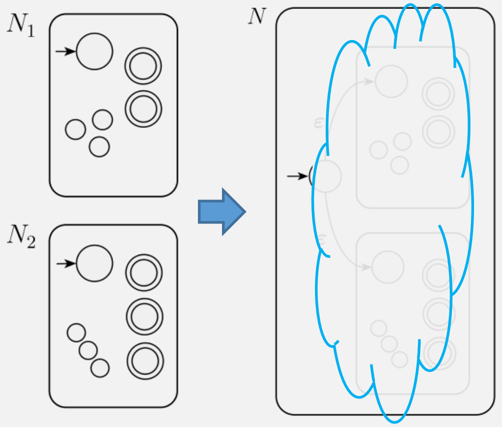
CS 420 / CS 620

Combining DFAs and Closed Operations

Monday, September 22, 2025

UMass Boston Computer Science



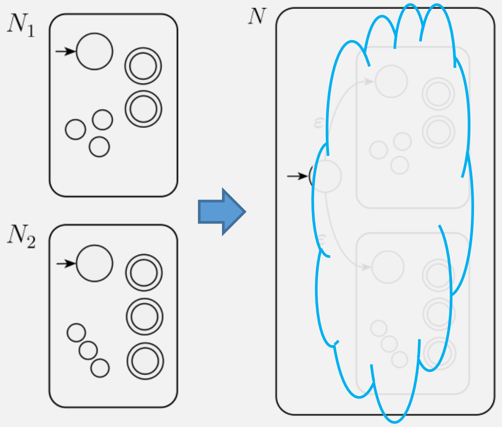
Announcements

- HW 2
  - ~~Due Mon 9/22 12pm (noon)~~

- HW 3
  - Out: Mon 9/22 12pm (noon)
  - Due: Mon 9/29 12pm (noon)

- Check previous Piazza posts before posting!

- All questions about languages or machines must use examples!



HW 1 Observations

- Problems must be <u>assigned to the correct pages</u> in GradeScope

- Proof format must be:
  Statements and **Justifications** table

- Questions / Complaints?
  - Open a GradeScope re-grade request ticket
  - Do not ask the instructor
    (they probably didn't grade it)!

- Extraneous defs,
  - $q_0=q_0$
  - $Q = \{1,2,3\}$
    - but no use of Q
  - $M = (Q,\Sigma,\delta,q_0,F)$
    - with unbound vars!

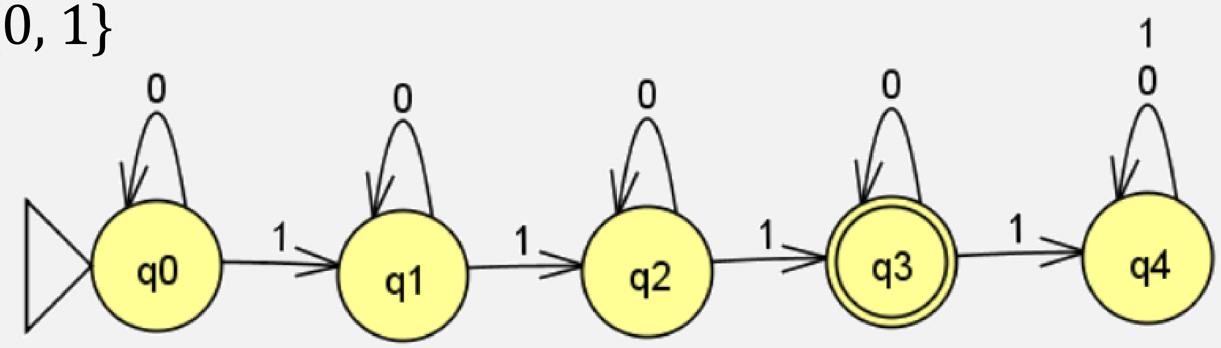
- Wrong types!
  - $M = \{Q,\Sigma,\delta,q_0,F\}$
    - Or some other non-tuple
    - Or forgot / omitted the tuple
  - $F = q_3$
  - $\Sigma=\{“1”, ”2”\}$

Last Time

In-class exercise Solution

- Design finite automata recognizing the language:
  $\{w \mid w \text{ has exactly three } 1\text{'s}\}$

- *States*:
  - Need one state to represent each count of 1's seen so far
  - $Q = \{q_0, q_1, q_2, q_3, q_{4+}\}$
- *Alphabet*: $\Sigma = \{0, 1\}$
- *Transitions*:



- *Start state*:
  - q_0
- *Accept states*:
  - $\{q_3\}$

So: a **DFA's computation** **recognizes simple string patterns**?

Yes!

Have you ever used **a programming language feature** for **recognizing simple string patterns**?

Regular Expressions! (stay tuned!)

Programming Advice: Break Down Complex Problems

2. Break down the problem

Complexity is really just a bunch of simple problems chained together. Try to break down your task into smaller chunks that are more manageable.

https://dev.to/nuxt-wimadev/7-powerful-principles-to-tackle-complex-coding-problems-40a5

- Breaking big scary unknown problems into small manageable ones is a **core skill** for developers. And unlike syntax, it can't be easily learned from google.

This last point should be obvious to anyone who's been coding for a while.

https://medium.com/@dannysmith/breaking-down-problems-its-hard-when-you-re-learning-to-code-f10269f4ccd5

2. Break it down

After understanding the problem, the next process is to break down the problem into smaller sub-problems.

https://javascript.plainenglish.io/5-step-process-to-solve-complex-programming-problems-8e4f74cfd88e

… and then **combine the** (small) **solutions**

Combining DFA Computation?

(Programmers do this all the time)

Password Requirements

DFA

- 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) DFA
  - DFA lower-case letters (a-z)
  - symbols or special characters (%, &, *, $, etc.) DFA
  - numbers (0-9) DFA
- Passwords cannot contain all or part of your email address DFA
- Passwords cannot be re-used DFA

To match all requirements,
combine smaller DFAs into one big DFA?

umb.edu/it/software-systems/password/

Password Checker DFAs

To **combine more than once**, this must be a **DFA**

- M_5: "AND"
  - M_3: "OR"
    - M_1: Check special chars
    - M_2: Check uppercase
  - M_4: Check length

Want: to **easily combine DFAs**, i.e., **composability**

We want these operations:

"OR" : DFA × DFA → DFA

"AND" : DFA × DFA → DFA

To **combine more than once**, **operations** must **preserve input type**, i.e., be **closed!**

