Universidad Rafael Landívar
Compiladores
Catedrático: Ing. Juan Carlos Soto
Sección: 01

# Laboratorio 2

Iván Alexander Canel García
1301019

Guatemala, 23 de octubre del 2021
Campus Central

# Laboratorio 02

Parser generator

## Objetivo

- Explorar las herramientas para generación de código para los analizadores léxico y sintáctico.
- Introducir la traducción basada en sintaxis en la determinación de un valor resultante.

## Requerimiento

$$\boldsymbol{bexpr} \rightarrow \boldsymbol{bexpr}\ or\ \boldsymbol{bterm} \mid \boldsymbol{bterm}$$
$$\boldsymbol{bterm} \rightarrow \boldsymbol{bterm}\ and\ \boldsymbol{bfactor} \mid \boldsymbol{bfactor}$$
$$\boldsymbol{bfactor} \rightarrow not\ \boldsymbol{bfactor} \mid (\boldsymbol{bexpr}) \mid true \mid false$$

- Generar el analizador LALR(1) de forma manual para la gramática indicada.
- Construir un programa con YACC/FLEX que permita calcular el resultado de la operación booleana según la gramática indicada.
- Cotejar el analizador creado manualmente con el generado mediante

### Entrada

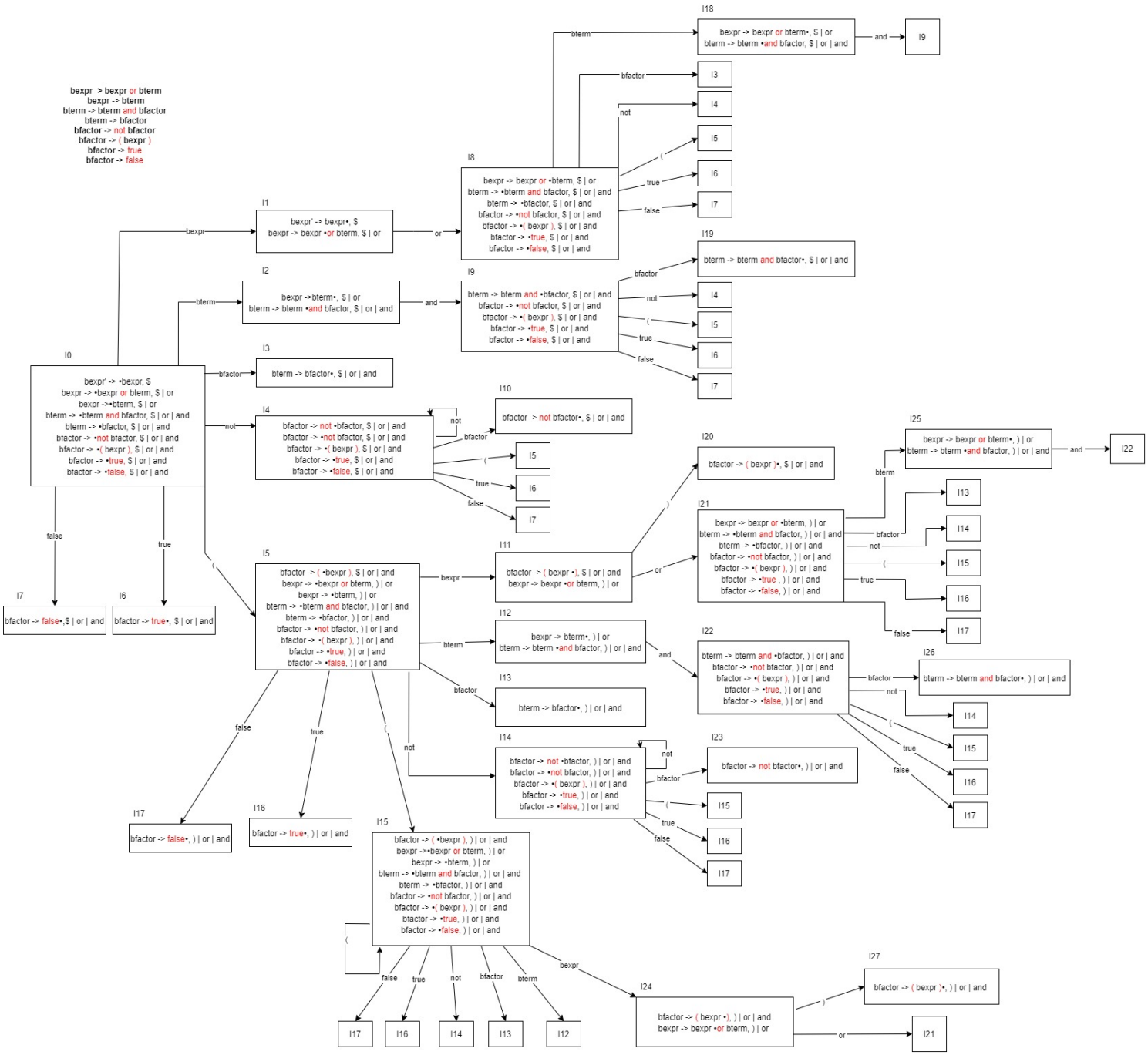
$$not(true\ or\ false)\ and\ true$$

### Salida

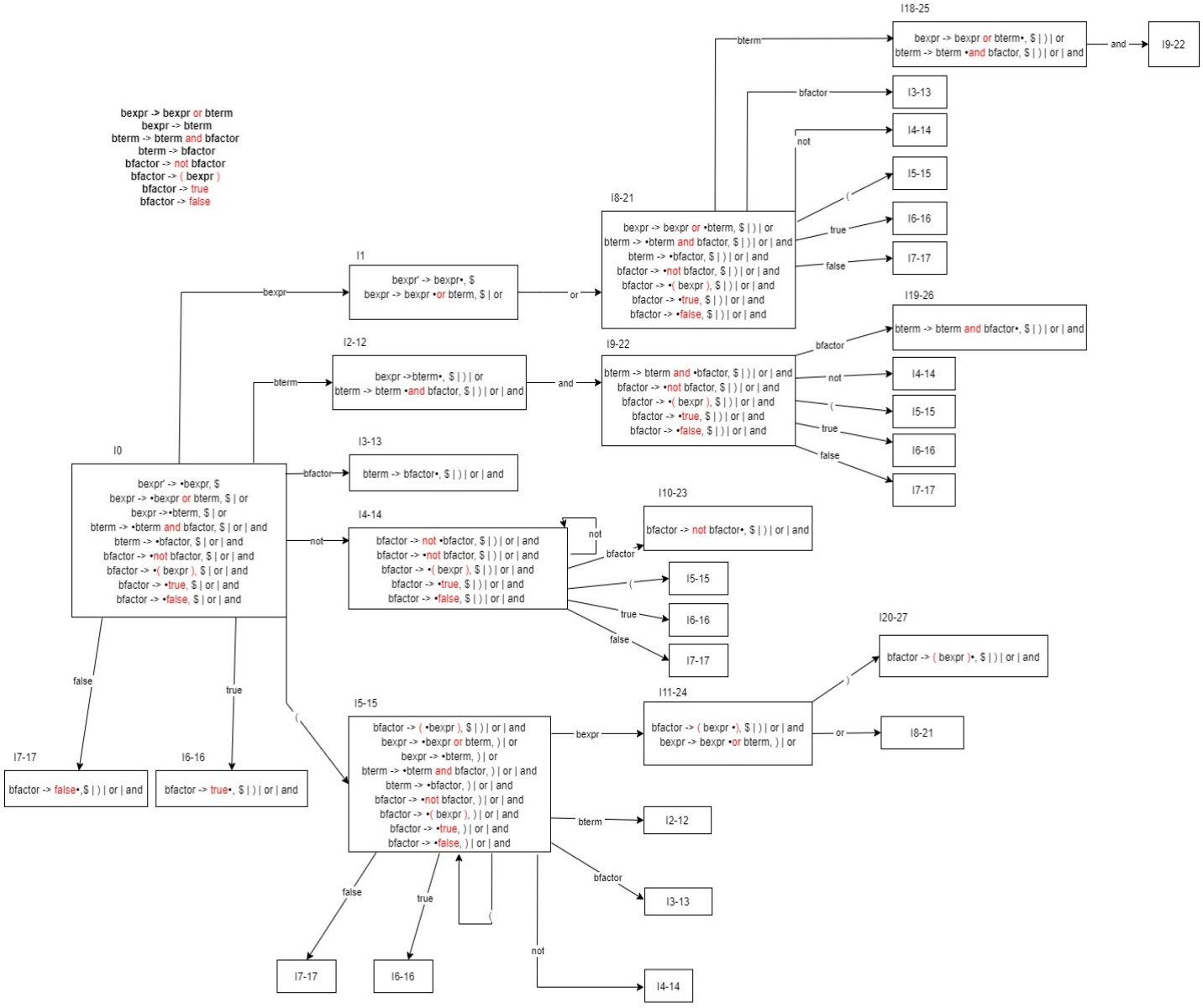
$$false\ (0)$$

- Generar el analizador LALR(1) de forma manual para la gramática indicada.

