Proving a Language is Regular

Friday, February 9, 2024

UMass Boston Computer Science


**Announcements**

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

---

- Turing Machines
- Linear bounded Automata
- Push-down Automata
- **Finite State Automata** = Regular Languages!
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!
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.

• If given: a Language $A$
  • Is $A$ a regular language?
    • Not necessarily!

Proof: ??????
In-class exercise 2: Language

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

Where \( \Sigma = \{0, 1\} \),

<table>
<thead>
<tr>
<th>String</th>
<th>In the language?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>11</td>
<td>No</td>
</tr>
<tr>
<td>111</td>
<td>Yes</td>
</tr>
<tr>
<td>1101</td>
<td>Yes</td>
</tr>
<tr>
<td>11011</td>
<td>No</td>
</tr>
</tbody>
</table>
Proving That a Language is Regular

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

Proof:

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

Previously

**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)

\textbf{Modus Ponens}

If we can prove these:

- \( \text{If } P \text{ then } Q \)
- \( P \)

Then we've proved:

- \( Q \)
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)
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:
  • $A = \{ w \mid w \text{ has exactly three 1’s } \}$

<table>
<thead>
<tr>
<th>String</th>
<th>In the language?</th>
<th>Accepted by machine?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>Reject</td>
</tr>
<tr>
<td>0</td>
<td>No</td>
<td>Reject</td>
</tr>
<tr>
<td>11</td>
<td>No</td>
<td>Reject</td>
</tr>
<tr>
<td>111</td>
<td>Yes</td>
<td>Accept</td>
</tr>
<tr>
<td>1101</td>
<td>Yes</td>
<td>Accept</td>
</tr>
<tr>
<td>11011</td>
<td>No</td>
<td>Reject</td>
</tr>
</tbody>
</table>

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**

1. DFA $M$ = \[
\]

   See state diagram
   (only if problem allows!)

2. DFA $M$ recognizes $A$

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

4. Language $A$ 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: Solution

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

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

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

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

(We do this with programs all the time)
Password Checker DFAs

$M_5$: AND

$M_3$: OR

$M_1$: Check special chars

$M_2$: Check uppercase

$M_4$: Check length

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

We want these operations:

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

To combine more than once, operations must be **closed**!
“Closed” Operations

• Set of Natural numbers = \{0, 1, 2, ...\}
  • 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 = {..., -2, -1, 0, 1, 2, ...}
  • Closed under addition and multiplication
  • Closed under subtraction?
    • yes
  • Closed under division?
    • no

• Rational numbers = \{x \mid x = \frac{y}{z}, y and z are Integers\}
  • Closed under division?
    • No?
    • Yes if \(z \neq 0\)

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

\[ M_3: \text{OR} \]
\[ M_1: \text{Check special chars} \]
\[ M_2: \text{Check uppercase} \]

(a) \[ \begin{array}{c}
A \ \ \ B
\end{array} \]

(b) \[ \begin{array}{c}
A \ \ \ B
\end{array} \]
Password Checker: “OR” = “Union”

$M_3$: OR

$M_1$: Check special chars

$M_2$: Check uppercase

(a)
Union of Languages

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

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

$$A \cup B = \{\text{fort, south, point, boston}\}$$
Submit 2/9 in-class work to gradescope