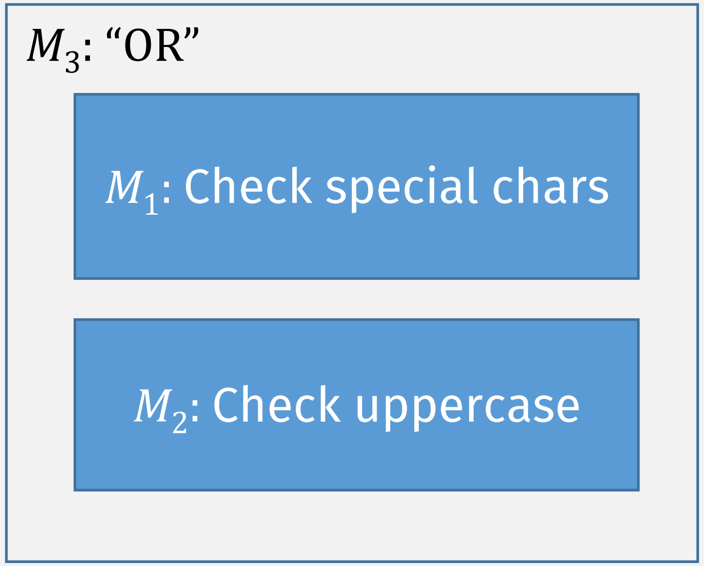
"Closed" Operations

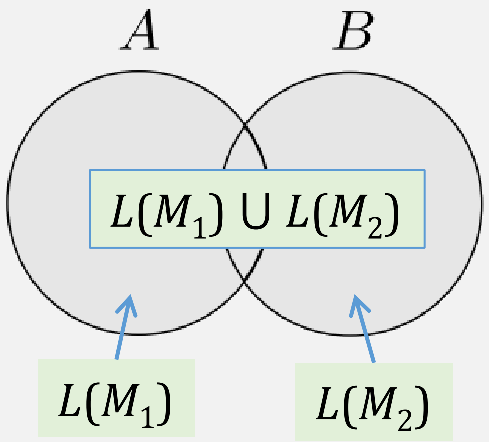
- Set of Natural numbers = $\{0, 1, 2, \ldots\}$
  - Closed under addition:
    - if x and y are Natural numbers,
    - then $z = x + y$ is a Natural number
  - Closed under multiplication?
    - yes
  - Closed under subtraction?
    - no
- Integers = $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
  - Closed under addition and multiplication
  - Closed under subtraction?
    - yes
  - Closed under division?
    - no
- Rational numbers = $\{x \mid x = y/z, y$ and z are Integers$\}$
  - Closed under division?
    - No?
    - Yes if z !=0

A set is **closed** under an operation if: the **result** of applying the operation to members of the set is **in the same set**

i.e., input set(s) = output set

Password Checker: “OR” = “Union”





A **DFA** is **equivalent** to a **language**
(the set of all strings accepted by the DFA)

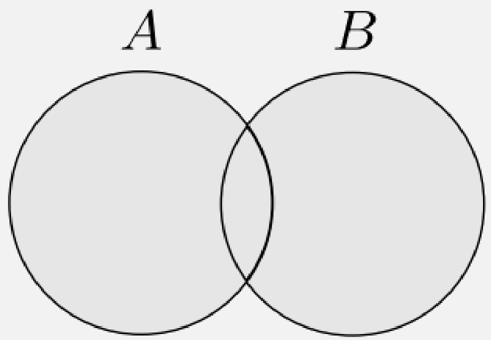
<u>Combining</u> DFAs is **equivalent** to **<u>combining</u> languages**
(the set of all strings accepted by the DFAs)

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

Union of Languages

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

If $A = \{\mathtt{fort}, \mathtt{south}\}$ $B = \{\mathtt{point}, \mathtt{boston}\}$



$$A \cup B = \{\mathtt{fort}, \mathtt{south}, \mathtt{point}, \mathtt{boston}\}$$

Want: "Closed" Ops For Regular Langs!

- Set of Regular Languages = $\{L_1, L_2, \ldots\}$
  - Closed under …?
    - OR (union)
    - AND (intersection)
    - …

A set is **closed** under an operation if:
the result of applying the operation to
members of the set is in the same set

i.e., input set(s) = output set

For Example:

OR: Regular Lang × Regular Lang → Regular Lang

Why Care About Closed Ops on Reg Langs?

- **Closed operations for regular langs preserve "regularness"**
- I.e., it **<u>preserves the same computation model</u>!**
- Can **"combine" smaller "regular" computations** to get **bigger ones:**

For Example:
OR: Regular Lang × Regular Lang → Regular Lang

- In general, this semester, **we want operations that are closed!**

Is Union Closed For Regular Langs?

In this course, **we are interested in closed operations for a** **<u>set of languages</u>** (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**)

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

Want to prove this statement

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

Or this (same) statement

Is Union Closed For Regular Langs?

(*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)

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

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**

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

Or this (same) statement

Flashback: # Mathematical Statements: **IF-THEN**

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 $A_1 \cup A_2$.

t Q), or

- If P is TRUE, then Q is TRUE (**modus ponens**)

Would have to prove: **there are <u>no</u>** <u>**regular languages**</u> (not true!)

Proving:

- To prove: $P \rightarrow Q$ is TRUE:
  - Prove P is FALSE (usually hard or impossible)
  - Assume P is TRUE, then prove Q is TRUE

p	q	$p \rightarrow q$
True	True	True
True	False	False
False	True	True
False	False	True

Is Union Closed For Regular Langs?

Definition of Regular Language

If a lang **has a DFA**, then it's **regular**

Statements

Do we know anything about A_1 and A_2?

Statements	Justifications
1. A_1 and A_2 are regular languages	1. Assumption of If part of If-Then ???
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1	2. Def of Regular Language
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2	3. Def of Regular Language
4. Construct DFA $M = (Q, \Sigma, \delta, q_0, F)$ (todo)	4. Def of DFA
5. M recognizes $A_1 \cup A_2$	5. S
6. $A_1 \cup A_2$ is a regular language	
7. The class of regular languages is closed under the union operation.	7. From stmt #1 and #6

How to create this M? Don't know what A_1 and A_2 are!

Definition of Regular Language ???

If a lang is **regular**, then it has a **DFA** ???

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

To prove $P \rightarrow Q$ is TRUE: Assume P is TRUE, then prove Q is TRUE

?? “If A Then B” =?= “If B Then A” ??

(Actual) Definition of Regular Language

If a lang has a DFA, then it's regular

1. A_1 and A_2 are regular languages	1. Assumption
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1	2. Def of Regular Language ???
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2	3. Def of Regular Language

Definition of Regular Language ???

If a lang is **regular**, then it has a **DFA** ???

Equivalence of Conditional Statements

- Yes or No? "If X then Y" **is equivalent to:**
    - "If Y then X" (**converse**)
        - No!

(Actual) Definition of Regular Language

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

If Regular, Then DFA?

- Prove: If L is a **regular language**, then a **DFA recognizes** L
- Proof (Sketch)

  Case analysis:
  - **Look at all** If-then **statements** of the form:
    - "If … **language** L, then L is a **regular language**"
  - (At least one is true!)
  - **Figure out which one(s)** led to conclusion:
    - "L is a **regular language**"
  - (There's only 1!)

- So it must be that: (because there was only 1 possible way to show that the language is regular)

"Corollary"

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

Is Union Closed For Regular Langs?

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. ??? $\boxed{\text{Construct DFA } M = (Q, \Sigma, \delta, q_0, F) \text{ (todo)}}$
5. M recognizes $A_1 \cup A_2$
6. $A_1 \cup A_2$ is a regular language
7. 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 to create this? Don't know what A_1 and A_2 are!

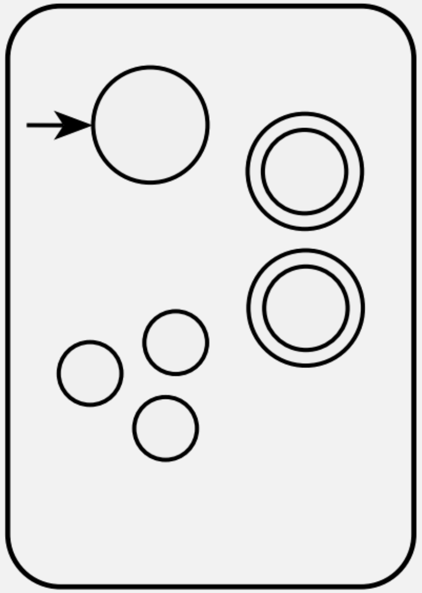
Justifications

1. Assumption
2. Def of Regular Language
3. Def of Regular Language
4. Def of DFA
5. See examples
6. Def of Regular Language
7. From stmt #1 and #6

"Corollary"

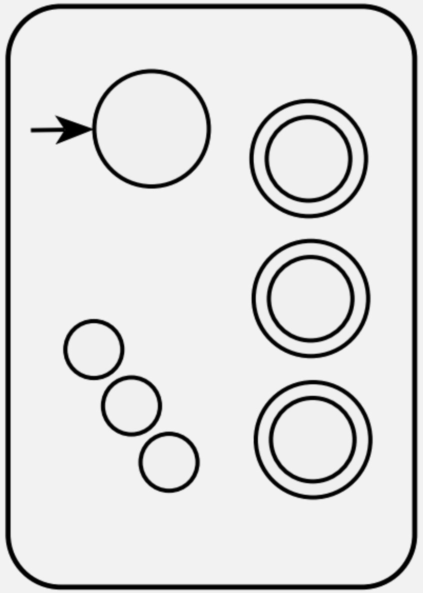
M_1

recognizes A_1



M_2

recognizes A_2



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

Regular language A_1
Regular language A_2

Even if we don't know what these languages are, we still know...

