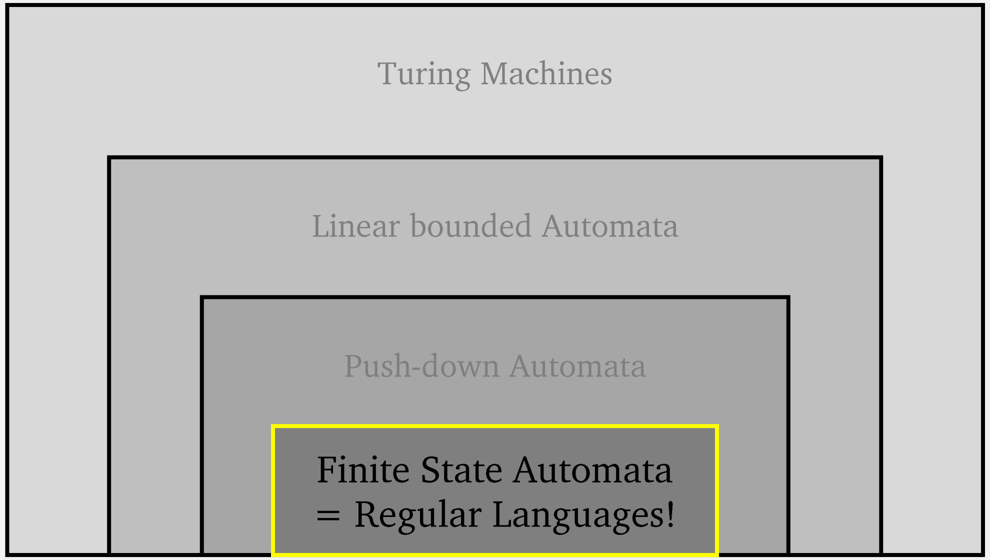
CS622

Proving a Language is Regular

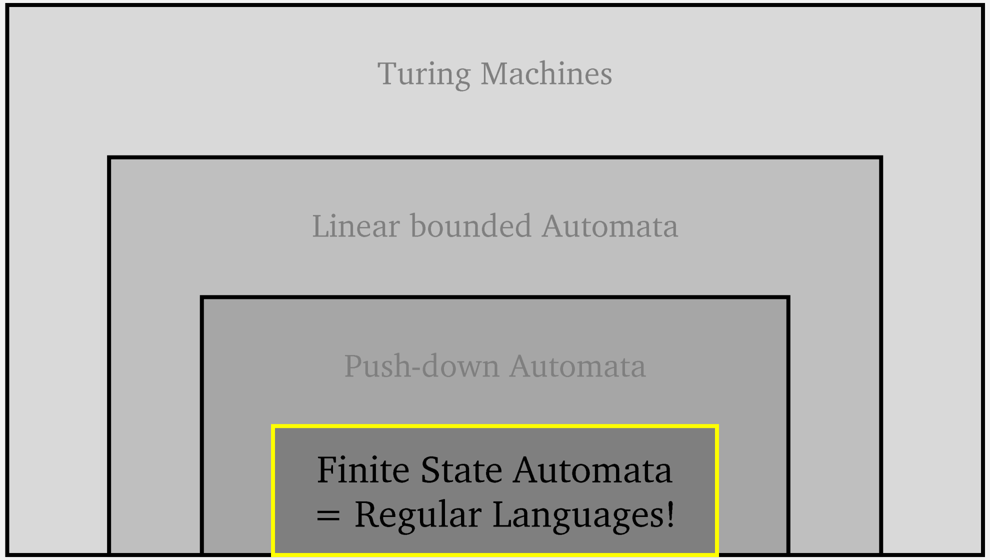
Friday, February 9, 2024

UMass Boston Computer Science



Announcements

- HW 1
  - <u>Due</u>: Mon 2/12 12pm (noon)



Previously

Languages Are Computation Models

- The **language** of a **machine** = set of strings that it **accepts**
  - E.g., a DFA **recognizes** a language

- A **computation model** = set of machines it **defines**
  - E.g., **all possible DFAs** are a **computation model**

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***.

= set of set of strings

Thus: a **computation model** equivalently = a set of languages

This class is really about studying **sets of languages!**

Previously

Regular Languages

- first **set of languages** we will study: **regular languages**

This class is <u>really</u> about studying **sets of languages!**

Previously

Regular Languages: Definition

If a **deterministic finite automata** (**DFA**) recognizes a language, then **that language** is called a **regular language**.

