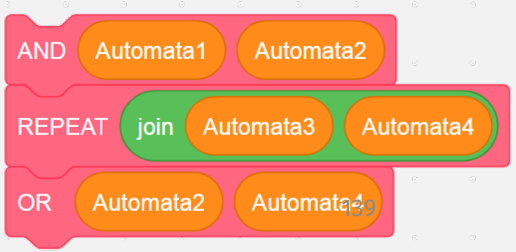
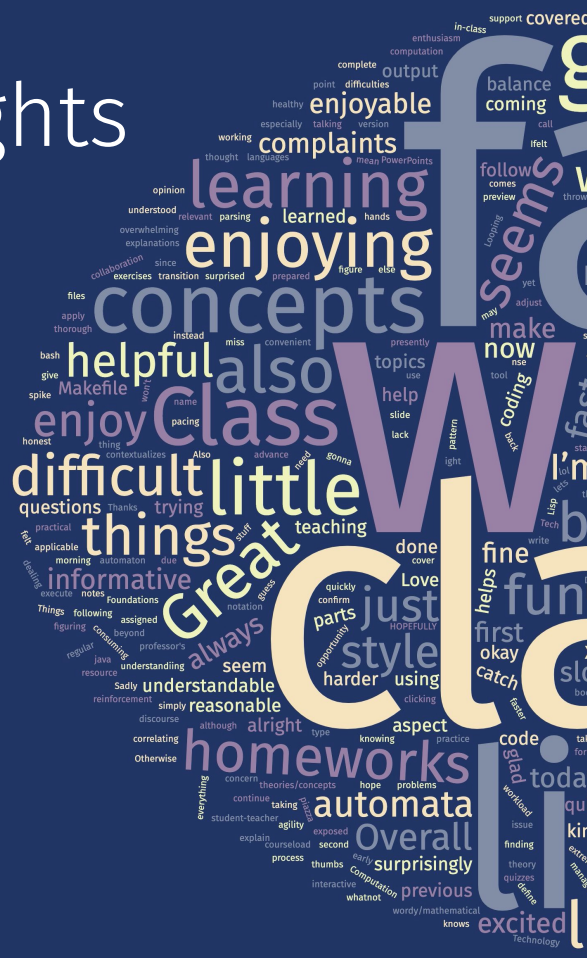
Combining Regular Languages



Your thoughts



Logistic Stuff

- Exam 1: 8AM – 10PM Saturday Oct 16
- Exam 2: 8AM – 10PM Saturday Nov 20 (where it has to be)
- See Schedule

- Enrichment page + PRs
- **HW Questions?** (HW1? HW2 out)
- General questions I can answer?

Last Time: Regular languages

- Definition: A language is ***regular*** if and only if it is decided by some dfa. (We will see other equivalent characterizations)

- Big question: Are there any decidable languages that are not regular?

- Answer: yes!
  - This is exciting. We have a mismatch between our first putative model of computation and classes of languages.

Last Time: In-class exercise

- Prove that this language is a regular language:
  - {w | w has exactly three 1's}
  - i.e., design a finite automata that recognizes it!

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

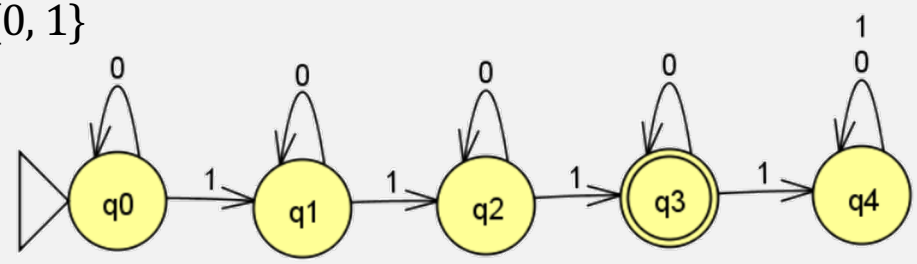
Last Time: In-class exercise

- Design finite automata recognizing:
  - {w | w has exactly three 1's}

So finite automata are used to recognize dumb patterns in strings???

Yes!

- *States*:
  - Need one state to represent how many 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*:
  - $\{q_3\}$

Seeking closure

“Closed” Operations on Sets

- Natural numbers = {0, 1, 2, ...}
  - Closed 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 **closed** under some operation if applying that operation to any members of the set always produces an element in the set.