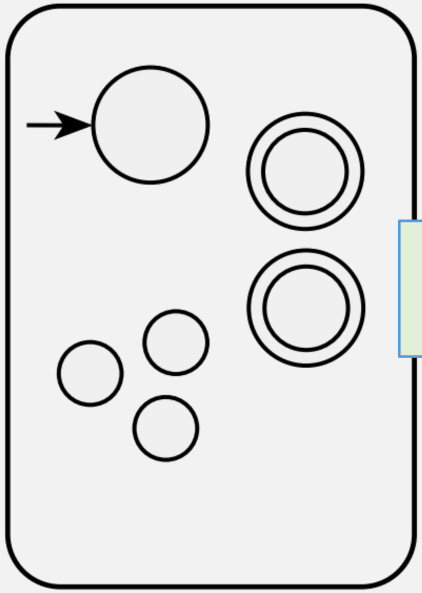
$$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1), \text{ recognize } A_1,$$
$$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2), \text{ recognize } A_2,$$

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

Union

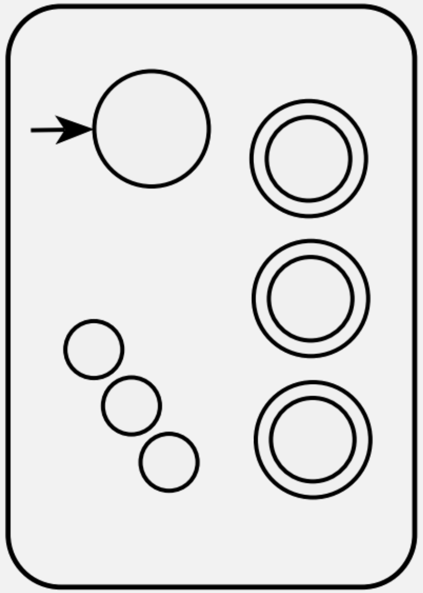
M_1

recognizes A_1



M_2

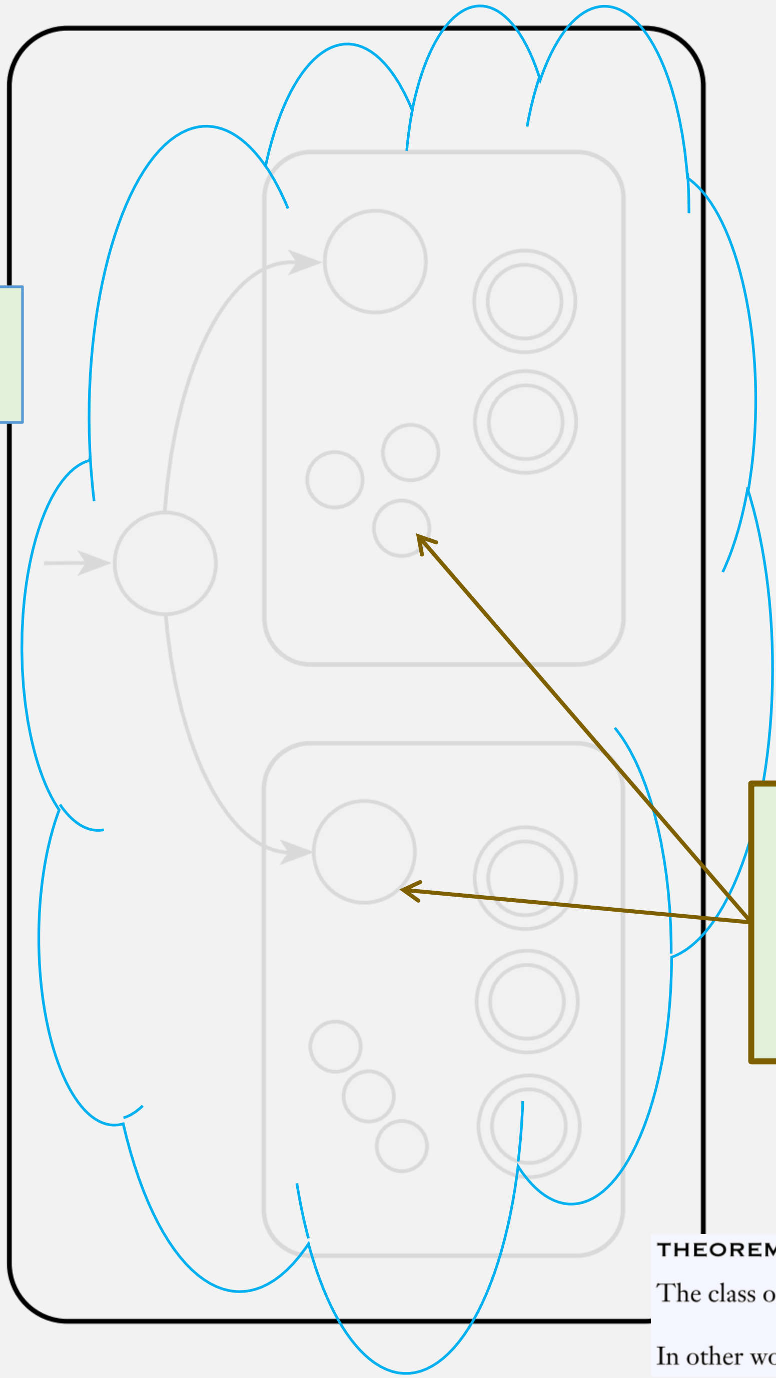
recognizes A_2



Want: M

Recognizes $A_1 \cup A_2$

(to prove $A_1 \cup A_2$ is regular)



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

- 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

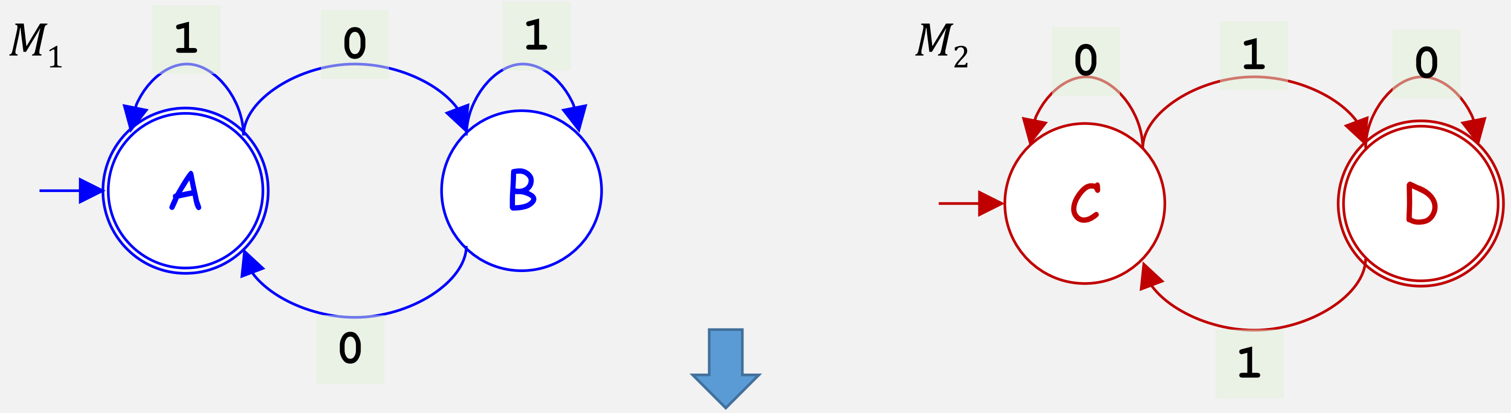
A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

