HW #6

Sam Quinn

CS321

02/22/2016

1)

To show that each of the **classes** in the Chomsky Hierarchy are closed under **union**, we must show that there is a grammar , which can generate a string that is exactly a string that is created by .

Let's assume that **G2,** and **G1** do not share any variables in common. We will let **S1** and **S2** be the start symbols of **G1** and **G2.**  We can form a new language we will assign the start symbol . Now if we look at the new language **G3** that we have just created we will find that all the start symbols **S** will be replaced by either **S1 or S2.** This means that and thus we can say , proving that every **class** in Chomsky’s Hierarchy is closed under a union**.**

2)

Prove that is not a regular language.

Let us assume that **L** is regular. Choose a , where n is strictly prime. We will use **p,** which can only be prime numbers, to be the pumping length for **L**. To prove a pumping lemma we can use the facts that there are 3 subsections of a string **xyz** such that that will imply . Because we can assume that is also true. If we set which we know is greater than 1 we can show that and if we expand the value of we will get . With and being at least 2 we can say that is not prime forming a contradiction with our original statement. **L** is not regular.

3)

1. Show that is a context free language.

To show that a language is context free, we must must show that the Language exhibits at least one closure property.

L = {acc, aacccc, aaacccccc}

s -> aScc

s -> acc

As you can see from this output a concatenation closure property would be an easy property to prove based on the fact that **a** always comes before **c**.

let us assume that strings generated by **L** can be split into two **x|y** such that . as we can see you will always see that there are strings of **c**’s (**y**) concatenated to the end of a string of all **a**’s (**x**). By showing that this language does not need to keep local memory for this specific language we can safely assume that L is a CFG.

1. Show that the above is not regular.

One of the properties of that a regular language is closed under is *complementation* so if we take a language that is know to be not regular let us call this . Now if we take the assumption that **L** is regular then when we take the complement to form **L’.** If **L** is truly a regular language ($ is null, a\*b\* is regular), but is actually , which is not a regular language.

4)

Show that is a context sensitive language, which is not a context free language.

S -> LSR | $

L -> a

L -> b

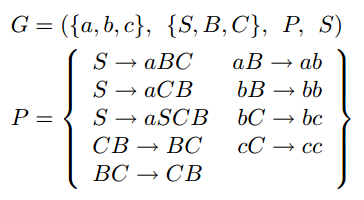
R -> c

R -> d

Because is context sensitive the strings will always be in the form **a|b|c|d** if the **b** or **c** strings are out of order then this would break the context sensitivity of this language. We can show that **L** is a context sensitive if it is accepted by a LBA. **L** is linearly bounded by **n** which would make this language have a finite length with the terminals in order with **a**’s to the left of **b**’s to the left of **c**’s to the left of **d**’s. Because the terminals have to be in order the language **L** can not be context free.

5)

Show that the following grammargenerates:



Showing start conditions:

aBC -> abC -> abc

aCB -> aBC -> abC -> abc

aSCB -> aaBCCB -> aabCCB -> aabCBC -> aabBCC -> aabbCC -> aabbcc

Now that the start conditions have been met we are able to see that this particular grammar is a CFL. Based on the CFL’s concatenation closure property we can say for certain that the grammaris created by the listed productions.

6)

Show that **CSL** is a proper subclass of **PRIM**, where **CSL** is the class of languages which have context sensitive grammars and **PRIM**  is the class of languages which have primitive recursive acceptors.

A Context sensitive language will always be a proper subclass of PRIM. PRIM is a recursive function that has a recognizer, which means it will halt every time. PRIM is a Type 0 language and every Type 0 languages with equivalent lengths on both sides will generate all of the Type 1 (CSL) languages. Because all CSL’s are linear bounded any recursively enumerable function could eventually get the CSL bound. This would suggest that the any CSL in a subset of PRIM because it is bounded by a number that would be able to be computed by a Primitive recursive function.