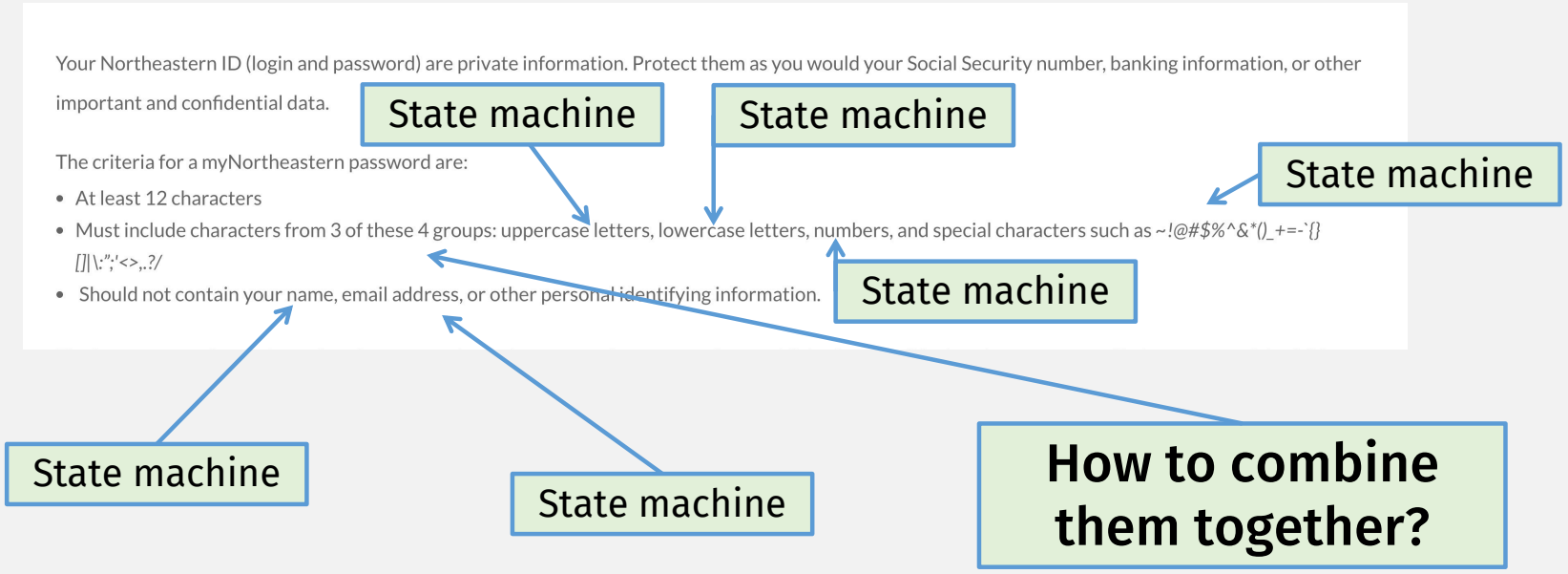
Closed Ops on Reg Langs:

Why might **you** actually care?

Why Care About Closed Ops on Reg Langs?

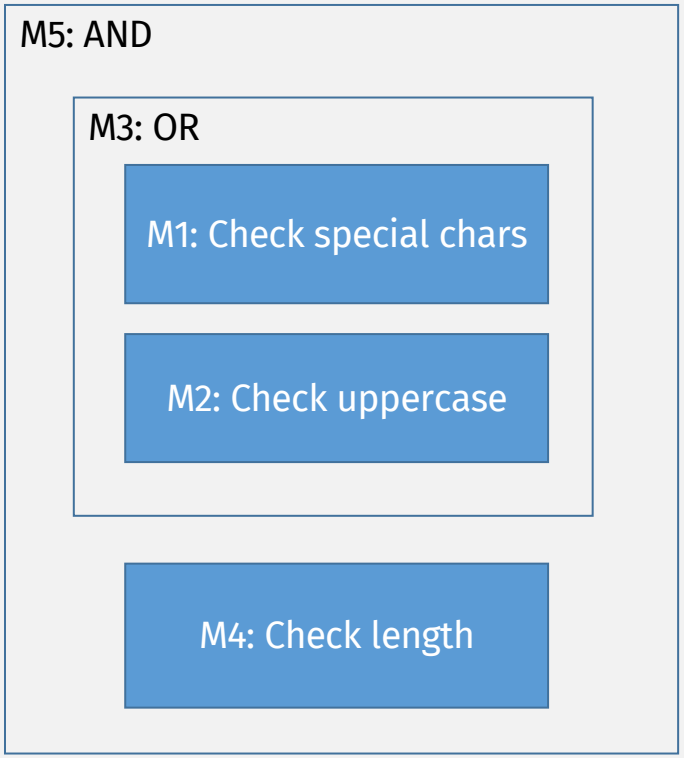
- Closed operations preserves “regularness”
- I.e., it preserves the same computation model
- So result of combining machines can be combined again
- A well-defined class

