Definición de la gramática

lexer

```
// DELETE THIS CONTENT IF YOU PUT COMBINED GRAMMAR IN Parser TAB
lexer grammar ExprLexer;

AND : 'and' ;
OR : 'or' ;
NOT : 'not' ;
EQ : '=' ;
COMMA : ',' ;
SEMI : ';' ;
LPAREN : '(' ;
RPAREN : ')' ;
LCURLY : '{' ;
RCURLY : '}' ;
SUM : '+' ;
SUB : '-' ;
MUL : '*' ;
DIV : '/' ;
POW : '^' ;
DOT : '.' ;
COLON : ':' ;
ASK : '?' ;

INT : [0-9]+ ;
ID: [a-zA-Z_][a-zA-Z_0-9]* ;
WS: [ \t\n\r\f]+ -> skip ;
```

Parser

```
parser grammar ExprParser;
options { tokenVocab=ExprLexer; }

program
    : stat EOF
    | def EOF
    ;

stat: ID '=' expr ';'
    | ID DOT ID '=' expr ';'
    | expr ';'
    ;

def : ID '(' ID (',' ID)* ')' '{' stat* '}' ;

expr: ID
    | INT
    | func
    | ID DOT ID
    | ID DOT func
    | 'not' expr
    | expr 'and' expr
    | expr 'or' expr
    | expr (MUL | DIV) expr
    | expr (SUM | SUB) expr
    | expr POW expr
    | expr '?' expr ':' expr
    | expr '?' expr
    ;

func : ID '(' expr (',' expr)* ')'
     | ID '('  ')'
;
```

Ejemplos de uso

Asignación como statement
Expresiones aritméticas con precedencia de operadores

```
f(x,y) {
    a = 3+foo;
    x* x + y * x;
    }
```

Condicional simple (exp1 ? exp2)
Condicional completo (exp1 ? exp2 : exp3)
Asignación y Condicionales

```
f(x,y) {

    x ? x+1 ;
    y ? y+ 1 : y-1;
    z = y ? y+ 1 : y-1;
}
```

Acceso a estructura

```
f(x,y) {
    x.foo;
    x.foo = 4;
}
```

Envio de mensajes sin argumento
Envio de mensajes con argumento
Envio de mensajes con argumentos

```
f(x,y) {
    x.foo();
    x.foo(4);
    x.bar(x.foo, 4);
}
```