Previously

A Language, Regular or Not?

- If given: **a DFA** M
  - We know: $L(M)$, the **language recognized by** M, is a **regular language**

Proof : If a **DFA** recognizes a **language**,
then **that language** is called a **regular language**.

(modus ponens)

- If given: **a Language** A
  - Is A a **regular language?**
    - **Not necessarily!**

Proof : ??????

In-class exercise 2: Language

Remember:
To understand a language, **always come up with string examples first** (in a table)! Both:
- **in the language**
- and **not in the language**

- Prove: the following **language** is a **regular language**:
  - $A = \{ w \mid w \text{ has exactly three 1's} \}$

(You will need this later in the proof anyways!)

String	In the language?
1	No
0	No
11	No
111	Yes
1101	Yes
11011	No

Where $\Sigma = \{0, 1\}$,

Previously

Proving That a Language is Regular

Prove: A **language** L = { … } is a **regular language**

Proof:

Statements	**Justifications**
1. DFA $M = (Q, \Sigma, \delta, q_0, F)$ (TODO: actually define M) (no unbound variables!)	1. Definition of a DFA
2. **DFA** M **recognizes** L	2. TODO: ???
3. **If a DFA recognizes** L**, then** L **is a regular language**	3. Definition of a regular language
4. **Language** L **is a regular language**	4. Stmts 2 and 3 (and **modus ponens**)

Sipser skips this step! (but you should not)

Proving = puzzle, i.e., "pieces" that "fit together"

Modus Ponens

If we can prove these:

- If P then Q
- P

Then we've proved:

- Q

Proving That a Language is Regular

Prove: A **language** L = { … } is a **regular language**

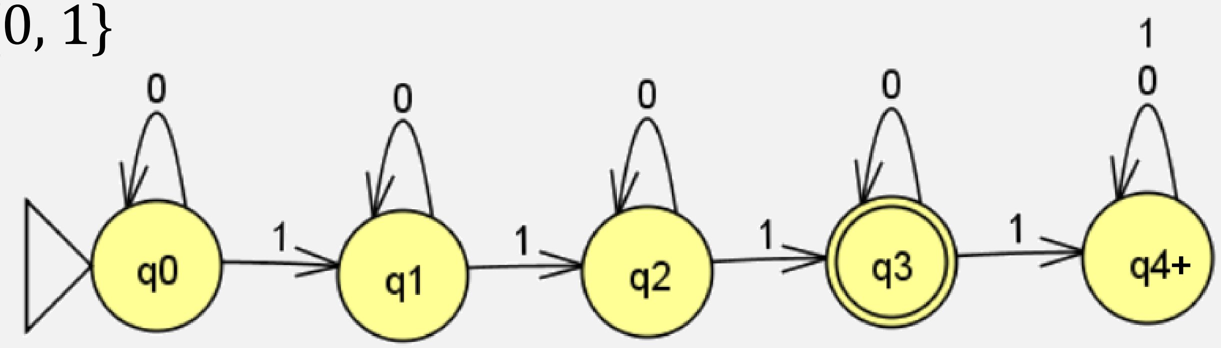
Proof:

Statements	**Justifications**
1. DFA $M = (Q, \Sigma, \delta, q_0, F)$ (TODO: actually define M) (no unbound variables!)	1. Definition of a DFA
2. DFA M recognizes L	2. TODO: ???
3. If a DFA recognizes L, then L is a regular language	3. Definition of a regular language
4. Language L is a regular language	4. Stmts 2 and 3 (and **modus ponens**)

In-class exercise 2: DFA

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

- *States*:
  - Need a separate state to represent: “seen zero 1s”, “seen one 1”, “seen two 2s”, …
  - $Q = \{q_0, q_1, q_2, q_3, q_{4+}\}$
- *Alphabet*: $\Sigma = \{0, 1\}$
- *Transitions*:



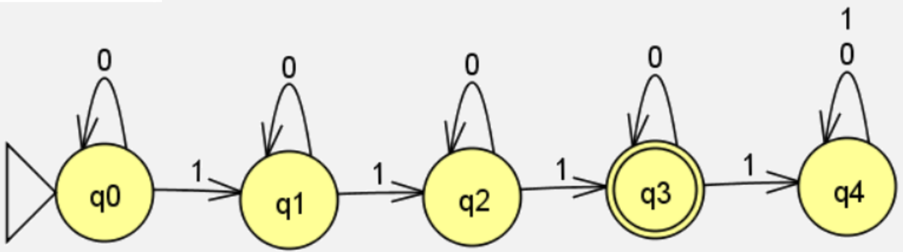
- *Start state*:
  - q_0
- *Accept states*:
  - $\{q_3\}$

