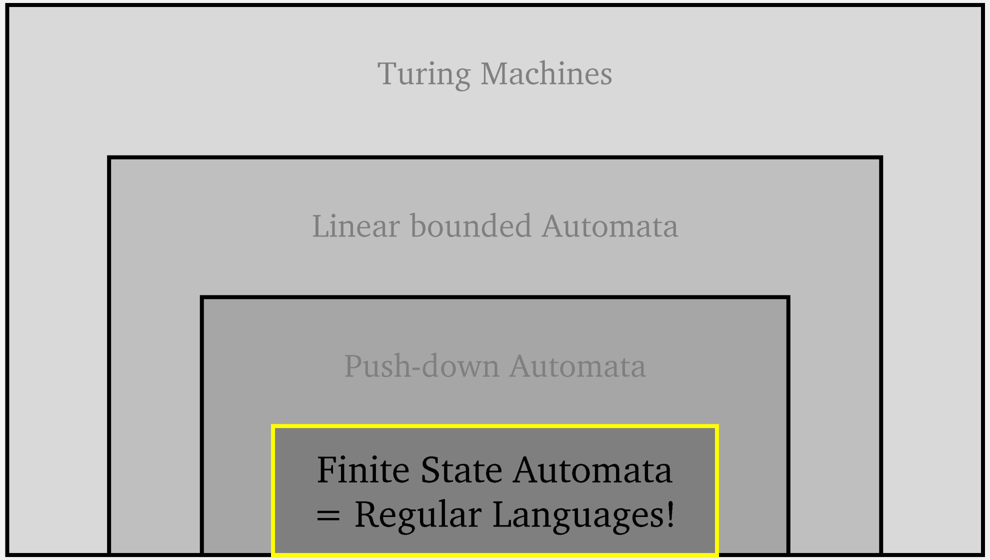
CS622

Regular Languages

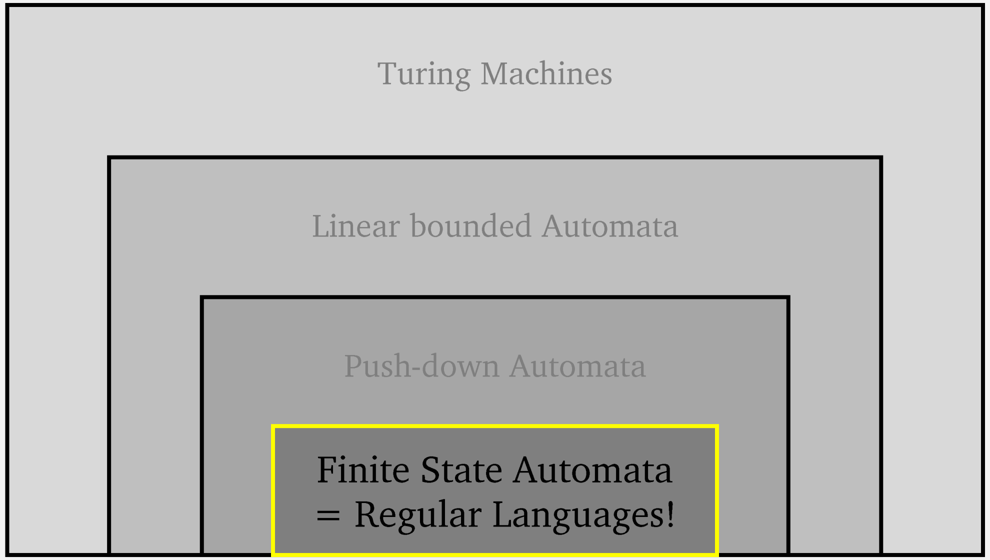
Wednesday, February 7, 2024

UMass Boston Computer Science



Announcements

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



Previously

Alphabets, Strings, Languages

An **alphabet** defines "**all possible strings**"

- An **alphabet** is a non-empty finite set of **symbols**

(strings with non-alphabet symbols are impossible)

$$\Sigma_1 = \{0,1\}$$

$$\Sigma_2 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$$

- A **string** is a finite sequence of **symbols** from an **alphabet**

01001 abracadabra ε

Empty string (length 0)

(ε symbol is not in the alphabet!)