From: https://its.northeastern.edu/2020/11/30/reminder-choose-a-secure-password-and-keep-it-safe/



Password checker

"Must contain either an uppercase letter or a 'special character' and must meet a minimum length."



Want to be able to easily combine finite automata machines

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

Composition

- of Machines
- of Programs
- of Computations

Operations on Regular Languages

DEFINITION 1.23

Let A and B be languages. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows:

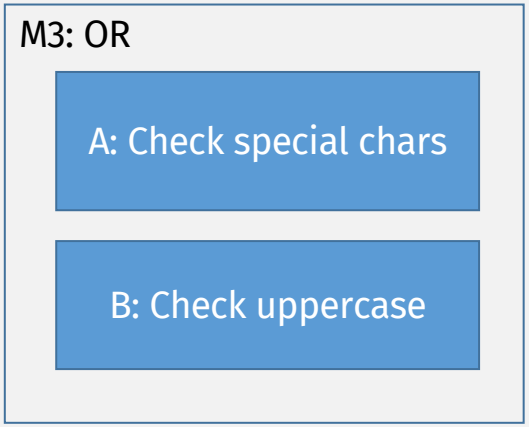
- **Union**: $A \cup B = \{x | \ x \in A \text{ or } x \in B\}$.
- **Concatenation**: $A \circ B = \{xy | \ x \in A \text{ and } y \in B\}$.
- **Star**: $A^* = \{x_1 x_2 \ldots x_k | \ k \geq 0 \text{ and each } x_i \in A\}$.

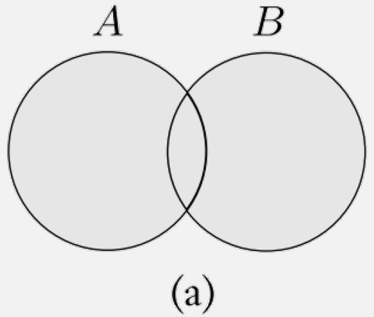
Operations on Languages

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

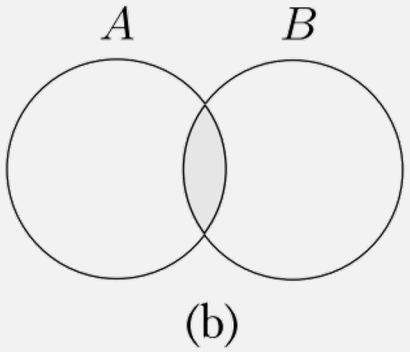
Union: {a,ab,aaaa} ∪ {bb,ab} = {a,ab,bb,aaaa}

Password checker: “Or” = “Union”

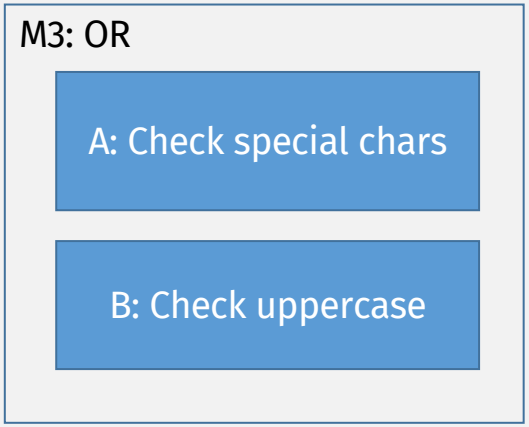


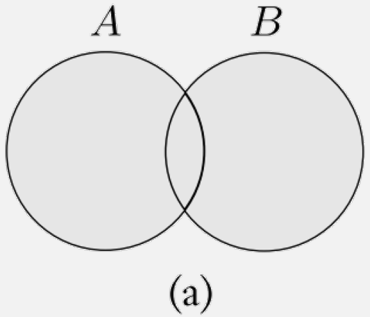


???



