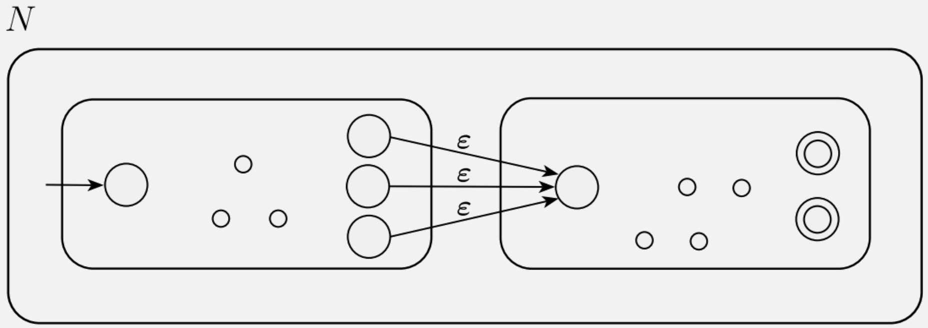
CS420

Combining Automata & Closed Operations

Tuesday, September 20, 2022

UMass Boston Computer Science



Announcements

- HW 1
  - Due Sun 9/25 11:59pm EST
  - Get started early!
  - Questions asked late on Sunday are less likely to be answered
- HW 0 grades returned
  - Use gradecope re-grade request for questions and/or complaints

Last Time: Tips on How to Create Finite Automata

Analogy for this class
Automata ~ `Programs` ::
Designing Automata ~ `Programming`!

1. Confirm understanding of the problem
   - Create examples … and expected results (**`accept`** / **`reject`**)

2. Decide information to "remember"
   - These are the machine states: some are accept states; one is start state

3. Determine transitions between states

4. Test machine behaves as expected
   - Use your examples; create additional ones if needed

Last Time: Is Union Closed For Regular Langs?

In this course, **we are interested in closed operations for a set of languages** (here the **set of regular languages**)

(*In general, a* **set** is **closed** under an operation if applying the **operation** to **members of the set** produces a **result in the same set**)

Want to prove this statement

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 $A_1 \cup A_2$.

Or this (same) statement

A **member of the set of regular languages** is …

… **a regular language,** which itself is a set (of strings) …

… so the **operations** we're interested in are **set operations**

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

Last Time: Union of Languages

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

If $A = \{\texttt{good}, \texttt{bad}\}$ and $B = \{\texttt{boy}, \texttt{girl}\}$, then

$$A \cup B = \{\texttt{good}, \texttt{bad}, \texttt{boy}, \texttt{girl}\}$$

Last Time: Is Union 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 $A_1 \cup A_2$.

Want to prove this statement

Need to show this language is regular (where A_1 and A_2 are regular)

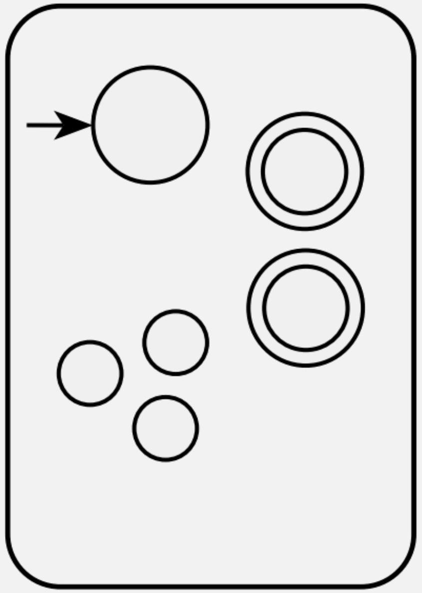
Given

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

- How do we prove that a language is regular?
  - Create a DFA recognizing it!
- So to prove this theorem … create a DFA that recognizes $A_1 \cup A_2$
  - But! We don't know what A_1 and A_2 are!
  - What do we know about A_1 and A_2???

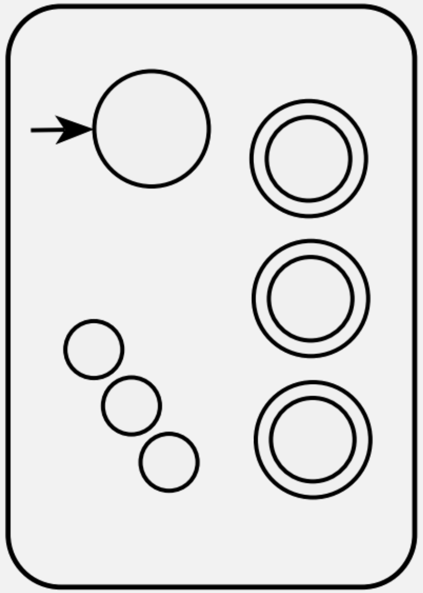
M_1

recognizes A_1



M_2

recognizes A_2



Regular language A_1
Regular language A_2

If we <u>don't know</u> what exactly these languages are, <u>we still know these facts</u> ...

