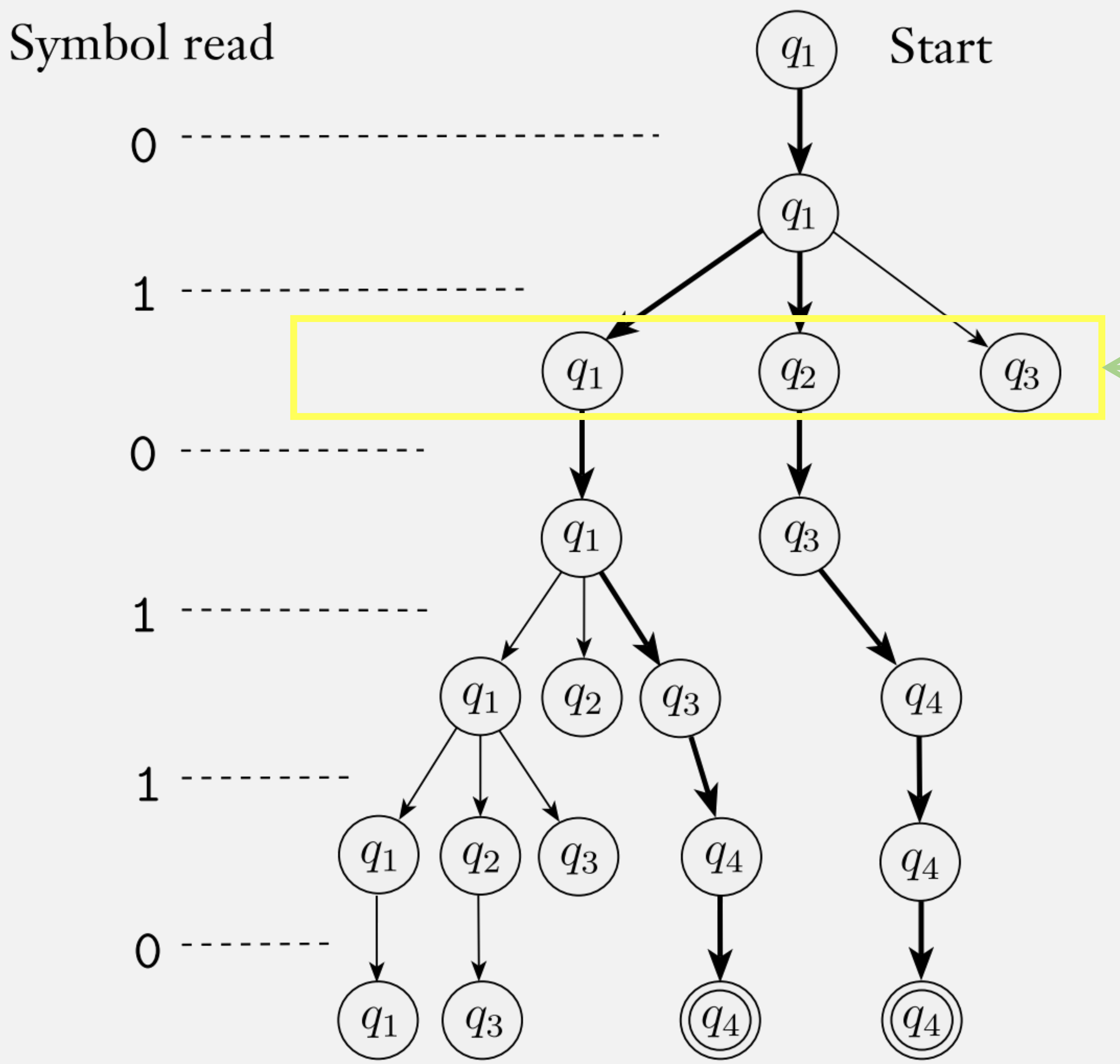
A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

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:
Each "state" of the DFA must be a set of states in the NFA



In a DFA, each step must be only **one** state

So design a state in the converted DFA to be a **set of NFA states**!

Convert NFA -> DFA

- Let NFA N = $(Q, \Sigma, \delta, q_0, F)$
- Then equivalent DFA M has states Q' = $\mathcal{P}(Q)$ (power set of Q)
- (do the rest for hw2)

Proving NFAs recognize regular langs

- Theorem:
  - A language is regular if and only if some NFA recognizes it.

How to prove theorem: X if and only if Y

- “X if and only if Y” = X iff Y = X <=> Y = X⇔Y
- Must prove both:

1. => if X, then Y
    - i.e., assume X, then use it to prove Y
2. <= if Y, then X
    - i.e., assume Y, then use it to prove X

Proving NFAs recognize regular langs

- *Theorem*:
  - A language A is regular if and only if some NFA N recognizes it.

