Chapter 4 Properties of Regular Languages

Closure Properties of RL

Elementary Questions about RL

Identifying Nonregular Languages

About RL …

- What is a regular language? Give a definition.
- Can you give the formal definition of RE?
- What is Kleen’s Theorem?

RL Review

- RL – a language that can be defined by a RE or FA.
- RE
  - 1. $\varnothing$, λ, and $a \in \Sigma$ are RE's.
  - 2. If r_1 and r_2 are RE, so are $r_1 + r_2$, $r_1 . r_2$, r_1^* and (r_1).
- Kleen's Theorem: L(FA)=L(TG)=L(RE)

Issues on Regular Languages

- Set operations on RL – would that result another RL?
- Is a given language finite or not?
- Is every finite language regular?
- How can we tell whether a given language is regular or not?

Motivations to study properties of RL

- One way ***to show a language is not regular*** is to study properties that are shared by all RL
  - If we know some such property and we can show a candidate language does not have it, then we can tell that the language is not regular
- FA is also an powerful **algorithm** to recognize membership, equality, and more

4.1 Closure Properties of RL

- Theorem 4.1
  - If L_1 and L_2 are RL, then so are $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 L_2$, L_1', and L_1^*. We say that the family of regular languages is closed under union, intersection, concatenation, complementation, and star-closure.

Theorem 4.1 Proof Outline - 1

- If L_1 and L_2 are RL, then there exist RE r_1 and r_2 such that L_1=L(r_1) and L_2=(r_2). By definition, we have
  - $L_1 \cup L_2$: $r_1 + r_2$
  - L_1L_2 : $r_1 \cdot r_2$
  - L_1* : r_1*

Theorem 4.1 Proof Outline - 2

- If L is a RL, then L′ is also a RL
- Proof – if L is a RL, then there is a FA M such that L = L(M). We can make a new FA M′ by
  - Change all final states of M to non-final states
  - Change all non-final states to final states
  - L(M′) = L′

Theorem 4.1 Proof Outline - 3

- If L_1 and L_2 are RL, then $L_1 \cap L_2$ is also RL.
- Proof (key step)
  - $L_1 \cap L_2 = (L_1' + L_2')'$

Example 4.1

- Show that RL is closed under difference
  - If L_1 and L_2 are regular so are $L_1 - L_2$
- Proof
  - $L_1 - L_2 = L_1 \cap L_2'$
  - L_2 is RL implies L_2' is RL (Theorem 4.1)
  - Then, because closure of RL under intersection, we have $L_1 \cap L_2'$ is regular

Theorem 4.2

- The family of RL is closed under reversal.
- Proof (try to give a outline)
  - Use RE or FA?

4.2 Elementary Questions about Regular Languages

- Given a language L and a string *w*, can we determine whether or not w is an element of L?
- Need a membership algorithm
- Algorithm – a method for which one can write a computer program (informal)

Theorem 4.5 FA as membership algorithm

- Given a standard representation of any RL L on Σ and any $w \in \Sigma^*$, there exists an algorithm for determining whether or not w is in L.
- Proof. Represent L by a dfa, then test w to see if it is accepted by this automaton

Theorem 4.6

RL empty, finite, or infinity determination using FA

- There exists an algorithm for determining whether a regular language L, given in standard representation, is **empty, finite, or infinite.**
- Proof. Represent L as a dfa.
  - If there is a path from the initial vertex to any final vertex, then L is not **empty**.
  - Find all the vertices that are the base of a cycle. If any of these are on a path from an initial to final vertex, L is **infinite**. Otherwise, it is **finite**.

Theorem 4.7
determine whether or not two RLs are equal

- Given two regular languages L_1 and L_2, there exists an algorithm to determine whether or not $L_1 = L_2$
- Proof
  - Using L_1 and L_2, we define the language
    - $L_3 = (L_1 \cap L_2') \cup (L_1' \cap L_2)$
    - By closure L_3 is regular and we can use the algorithm in Theorem 4.6 to determine if L_3 is empty
    - If L_3 is empty then $L_1 = L_2$

4.3 Identifying Nonregular Languages

- How do you prove a language to be regular?
  - Constructive proof
- How do you prove a language to be nonregular?
  - Proof by negation
    - **Using the Pigeonhole Principle**
    - **A Pumping Lemma**

Using the Pigeonhole Principle

- **Case 1**. Finite number of pigeonholes, H, for finite number of pigeons, P.
  - When $|H| < |P|$ and all of pigeons went into the pigeonholes, then at least one hole contains at least two pigeons
- **Case 2**. When $|H| = N$, and $|P| = \infty$, then at least one hole contains infinite number of pigeons
- Applying the principle to an FA that has <u>limited memory</u> and can accept an <u>infinite</u> RL.
  - Pumping Lemma = another form of the pigeonhole principle

A Pumping Lemma

- Let L be an infinite RL. Then there exists some positive integer m such that any $w \in L$ with $|w| \geq m$ can be decomposed as
  - $w = xyz$
  - with $|xy| \leq m$
  - and $|y| \geq 1$
  - such that $w_i = xy^iz$ is also in L for all i = 0, 1, 2, …

A Pumping Lemma
-- in other words

- Every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the **middle part** (y) yields another string in L. We say that the middle string is "**pumped**".
- w = xyz
  - $|w| \geq m$ (m is the integer in Pumping Lemma)
  - x or z can be empty but y must be nonempty
  - You can think m to be the number of the states of the FA that accepts L

Example 4.7
Show $L = \{a^n b^n : n \geq 0\}$ is not regular

- Proof.
  - **Assume** that L is regular, so that the pumping lemma must hold. **Let m** be the integer in the pumping lemma.
    - **Let w** = $a^m b^m$ ($|w| \geq m$)
    - By the pumping lemma, w may be written as xyz, where $|y| \geq 1$ and $|xy| \leq m$ and xy^iz is also in L for all i = 0, 1, 2, …
    - **Let i** = 2, $xy^2z = a^{m+k}b^m \notin L$ with $1 \leq k \leq m$
    - This string brings at least one more a into the a's segment, and is not a string in L. **This contradicts** the pumping lemma and thereby indicates that the assumption that L is regular must be false.