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

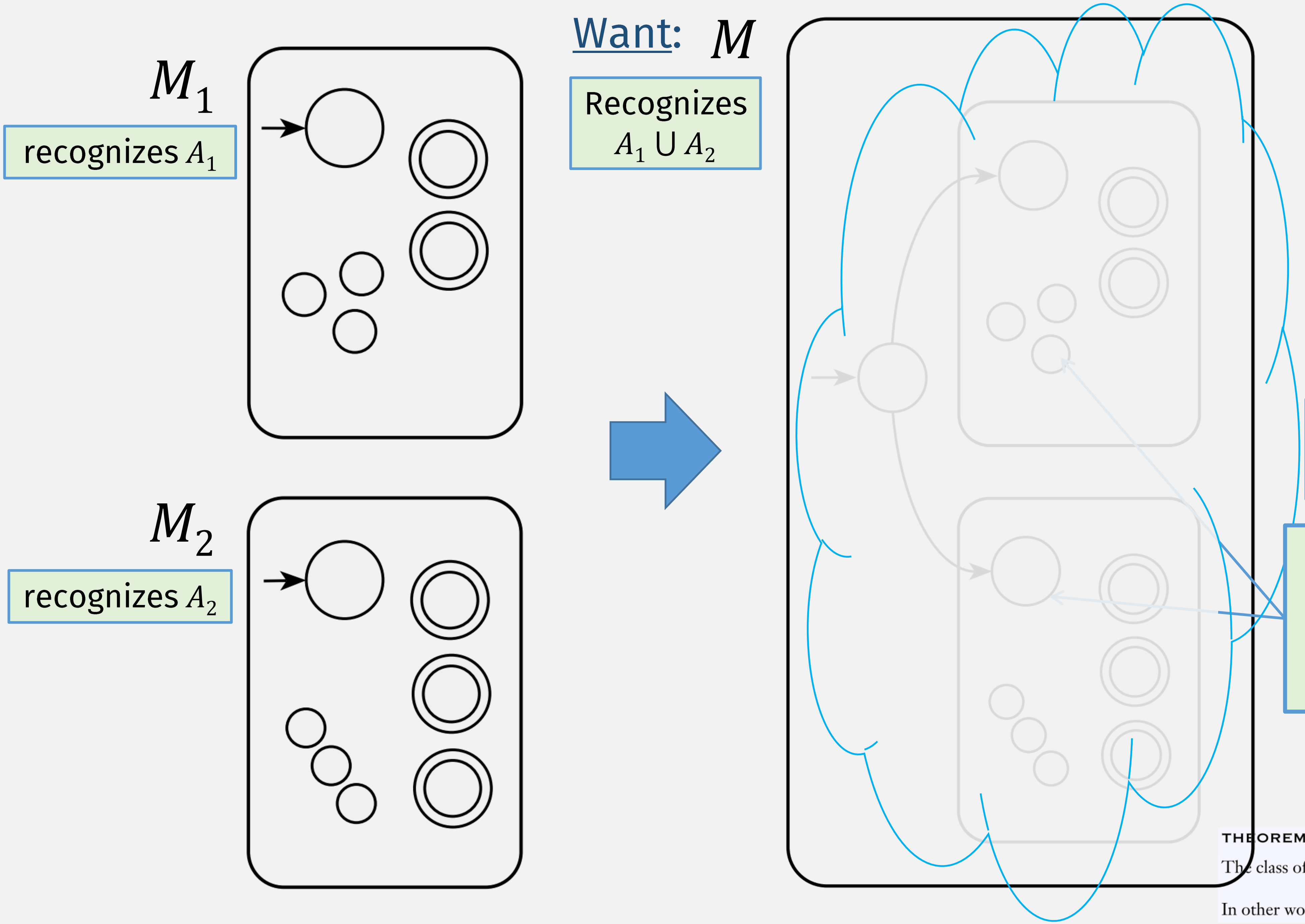
DEFINITION

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***.

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

Union



Rough sketch Idea:
M is a combination of M_1 and M_2 that checks whether its input is accepted by either M_1 and M_2

But, a DFA can only read its input once!

Need to somehow simulate "being in" both an M_1 and M_2 state simultaneously

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 $A_1 \cup A_2$.

