**CMPS 455, Handout No.2, Assignment No.2**

**Grammar of some rules in C++**

**Recall:** to find a grammar of a statement, the best is to construct its FA and then use FA to write its grammar.

**Examples.** Write the grammar of the following FAs.

|  |  |
| --- | --- |
| **FA( Finite Automata)** | **CFG( Context Free Grammar)** |
| Σ={a,b }, L=aba\*  a b a  A B C | A🡪aB  B🡪bC  C🡪aC  C🡪λ  Terminals={a,b }, non-terminals={A,B,C} |
| L = “powers of a “ or “powers of b” = a\* + b\*  1st, construct FA for the language. 2nd use the FA to write the CFG of the language  b a    b a  B X A  X is final: L1 = λ  A is final: L2=aa\* , B is final: L3=bb\*  L=L1+L2+L3 = λ + aa\* + bb\* , λ= λ + λ  = (λ +aa\*) + (λ+bb\*)=a\* + b\* | X🡪aA  X🡪bB  X🡪λ  A🡪aA  A🡪λ  B🡪bB  B🡪λ  We may have more than one grammar per line. If all have the same state on the left. We separate the right hand side item with “|” reads or  X🡪bB | aA | λ  A🡪aA |λ  B🡪bB |λ |

**Tracing grammars: T**o determine whether a given word is accepted or rejected by a grammar, we use one of the following two methods:

**Example.** Trace the following CFG to determine whether word w is accepted or rejected:

CFG: A🡪aB, B🡪bC, C🡪aC | λ. Terminals={a,b,c}, Non-terminals={A,B,C}

|  |  |
| --- | --- |
| **Parsing tree method** | **Left-most-derivation method** |
| w1 = abaaa: A rule used  A🡪aB  a B  B🡪bC  b C  C🡪aC  a C  C🡪aC  a C  C🡪aC  a C  a b a a a λ =w1  w2=abaab:  A  a B  b C  a C  `  a C    a b a a does not generate b  CFG rejects w2 | A🡪 aB   * abC * abaC * abaaC * abaaaC * abaaaλ = w1   **Note.** In left-most-derivation you ***must*** do only one substitution at time. For example  A🡪 aB , use both B🡪aB and B🡪bC to get   * abaC   is illegal, two substitutions at once  A🡪 aB   * abC * abaC * abaaC * abaa(does not generate b)   CFG rejects w2 |

**Example.** An Identifiers in C++ could be string of letters, digits, or underscores. Identifiers must begin with a letter or underscore. Construct a grammar for identifiers in C++

|  |  |
| --- | --- |
| **FA representing the grammar of identifiers in C++** | **CFG** |
| Letter(L)  Letter (L)  Digit (D)      Underscore(U) Underscore(U)  Ω ɸ  Letter = LowerCase + UpperCase: L🡪 |  Theta( Θ ),Gamma(Γ ) 🡪a|b|c|…….|z  🡪A|B|…….|Z  Digits: D🡪0|1|2|……|9  Underscore: U🡪\_ | Ω🡪Lɸ | Uɸ  L🡪 | Γ  🡪a|b|c|….|z  🡪A|B|C|….|Z  ɸ🡪Lɸ | Dɸ | Uɸ | λ  D🡪0|1|2|……..|9  U🡪\_  Terminals={a..z, A..Z, 0..1, \_}  Non-terminals={ D,U} |

**Example:** Trace the identifiers wi to determine whether each identifier is accepted or not:

|  |  |
| --- | --- |
| w1=Tax\_20  Ω …………………………. Ω🡪Lɸ, L🡪  L ɸ…………………… ɸ🡪Lɸ, L🡪  Γ L ɸ  Θ L ɸ  Θ U ɸ…….……. ɸ 🡪Dɸ  D ɸ  D ɸ  T a x \_ 2 0 λ =accepted | w2=\_ \_ \_ 2  Ω……………………….Ω🡪Uɸ  U ɸ………………….ɸ🡪Uɸ  U ɸ…………..ɸ🡪Uɸ  U ɸ…….ɸ🡪Uɸ  D ɸ..ɸ🡪Dɸ  D🡪2,ɸ🡪λ  \_ \_ \_ 2 λ =w2 accepted |

