ECS120 Fall 2006

Discussion Notes

November 1, 2006

Announcements

The **midterm** is on **Thursday, November 2nd** during class.

- Any examples covered in class, the book, or the homeworks (up to homework 5) can be used as common knowledge.
- Closed book exam.
- Allowed a single page of notes (front and back).

Homework 5 Resources

Problem 1: Decision Procedures

- **Recap:** A decision procedure is an algorithm/procedure that provides a yes or no answer for a decision problem.
- **Bounded:** These procedures must be bounded. If a procedure could loop infinitely, there would be some cases where it would never output an answer.
- **Efficient:** These procedures should be efficient.

However, sometimes designing an efficient procedure (meaning it runs in polynomial time) is not trivial. Focus more on the ability to provide a bounded algorithm than an efficient one.

So far we have covered decision procedures during lecture:

- **Membership Problem (Regular):** Let L be a regular language. Decide if $w \in L$. (We covered this procedure in class 10/17.)
- **Emptiness Problem (Regular):** Let L be a regular language. Decide if $L = \emptyset$. (We covered this procedure in class 10/17.)
- **Complement of Emptiness Problem (Regular):** Let L be a regular language. Decide if $L = \Sigma^*$. (We covered this procedure in class 10/17.)

- **Subset Problem (Regular):** Let L_1 and L_2 be regular languages. Decide if $L_1 \subseteq L_2$. (We covered this procedure in class 10/17.)
- **Equivalence Problem (Regular):** Let L_1 and L_2 be regular langauges. Decide if $L_1 = L_2$. (We covered this procedure in class 10/17.)
- **Membership Problem (Context-Free):** Let G be a context-free grammar. Decide if $w \in L(G)$. (We covered this procedure in class 10/31.)
- **Emptiness Problem (Context-Free):** Let G be a context-free grammar. Decide if $L(G) = \emptyset$. (We covered this procedure in class 10/31.)

Ideally, you should have lecture notes from these days. Many of these are also described in the decidability chapter (chapter 4) of your book. However, this chapter uses concepts and notation we have not yet covered in class. You should be able to understand the general idea behind these decision procedures given in this chapter. (Not all of the decision procedures given in class appear in the book.)

We've also seen the following decision procedures so far:

- Let L_1 and L_2 be regular languages. Decide if L_1 and L_2 have at least 1 string in common. (We covered this in discussion 10/18.)
- Let L_1 and L_2 be regular languages. Decide if $w \notin (L_1$ or $L_2)$. (We covered this in discussion 10/18.)
- Let A be a finite automaton. Decide if u is a prefix of some string $w \in L(A)$. (This is from homework 3.)
- Let D be a DFA. Decide if u is a substring of some $w \in L(D)$ such that u is your name. (This is from the sample midterm.)

Problem 2: Closure

We have seen lots of examples of closure. Let L_1 and L_2 be regular languages. Then L_3 is also regular when:

- **Union:** $L_1 \cup L_2 = L_3$. (Book)
- **Intersection:** $L_1 \cap L_2 = L_3$. (Discussion 10/10)
- **Difference:** $L_1 - L_2 = L_3$. (Homework 2)
- **Concatenation:** $L_1 \circ L_2 = L_3$. (Book, Discussion 10/10)
- **Exclusive Or:** $L_1 \oplus L_2 = L_3$. (Homework 2)
- **Complement:** $\overline{L_1} = L_3$. (Lecture)
- **Star:** $(L_1)^* = L_3$. (Book)
- **Echo(L_1):** $L_3 = \{a_1a_1a_2a_2 \dots a_na_n \in \Sigma^* \mid a_1a_2 \dots a_n \in L_1\}$. (Homework 2)

- **NOPREFIX(L_1):** $L_3 = \{w \in L_1 \mid$ no proper prefix of w is a member of $A\}$. (Book Exercise 1.40, Lecture)
- **NOEXTEND(L_1):** $L_3 = \{w \in L_1 \mid w$ is not the proper prefix of any string in $A\}$. (Book Exercise 1.40, Homework 2)

Homework 4 illustrates closure of operations for non regular languages.

Let L_1 and L_2 be context-free languages. Then L_3 is also context-free when:

- **Union:** $L_1 \cup L_2 = L_3$. (Lecture)
- **Concatenation:** $L_1 \circ L_2 = L_3$. (Lecture)
- **Star:** $(L_1)^* = L_3$. (Homework 5)

However, homework 5 illustrates that intersection and complement are not closed under context-free languages.

You need to be familiar with logic rules to use these statements. Remember De Morgan's Law states that $\overline{A \wedge B} = \overline{A} \vee \overline{B}$. Observe the following:

Proposition:	$A \wedge B \rightarrow C$	(Given, True)
Inverse:	$\overline{A} \vee \overline{B} \rightarrow \overline{C}$	(Not Necessarily True)
Converse:	$C \rightarrow A \wedge B$	(Not Necessarily True)
Contrapositive:	$\overline{C} \rightarrow \overline{A} \vee \overline{B}$	(Always True)

Consider the following example. Suppose we have $L_1 \circ L_2 = L_3$. We know that if L_1 is regular and L_2 is regular, then L_3 is regular. The contrapositive of this is that if L_3 is non regular, then L_1 is non regular or L_2 is non regular.

Can we claim that if L_1 is non regular or L_2 is non regular, then L_3 is non regular? No, this statement is in the form $\overline{A} \vee \overline{B} \rightarrow \overline{C}$ which is the inverse.

Can we claim that if L_3 is regular, then L_1 is regular and L_2 is regular? No, this statement is in the form $C \rightarrow A \wedge B$ which is the converse.