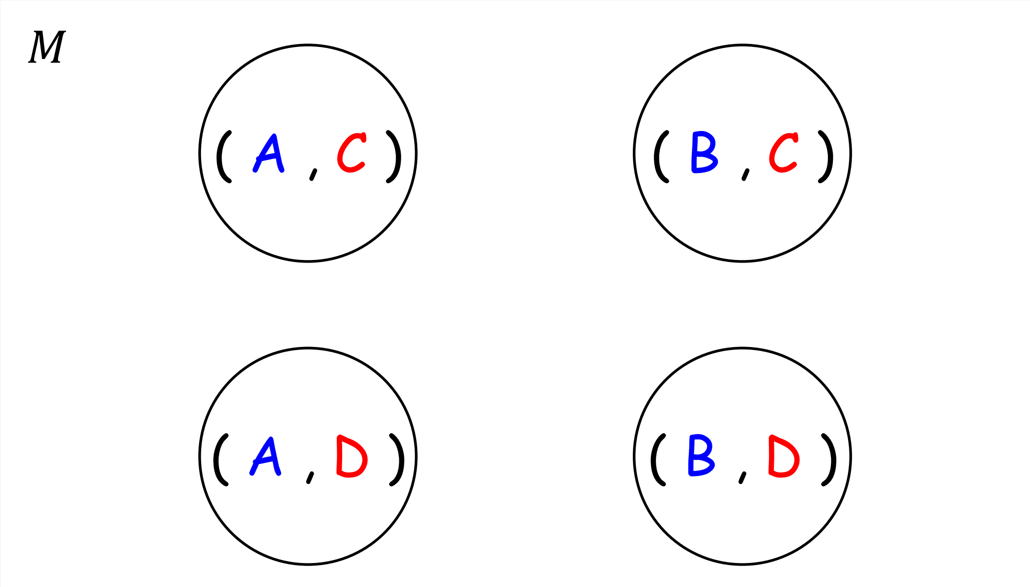
1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta \colon Q \times \Sigma \longrightarrow Q$ is the ***transition function***,[1]
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

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

states of M: all possible pair combinations of states of M_1 and M_2

DFA Union Example





Union is Closed For Regular Languages

Proof (continuation)

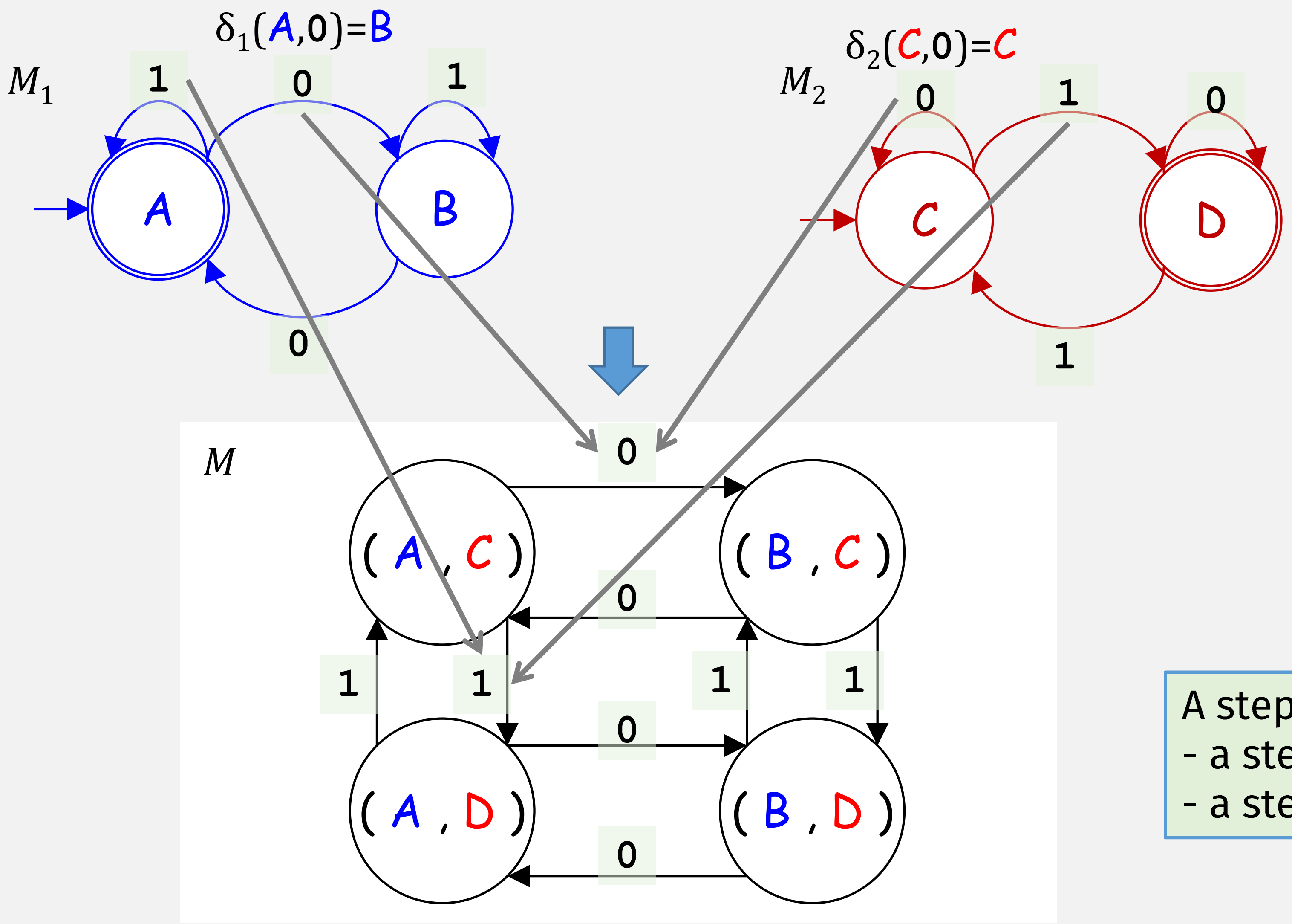
- Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1,
  $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2,
- Construct: $M = (Q, \Sigma, \delta, q_0, F)$, using M_1 and M_2, that recognizes $A_1 \cup A_2$
- states of M: $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$ $= Q_1 \times Q_2$
  This set is the ***Cartesian product*** of sets Q_1 and Q_2

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

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. $\delta \colon Q \times \Sigma \longrightarrow Q$ 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 step in M is <u>both</u>:
- a step in M_1, and
- a step in M_2