In-class exercise 2: DFA Recognizes Lang

- Prove: the following **language** is a **regular language**:
  - $A = \{ w \mid w \text{ has exactly three 1's} \}$

String	In the language?	Accepted by machine?
1	No	Reject
0	No	Reject
11	No	Reject
111	Yes	Accept
1101	Yes	Accept
11011	No	Reject

Where $\Sigma = \{0, 1\}$,



Proving That a Language is Regular

- Prove: language $A = \{ w \mid w \text{ has exactly three 1's} \}$ is a regular language

Proof:

Statements	Justifications
☑ 1. DFA $M =$	1. Definition of a DFA
See state diagram (only if problem allows!)	
2. DFA M recognizes A	☑ 2. See examples table
3. If a DFA recognizes A, then A is a regular language	3. Definition of a regular language
4. Language A is a regular language	4. Stmts 2 and 3 (and modus ponens)

■

In-class exercise 2: Solution

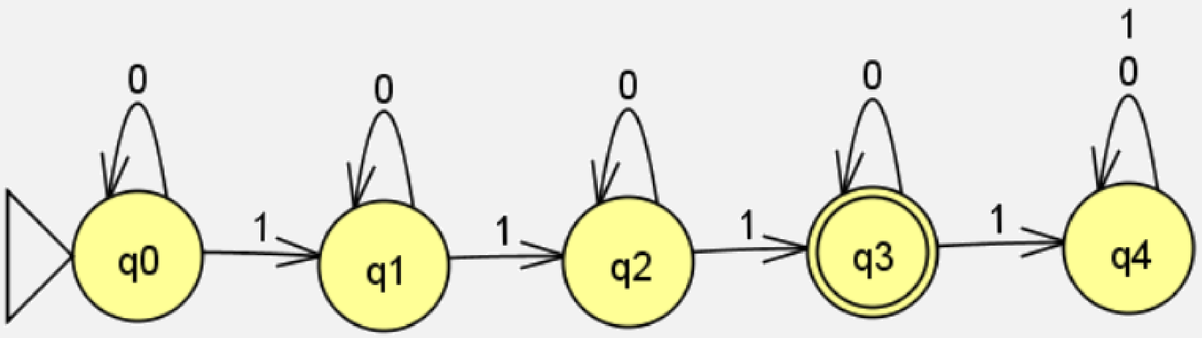
- Design finite automata recognizing:
  - $\{w \mid w$ has exactly three 1's$\}$

So: a DFA's computational model (regular languages) represents string matching computations??

Yes!

programming language (feature) to recognize simple string patterns?

Regular expressions!



Combining DFA computations?