Union is Closed For Regular Languages

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,

Want: M that can simultaneously be in both an M_1 and M_2 state

- Construct: $M = (Q, \Sigma, \delta, q_0, F)$, using M_1 and M_2, that recognizes $A_1 \cup A_2$
- 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

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***,[1]
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

A state of M is a pair:
- the first part is a state of M_1 and
- the second part is a state of M_2

So the states of M is all possible combinations of the states of M_1 and M_2

Union is Closed For Regular Languages

<u>*Proof*</u>

- 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: $M = (Q, \Sigma, \delta, q_0, F)$, using M_1 and M_2, that recognizes $A_1 \cup A_2$
- states of M: $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$ $= Q_1 \times Q_2$
  This set is the ***Cartesian product*** of sets Q_1 and Q_2

$a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

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 step in M is includes <u>both</u>:
- a step in M_1, and
- a step in M_2

Union is Closed For Regular Languages

<u>*Proof*</u>

- 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: $M = (Q, \Sigma, \delta, q_0, F)$, using M_1 and M_2, that recognizes $A_1 \cup A_2$
- states of M: $Q = \{(r_1, r_2) | \, r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$ $= Q_1 \times Q_2$
  This set is the ***Cartesian product*** of sets Q_1 and Q_2
- M transition fn: $\delta\big((r_1, r_2), a\big) = \big(\delta_1(r_1, a), \delta_2(r_2, a)\big)$
- M start state: (q_1, q_2)

Start state of M is both start states of M_1 and M_2

Union is Closed For Regular Languages

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: $M = (Q, \Sigma, \delta, q_0, F)$, using M_1 and M_2, that recognizes $A_1 \cup A_2$
- states of M: $Q = \{(r_1, r_2) | \, r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$ $= Q_1 \times Q_2$
  This set is the ***Cartesian product*** of sets Q_1 and Q_2
- M transition fn: $\delta\big((r_1, r_2), a\big) = \big(\delta_1(r_1, a), \delta_2(r_2, a)\big)$
- M start state: (q_1, q_2)

Remember:
Accept states must be subset of Q

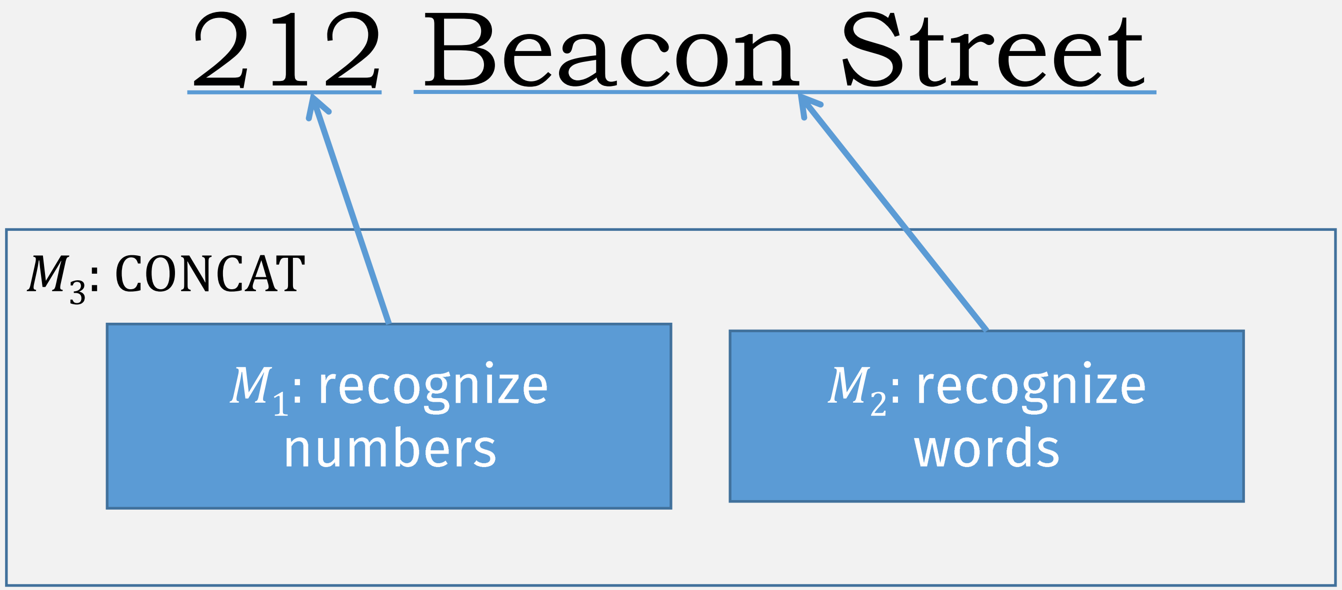
Accept if <u>either</u> M_1 or M_2 accept

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

(Q.E.D.) ■

Another operation: Concatenation

Example: **Recognizing street addresses**



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

Concatenation of Languages

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

If $A = \{\mathtt{good}, \mathtt{bad}\}$ and $B = \{\mathtt{boy}, \mathtt{girl}\}$, then

$$A \circ B = \{\mathtt{goodboy}, \mathtt{goodgirl}, \mathtt{badboy}, \mathtt{badgirl}\}$$

Is Concatenation Closed?