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

https://cs.wellesley.edu/~cs235/fall10/lectures/14_DFA_operations_revised_2.pdf

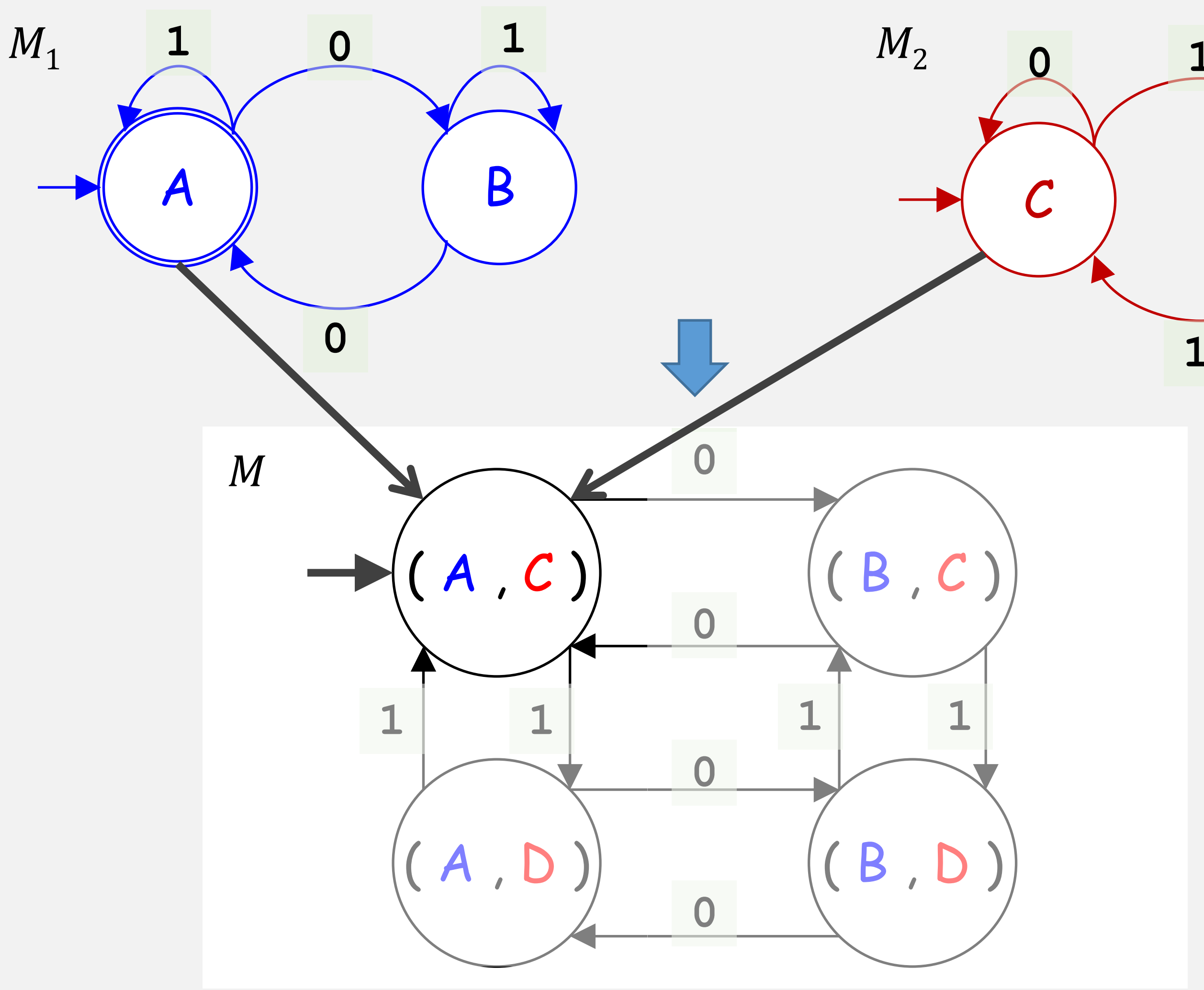
Union is Closed For Regular Languages

Proof (continuation)

- 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

DFA Union Example



Start state of M is <u>both</u> start states of M_1 and M_2

Union is Closed For Regular Languages

Proof (continuation)

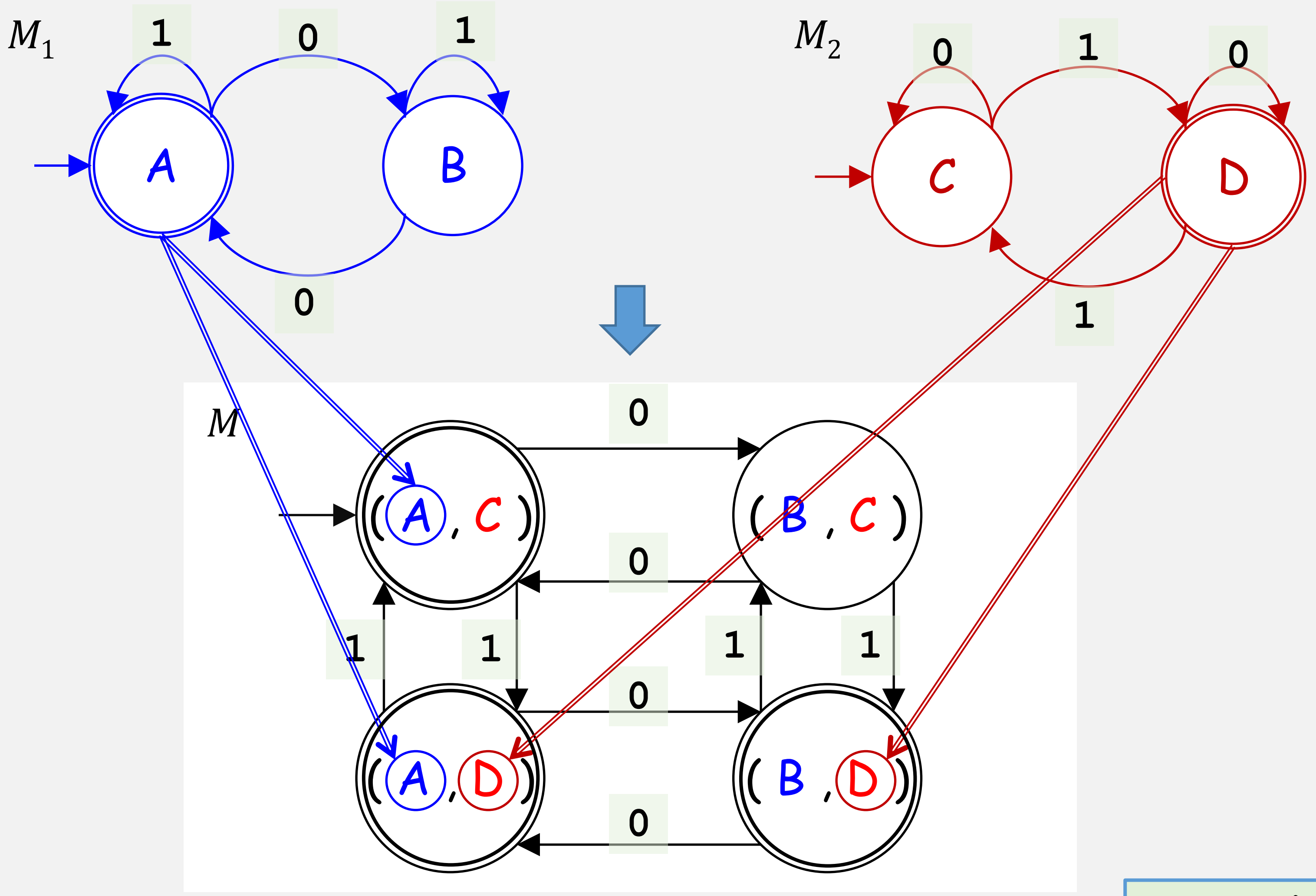
- Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1,
  $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2,
- Construct: $M = (Q, \Sigma, \delta, q_0, F)$, using M_1 and M_2, that recognizes $A_1 \cup A_2$
- states of M: $Q = \{(r_1, r_2) | \, r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$ $= Q_1 \times Q_2$
  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

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

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

DFA Union Example



Accept if either M_1 or M_2 accept

https://cs.wellesley.edu/~cs235/fall10/lectures/14_DFA_operations_revised_2.pdf

Union is Closed For Regular Languages

Proof (continuation)

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

Q.E.D.? ■

Previously

Is Union Closed For Regular Langs?