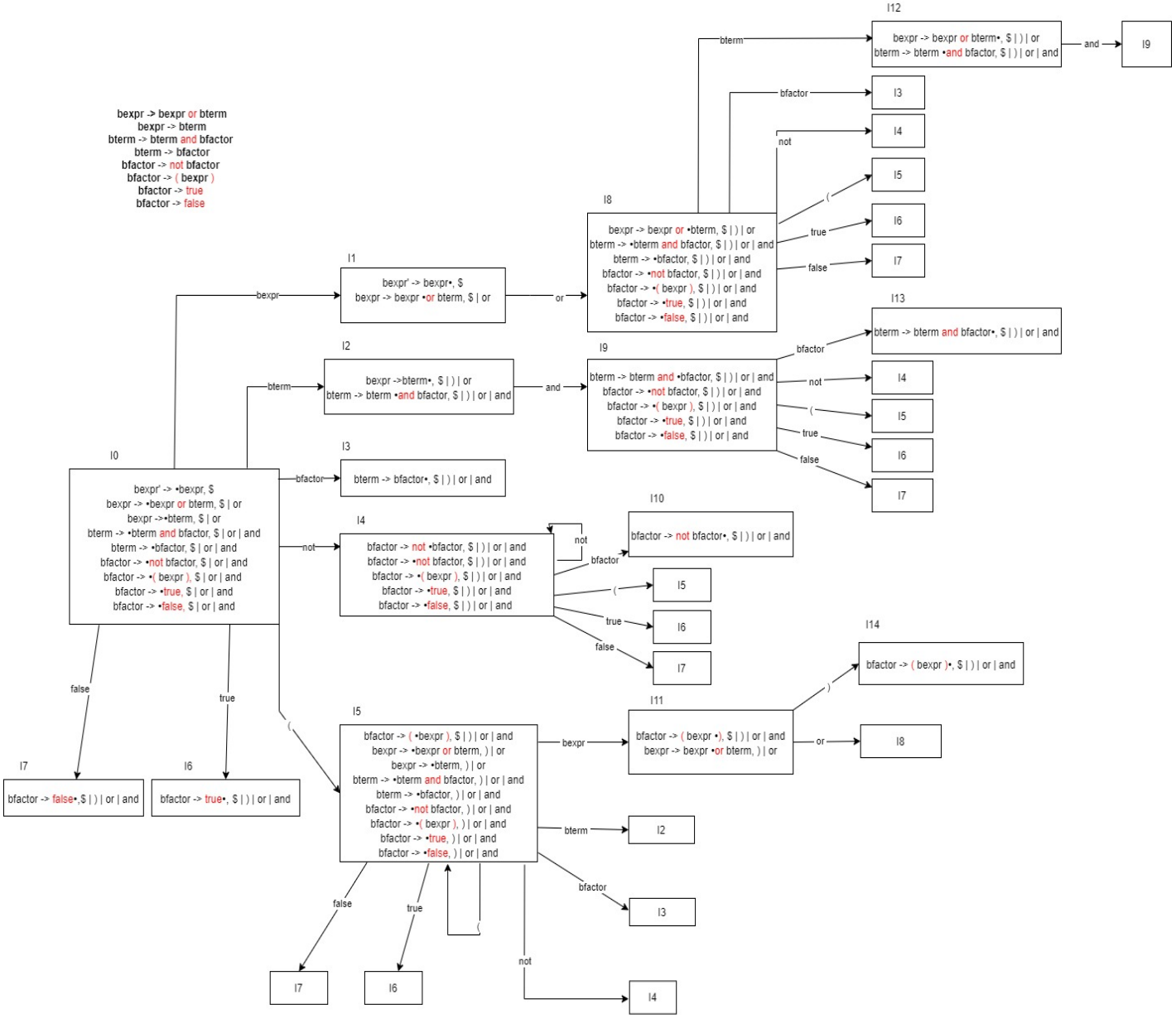
# CLR(1)