- A **language** is a set of **strings**

Languages can be **infinite**

$$A = \{\text{good}, \text{bad}\}$$

$$\emptyset \qquad \{\ \}$$

The **Empty set** is a **language**

$$A = \{w \mid w \text{ contains at least one 1 and an even number of 0s follow the last 1}\}$$

"**the set of all …**"

"**such that …**"

Computation and Languages

- The **language** of a **machine** = <u>set of strings</u> that it **accepts**

- E.g., A DFA M ***accepts*** w if $\hat{\delta}(q_0, w) \in F$

Machine and Language Terminology

- The **language** of a **machine** = set of strings that it **accepts**

- E.g., A DFA M ***accepts*** w ← string

  M ***recognizes language*** A ← Set of strings

  if $A = \{w|\ M \text{ accepts } w\}$

  "the set of all …" "such that …"

Machine and Language Terminology

- The **language** of a **machine** = set of strings that it **accepts**

- E.g., A DFA M ***accepts*** w

  M ***recognizes language*** $L(M)$

  $L(M) = \{w \mid M \text{ accepts } w\}$

Using L as function mapping **Machine** → **Language** is common notation

Machine and Language Terminology

- The **language** of a **machine** = set of strings that it **accepts**

- E.g., A DFA M ***accepts*** w

  M ***recognizes language*** $L(M)$

- **Language** of $M = L(M) = \{w \mid M \text{ accepts } w\}$

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

= set of set of strings

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

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

Regular Languages

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

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

Regular Languages: Definition

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

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** <u>recognizes</u> 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 : ??????

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**)

Modus Ponens

If we can prove these:

- If P then Q
- P

Then we've proved:

- Q

A Language: strings with odd # of **1**s

- **In-class exercise** (submit to gradescope)**:**

String	In the language?
1	Yes
0	No
01	Yes
11	No
1101	Yes
ε	no

Come up with string examples (in a table), **both**
- **in the language**
- and **not in the language**

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

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

How to prove the language is regular?

Prove there's a DFA recognizing it!

Proving That a Language is Regular

<u>Prove</u>: A **language** L = { … } is a **regular language**

<u>Proof</u>:

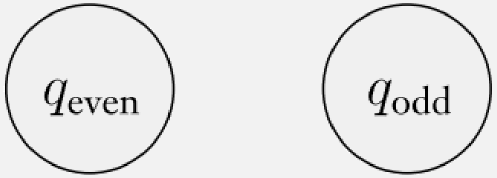
Statements	**Justifications**
1. DFA $M = (Q, \Sigma, \delta, q_0, F)$ (TODO: actually define M) (no unbound variables!)	1. Definition of a DFA
2. DFA M recognizes L	2. TODO: ???
3. <u>If</u> a DFA recognizes L, <u>then</u> L is a regular language	3. Definition of a regular language
4. Language L is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

Designing Finite Automata: Tips

- Input is read only once, one char at a time (cant go back)
- Must decide accept/reject after that
- States = the machine's "memory"!
  - # states must be decided in advance
  - Think about what information must be "remembered".
- Every state/symbol pair must have a defined transition (for DFAs)
- Come up with examples to help you!