Statements	**Justifications**
1. A_1 and A_2 are regular languages	1. Assumption
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1	2. Def of Regular Language
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2	3. Def of Regular Language
??? 4. Construct DFA $M = (Q, \Sigma, \delta, q_0, F)$ ☑	4. Def of DFA
5. M recognizes $A_1 \cup A_2$	5. See examples (TODO!)
6. $A_1 \cup A_2$ is a regular language	6. Def of Regular Language
7. 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 $A_1 \cup A_2$.	7. From stmt #1 and #6

How to create this? Don't know what A_1 and A_2 are!

“Prove” that DFA recognizes a language

Let $s_1 \in A_1$ and $s_2 \in A_2$

Let $s_3 \notin A_1$ and $s_4 \notin A_2$

Be careful when choosing examples!

In this class, **a table like this is sufficient to “prove” that a DFA recognizes a language**

String	In lang $A_1 \cup A_2$?	Accepted by M ?
s_1	Yes	
s_2	Yes	
s_3	???	
s_4	???	

Don’t know A_1 and A_2 exactly … … but we know … … they are **sets of strings**!

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

constructed $M = (Q, \Sigma, \delta, q_0, F)$ recognizes $A_1 \cup A_2$?

"Prove" that DFA recognizes a language

Let $s_1 \in A_1$ and $s_2 \in A_2$

~~Let $s_3 \notin A_1$ and $s_4 \notin A_2$~~

Let $s_5 \notin A_1$ and $\notin A_2$

String	In lang $A_1 \cup A_2$?	Accepted by M ?
s_1	Yes	
s_2	Yes	
~~s_3~~	~~???~~	
~~s_4~~	~~???~~	
s_5	No	

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

constructed $M = (Q, \Sigma, \delta, q_0, F)$ recognizes $A_1 \cup A_2$?

Union is Closed For Regular Languages

Proof (continuation)

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

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

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

"Prove" that DFA recognizes a language

Let $s_1 \in A_1$ and $s_2 \in A_2$

(this column needed when **machine is not concrete**, i.e., **can't directly run machine** to check if string is accepted)

Let $s_5 \notin A_1$ and $\notin A_2$

String	In lang $A_1 \cup A_2$?	Accepted by M ?	Justification
s_1	Yes	Accept ??	(J1)
s_2	Yes	Accept	(J1)
~~s_3~~	~~???~~	???	
~~s_4~~	~~???~~	???	
s_5	No	Reject ??	(J2)

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

a string $\in A_1 \rightarrow$ accepted by $M_1 \rightarrow$ accepted by M (J1)

string $\notin A_1$ and $\notin A_2 \rightarrow M_1$ and M_2 rejects $\rightarrow M$ rejects (J2)

constructed $M = (Q, \Sigma, \delta, q_0, F)$ to Accept if <u>either</u> M_1 or M_2 accept Else reject

$$F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$$

Previously

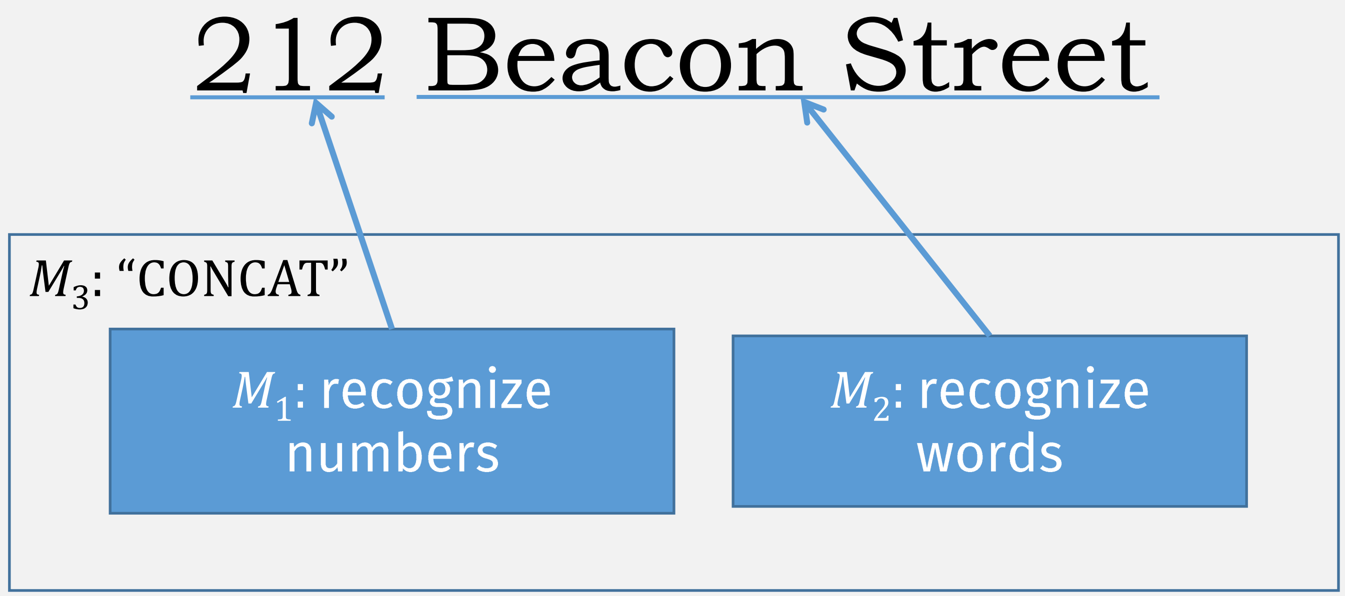
Is Union Closed For Regular Langs?

Statements	**Justifications**
1. A_1 and A_2 are regular languages	1. Assumption
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1	2. Def of Regular Language
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2	3. Def of Regular Language
4. Construct DFA $M = (Q, \Sigma, \delta, q_0, F)$ ☑	4. Def of DFA
5. M recognizes $A_1 \cup A_2$	5. See Examples Table ☑
6. $A_1 \cup A_2$ is a regular language	6. Def of Regular Language
7. 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 $A_1 \cup A_2$.	7. From stmt #1 and #6

Q.E.D. ∎

Another (common string) 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 $A = \{\mathtt{fort}, \mathtt{south}\}$ $B = \{\mathtt{point}, \mathtt{boston}\}$

$$A \circ B = \{\mathtt{fortpoint}, \mathtt{fortboston}, \mathtt{southpoint}, \mathtt{southboston}\}$$

Is Concatenation Closed?

