Amir Ayoub

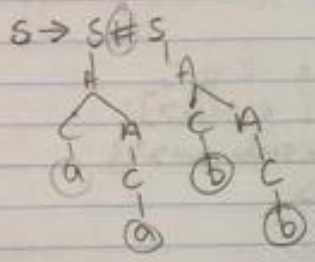
Class: 314 -07

Homeork 1

Grammer G:

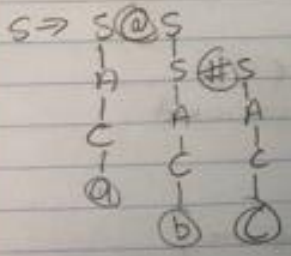
S → A | S#S | S@S
A → C | CA
C → a | b | c

I. aa##bb



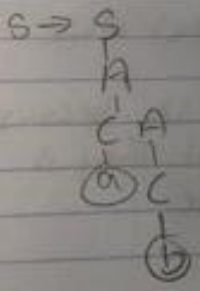
∴ we prove that this can not be drived because ## is not provided in the language

II. a@b#c



∴ we prove that this is Unambiguous

III. ab



∴ we prove that with the language provided it is unambiguous

Rewrite grammer 6

S → S@S | P ∞ this removes the ambiguity
P → P#P | A
A → C | CA
C → a | b | c

2. Grammer #

<Statement> → Assignment | while
<Assignment> → <var> = <value> [, <value>];
<while> → while(<value>) {{<statement>}}
<Value> → <var> | <Number>
<Var> → a | b | c
<Number> → 0 | 1

Productions:

I. a=0, b, yes it is in the language because when you choose Assignment they both exist

II. a=b, c, 1, No, because there is no way we can get 1

III. while(a){b=0; while(b){}} No, there is no way to go to the last while

IV. a=1; while(a){a=1; while (a=0;}, No, because after you choose statement. they is no way to get back to while

3. Write a grammer for BNF in BNF
(Not EBNF, e.g. no curly Braces)

Example

Assign → id := expr
id → A|B|C
expr → id + expr | id * expr | (expr) | id

A := B*(A+C)