Problem 3 & 4: Context-Free Grammars

We've gone over several examples of grammars in class. You can also find examples in the book, discussion notes, and in homework 4.

Part 4(a):
Here are some example descriptions of languages in an easy-to-recognize form:

$$L = \{w \in \Sigma^* \mid w \text{ starts and ends with the same symbol}\}$$
$$L = \{w \in \Sigma^* \mid \text{the length of } w \text{ is odd}\}$$
$$L = \text{ the set of strings in } \Sigma^* \text{ with more } \texttt{a}\text{s than } \texttt{b}\text{s}$$

Look at how languages are specified in the book for more examples.

Part 4(b):
Do this the same way you have for previous homework assignments.

Part 4(c):
We discussed ambiguity in class, and it is also discussed in the book. You can just provide two different parse trees for a single string to prove that is ambigious. To prove that it is not ambiguous, you need to prove that there is only one possible parse tree for every string in the language.

Problem 5: Pumping Lemma

Using the pumping lemma for context-free languages is very similar to that for regular languages. However it is more tricky to work with.

To prove that a language is not context-free using the pumping lemma, you must show that there is some $w \in L$ with $|w| \geq p$ that contradicts the pumping lemma. You only need to show this for a single w, not all possible $w \in L$.

To show that there is some $w \in L$ with $|w| \geq p$ that breaks the pumping lemma, you must show that it is impossible to split the string w into $w = uvxyz$ such that it satisfies the pumping lemma conditions. This means you have to show that for every possible way you can split w, there is a contradiction. If there exists one way to split w into $uvxyz$ that satisfies the pumping lemma, then you need to choose another w or try to prove this language is non context-free with a different method.

Problem 6: Chomsky Normal Form

The algorithm for this was given in class, and is also in the book. The book also contains several examples. (We'll cover one in discussion, time permitting.)

Example: Pumping Lemma for CFL

Let $L = \{ww \mid w \in \{0,1\}^*\}$. Use the pumping lemma to show that L is not a CFL.

Front Matter:
Assume L is a CFL for the sake of contradiction. Then there exists some pumping lenth p such that for any $s \in L$ where $|s| \geq p$ the string s may be split into $s = uvxyz$ such that: (1) $uv^ixy^iz \in L$ $\forall i \geq 0$, (2) $|vy| > 0$, and (3) $|vxy| \leq p$.

Middle Matter (Attempt 1):
Let $s = 0^p10^p1$. Since this may be written as ww where $w = 0^p1$, we know $s \in L$. Also, $|s| = 2p + 2 \geq p$. Therefore, we should be able to split s into $s = uvxyz$ satisfying the pumping lemma conditions.

Lets go through some possible ways to split s. Since $|vxy| \leq p$, the possible strings vxy are:

Case 1: $xyz = 0^k$ where $1 \leq k \leq p$. This occurs when our string looks like:

$$\underbrace{0\,0\cdots0\,0}_{vxy}1\,0\,0\cdots0\,0\,1 \text{ or } 0\,0\cdots0\,0\,1\underbrace{0\,0\cdots0\,0}_{vxy}1$$

Since $|vy| > 0$, vy must be one or more 0s. Pumping down would cause our string to look like $0^{k-1}\,1\,0^p\,1$. We know that k is at least 1, and at most equal to p. Therefore $k-1$ ranges from 0 to $p-1$, and we can say $k-1 \neq p$. Therefore $uxz \notin L$ causing a contradiction.

Case 2: $xyz = 0^k1$ where $0 \leq k \leq p-1$. This occurs when our string looks like:

$$0\underbrace{0\cdots0\,0\,1}_{vxy}0\,0\cdots0\,0\,1 \text{ or } 0\,0\cdots0\,0\,1\,0\underbrace{0\cdots0\,0\,1}_{vxy}$$

Since $|vy| > 0$, then v or y must contain at least one symbol. Whenever y is non empty, then it must contain at least a single 1. Therefore $uxz \notin L$ since removing y results in a string 0^n1 where $n > p$.

If y is empty, then v must contain at least one 0. Therefore $uxz \notin L$ since removing v offbalances the number of zeros.

Case 3: $xyz = 10^k$ where $0 \leq k \leq p-1$. This occurs when our string looks like:

$$0\,0\cdots0\,0\underbrace{1\,0\,0\cdots0}_{vxy}0\,1$$

Similarly to case 2, this results in a contradiction.

Case 4: $xyz = 0^j10^k$ where $0 \leq j+k \leq p-1$. This occurs when our string looks like

$$0\underbrace{0\cdots0\,0\,1\,0\,0\cdots0}_{vxy}0\,1$$

Unfortunately, this case does not result in a contradiction. We can split the string as follows:

$$\underbrace{0\cdots0}_{u}\;\underbrace{0}_{v}\;\underbrace{1}_{x}\;\underbrace{0}_{y}\;\underbrace{0\cdots0}_{z}$$

Notice that $uv^ixy^iz = 0^{p-1+i}\,1\,0^{p-1+i}\,1 \in L$.

Since there does exist a way to split the string 0^p10^p1, this does *not* contradict the pumping lemma. Therefore, we need to look for another string.

Middle Matter (Attempt 2):
Instead, let $s = 0^p1^00^p1^p$. The proof using this string is in your book on page 127 (example 2.38). No matter how you split s, it results in a contradiction of the pumping lemma.

End Matter:
There is no way to split s such that it satisfies the pumping lemma conditions. This is a contradiction, therefore L is not a CFL.