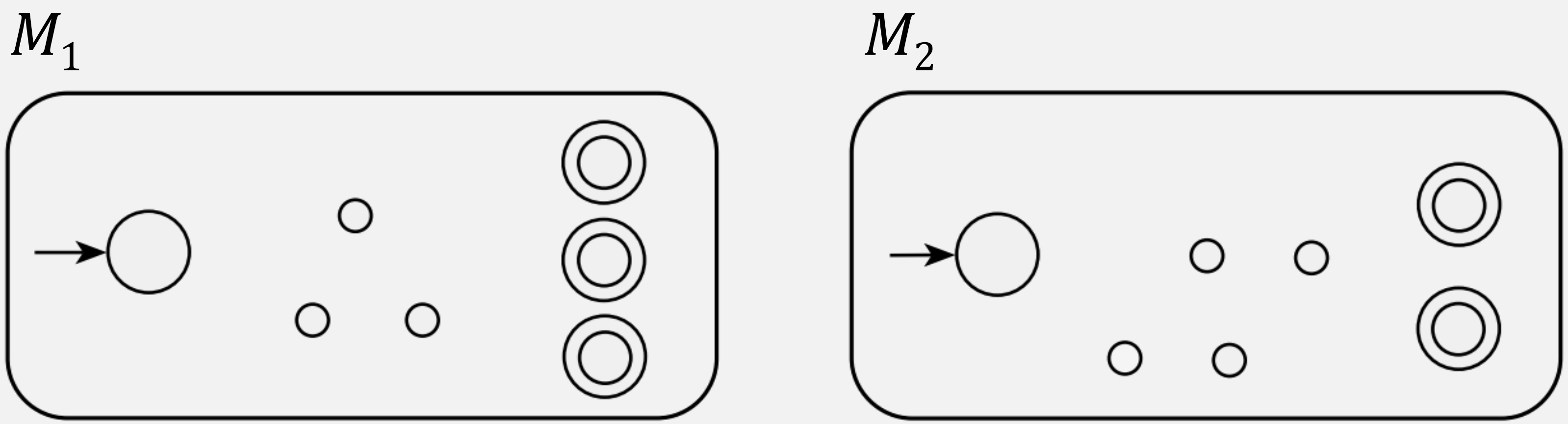
THEOREM

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

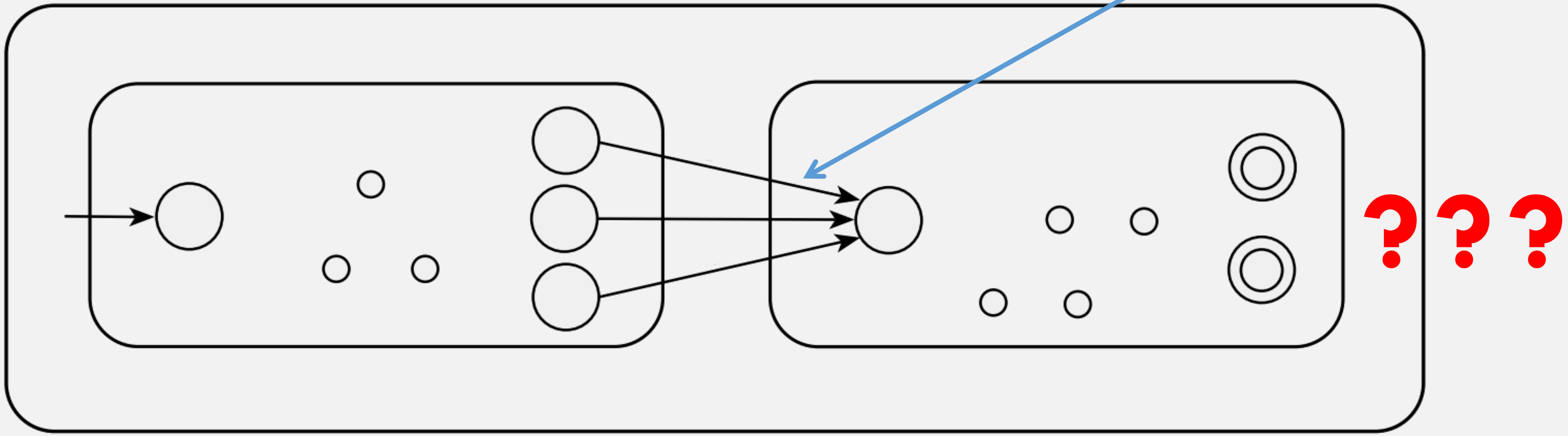


PROBLEM: Can only read input once, can't backtrack

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 <u>either</u>:

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 either:
  - stay in M_1 (correct, if full input is jenssmith)
  - or switch to M_2 (correct, if full input is jensmith)
- But to recognize $A \circ B$, it needs to handle both cases!!
  - Without backtracking

A **DFA** can't do this!

Is Concatenation Closed?

FALSE?

THEOREM

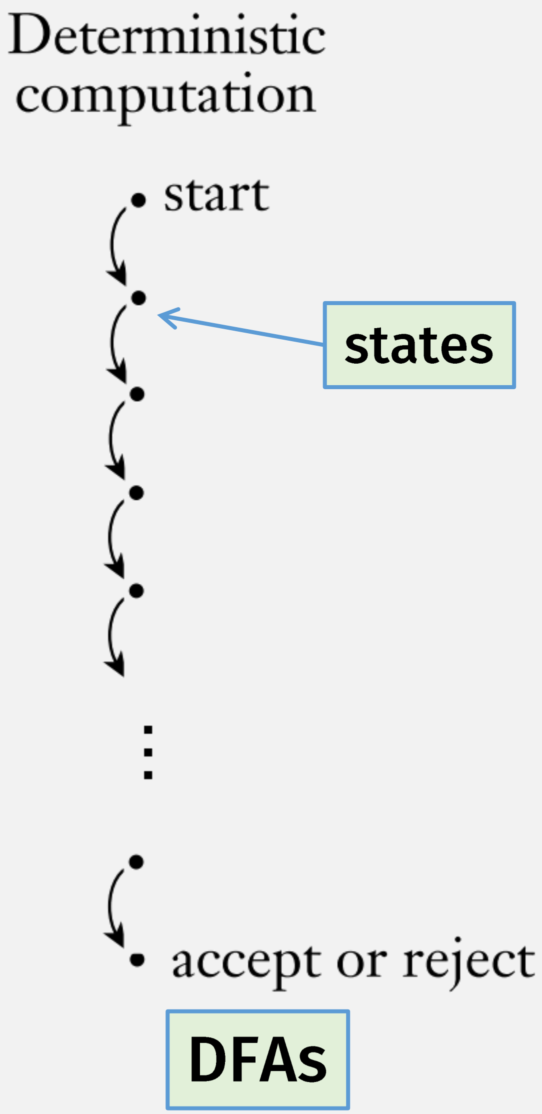
The class of regular languages is closed under the concatenation operation.

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

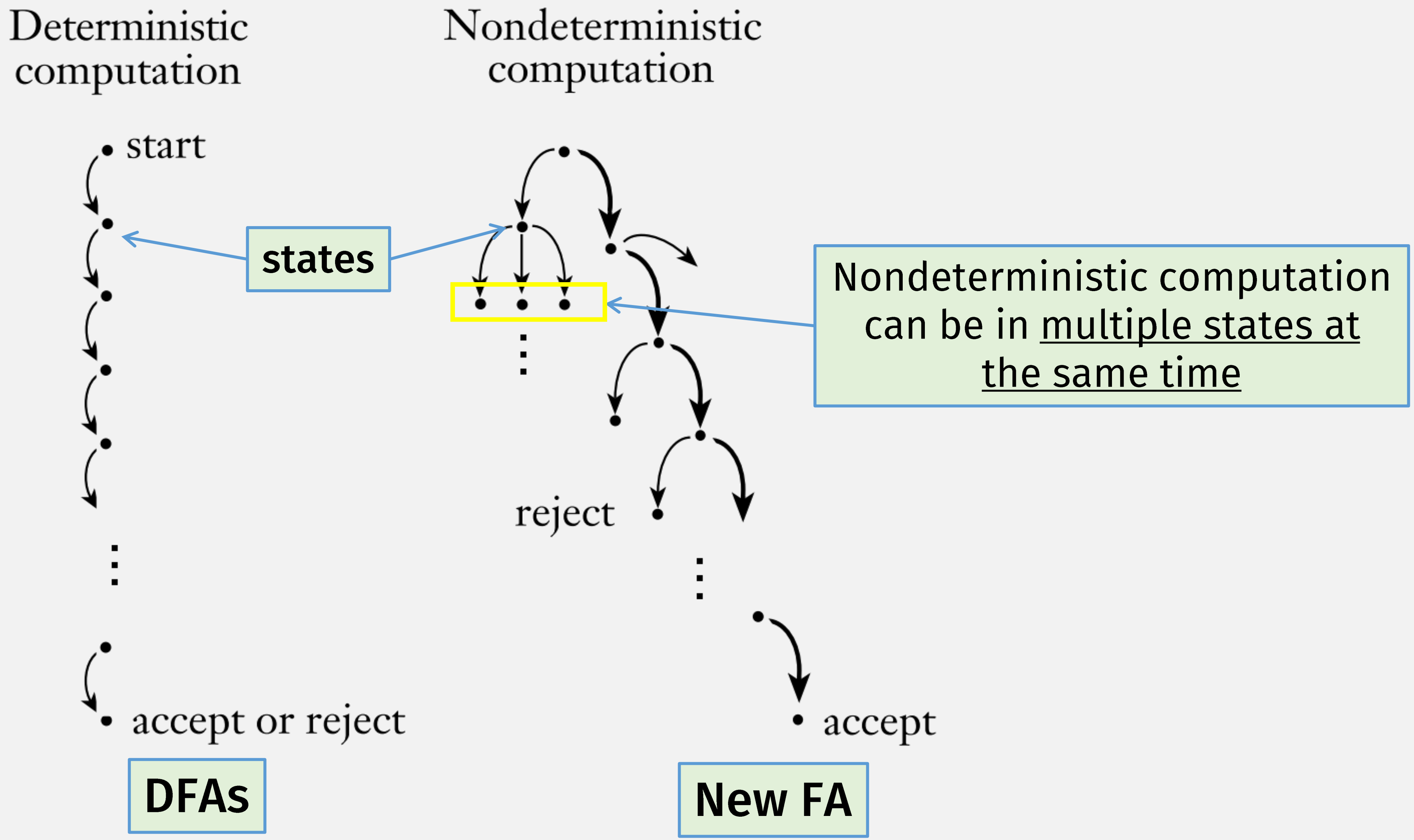
- Cannot **combine A_1 and A_2's machine because:**
  - **Need to switch from A_1 to A_2 at some point …**
  - … but **we don't know when!** (we can only read input once)
- **This requires a new kind of machine!**
- But **does this mean concatenation is not closed for regular langs?**