**Example.** Use the given CFG. Trace the grammar to determine whether w=a\*(b+a) is accepted or not.

|  |  |  |
| --- | --- | --- |
| **CFG** | **Parsing tree** | **Left-most-derivation** |
| E🡪E+T  E🡪 E- T  E🡪 T  T🡪T\*F  T🡪T/F  T🡪 F  F🡪(E )  F🡪 a  F🡪 b | E  |  T  / | \  T \* F  | / | \  F ( E )  / | \  E + T  | |  T F  |  F  a \* ( b + a ) | E🡪 T  🡪T \* F  🡪F \* F  🡪a \* F  🡪a \* ( E )  🡪a \* ( E + T )  🡪a \* ( T + T )  🡪a \* (F + T )  🡪a \* ( b + T)  🡪a \* ( b + F )  🡪a \* ( b + a )  Note: ONLY one substitution at each step |

**Example**. Given the following FA with Σ={a,b,c}. We want to write a program to determine whether a given word w=abbcc$ ($ marks the end of string) is accepted or not.

|  |  |
| --- | --- |
| **FA** | **Transition table** |
| a a  s0 s1  b,c  s2 c b,c  a,b s3  a,b,c a,b,c  s4 a,b,c s5 | |  |  |  |  | | --- | --- | --- | --- | | Input  State(s) | a=0 | b =1 | c = 2 | | 0 | 1 | 2 | 2 | | 1 | 1 | 3 | 3 | | 2 | 4 | 4 | 2 | | 3 | 5 | 5 | 5 | | 4 | 4 | 4 | 4 | | 5 | 4 | 4 | 4 |   Each state must be **complete**, means at each state we have to know what happens when the input is a,b,or c  **Initial state=s0 or just 0**  **Final states=s2 and s4 or just 2 , 4** |

introduce the transition table to your program using a 2 dimensional array: Table

|  |  |
| --- | --- |
| int Table[6][3]={ {1,2,2}, { 1,2,3},{4,4,2},  {5,5,5},{4,4,4},{4,4,4} };  int state=0, i=0, col=0; // start at state 0  string w=”abbcc$”;  while ( i < w.length() )  { switch( w[i] )  { case ‘a’ : col=0; break;  case ‘b’ : col=1; break;  case ‘c’ : col=2; break; | case ‘$’: if (state==2||state==4)  cout<<w<<” is accepted\n”;  else  cout<<w<<” is rejected\n”;  break;  }//end of switch  state= Table[state][col]; //go to the next state  ++i; //go to the next input letter  }// end of while |

**Example.** Write a CFG for numbers of types int and float in c++

|  |  |
| --- | --- |
| **FA** | **CFG** |
| Unsigned int (only positive integers, such as 123 )  U X  Digit(D) Digit(D)  D🡪0|1|2,….|9 | U🡪D X  X🡪DX | λ  D🡪0|1|2|3|…..|9 |
| int (such as : +25, -25, or just 25 )  D  Sign(+,- , λ) D  S  I H K  U=unsigned int | I🡪 SH; H🡪DK; K🡪DK|λ ; S🡪+ | - | λ  D🡪0|1|2|….|9  **OR:** or a sign followed by an unsigned int  I 🡪SU ; U🡪DX; X🡪DX | λ  D🡪0|1|2|……..|9  S🡪+|-|λ |
| float (such as: +25., -25., 25.3, +.123)  +25.13, -25.13, 25.13  25. , +25., -25.    +.25, -.25, .25  F🡪 I. | I. U | I | S.U  I🡪SU; U🡪DX; X🡪DX|λ;  D🡪 0|1|2|3|………|9  Integer numbers are also considering as floating numbers  S🡪+| - | λ | |

**Example.** Trace the above CFG for n1=+123, n2= -.02, n3= -2.5

|  |  |
| --- | --- |
| CFG: F is the initial state  F🡪 I. | I. U | S.U | I | U  I🡪SU  U🡪DX  X🡪DX | λ  D🡪0| 1| 2|…………|9  S🡪 + | - | λ | n1=+123 F  | ……………………F🡪 I  I  / \ ……………………..I🡪SU  S U …………………….U🡪DX  D X………………….X🡪DX  D X………….X🡪DX  D X  + 1 2 3 λ=+123=n1 |
| n2 = -.25 F  / | \  S . U  / \  D X  / \  D X   * . 2 5 λ = -.25=n2 | n3=-2.5 F  / | \  I . U  / \ / \  S U D X  D X  - 2 λ . 5 λ =-2.5= n3 |