THEOREM

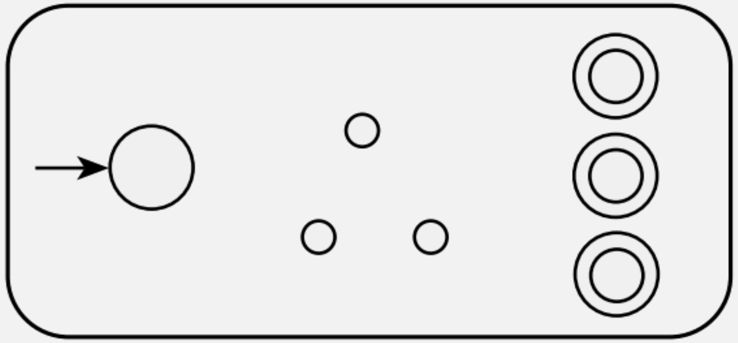
The class of regular languages is closed under the concatenation operation.

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

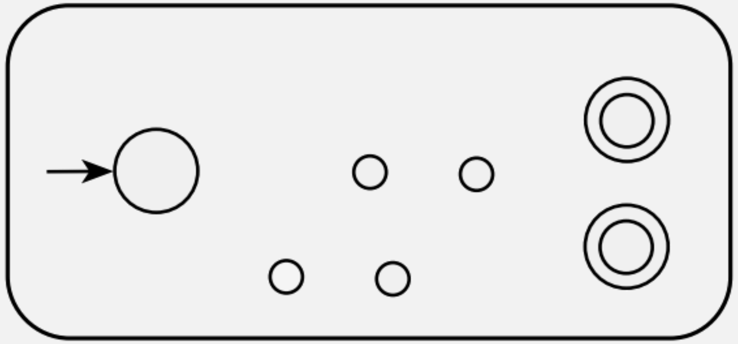
- **Construct a <u>new</u> machine M recognizing $A_1 \circ A_2$?** (like union)
  - Using **DFA M_1** (which recognizes A_1),
  - and **DFA M_2** (which recognizes A_2)

Concatentation

M_1



M_2

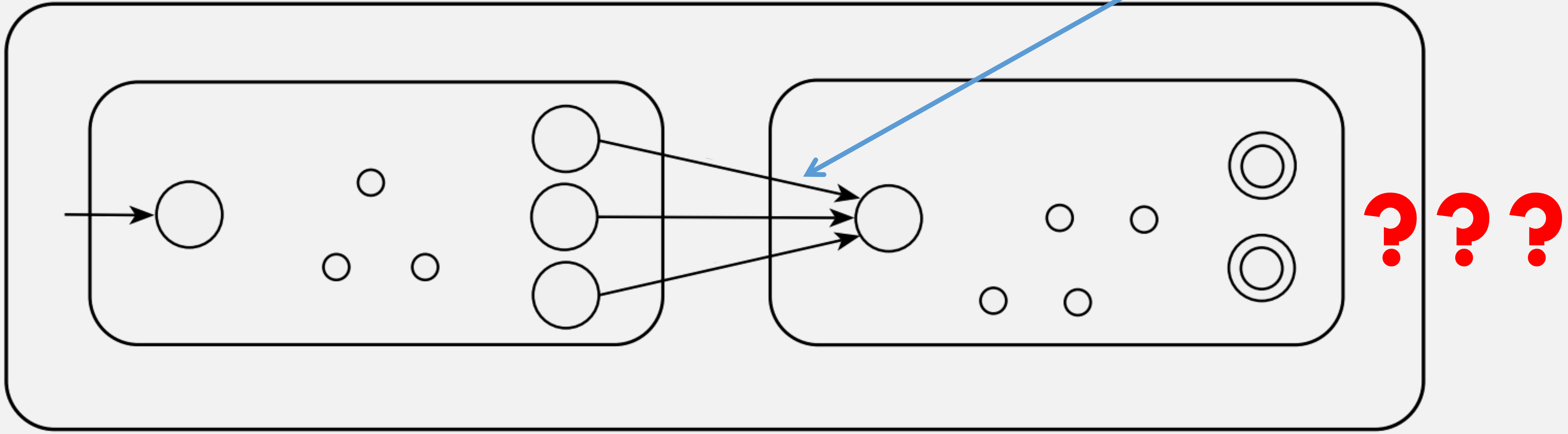


PROBLEM: Can only read input once, can't backtrack

Let M_1 recognize A_1, and M_2 recognize A_2.

