Regular Languages

Wednesday, February 7, 2024
UMass Boston Computer Science

Finite State Automata
= Regular Languages!
Announcements

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

Alphabets, Strings, Languages

- An **alphabet** is a **non-empty finite set** of symbols
  \[ \Sigma_1 = \{0, 1\} \]
  \[ \Sigma_2 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\} \]

- A **string** is a **finite sequence** of symbols from an alphabet
  
  \[
  01001 \quad \text{abracadabra} \quad \varepsilon
  \]

- A **language** is a **set** of strings
  
  \[
  A = \{\text{good, bad}\}
  \]

  \[
  \emptyset \quad \{\}\n  \]

  The **Empty set** is a language

  Languages can be **infinite**

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

  “the set of all ...”
  “such that ...”

  An alphabet defines “all possible strings”

  (strings with non-alphabet symbols are impossible)

  Empty string (length 0)

  (\(\varepsilon\) symbol is not in the alphabet!)

  103
Computation and Languages

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

• E.g., A DFA $M$ **accepts** $w$ if $\delta(q_0, w) \in F$
Machine and Language Terminology

• The language of a machine = set of strings that it accepts

• E.g., A DFA \( M \) accepts \( w \)

\[ M \text{ recognizes language } A \]

if \( A = \{ w \mid M \text{ accepts } w \} \)

“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** $w$

\[ M \text{ recognizes language } L(M) = \{ w \mid M \text{ accepts } w \} \]

Using $L$ as function mapping Machine $\rightarrow$ 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 $M = L(M) = \{ w | M$ 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

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 really 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 **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:** ?????
Proving That a Language is Regular

**Prove:** A language \( L = \{ \ldots \} \) is a regular language

**Proof:**

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

**Modus Ponens**
If we can prove these:
- If \( P \) then \( Q \)
- \( P \)

Then we've proved:
- \( Q \)
A Language: strings with odd # of 1s

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

<table>
<thead>
<tr>
<th>String</th>
<th>In the language?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>01</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>No</td>
</tr>
<tr>
<td>1101</td>
<td>Yes</td>
</tr>
<tr>
<td>ε</td>
<td>No</td>
</tr>
</tbody>
</table>

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

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

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

How to prove the language is regular?

Prove there’s a DFA recognizing it!
Proving That a Language is Regular

Prove: A language $L = \{ \ldots \}$ is a regular language

Proof:

**Statements**

1. DFA $M = (Q, \Sigma, \delta, q_0, F)$
   (TODO: actually define $M$)
   (no unbound variables!)

2. DFA $M$ recognizes $L$

3. If a DFA recognizes $L$, then $L$ is a regular language

4. Language $L$ is a regular language

**Justifications**

1. Definition of a DFA

2. TODO: ???

3. Definition of 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 (can't 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 # 1s

- **States:**
  - 2 states:
    - seen even 1s so far
    - seen odds 1s so far

- **Alphabet:** 0 and 1

- **Transitions:**

- **Start / Accept states:**
Proving That a Language is Regular

Prove: A language $L = \{ \ldots \}$ is a regular language

Proof:

**Statements**

1. DFA $M =$

   See state diagram
   (only if problem allows!)

2. DFA $M$ recognizes $L$

3. If a DFA recognizes $L$, then $L$ is a regular language

4. Language $L$ is a regular language

**Justifications**

1. Definition of a DFA

2. TODO: ???

3. Definition of a regular language

4. Stmts 2 and 3
   (and modus ponens)
“Prove” that DFA recognizes a language

- In-class exercise (part 2):

<table>
<thead>
<tr>
<th>String</th>
<th>In the language?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>01</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>No</td>
</tr>
<tr>
<td>1101</td>
<td>Yes</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>No</td>
</tr>
</tbody>
</table>

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

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

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 \( L = \{ \ldots \} \) is a regular language

Proof:

Statements

1. DFA \( M = \)

See state diagram
(only if problem allows!)

2. DFA \( M \) recognizes \( L \)

3. If a DFA recognizes \( L \),
then \( L \) is a regular language

4. Language \( L \) is a regular language

Justifications

1. Definition of a DFA

2. See examples table

3. Definition of a regular language

4. Stmts 2 and 3
(and modus ponens)
In-class exercise 2

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

• Where $\Sigma = \{0, 1\}$,

**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. $\Sigma$ is a finite set called the **alphabet**, 
3. $\delta: \Sigma \times Q \rightarrow Q$ is the **transition function**, 
4. $q_0 \in Q$ is the **start state**, and 
5. $F \subseteq Q$ is the **set of accept states**.

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

**You will need this later in the proof anyways!**
**Proving That a Language is Regular**

**Prove:** A language \( L = \{ \ldots \} \) is a regular language

**Proof:**

<table>
<thead>
<tr>
<th>Statements</th>
<th>Justifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. DFA ( M = (Q, \Sigma, \delta, q_0, F) )</td>
<td>1. Definition of a DFA</td>
</tr>
<tr>
<td>(TODO: actually define ( M ))</td>
<td></td>
</tr>
<tr>
<td>(no unbound variables!)</td>
<td></td>
</tr>
<tr>
<td>2. DFA ( M ) recognizes ( L )</td>
<td>2. TODO: ???</td>
</tr>
<tr>
<td>3. If a DFA recognizes ( L ), then ( L ) is a regular language</td>
<td>3. Definition of a regular language</td>
</tr>
<tr>
<td>4. Language ( L ) is a regular language</td>
<td>4. Stmts 2 and 3 (and modus ponens)</td>
</tr>
</tbody>
</table>
In-class exercise Solution

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

  - 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 recognizes simple string patterns?

Yes!

Have you ever used a programming language feature to recognize simple string patterns?
Submit 2/7 in-class work to gradescope