# LALR(1)


bexpr -> bexpr or bterm
bexpr -> bterm
bterm -> bterm and bfactor
bterm -> bfactor
bfactor -> not bfactor
bfactor -> ( bexpr )
bfactor -> true
bfactor -> false
I0
bexpr' -> •bexpr, $
bexpr -> •bexpr or bterm, $ | or
bexpr ->•bterm, $ | or
bterm -> •bterm and bfactor, $ | or | and
bterm -> •bfactor, $ | or | and
bfactor -> •not bfactor, $ | or | and
bfactor -> •( bexpr ), $ | or | and
bfactor -> •true, $ | or | and
bfactor -> •false, $ | or | and
I1
bexpr' -> bexpr•, $
bexpr -> bexpr •or bterm, $ | or
I2-12
bexpr ->bterm•, $ | ) | or
bterm -> bterm •and bfactor, $ | ) | or | and
I3-13
bterm -> bfactor•, $ | ) | or | and
I4-14
bfactor -> not •bfactor, $ | ) | or | and
bfactor -> •not bfactor, $ | ) | or | and
bfactor -> •( bexpr ), $ | ) | or | and
bfactor -> •true, $ | ) | or | and
bfactor -> •false, $ | ) | or | and
I5-15
bfactor -> ( •bexpr ), $ | ) | or | and
bexpr -> •bexpr or bterm, ) | or
bexpr -> •bterm, ) | or
bterm -> •bterm and bfactor, ) | or | and
bterm -> •bfactor, ) | or | and
bfactor -> •not bfactor, ) | or | and
bfactor -> •( bexpr ), ) | or | and
bfactor -> •true, ) | or | and
bfactor -> •false, ) | or | and
I6-16
bfactor -> true•, $ | ) | or | and
I7-17
bfactor -> false•,$ | ) | or | and
I8-21
bexpr -> bexpr or •bterm, $ | ) | or
bterm -> •bterm and bfactor, $ | ) | or | and
bterm -> •bfactor, $ | ) | or | and
bfactor -> •not bfactor, $ | ) | or | and
bfactor -> •( bexpr ), $ | ) | or | and
bfactor -> •true, $ | ) | or | and
bfactor -> •false, $ | ) | or | and
I9-22
bterm -> bterm and •bfactor, $ | ) | or | and
bfactor -> •not bfactor, $ | ) | or | and
bfactor -> •( bexpr ), $ | ) | or | and
bfactor -> •true, $ | ) | or | and
bfactor -> •false, $ | ) | or | and
I10-23
bfactor -> not bfactor•, $ | ) | or | and
I11-24
bfactor -> ( bexpr •), $ | ) | or | and
bexpr -> bexpr •or bterm, ) | or
I18-25
bexpr -> bexpr or bterm•, $ | ) | or
bterm -> bterm •and bfactor, $ | ) | or | and
I19-26
bterm -> bterm and bfactor•, $ | ) | or | and
I20-27
bfactor -> ( bexpr )•, $ | ) | or | and
bexpr
bterm
bfactor
not
(
true
false
or
and
)

## Modificando estados



# Tabla LALR(1)

| State | ACTION | | | | | | | | GOTO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | or | and | not | ( | ) | true | false | $ | bexpr' | bexpr | bterm | bfactor |
| 0 | | | S4 | S5 | | S6 | S7 | | | 1 | 2 | 3 |
| 1 | S8 | | | | | | | Accept | | | | |
| 2 | R2 | S9 | | | R2 | | | R2 | | | | |
| 3 | R4 | R4 | | | R4 | | | R4 | | | | |
| 4 | | | S4 | S5 | | S6 | S7 | | | | | 10 |
| 5 | | | S4 | S5 | | S6 | S7 | | | 11 | 2 | 3 |
| 6 | R7 | R7 | | | R7 | | | R7 | | | | |
| 7 | R8 | R8 | | | R8 | | | R8 | | | | |
| 8 | | | S4 | S5 | | S6 | S7 | | | | 12 | 3 |
| 9 | | | S4 | S5 | | S6 | S7 | | | | | 13 |
| 10 | R5 | R5 | | | R5 | | | R5 | | | | |
| 11 | S8 | | | | S14 | | | | | | | |
| 12 | R1 | S9 | | | R1 | | | R1 | | | | |
| 13 | R3 | R3 | | | R3 | | | R3 | | | | |
| 14 | R6 | R6 | | | R6 | | | R6 | | | | |

- Construir un programa con YACC/FLEX que permita calcular el resultado de la operación booleana según la gramática indicada.

