CSE 401 - LL Semantics - Week 5

Canonical LL(1) Problems and their Solutions:

FIRST Conflict:
Both productions of A have α in their FIRST sets
0. A ::= αβ | αγ

Solution:
Factor out the prefix (α)
0. A ::= α Tail
1. Tail ::= β | γ

FIRST FOLLOW Conflict:
B is nullable, α in FIRST & FOLLOW
0. A ::= B α
1. B ::= α | ε

Solution:
Substitute B into A
0. A ::= αα | α
Factor out the prefix (α)
0. A ::= α Tail
1. Tail ::= α | ε

Left Recursion:
Special FIRST conflict: β in FIRST for both productions
0. A ::= A α | β

Solution:
Create recursive tail from suffix of recursive production
1. Tail ::= α Tail
Append Tail to non recursive productions
0. A ::= β Tail
1. Tail ::= α Tail
Add empty string (ε) as a rhs for the tail production
0. A ::= β Tail
1. Tail ::= α Tail | ε

Indirect Left Recursion:
Recursively alternates between A & B
0. A ::= B β
1. B ::= A | α

Solution:
Substitute B into A
0. A ::= A β | α β
Solve like normal Left Recursion
0. A ::= α β Tail
1. Tail ::= β Tail | ε

Edit the following Grammars to make them LL(1). Then walk through the top down parse for the string given in the parenthesis.

Grammar 1 (“azx”)

0. S ::= a B | a w
1. B ::= C x | y
2. C ::= ε | z

Grammar 2 (“ax”)

0. S ::= a B
1. B ::= C x | y
2. C ::= ε | x

Grammar 3 (“azx”)

0. S ::= S B | a | w
1. B ::= C x | y
2. C ::= ε | z

Grammar 4 (“azx”)

0. S ::= B w | a B
1. B ::= S | z x

Example pseudocode of a recursive-descent parser for grammar 2: (Helpful example for Homework 3: Programming!!)

```
// S ::= aB
void S() {
    match(a)
    B()
}
```

```
// B ::= Cx | y
void B() {
    if next is y {
        match(y)
    } else {
        C()
        match(x)
    }
}
```

```
// C ::= ε | x
// *One* implementation of
// the C() method
void C() {
    if next is x {
        match(x)
    } else if (lookahead is
        not in follow set for C){
        error
    }
    // else, epsilon so do
    // nothing
}
```