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

Need to switch machines at some point, but when?



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

Overlapping Concatenation Example

- Let M_1 recognize language A = { `jen`, `jens` }
- and M_2 recognize language B = { `smith` }
- Want: Construct M to recognize $A \circ B$ = { `jensmith`, `jenssmith` }

- If M sees `jen` …
- M must decide to either:

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

Overlapping Concatenation Example

- Let M_1 recognize language A = { `jen`, `jens` }
- and M_2 recognize language B = { `smith` }
- Want: Construct M to recognize $A \circ B$ = { `jensmith`, `jenssmith` }

- If M sees `jen` ...
- M must decide to <u>either</u>:
  - stay in M_1 (correct, if full input is `jenssmith`)

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

Overlapping Concatenation Example

- Let M_1 recognize language A = { `jen`, `jens` }
- and M_2 recognize language B = { `smith` }
- Want: Construct M to recognize $A \circ B$ = { `jensmith`, `jenssmith` }

- If M sees `jen` …
- M must decide to <u>either</u>:
  - stay in M_1 (correct, if full input is `jenssmith`)
  - or switch to M_2 (correct, if full input is `jensmith`)
- <u>But</u> to recognize $A \circ B$, it needs to handle <u>both cases</u>!!
  - Without backtracking

A DFA can't do this!

Is Concatenation Closed?

FALSE?

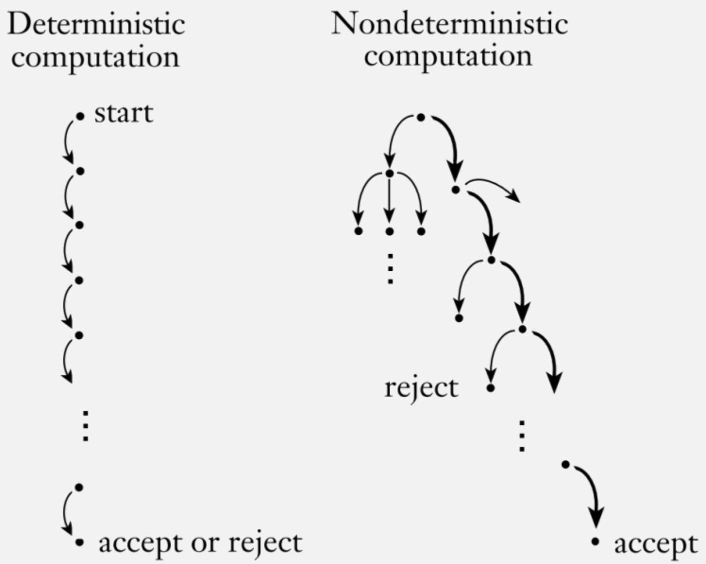
THEOREM

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

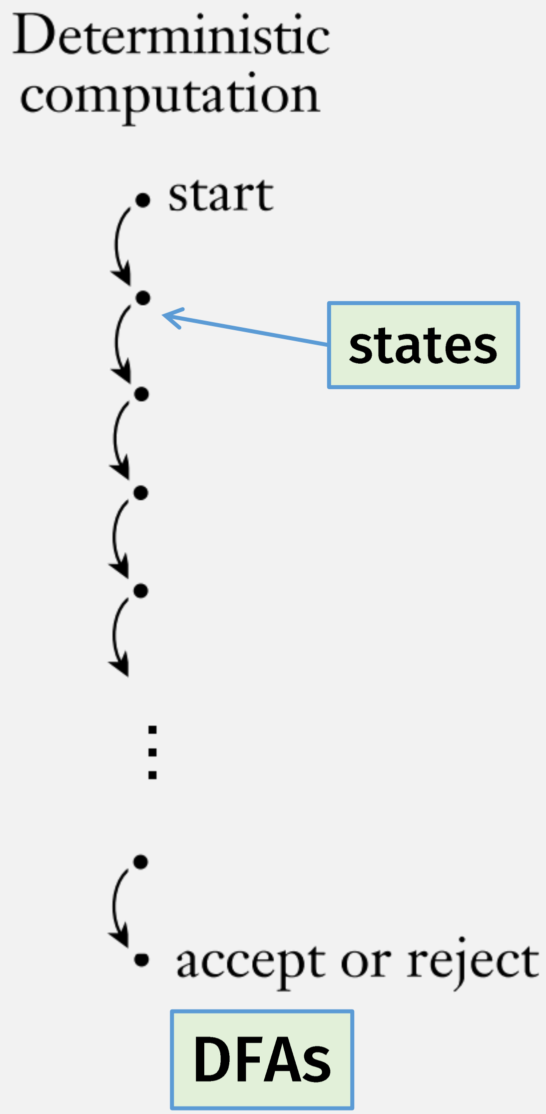
In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

