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.

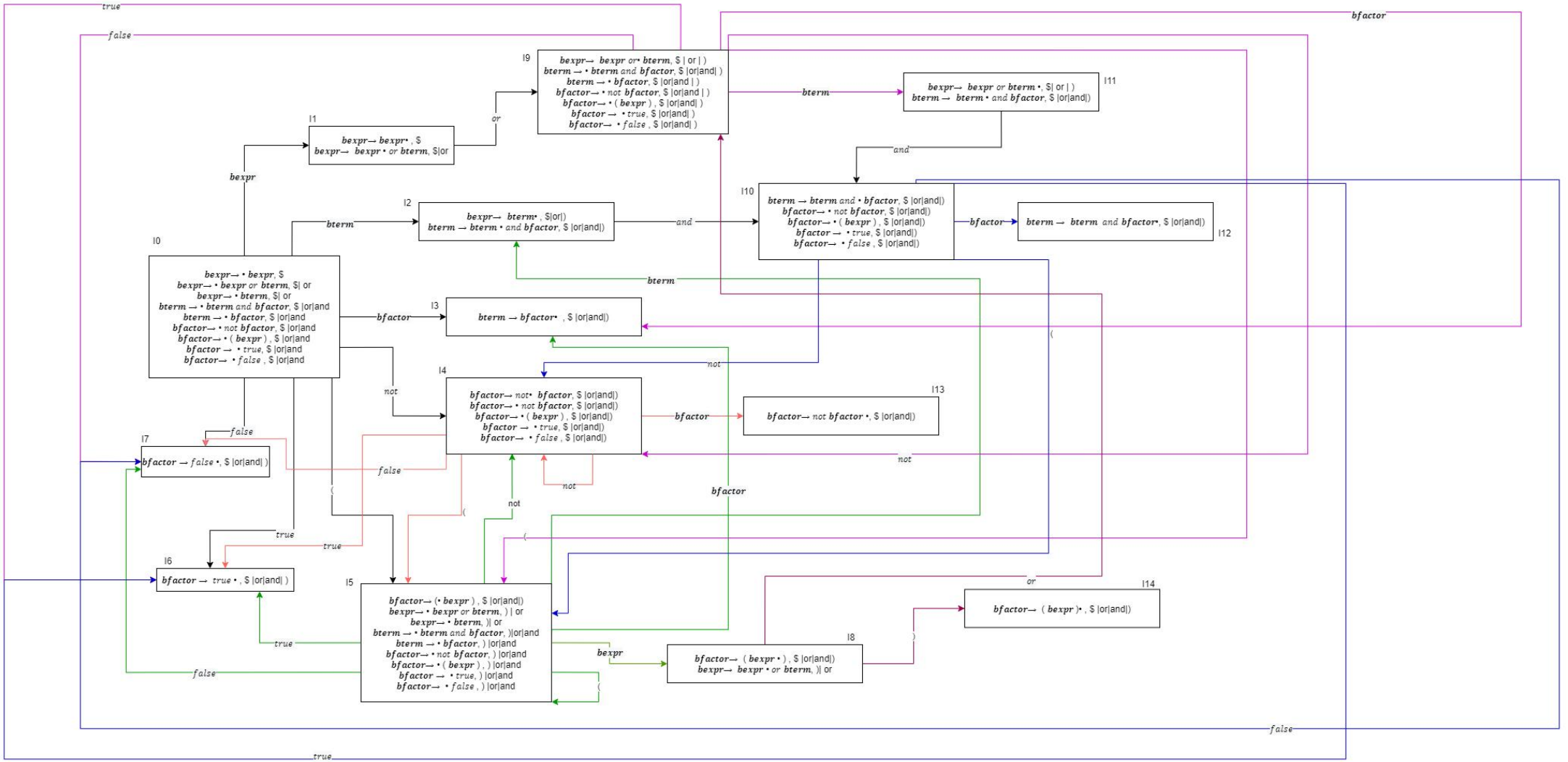
Requerimiento

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

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

	or	**and**	**not**	**(**	**)**	**true**	**false**	**$**	**bexpr**	**bterm**	**bfactor**
0			S4	S5		S6	S7		GOTO 1	GOTO 2	GOTO 3
1	S9							ACCEPT			
2	R2	S10			R2			R2			
3	R4	R4			R4			R4			
4			S4	S5		S6	S7				GOTO 13
5			S4	S5		S6	S7		GOTO 8	GOTO 2	GOTO 3
6	R7	R7			R7			R7			
7	R8	R8			R8			R8			
8	S9				S14						
9			S4	S5		S6	S7			GOTO 11	GOTO 3
10			S4	S5		S6	S7				GOTO 12
11	R1	S10			R1			R1			
12	R3	R3			R3			R3			
13	R5	R5			R5			R5			
14	R6	R6			R6			R6			

