Nonregular Languages

• 2017/09/07

• Chapter 1.4 in Sipser

➤ Announcement:

- Slides for this lecture are here:
  http://www.public.asu.edu/~yzhan442/teaching/CSE355/Lectures/NREG.pdf

- HW3 will be announced by Friday, due next Friday (September 15)

- Mid-term #1 is scheduled to be on September 19 (Chapter 1)
  - In class; 75 min; a one-page guide will be provided soon
  - Difficulty level will be similar to exercises & problems in HWs and textbook
  - A one-page (maximum size A4) hand-written cheat sheet is allowed
  - Contact me by Friday, September 15 if you have any special requirements

ASU
Ira A. Fulton
Schools of Engineering
Last time

- Regular expressions (REs)
  - Regular operations and FA
  - RE examples
  - Equivalence of FA and REs

➢ Previous goals:
  - Learn that FA and regular expressions are equivalent!
  - Learn how to convert REs into FA and vice versa!
  - Learn to use proof by induction
Outline for today

- Nonregular languages
  - Limitations of regular languages
  - Pumping lemma
  - Proving for nonregularity
    - Using pumping lemma
    - Using closure properties of RLs

➢ Goals:
  - Learn the limitations of regular languages
  - Learn how to prove a language is not regular
  - Learn to use proof by contradiction
Limitations of regular languages

We have seen regular languages being used for

- Lexical analysis
- Controller design
- ...

What it can’t do (compute)?

- Consider the language \( \{0^n1^n | n \geq 0\} \)? How to design a FA for it?
Limitations of regular languages

We have seen regular languages being used for

- Lexical analysis
- Controller design
- ...

What it can’t do (compute)?

- Consider the language \( \{0^n1^n| n \geq 0\} \)? How to design a FA for it?

Step 1: For a given \( n \)

\[q_0 \xrightarrow{0} q_1 \quad \ldots \quad q_{-2n} \xrightarrow{1} q_{-2n+} \]

Step 2: Union them all
Limitations of regular languages

We have seen regular languages being used for

- Lexical analysis
- Controller design
  - ...

What it can’t do (compute)?

- Consider the language \( \{0^n1^n | n \geq 0\} \)? How to design a FA for it?

Step 1: For a given \( n \)

- \( q_0 \) \( \xrightarrow{0} \) \( q_1 \)
- \( q_2n \) \( \xrightarrow{1} \) \( q_{2n+} \)

Step 2: Union them all

does’t work for FA!
Limitations of regular languages

We have seen regular languages being used for

- Lexical analysis
- Controller design
- ... 

What it can’t do (compute)?

- Consider the language \( \{0^n1^n | n \geq 0\} \)? How to design a FA for it?
- How about \( \{0^n\} \) – connection to nonregularity of the language of valid C programs

Step 1: For a given \( n \)

Step 2: Union them all

\[ \begin{array}{cccc}
q_0 & 0 & q_1 & \\
& & \cdots & \\
& & & q_{-2n+1}
\end{array} \]

does’t work for FA!
Limitations of regular languages

**Definition 1.16**
A language is called a *regular language* if some finite automaton recognizes it.

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

How do we decide whether a language is regular?

How do we decide if a language is not regular?

**Definition 1.16**
A language is called a *regular language* if some finite automaton recognizes it.
Limitations of regular languages

How do we decide whether a language is regular?

How do we decide if a language is not regular?

Proof by contradiction:

- $A \rightarrow B \quad \text{then} \quad \neg B \rightarrow \neg A$
- if $A$ is a regular language, then we have $B$ satisfied
- for any language $A'$, assume it is regular, then $B$ must holds
- show that $A'$ satisfies $\neg B$

$B$ must be a necessary property/condition of any regular language
Outline for today

• Nonregular languages
  o Limitations of regular languages
  o **Pumping lemma**
  o Proving for nonregularity
    • Using pumping lemma
    • Using closure properties of RLs

➢ Goals:
  o Learn the limitations of regular languages
  o **Learn how to prove a language is not regular**
  o Learn to use proof by contradiction
Pumping lemma

**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. \(\Sigma\) is a finite set called the *alphabet*,
3. \(\delta: Q \times \Sigma \rightarrow Q\) is the *transition function*,
4. \(q_0 \in Q\) is the *start state*, and
5. \(F \subseteq Q\) is the *set of accept states*.\(^1\)

Consider an accept string:

\[(baba) \quad (baba)(baba) \quad (baba)(baba)(baba)\]
\[baaba(aa) \quad baaba(aa)(aa) \quad baaba(aa)(aa)(aa)\]

