ECS120 Introduction to the Theory of Computation
Fall Quarter 2007

Discussion Notes
Wednesday, October 24, 2007

Proving Regular

There are two methods you could use to prove the languages are not regular. The first method is to use the pumping lemma and proof by contradiction. The second method is to use the closure properties of regular languages, known regular languages, and proof by contradiction.

Method 1: Pumping Lemma

The pumping lemma is stated in your book (page 78). Essentially, it states:

If L is a regular language, then there is a pumping length p where, if $w \in L$ and $|w| \geq p$, then w may be divided into three pieces $w = xyz$ satisfying the following:

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

For example, consider the language $L = 0^*1$. Since it is a regular expression, we know L is regular and the pumping lemma holds. Consider the string $w = 0^p1$. We know that $w \in L$ and $|w| = p + 1 \geq p$. We can split w into $x = \epsilon$, $y = 0^p$, and $z = 1$. Notice that these satisfy the conditions of the pumping lemma.

However, just because we can split w into pieces xyz which satisfy the conditions of the pumping lemma, does not mean L is regular. We know that if a language is regular, the pumping lemma holds. The contrapositive of this is if the pumping lemma does not hold, then the language is not regular. You can use this to try and prove that the given language is non-regular with the following steps:

1. Assume for the sake of contradiction that L is regular.
2. Since L is regular, there exists a p where for any string $w \in L$ with length $|w| \geq p$, we may divide w into $w = xyz$ such that the conditions of the pumping lemma hold.
3. Choose a string $w \in L$ where $|w| \geq p$.
4. Show that this string can not be divided into $w = xyz$ such that all of the pumping lemma conditions hold.

5. This is a contradiction of the pumping lemma, therefore L is not regular.

Steps 3 and 4 are where it gets tricky. First, you need to choose the string w. Not all strings will work, so you may have to try a few strings before you get one that works.

Second, you have to show that with this string, the pumping lemma does not hold. You want to show that it is impossible for all three conditions to hold for your given string.

Method 1: Example

Prove that $L = \{ss \mid s \in \{0,1\}^*\}$ is not regular using the pumping lemma.

(Sipser Example 1.75)

Assume for the sake of contradiction that L is regular.

Let $w = 0^p10^p1$. Since $w \in L$ and $|w| = 2p + 2 \geq p$, then by the pumping lemma we can split w into $w = xyz$ such that:

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

Lets figure out what possible values x, y, and z may take. Notice that $|xy| \leq p$. The first p characters of our string are all 0s:

$$\underbrace{00\cdots0}_{p}1\underbrace{00\cdots0}_{p}1$$

This tells us that xy must be a string of 0s, looking similar to:

$$\underbrace{00\cdots0}_{xy}100\cdots01$$

Also, $|y| > 0$. Therefore y must be at least one 0. However, $xy^iz \in L$ for each $i \geq 0$. Therefore the string xz should be in L (for $i = 0$). This yields in a contradiction since $xz \notin L$.

Is this clear? There is no way to write xz as ss therefore $xz \notin L$. Lets do a specific example to convince you.

Let $p = 2$ giving us $w = 001001$. There are three ways we can split this string under conditions (2) and (3) of the pumping lemma:

	x	y	z	xz
(i)	ϵ	0	01001	01001
(ii)	ϵ	00	1001	1001
(iii)	0	0	1001	01001

Notice that (i) and (iii) are odd, and can't even be split into two equal length strings. Also, for (ii) string $10 \neq 01$ so this does not belong in our language either.

Therefore there is no way to split this string and satisfy the three conditions of the pumping lemma. This is a contradiction. Thus L must be non-regular.

The book pumps up and argues that $xyyz \notin L$ which works as well. This gives an example of pumping down.

Method 2: Closure Properties

We know that certain operations are closed under the regular languages. In problem 2, we prove that certain operation(s) are closed under non-regular langauges. We also have several languages that we have already proven to be non-regular. We can use this knowledge to our advantage and build a proof by contradiction.

For example, suppose we want to prove that L_1 is non-regular. We could:

1. Assume for the sake of contradiction that L_1 is regular.
2. Suppose we know that $L_1 \cap L_2 = L_3$.
3. Suppose we already know that L_2 is regular, and that L_3 is non-regular.
4. Since L_1 and L_2 are regular, and intersection is closed under the regular langauges, then L_3 must be regular.
5. However, we already know that L_3 is non-regular. This is a contradiction of the closure properties, therefore L_1 must be non-regular.

The difficult part is finding languages L_1, L_2, and an operation that this will work for. Therefore the bulk of the work is in determining steps 2 and 3.

*** These are just general examples of how you might want to approach these problems.

Counterexample Proof

To disprove a statement, you just need to provide a counterexample. Choose a L_1 and L_2 which disproves the statement. Use simple langauges that we have already proven to be regular or non-regular.

Example

Let L_1 and L_2 be non-regular languages, and let $L_1 - L_2 = L_3$. Is L_3 non-regular (i.e. is set difference closed under non-regular langauges)?

No. Consider the case when L_1 and L_2 are equivalent, i.e. when $L_1 = L_2$. Then $L_1 - L_2 = L_1 - L_1 = \emptyset$. Since $\emptyset$ is regular, set difference is not closed under non-regular languages.

Context Free Grammars

Example 1

What language is represented by the following grammar?

$$\begin{aligned} \boldsymbol{S} &\rightarrow \boldsymbol{A} \mid \boldsymbol{B} \mid \boldsymbol{AB} \\ \boldsymbol{A} &\rightarrow \boldsymbol{aA} \mid \boldsymbol{\epsilon} \\ \boldsymbol{B} &\rightarrow \boldsymbol{bB} \mid \boldsymbol{\epsilon} \end{aligned}$$

What are some example derivations?

$$S \Rightarrow A \Rightarrow aA \Rightarrow aaA \Rightarrow aa$$
$$S \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aab$$

This generates the language represented by the regular expression a^*b^*.

Example 2

Let L be the set of strings over the alphabet $\{a, b\}$ with more a's than b's. Give a context-free grammar for L.

(Sipser Example 2.6a)

Suppose for a moment that our grammar is as follows:

$$\begin{aligned} S &\rightarrow TaT \\ T &\rightarrow aTb \mid bTa \mid a \mid \epsilon \end{aligned}$$

However, string $aaaa \in L$. How would we derive this? We can't, therefore we need to add one more rule:

$$\begin{aligned} S &\rightarrow TaT \\ T &\rightarrow TT \mid aTb \mid bTa \mid a \mid \epsilon \end{aligned}$$

Then we get the derivation:

$$S \Rightarrow TaT \Rightarrow TTaT \Rightarrow aTaT \Rightarrow aaaT \Rightarrow aaaa$$