Password Requirements

DFA

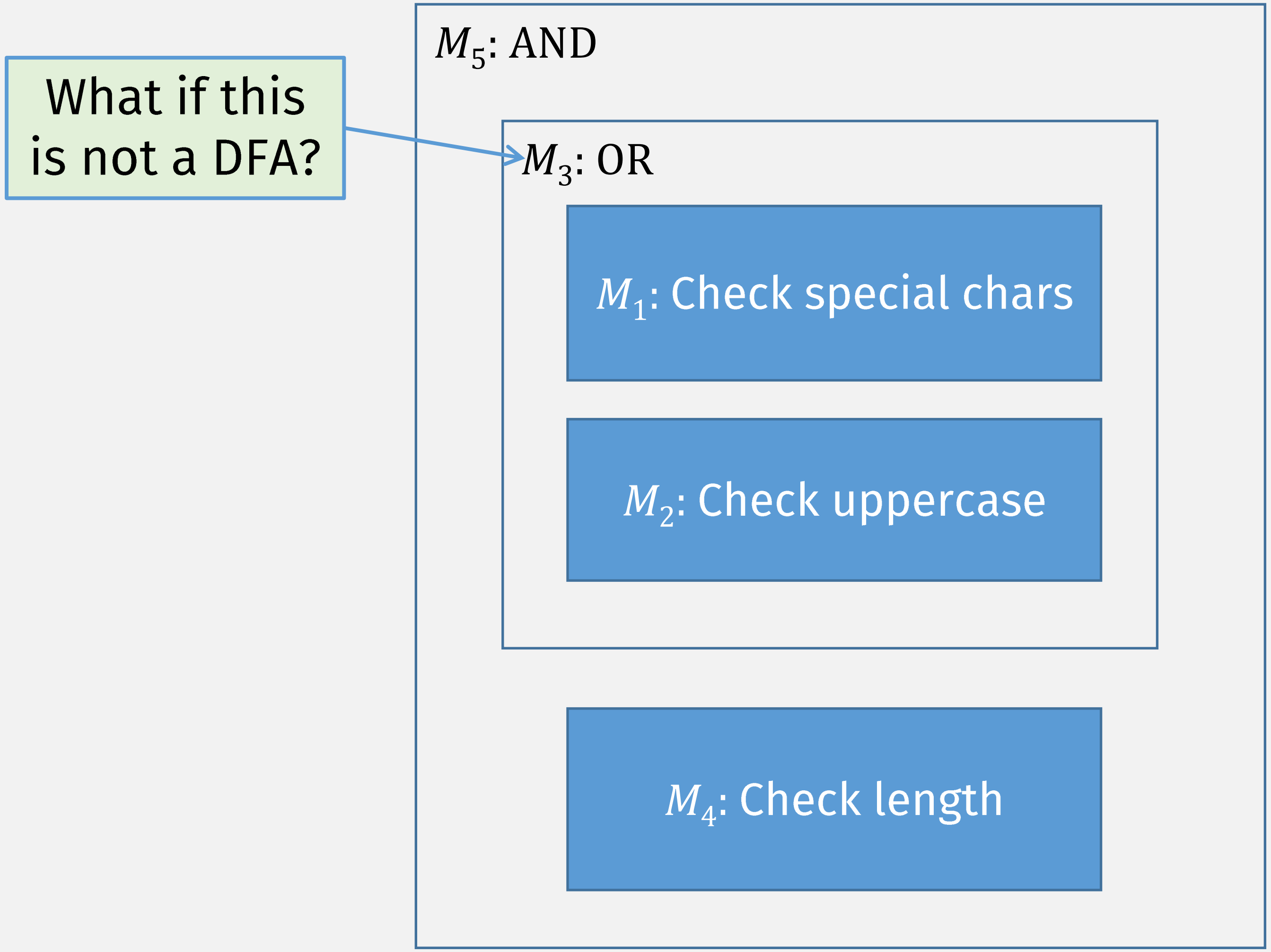
- Passwords must have a minimum length of ten (10) characters - but more is better!
- Passwords **must include at least 3** different types of characters:
  - upper-case letters (A-Z) ← DFA
  - DFA → lower-case letters (a-z)
  - symbols or special characters (%, &, *, $, etc.) ← DFA
  - numbers (0-9)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

To match <u>all</u> requirements, <u>combine</u> smaller DFAs into one big DFA?

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

(We do this with programs all the time)

Password Checker DFAs



Want to be able to easily combine DFAs, i.e., composability

We want these operations:

$\text{OR} : \text{DFA} \times \text{DFA} \rightarrow \text{DFA}$

$\text{AND} : \text{DFA} \times \text{DFA} \rightarrow \text{DFA}$

To combine more than once, operations must be **closed**!

"Closed" Operations

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

- 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** $z\ !{=}0$

Why Care About Closed Ops on Reg Langs?

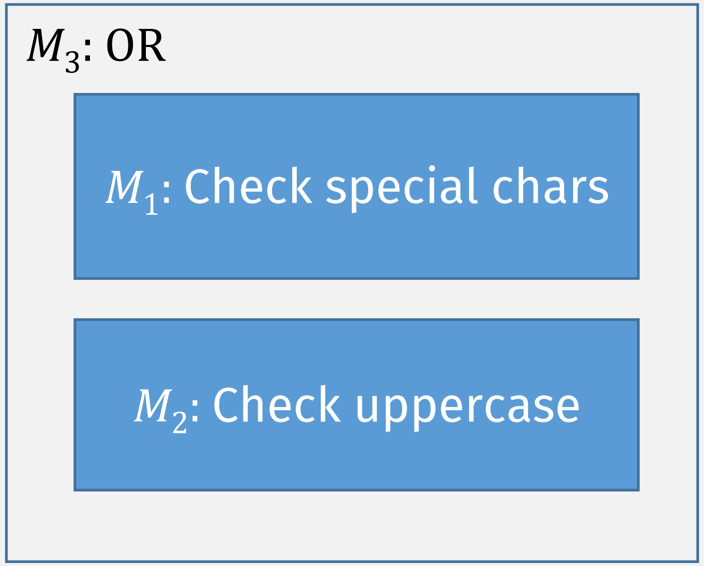
- Closed operations preserve "regularness"
- I.e., it preserves the same computation model!
- This way, a "combined" machine can be "combined" again!