In applying the Pumping Lemma…

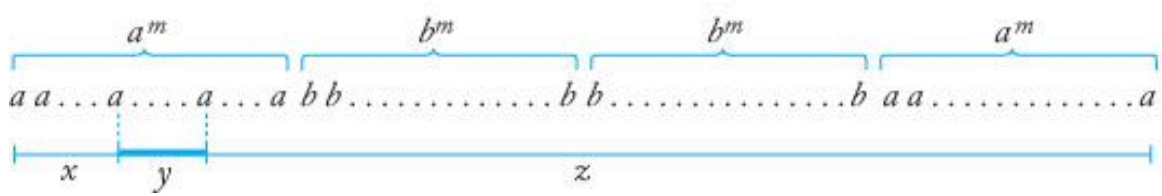
- The correct argument can be viewed as a **game** we play against an opponent
- Our goal is to win the game by establishing a contradiction of the pumping lemma, while the opponent tries to foil us. There are **4 moves**:
  - The opponent picks m
  - Given m, **we pick w** in L with $|w| \geq m$
  - The opponent chooses the decomposition xyz, subject to $|xy| \leq m$, $|y| \geq 1$
  - **We pick i** in such a way that the pumped string w_i, is not in L. If we can do so, we win the game

Key elements
apply Pumping Lemma to prove L is non-regular

1. Assume L is regular and m is integer in Pumping Lemma
2. Let w = f(m) in L and $|w| \geq m$
3. W =xyz, subject to $|xy| \leq m$, $|y| \geq 1$
4. Let i = ? to show resulting Wi make Pumping property fail, and therefore a contradiction. Conclusion: L is not regular

Example 4.8
show L = {ww^R: $w \in \Sigma^*$} is not regular

- Proof.
  - Assume that L is regular, so that the pumping lemma must hold. Let m be the integer in the pumping lemma. Let
    - $w = a^m b^m b^m a^m$ ($|w| \geq m$) -- **step 2**
    - Because this choice of w, in **step 3,** y can only consists of a's; in **step 4**, we may use i = 0 and the contradiction follows – the resulting string has fewer a's on the left.



Key steps:
pick the w and i to make the argument easier for us

- Ex. 4.9 $L = \{w \in \Sigma^*, n_a(w) < n_b(w)\}$
  - $w = a^m b^{m+1}$ $i = 2$, $w_2 = ?$ for $y = a^k$
- Ex. 4.10 $L = \{(ab)^n a^k : n > k, k \geq 0\}$
  - $w = (ab)^{m+1} a^m$ $i = 0$, $w_0 = ?$ for all possible y's
- Ex. 4.11 $L = \{a^n : n \text{ is a perfect square}\}$
  - Pick $n = m^2$ $i = 0$, $w_0 = ?$ for $y = a^k$
  - You may try $i = 2$, and it works too

Example 4.13

For $L = \{a^n b^l : n \neq l\}$, there is a better way than the Pumping Lemma to show L is not regular

- Suppose L is regular. Then by Theorem 4.1, L′ is also a RL, and
  - $L_1 = L' \cap L(a^*b^*)$ would also regular.
  - But $L_1 = \{a^n b^n : n \geq 0\}$ is nonregular.
  - Consequently, L cannot be regular.

  - Note: this is also “prove by contradiction”

More examples/exercises
in proving a language to be non regular

- PALINDROME
- PRIME

The Pumping Lemma is …

- Difficult to understand
- Easy to make mistakes when applying it
- Common mistakes
  - Using it to show a language is regular
  - Pick w is not in L or not easy to argue
  - Mix up m and i
  - Make some assumption about the decomposition xyz

To apply the Pumping Lemma well:
Knowledge of the rule + a good strategy

- Knowledge of the rules is essential, but that alone is not enough to play a good game
- Need a good strategy to win
- Read more examples + do more exercises will help – just like playing any games!

The Pumping Lemma: Poem

- Any regular language *L* has a magic number *p*
  And any long-enough word in *L* has the following property:
  Amongst its first *p* symbols is a segment you can find
  Whose repetition or omission leaves *x* amongst its kind.

  So if you find a language *L* which fails this acid test,
  And some long word you pump becomes distinct from all the rest,
  By contradiction you have shown that language *L* is not
  A regular guy, resiliant to the damage you have wrought.

  But if, upon the other hand, *x* stays within its *L*,
  Then either *L* is regular, or else you chose not well.
  For *w* is *xyz*, and *y* cannot be null,
  And *y* must come before *p* symbols have been read in full.

  As mathematical postscript, an addendum to the wise:
  The basic proof we outlined here does certainly generalize.
  So there is a pumping lemma for all languages context-free,
  Although we do not have the same for those that are r.e.