**Scanner.lex**

```
OR      [Oo][Rr]
AND     [Aa][Nn][Dd]
NOT     [Nn][Oo][Tt]
TRUE    [Tt][Rr][Uu][Ee]
FALSE   [Ff][Aa][Ll][Ss][Ee]

%%
[\n()]    return yytext[0];
[ \t]     ; { /* Espacios en blanco */ }
{AND}   {return AND; }
{OR}    {return OR; }
{NOT}   {return NOT; }
{TRUE} {yylval = 1; return (TRUE);}
{FALSE} {yylval = 0; return (FALSE);}
.                   printf("Error\n");
%%

int yywrap(void)
{
        return 0;
}
```

**Grammar.y**

```
%{
        #include <ctype.h>
        #include <stdio.h>
        #define YYSTYPE double /* double type for Yacc stack */

        extern int yylex();
        void yyerror(char *msg);
%}

%token OR
%token AND
%token NOT
%token TRUE
%token FALSE

%%
lines    : lines bexpr '\n' { printf($2 ? "TRUE(1)\n" : "FALSE(0)\n") ;}
                | lines '\n'
                | /* empty */
                ;


bexpr  : bexpr OR bterm { $$ = $1 || $3 }
                | bterm;
```

```
bterm  : bterm AND bfactor { $$ = $1 && $3 }
                | bfactor;

bfactor: NOT bfactor { $$ = !$2 }
                | '(' bexpr ')' { $$ = $2 }
                | TRUE { $$ = $1 }
                | FALSE { $$ = $1 }
%%

#include "lex.yy.c";

void yyerror(char *msg){
        printf("Error de sintaxis");
        exit(0);
}

main(){
        yyparse();
        return 0;
}
```

- Cotejar el analizador creado manualmente con el generado mediante

$$not(true\ or\ false)\ and\ true$$

**Manual**

| Pila de Estados | Pila | Por Leer | Acción |
|---|---|---|---|
| I0 | # | NOT(TRUE OR FALSE) AND TRUE $ | Shift I4 a la pila |
| I0 I4 | NOT# | (TRUE OR FALSE) AND TRUE $ | Shift I5 a la pila |
| I0 I4 I5 | (NOT# | TRUE OR FALSE) AND TRUE $ | Shift I6 a la pila |
| I0 I4 I5 I6 | TRUE ( NOT# | OR FALSE) AND TRUE $ | Reduce con la regla 7 |
| I0 I4 I5 I3 | bfactor ( NOT# | OR FALSE) AND TRUE $ | Reduce con la regla 4 |
| I0 I4 I5 I2 | bterm ( NOT# | OR FALSE) AND TRUE $ | Reduce con la regla 2 |
| I0 I4 I5 I11 | bexpr ( NOT# | OR FALSE) AND TRUE $ | Shift I8 a la pila |
| I0 I4 I5 I11 I8 | OR bexpr ( NOT# | FALSE) AND TRUE $ | Shift I7 a la pila |
| I0 I4 I5 I11 I8 I7 | FALSE OR bexpr ( NOT# | ) AND TRUE $ | Reduce con la regla 8 |
| I0 I4 I5 I11 I8 I3 | bfactor OR bexpr ( NOT# | ) AND TRUE $ | Reduce con la regla 4 |
| I0 I4 I5 I11 I8 I12 | bterm OR bexpr ( NOT# | ) AND TRUE $ | Reduce con la regla 1 |
| I0 I4 I5 I11 | bexpr ( NOT# | ) AND TRUE $ | Shift I14 a la pila |
| I0 I4 I5 I11 I14 | ) bexpr ( NOT# | AND TRUE $ | Reduce con la regla 6 |
| I0 I4 I10 | bfactor NOT# | AND TRUE $ | Reduce con la regla 5 |
| I0 I3 | bfactor# | AND TRUE $ | Reduce con la regla 4 |
| I0 I2 | bterm# | AND TRUE $ | Shift 9 a la pila |
| I0 I2 I9 | AND bterm# | TRUE $ | Shift 6 a la pila |
| I0 I2 I9 I6 | TRUE AND bterm# | $ | Reduce con la regla 7 |
| I0 I2 I9 I13 | bfactor AND bterm# | $ | Reduce con la regla 3 |
| I0 I2 | bterm# | $ | Reduce con la regla 2 |
| I0 I1 | bexpr# | $ | ACCEPT |

**Generado**

```
D:\Universidad Rafael Landívar\TERCER AÑO\3. Segundo Ciclo\Compiladores (LAB)\Lab lex\Laboratorio2>a

NOT (TRUE OR FALSE ) AND TRUE
FALSE(0)
```

Tanto el analizador generado manualmente como el analizador realizado en YACC/FLEX presentan las mismas características y aceptan de forma correcta la gramática generada.

La comparación realizada de nuestra tabla LALR respecto al resultado (log) generado en bison/yacc “grammar.output”, presenta una tabla igual a la generada manualmente, a diferencia que esta última contiene unos cuantos estados extras por si existiese un error.