Password checker: “Or” = “Union”





A Closed Operation: Union

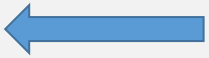
THEOREM 1.25

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

Un A Closed Operation?

25

The cl uages is closed under on.

In other words, i is $A_1 \cup A_2$.

Proof

- Given:
  - machine **M_1** (with st recognizing A_1, and
  - machine **M_2** (nd trans gnizing A_2
- Construc ine **M**, using **M_1** an
- Given runs it on **M_1**. If **M_1** rejects, **M** hat input
- So a s of **M** is in $Q_1 \cup Q_2$ (union **M_1** and **M_2**'s state
- **M**'s transition fn …

Union Closed?

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 hw2)

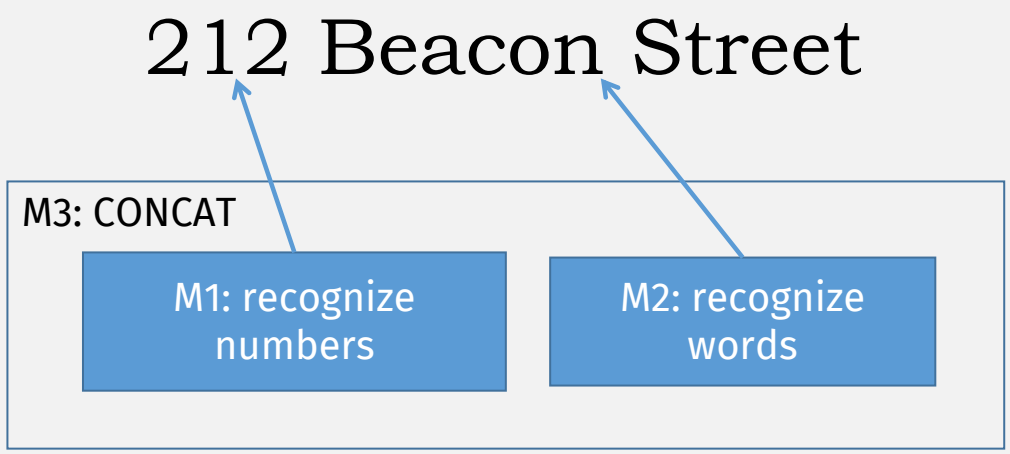
- Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1,
  $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2,
- Construct a new machine $M = (Q, \Sigma, \delta, q_0, F)$ using M_1 and M_2

M runs its input on both M_1 and M_2 in parallel; accept if either accepts

- states of M: $Q = \{(r_1, r_2) | \, r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
  This set is the ***Cartesian product*** of sets Q_1 and Q_2
- M's 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: $F = \{(r_1, r_2) | \, r_1 \in F_1 \text{ or } r_2 \in F_2\}$

Another operation: Concatenation

- Example: Matching street addresses



Concatenation

- Definition: $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

- Example: $\{a, ab, ba\}\{b, aa\}$

$$= \{ab, aaa, abb, abaa, bab, baaa\}$$

- $L = \{a^n b^n : n \geq 0\}$

$$L^2 = \{a^n b^n a^m b^m : n, m \geq 0\}$$

$$aabbaaabbb \in L^2$$

Another Operation

$$L^n = \underbrace{LL\cdots L}_{n}$$

- Definition:

$$\{a,b\}^3 = \{a,b\}\{a,b\}\{a,b\} =$$

$$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

- Special case: $L^0 = \{\varepsilon\}$

$$\{a, bba, aaa\}^0 = \{\varepsilon\}$$

Is Concatenation Closed?

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 $A_1 \circ A_2$.

- Construct a <u>new</u> machine M? (like union)
  - From DFA M_1 (which recognizes A_1),
  - and DFA M_2 (which recognizes A_2)

Is Concatenation Closed?

