HW #5

Sam Quinn

CS321

02/15/2016

2.4)

Every finite language has a grammar

let sigma be our alphabet which contains all of the letters in the english alphabet, with the addition of the numbers from 0 to 9.

The productions necessary for this language could be;

The productions will allow any of the symbols with in to be added until the length of 10 is reached, which would make the string end with a terminal.

There are 26 letters and 10 numbers in , which means . To find the total number of strings contained in we will just need to enumerate all of the possible combinations of the 36 characters up to length 10, which is equal to 36 choose 10.

**Number of strings:**

**Algorithm:**

Check(s):

if length of s is > 10

Fail

for i = 0 until 9:

if s[i] is not in :

Fail

Pass

2.6)

**THE BOY PLAYS WITH THE GIRL**

S -> NP(THE) -> N(BOY) -> VP -> V(PLAYS) -> PREP(WITH) -> NP(THE) -> N(GIRL)

**THE BOY PLAYS WITH THE BALL**

S -> NP(THE) -> N(BOY) -> VP -> V(PLAYS) -> PREP(WITH) -> NP(THE) -> N(BALL)

**THE BALL PLAYS WITH THE BALL**

S -> NP(THE) -> N(BALL) -> VP -> V(PLAYS) -> PREP(WITH) -> NP(THE) -> N(BALL)

**WITH THE BALL THE BOY PLAYS**

S !-> PREP(WITH)

**THE GIRL PLAYS WITH THE PLAYS**

S -> NP(THE) -> N(GIRL) -> VP -> V(PLAYS) -> PREP(WITH) -> NP(THE) !-> V(PLAYS)

**THE BALL PLAYS WITH THE BAT**

S -> NP(THE) -> N(BALL) -> VP -> V(PLAYS) -> PREP(WITH) -> NP(THE) !-> ?(BAT)

**THE BOY PLAYS THE GUITAR**

S -> NP(THE) -> N(BOY) -> VP -> V(PLAYS) !-> NP(THE)

2.8)

**( A + (( B + C ) - D ))**  Yes

**(((((( B )))))**  Yes

**( (A + ( + ( B + ( - C + D ))))** No

**(A + A + A )** No

**(A+(A)+(A+(A)))**  No

2.9)

**Type 0 which is not Type 1:**

a -> b Any set of terminals or nonterminals can go to any set of terminals or nonterminals.

**Type 1 which is not Type 2:**

aBc -> abc Strings that are put in the right context can be transformed into any string of terminals or nonterminals.

**Type 2 which is not Type 3:**

S -> a A nonterminal string can be transformed into any string containing terminals and nonterminals.

210)

1. **{S -> aSb, B -> SSa, aB -> Ba}**  = type 0
2. **{S -> a, A -> aS, B -> bS, S -> A}** = type 3
3. **{S -> a, aSBa -> S, aaS -> aa, B -> b}**  = type 1
4. **{S -> A, A -> aSS, B -> SSa, S -> b, A -> B}** = type 2

2.12)

The language will produce only strings of even sets of a’s increasing by 2n each time. For example .

For L to be a regular language there must be restricted production rules, such that there is only one nonterminal on the lefthand side and only a single terminal possible followed by a nonterminal.

To show that the language L is a regular language we must show that the you can only create the strings adhering to the productions listed above. To show this easily we can show that if the strings created with L are all even since according to L there are only strings of a’s that are of 2n length. Every number that is multiplied by 2 is an even number, which states that there can only be a’s in even subsets. With the production rules listed above there is no way you could create a string including terminals and nonterminals that would produce an odd number of a’s. With that said the language L is a “regular language” since it only has productions that are terminals potentially followed by one nonterminal.

**Proof:**

Assume that the language L(G) is regular, show that L(G+1) is true.

**Base case:**

L(1) = aa L(2) = aaaa

**Inductive step:**

let k = n + 1, show that L(k) is still in the language L.

Because the language L multiplies the input value by 2 every value will be equal. The values will also be multiplied by 2 which will allow us to change the language a little.

Because we have split up the section of the original the output from the new modified language will be a string of all a’s followed by all b’s. Circling back to the original assumption that the string produced by L(G) is all evens still holds. Even if the new value of k is odd and is inputted into the language L, 2 odd strings concatenated together will produce an even string length. The only way for the language L to get an odd length string returned would be from one of the sides being odd while the other is even, which will never happen since they share the same n value being inputted. Therefore any string produced by L(G) is in the Language L.