1. Because we can write a;b;c this CFG is ambiguous. Firstly a;b;c. are all expressions but before we parse to find that out we are met with the dilemma. Should we parse as so, (a;b);c with <a;b>= <Expr> and c = <Expr> or should we parse as <a>;<b;c> with b;c being the expression thats together.

2. To get rid of ambiguity we can make an expression list
↓
<id> ::= a\|b\|c\| . . . z
<dig> ::= 0\|1\|2\|. . . 9
<expr> ::= (<>)\| <dig>\| <id> \| let <id> = <expr> in <expr> \|
<exprlst>

<Exprlst> ::= <exprlst>; <expr> \| <Expr> \| begin <Expr> end

3. let x=5 in a;b;c
We know that there will be no error because a will be looked at as the <exprlist> and then the other two (b;c) will be the expression. This allows the code to work from left to right.