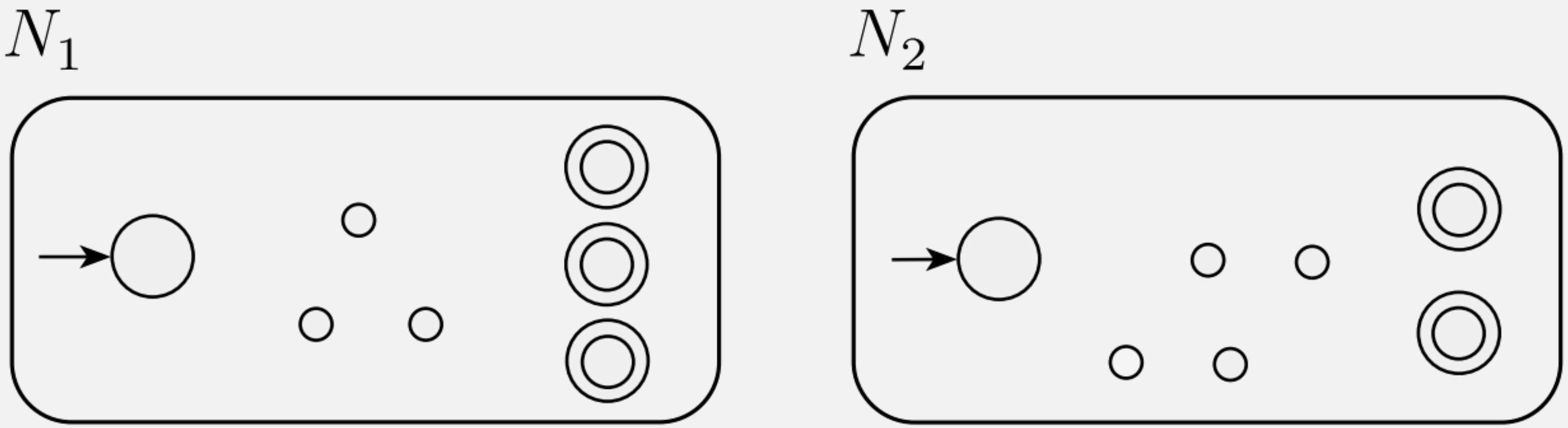
- Must prove:
  - => If A is regular, then some NFA N recognizes it
    - If A is regular, then a DFA recognizes it. But a DFA is also an NFA!
  - <= If an NFA N recognizes A, then A is regular.
    - Convert N to DFA

Regular Operations, Revisited

- Regular languages are closed under the following operations:
  - Union
  - Concatenation
  - Kleene Star

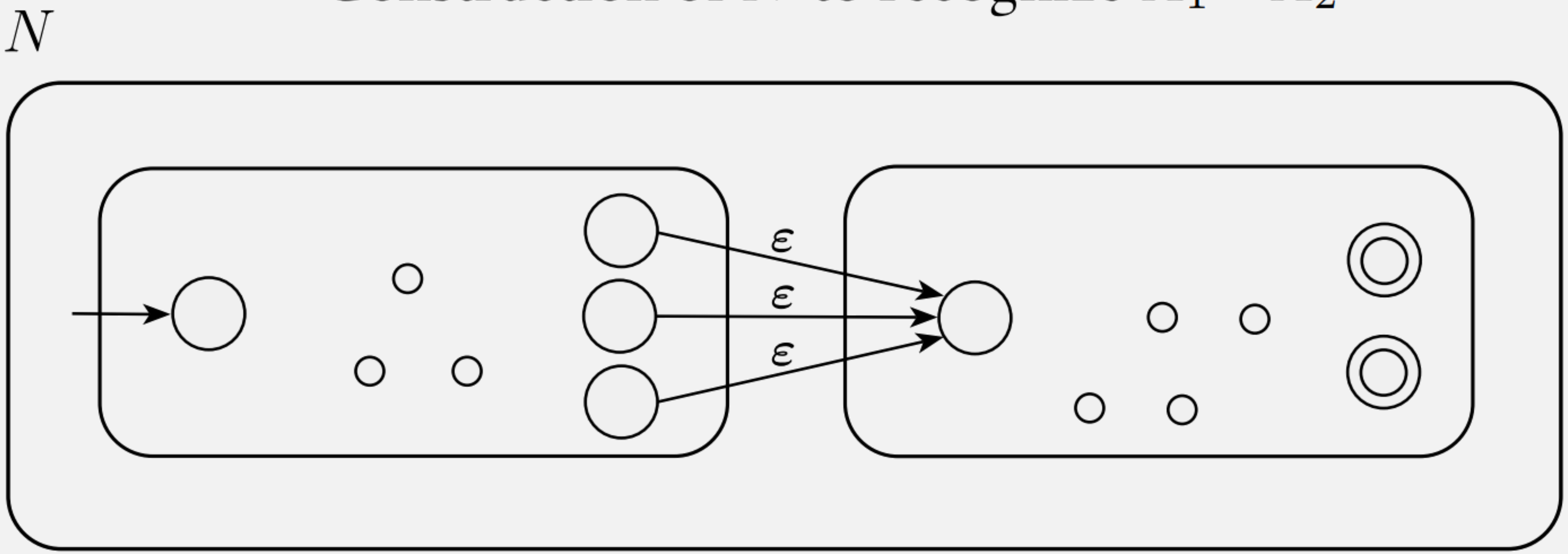
- Easy to prove (by construction) using NFAs

Concatentation



Let N_1 recognize A_1, and N_2 recognize A_2.

Construction of N to recognize $A_1 \circ A_2$



Why do we care?

- These three operations can describe all regular languages!
  - Union
  - Concatenation
  - Kleene Star
- Ie, they define **regular expressions**

Check-in Quiz 2

End of Class Survey