**Example**. Write a CFG for simple function’s prototypes in c++

|  |  |
| --- | --- |
| Examples of function’s prototype in C++ | CFG for function’s prototype in C++ |
| 1. void f( ) 2. void f(int a, int b, int c); 3. int f(int a, int b int c);     ) at the end  List of 0 or more arguments  ( left parenthesis  Identifier  type | <Fun>🡪<T> <id> ( <arguments> ) ;  <T>🡪void| int | char|…….  <id>🡪 (few examples before )  <arguments> 🡪<A>  <A> 🡪<T><id>  🡪<T><id>, <A> |λ |

**Example.** Trace the following function’s prototype: int f( int a, float b, char c);

|  |
| --- |
| <Fun>  <type> <id> ( <arguments> ) ;    <A>  <T> <id> , <A>  <T> <id> , <A>  <T><id>  int f ( int a , float b , char c ) ; |

Example. Write a CFG for if-else statements in C++

|  |  |
| --- | --- |
| Examples of if-else statement in C++ | CFG for if-else statements in c++ |
| 1. if( a )   cout<<a;   1. if( a> b )   {  Max=a;  cout<<Max<<endl;  }  else  {  <statements>  }   1. if( <condition> )   { <statements> }  else if ( <condition> )  { <statements> }  else if ( <condition> )  { <statement> }  else  { <statements> } | <if-else>🡪if ( <condition> ) {<statements> }  🡪if( <condition> ){statements>}else{<statements>}  🡪if( <condition> ){ <statements> }  else if (<condition>) { <statement> }  else if (<condition> ){ <statement> }  else {<statements> }  **or**  <if-else> 🡪if ( <C> ) <S> <X>  <X> 🡪 λ  🡪else <S>  🡪 else if( <C> )<S><X>  <C> 🡪……  <S> 🡪……. |

Example. Trace the following :

|  |  |
| --- | --- |
| if( a>b && a > c )  { max= a; }  else if (b > c )  { max= b; }  else  { max= c; } | <if-else >  if ( <C> ) <S> <X>  else if (<C> ) <S> <X>  else <S>  if ( a>b && a >c ){ max=a;} else if ( b>c ) {max=b;}else {max=c;} |

CMPS 455, Assignment No.2 Names: …………………………………………………………

(CFG of some statements in C++)

1. Consider the following grammar:

**S🡪aSbB | A | c**

**A🡪cA | c**

**B🡪 d | A**

Trace the grammar to determine which of the following words are accepted or rejected?

i. accbc (use parsing tree) ii. acccdd (use left-most-derivation)

2. Given the following CFG:

S🡪I = E

E🡪E+T | E-T | T

T🡪T\*F | T/F | F

F🡪( E ) | I

I🡪 a | b

Use parsing to trace the grammar and decide which of the following statements are accepted or

rejected

1. a=a\*( b- a\*a) ii. b=a\*b – b\*(a + b)
2. Find the language of the following grammars:

|  |  |
| --- | --- |
| * 1. S🡪aS | bB | aA |λ   B🡪 bB | aA  A🡪aA | bA | λ | * 1. S🡪aS | bA |λ   A🡪aA | bX |λ  X🡪aX | bX |λ |

1. Find a CFG for each of the following languages

(i) L=a\* + b\* (ii) L=a\*b\*c\* (iii) L=ab\* + ba\*+c

1. Find a CFG for

|  |  |
| --- | --- |
| 1. ***Switch statement*** in C++   switch ( <condition> )  {  case v1: <statements> break;  case v2: <statements> break;  ……  default : <statements> break;  }  Where vi are int of char type | 1. ***Do-while loop*** in C++   do{  <statements>  }while (<condition> ) |

**Programming assignment**

Write a program to find the value of a postfix expression. Variables are one or more characters.

**Sample I/O:**

|  |  |
| --- | --- |
| Enter a postfix expression with a $ at the end:  20 num1 45 + tom - \* $  Enter the value of num1: 10  Enter the value of tom: 5  Expression’s value is 1000  CONTINUE(y/n)? y | Enter a postfix expression with a $ at the end: myscore yourscore 45 + 100 + \* $  Enter the value of myscore: 3  Enter the value of yourscore: 5  Expressions value is 450  CONTINUE(y/n)? n |

**CMPS 455 quiz No.2 Name………………………………………………..**

**Starting from this quiz, all quizzes must be answered individually and will be graded individually as well.**

Given the following CFG

E🡪E+T | E-T | T

T🡪T\*F | T/F | F

F🡪( E ) | a | b

Trace statement: (a+b) \* (a-b) using both parsing and left-most-derivation methods

1. Parsing technique
2. Left-most-derivations method