- Cannot **combine A_1 and A_2's machine because:**
  - **Need to switch from A_1 to A_2 at some point …**
  - **… but we don't know when!** (we can only read input once)
- **This requires a new kind of machine!**
- But **does this mean concatenation is not closed for regular langs?**

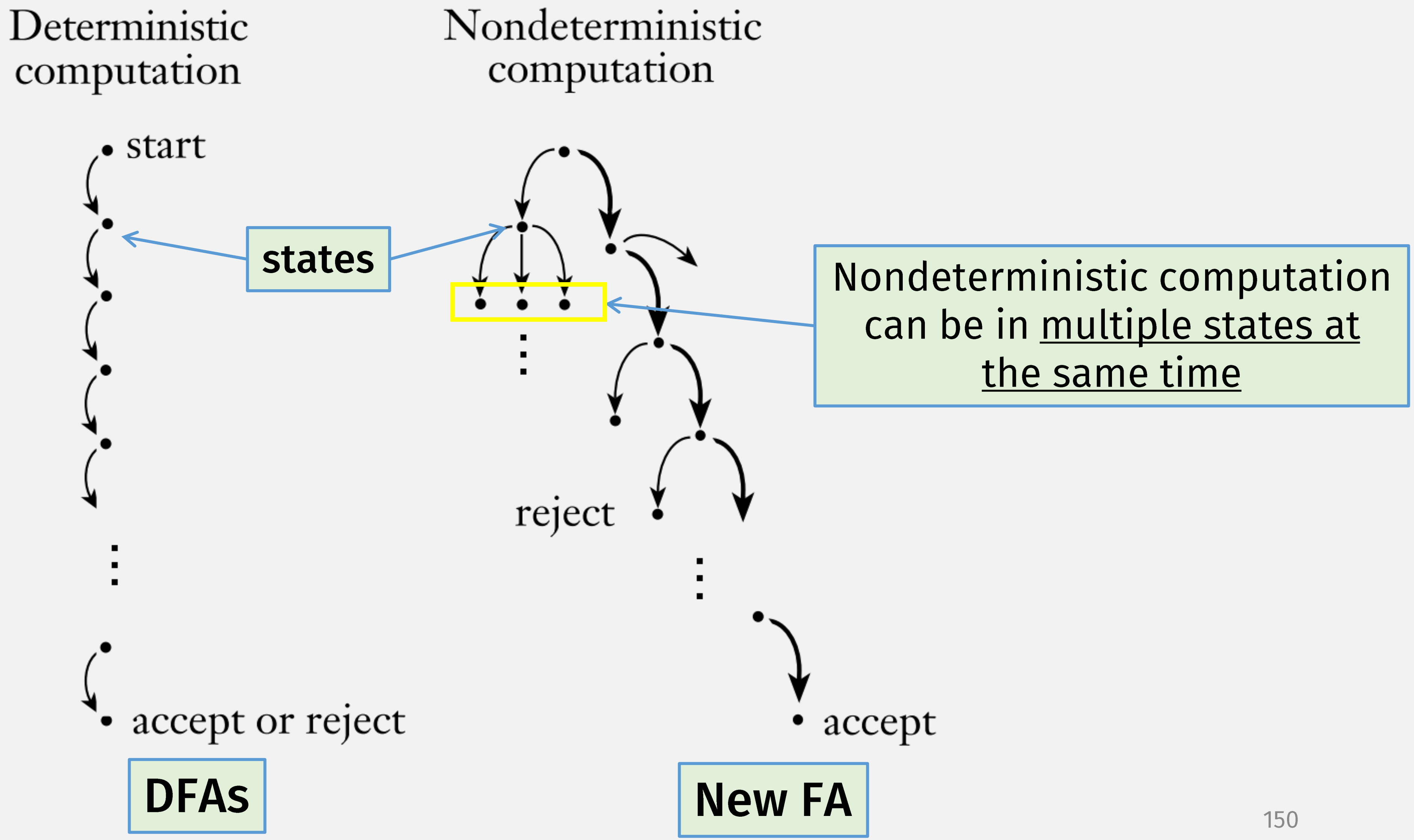
Nondeterminism



Deterministic vs Nondeterministic



Deterministic vs Nondeterministic



Finite Automata: The Formal Definition

DEFINITION

deterministic

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***.