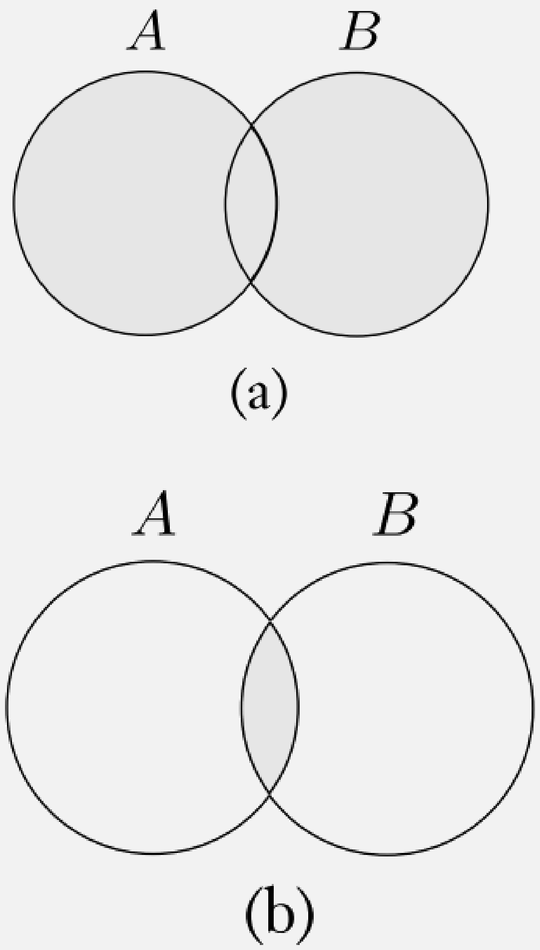
We want:

OR, AND : DFA × DFA → DFA

- So this semester, we will look for operations that are closed!

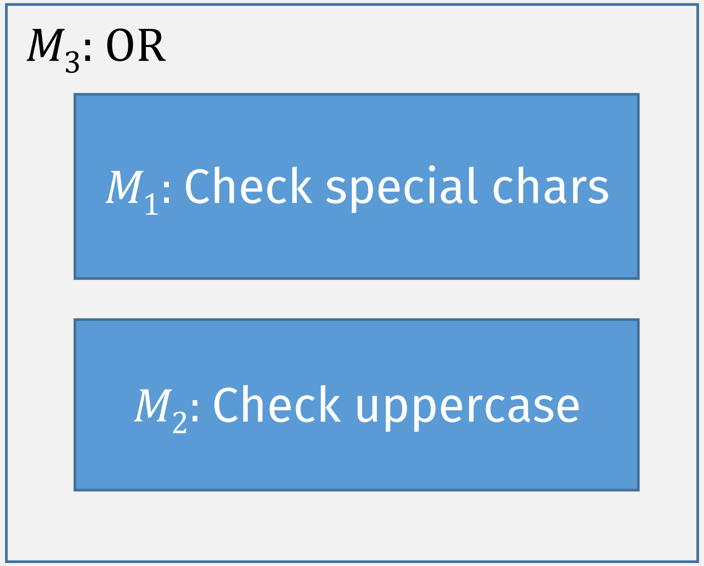
Password Checker: “**OR**” = “Union”

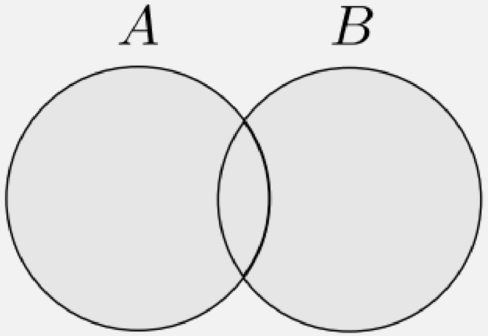




???

Password Checker: “**OR**” = “Union”





(a)

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 $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$.

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

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

Submit 2/9 in-class work to gradescope