The corresponding state sequence:

\[q_1q_2q_3q_1\]
\[q_1q_2q_2q_3q_1q_3q_1\]
**Pumping lemma**

**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. \(\Sigma\) is a finite set called the **alphabet**, 
3. \(\delta : Q \times \Sigma \rightarrow Q\) is the **transition function**, \(^1\)
4. \(q_0 \in Q\) is the **start state**, and 
5. \(F \subseteq Q\) is the **set of accept states**. \(^2\)

Consider an accept string:

\[(baba) \quad (baba)(baba) \quad (baba)(baba)(baba)\]

\[baaba(aa) \quad baaba(aa)(aa) \quad baaba(aa)(aa)(aa)\]

The corresponding state sequence:

\(q_1 q_2 q_3 q_1\)

\(q_1 q_2 q_2 q_3 q_1 q_3 q_1\)

**Certain substrings can be pumped!**
Pumping lemma

**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. \(\Sigma\) is a finite set called the **alphabet**, 
3. \(\delta : Q \times \Sigma \rightarrow Q\) is the **transition function**,\(^1\) 
4. \(q_0 \in Q\) is the **start state**, and 
5. \(F \subseteq Q\) is the **set of accept states**.\(^2\)

\[ S = S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_5 \rightarrow S_6 \rightarrow \ldots \rightarrow S_n \]

\(q_1 \rightarrow q_3 \rightarrow q_{20} \rightarrow q_9 \rightarrow q_{17} \rightarrow q_9 \rightarrow q_6 \rightarrow q_{35} \rightarrow q_{13}\)
Pumping lemma

THEOREM 1.70

Pumping lemma If $A$ is a regular language, then there is a number $p$ (the pumping length) where if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$. 
Proof

**Theorem 1.70**

**Pumping lemma** If $A$ is a regular language, then there is a number $p$ (the pumping length) where if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

1. Consider an DFA $M = (Q, \Sigma, \delta, q_0, F)$ and let $p = |Q|$
2. For any $w = w_1w_2...w_n \in L(M)$ such that $|w| = n \geq p$
3. Let the state sequence be $r_1r_2...r_nr_{n+1}$
4. By the pigeon hole principle $\exists i, j \in [1, p + 1] \ r_i = r_j (i \neq j)$
5. Let $x = w_1...w_{i-1}$, $y = w_i...w_{j-1}$, $z = w_j...w_n$

---

[Diagram of a DFA with states $q_1$, $q_9$, $q_{13}$, and transitions $r_1$, $r_i/r_j$, $r_{n+1}$]
Proof

**Theorem 1.70**

**Pumping lemma** If \( A \) is a regular language, then there is a number \( p \) (the pumping length) where if \( s \) is any string in \( A \) of length at least \( p \), then \( s \) may be divided into three pieces, \( s = xyz \), satisfying the following conditions:

1. for each \( i \geq 0 \), \( xy^iz \in A \),
2. \( |y| > 0 \), and
3. \( |xy| \leq p \).

**NOTE:** \( p \) can be arbitrarily chosen in PL

Does PL satisfy for any finite (and thus regular) language?

1. Consider an DFA \( M = (Q, \Sigma, \delta, q_0, F) \) and let \( p = |Q| \)
2. For any \( w = w_1w_2...w_n \in L(M) \) such that \( |w| = n \geq p \)
3. Let the state sequence be \( r_1r_2...r_nr_{n+1} \)
4. By the pigeon hole principle \( \exists i, j \in [1, p + 1] \) \( r_i = r_j (i \neq j) \)
5. Let \( x = w_1...w_{i-1}, y = w_i...w_{j-1}, z = w_j...w_n \)
Proof

**Theorem 1.70**

**Pumping lemma** If $A$ is a regular language, then there is a number $p$ (the pumping length) where if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

**Note:** $p$ can be arbitrarily chosen in PL. Does PL satisfy for any finite (and thus regular) language? **Yes!**

1. Consider an DFA $M = (Q, \Sigma, \delta, q_0, F)$ and let $p = |Q|$
2. For any $w = w_1w_2...w_n \in L(M)$ such that $|w| = n \geq p$
3. Let the state sequence be $r_1r_2...r_nr_{n+1}$
4. By the pigeon hole principle $\exists i, j \in [1, p + 1] \ r_i = r_j (i \neq j)$
5. Let $x = w_1...w_{i-1}$, $y = w_i...w_{j-1}$, $z = w_j...w_n$
Pumping lemma

**Theorem 1.70**

Pumping lemma If \( A \) is a regular language, then there is a number \( p \) (the pumping length) where if \( s \) is any string in \( A \) of length at least \( p \), then \( s \) may be divided into three pieces, \( s = x y z \), satisfying the following conditions:

1. for each \( i \geq 0 \), \( x y^i z \in A \),
2. \( |y| > 0 \), and
3. \( |xy| \leq p \).