Also called a **Deterministic Finite Automata (DFA)**

Precise Terminology is Important

- A **finite automata** or **finite state machine** (**FSM**) defines …
  … computation with a finite number of states

- There are many kinds of FSMs

- We've learned one kind, the **Deterministic Finite Automata (DFA)**
  - (So currently, the terms DFA and FSM refer to the same definition)

- We will learn other kinds, e.g., **Nondeterministic Finite Automata (NFA)**

- Be careful with terminology!

Nondeterministic Finite Automata (NFA)

DEFINITION

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.

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

Difference

Power set, i.e. a transition results in <u>set</u> of states

Power Sets

- A power set is the set of all subsets of a set

- Example: S = {a, b, c}

- Power set of S =
  - {{ }, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}}
  - Note: includes the empty set!

Nondeterministic Finite Automata (NFA)

DEFINITION

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

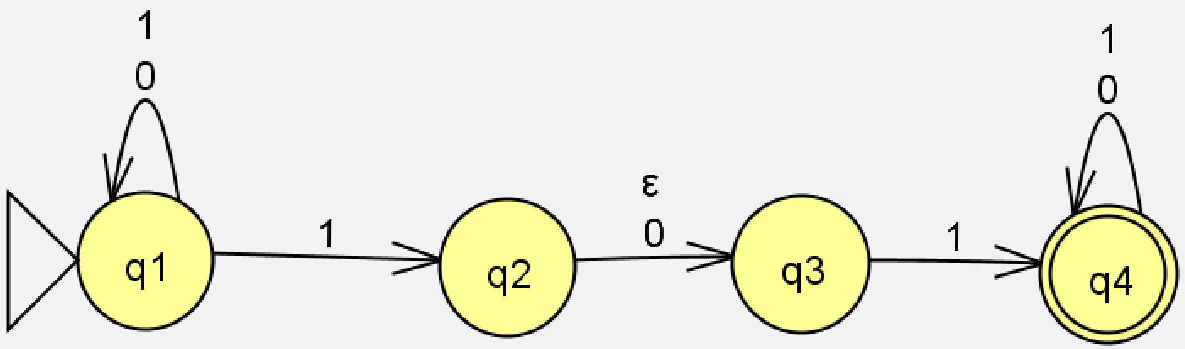
1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta : Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Transition label can be "empty", i.e., machine can transition without reading input

$$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$$

NFA Example

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



DEFINITION

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.

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

$$\delta : Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$$

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

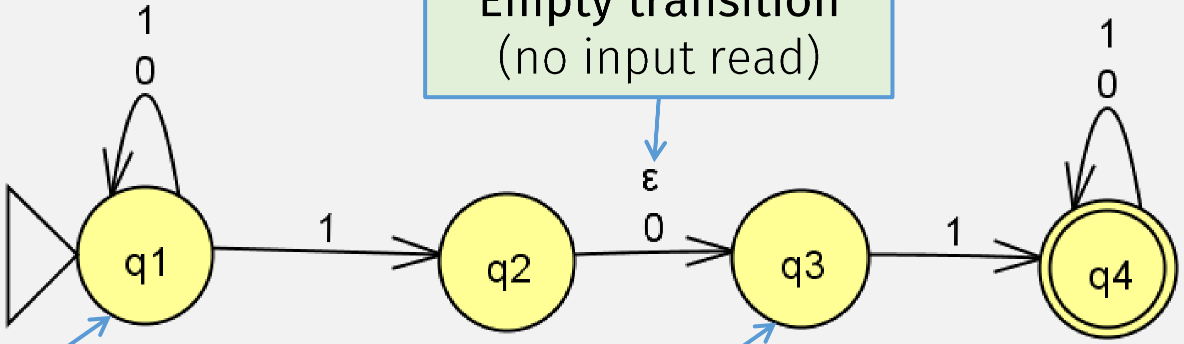
Empty transition (no input read)

	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	$\{q_4\}$	$\{q_4\}$	$\emptyset$,

Result of transition is a set

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

Empty transition (no input read)