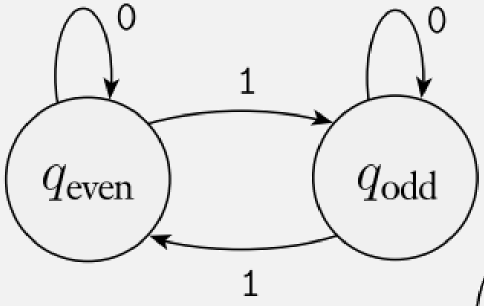
Design a DFA: accept strs with odd # **1**s

- <u>States</u>:
  - 2 states:
    - seen even **1**s so far
    - seen odds **1**s so far

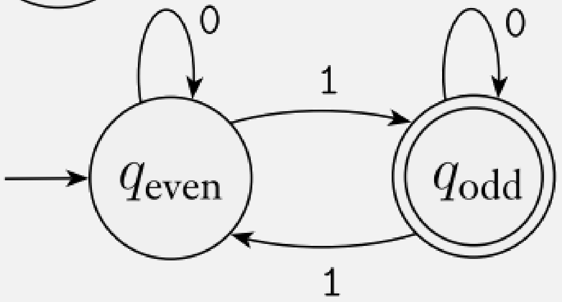


- <u>Alphabet</u>: **`0`** and **`1`**

- <u>Transitions</u>:



- <u>Start</u> / <u>Accept</u> states:



Proving That a Language is Regular

Prove: A **language** L = { … } 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 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**)

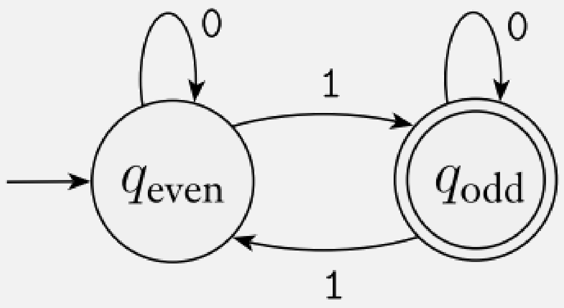
“Prove” that DFA recognizes a language

- **In-class exercise** (part 2):

String	In the language?
1	Yes
0	No
01	Yes
11	No
1101	Yes
ε	no

Confirm the DFA:
- Accepts strings in the language
- Rejects strings not in the language

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



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

Proving That a Language is Regular

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

Proof:

Statements	**Justifications**
1. DFA $M =$ See state diagram (only if problem allows!)	1. Definition of a DFA
2. DFA M recognizes L	☑ 2. See examples table
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**)

$\blacksquare$

In-class exercise 2

Remember:
To understand the 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 | w has exactly three 1's }

You will need this later in the proof anyways!

- Where Σ= {0, 1},

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***.

Proving That a Language is Regular

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

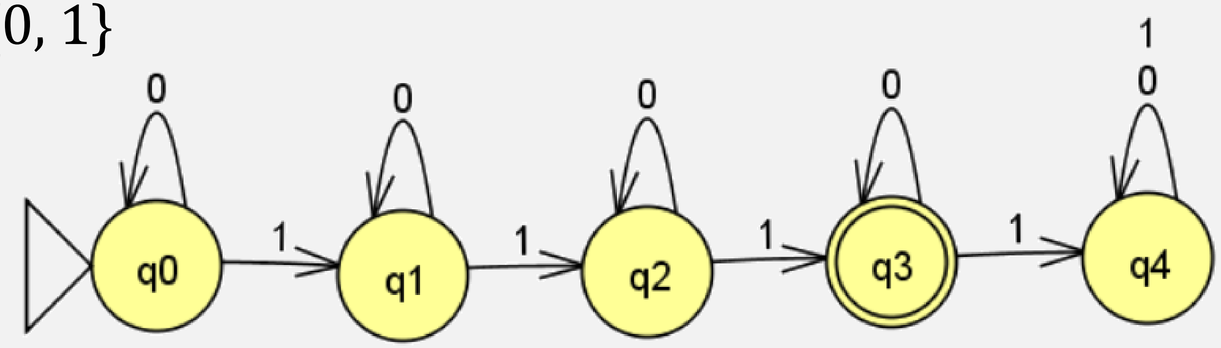
Proof:

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 Solution

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

- *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*:
  - $\{q_3\}$

So a DFA's computation <u>recognizes simple string patterns</u>?

Yes!

Have you ever used a programming language feature to <u>recognize simple string patterns</u>?

Submit 2/7 in-class work to gradescope