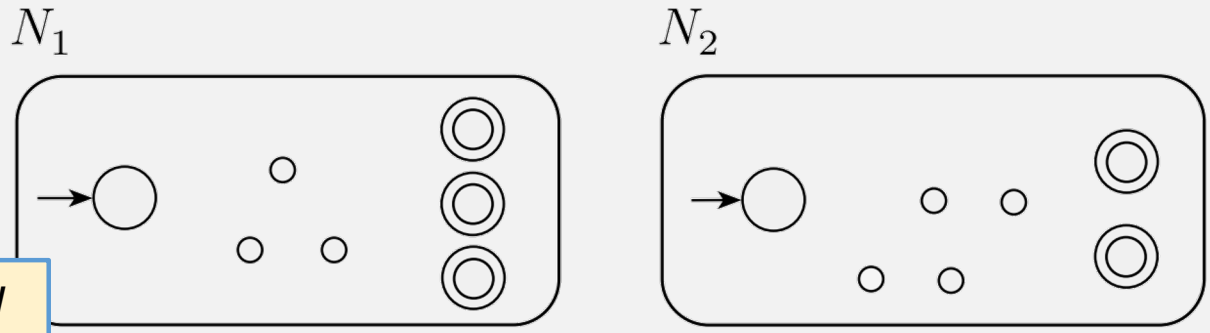
THEOREM 1.26

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)
- Need a new kind of machine!
- So is concatenation not closed for reg langs???

Concatentation



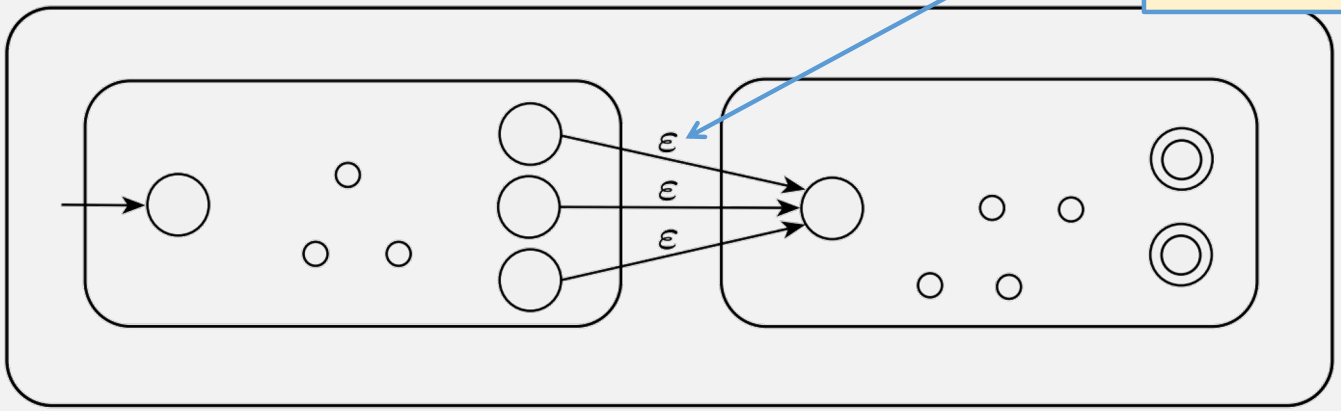
N is a new kind of machine, an NFA! (next time)

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



Star-Closure (Kleene *)
AKA “Unbounded repetition”

- All strings that can be constructed from L
- Definition: $L^* = L^0 \bigcup L^1 \bigcup L^2 \cdots$
- Example: {a,bb}* = {ε,
  a, bb
  aa, abb, bba, bbbb
  aaa, aabb, abba, abbbb … }

Positive Closure

$$L^+ = L^1 \cup L^2 \cup \cdots$$

- Definition: Same with L^* but without the ε

$$\{a, bb\}^+ = \left\{ \begin{array}{l} a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array} \right\}$$

More Operations on Languages

- The other usual set operations

{a,ab,aaaa} ∪ {bb,ab} = {a,ab,bb,aaaa}

{a,ab,aaaa} ∩ {bb,ab} = {ab}

{a,ab,aaaa} – {bb,ab} = {a,aaaa}

More Operations on Languages

- Complement: $\overline{L} = \Sigma^* - L$

$$\overline{\{a, ba\}} = \{\lambda, b, aa, ab, bb, aaa, \ldots\}$$

More Operations on Languages

Reverse $L^R = \{w^R : w \in L\}$

- Definition: $\{ab, aab, baba\}^R = \{ba, baa, abab\}$

- Examples: $L = \{a^n b^n : n \geq 0\}$

$$L^R = \{b^n a^n : n \geq 0\}$$

Check-in Quiz

On gradescope