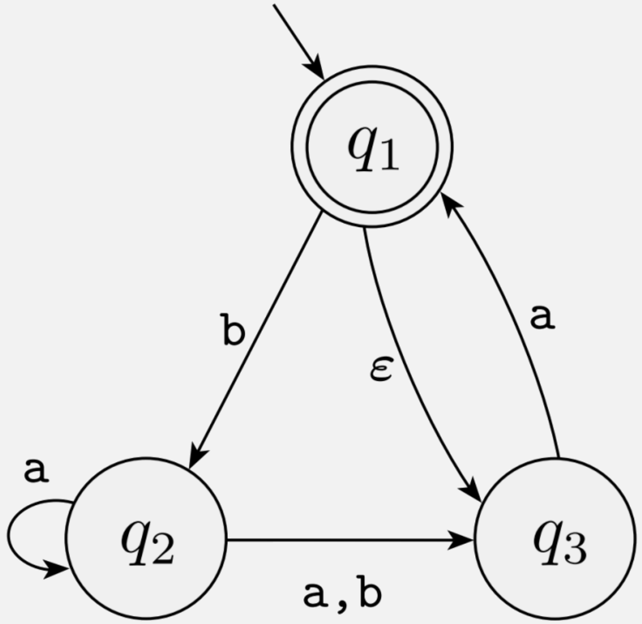


Multiple 1 transitions

No 0 transition

In-class Exercise

- Come up with a formal description for the following NFA
  - $\Sigma = \{ \texttt{a}, \texttt{b} \}$



DEFINITION

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

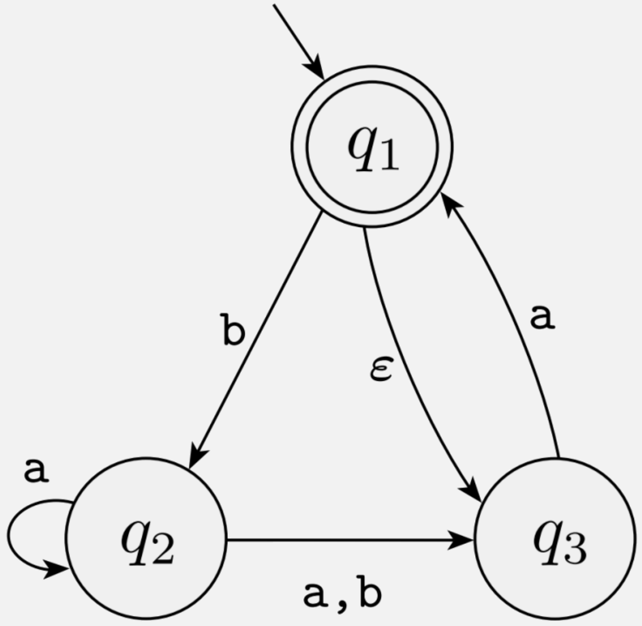
1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta \colon Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

In-class Exercise Solution

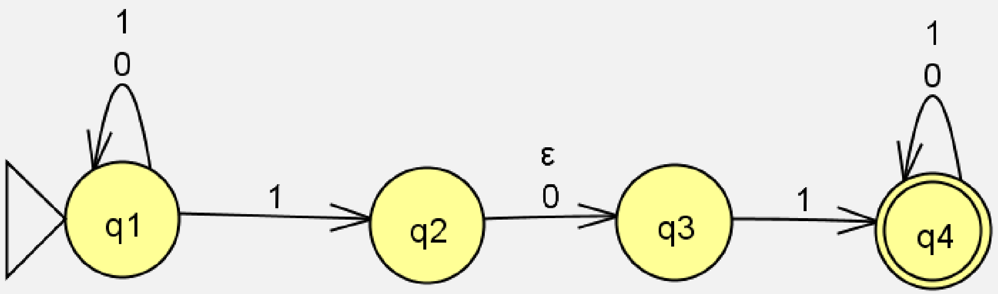
Let $N = (Q, \Sigma, \delta, q_0, F)$

- $Q = \{ q_1, q_2, q_3 \}$
- $\Sigma = \{ \texttt{a}, \texttt{b} \}$
- δ ...
- $q_0 = q_1$
- $F = \{ q_1 \}$

$\delta(q_1, \texttt{a}) = \{ \}$
$\delta(q_1, \texttt{b}) = \{ q_2 \}$
$\delta(q_1, \varepsilon) = \{ q_3 \}$
$\delta(q_2, \texttt{a}) = \{ q_2, q_3 \}$
$\delta(q_2, \texttt{b}) = \{ q_3 \}$
$\delta(q_2, \varepsilon) = \{ \}$
$\delta(q_3, \texttt{a}) = \{ q_1 \}$
$\delta(q_3, \texttt{b}) = \{ \}$
$\delta(q_3, \varepsilon) = \{ \}$



Next Time: Running Programs, NFAs (JFLAP demo): `010110`



Check-in Quiz 9/20

On gradescope