How do we decide if a language is not regular?

Proof by contradiction:

- \( A \rightarrow B \rightarrow \neg B \rightarrow \neg A \)
- if \( A \) is a regular language, then we have \( B = PL \) satisfied
- for any language \( A' \), assume it is regular, then \( B = PL \) must holds
- show that \( A' \) satisfies \( \neg PL \)

**Pumping lemma (PL)** is a necessary property for any regular language

**B** must be a necessary property/condition of any regular language
Outline for today

• Nonregular languages
  o Limitations of regular languages
  o Pumping lemma
  o Proving for nonregularity
    • Using pumping lemma
    • Using closure properties of RLs

➢ Goals:
  o Learn the limitations of regular languages
  o Learn how to prove a language is not regular
  o Learn to use proof by contradiction
Example

Book example 1.73: Show that $A = \{0^n1^n | n \geq 0\}$ is nonregular

1. Assume that $A$ is regular, then $A$ must satisfy the pumping lemma
2. Since $p$ is not set in the lemma, we cannot arbitrarily set $p$; consider it as a variable
3. Choose $w = 0^p1^p$ with $p$ as a variable: the most innovative step; may not work for your first choice!
4. $w = xyz$  
   Bad news: we must consider all possible way to decompose $w$
   Good news: often we only need to consider a few cases
5. Decompositions:
Example

Book example 1.73: Show that \( A = \{0^n1^n | n \geq 0\} \)
is nonregular

1. Assume that \( A \) is regular, then \( A \) must satisfy the pumping lemma
2. Since \( p \) is not set in the lemma, we cannot arbitrarily set \( p \); consider it as a variable
3. Choose \( w = 0^p1^p \) with \( p \) as a variable: the most innovative step; may not work for your first choice!
4. \( w = xyz \)
   Bad news: we must consider all possible way to decompose \( w \)
   Good news: often we only need to consider a few cases
5. Decompositions:
   • Case 1: if \( y \) contains only 0
   • Case 2: if \( y \) contains only 1
   • Case 3: if \( y \) contains both 0 and 1
Using closure properties

How do we decide whether a language is regular?

How do we decide if a language is not regular?

Proof by contradiction:

- \( A \rightarrow B \implies \neg B \rightarrow \neg A \)
- if \( A \) is a regular language, then we have \( B \) satisfied
- for any language \( A' \), assume it is regular, then \( B \) must hold
- show that \( A' \) satisfies \( \neg B \)

B must be a necessary property/condition of any regular language

**Definition 1.23**

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

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

These are also properties of regular languages!
Example

Book example 1.74: Show that \( A = \{ w \mid w \text{ contains an equal number of 0s and 1s} \} \) is nonregular

1. Assume that \( A \) is regular, **choose another regular language \( B \): key step**
2. \( L = A \{ \cup, \cap, \circ, \backslash \} \) \( B \) must also be regular. **Why don’t we use \(*\)?**
3. Derive a contradiction by showing that \( L \) is not regular

- Let us choose \( B = \{ 0^*1^* \} \)
- What is \( L \)?
Example

Book example 1.74: Show that $A = \{w \mid w$ contains an equal number of 0s and 1s$\}$ is nonregular

1. Assume that $A$ is regular, choose another regular language $B$: key step
2. $L = A \{\cup, \cap, \circ, \backslash\}$ B must also be regular. Why don’t we use $*$?
3. Derive a contradiction by showing that $L$ is not regular

- Let us choose $B = \{0^*1^*\}$
- What is $L$? $\{0^n1^n\}$!
- Easy to use a similar process to prove that valid C programs are not regular
Outline for today

• Nonregular languages
  o Limitations of regular languages
  o Pumping lemma
  o Proving for nonregularity
    • Using pumping lemma
    • Using closure properties of RLs

➢ Goals:
  o Learned the limitations of regular languages
  o Learned how to prove a language is not regular
  o Learned to use proof by contradiction

• Reading assignment for the next class:
  o Sipser Sec. 2.1 – Quiz link will be sent out; due date is before the beginning of the next class