Nondeterminism

Deterministic vs Nondeterministic



Deterministic vs Nondeterministic



Previously

DFAs: 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***.

Deterministic Finite Automata (DFA)

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

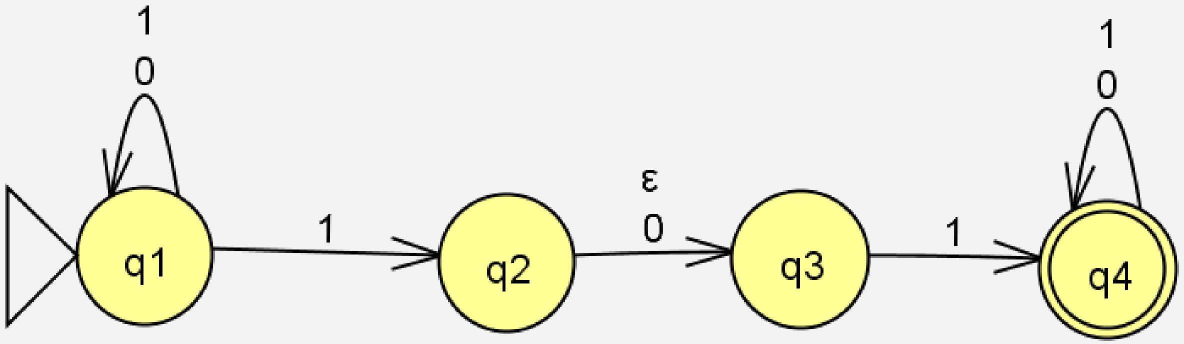
Transition label can be "empty", i.e., machine can transition without reading input

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

CAREFUL:
ε symbol is reused here, as a transition label (ie, an argument to δ)
- **It's not the empty string!**
- And it's (still) not a character in the alphabet Σ!

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

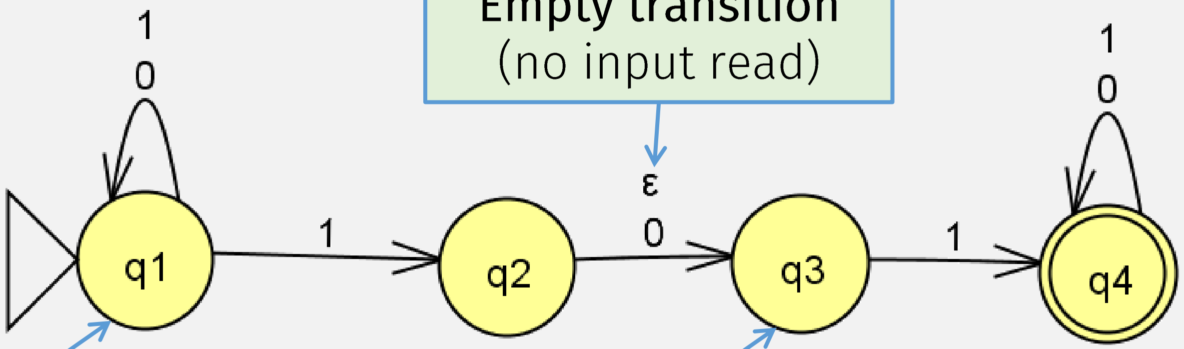
Empty transition (no input read)

	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)

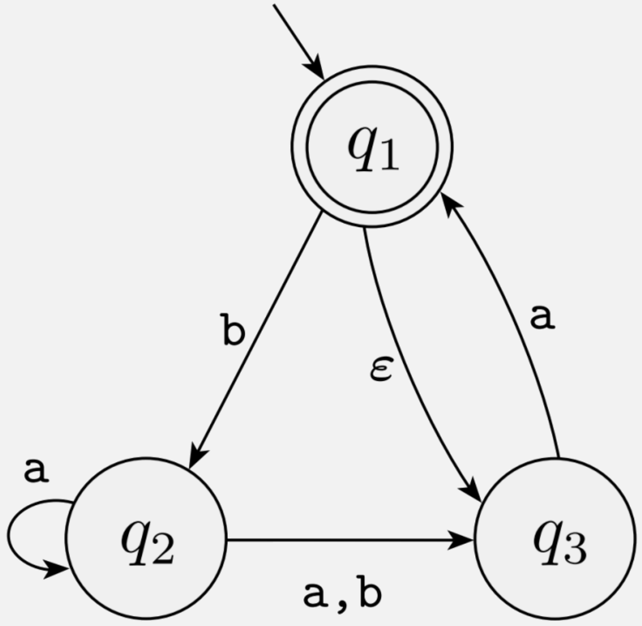


Multiple 1 transitions

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

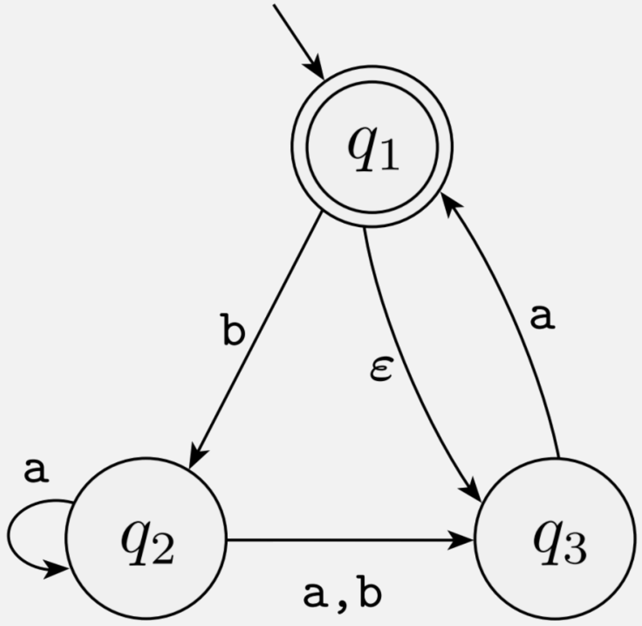
1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta \colon Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

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 ...
- $q_0 = q_1$
- $F = \{ q_1 \}$

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



NFA Computation (JFLAP demo): `010110`