1	bexpr -> bexpr or bterm
2	bexpr -> bterm
3	bterm -> bterm and bfactor
4	bterm -> bfactor
5	bfactor -> not bfactor
6	bfactor -> (bexpr)
7	bfactor -> true
8	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
bexpr→ bterm• , $|or|)
bterm → bterm • and bfactor, $ |or|and|)
I3
bterm → bfactor• , $ |or|and|)
I4
bfactor→ not• bfactor, $ |or|and|)
bfactor→ • not bfactor, $ |or|and|)
bfactor→ • (bexpr) , $ |or|and|)
bfactor → • true, $ |or|and|)
bfactor→ • false , $ |or|and|)
I5
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
bfactor → true • , $ |or|and|)
I7
bfactor → false •, $ |or|and|)
I8
bfactor→ (bexpr •) , $ |or|and|)
bexpr→ bexpr • or bterm,)| or
I9
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|)
I10
bterm → bterm and • bfactor, $ |or|and|)
bfactor→ • not bfactor, $ |or|and|)
bfactor→ • (bexpr) , $ |or|and|)
bfactor → • true, $ |or|and|)
bfactor→ • false , $ |or|and|)
I11
bexpr→ bexpr or bterm •, $| or |)
bterm → bterm • and bfactor, $ |or|and|)
I12
bterm → bterm and bfactor•, $ |or|and|)
I13
bfactor→ not bfactor •, $ |or|and|)
I14
bfactor→ (bexpr)• , $ |or|and|)
bexpr
bterm
bfactor
not
and
or
true
false
(
)

- 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 YACC/FLEX.

R/Al momento de colocar bison -v grammar.y se genera un archivo grammar.output, el cual se encarga de crear la tabla del LALR(1). Creando cada uno de los estados y terminales con su shift y reduce y Goto. El programa creado en grammar.y tiene estados para los casos que se lea un error o un salto de linea, por lo tanto, contiene mayor cantidad de estados que el creado manualmente. A continuación se mostrará la tabla creada por el YACC/FLEX

Utilizando la grámatica con su respectiva numeración de reglas:
Grammar

```
 0 $accept: lines $end
 1 lines: lines expr '\n'
 2      | lines '\n'
 3      | lines errorr '\n'
 4      | /* empty */

 5 errorr: ERROR

 6 expr: expr OR term
 7     | term

 8 term: term AND factor
 9     | factor

10 factor: NOT factor
11       | '(' expr ')'
12       | TRUE
13       | FALSE
```

Y la tabla que genera el grammar.output:

	or	and	not	(	)	true	false	$	/n	error	bexpr	bterm	bfactor	error	lines
0								R4							GOTO 1
1			S3	S8		S4	S5	S2	S7	S6	GOTO 10	GOTO 11	GOTO 12	GOTO 9	
2								ACC							
3			S3	S8		S4	S5						GOTO 13		
4								R12							
5								R13							
6								R5							
7								R2							
8			S3	S8		S4	S5				GOTO 14	GOTO 11	GOTO 12		
9									S15						
10	S16								S17						
11		S18						R7							
12								R9							
13								R10							
14	S16				S19										
15								R3							
16			S3	S8		S4	S5					GOTO 20	GOTO 12		
17								R1							
18			S3	S8		S4	S5						GOTO 21		
19								R11							
20		S18						R6							
21								R8							

Y comparándola con la generada manualmente:

	or	and	not	(	)	true	false	$	bexpr	bterm	bfactor
0			S4	S5		S6	S7		GOTO 1	GOTO 2	GOTO 3
1	S9							ACCEPT			
2	R2	S10			R2			R2			
3	R4	R4			R4			R4			
4			S4	S5		S6	S7				GOTO 13
5			S4	S5		S6	S7		GOTO 8	GOTO 2	GOTO 3
6	R7	R7			R7			R7			
7	R8	R8			R8			R8			
8	S9				S14						
9			S4	S5		S6	S7			GOTO 11	GOTO 3
10			S4	S5		S6	S7				GOTO 12
11	R1	S10			R1			R1			
12	R3	R3			R3			R3			
13	R5	R5			R5			R5			
14	R6	R6			R6			R6			

Y aunque los estados tengan distinta numeración podemos observar que son similares. Considerando que se crearon estados para los saltos de línea y los errores. Así mismo tienen la diferencia que los reduce no se encuentran en todos sus follows, pero si realiza reduce en la regla tal como lo hace el analizador manual del LALR(1).

A continuación se presenta la tabla del analizador LALR(1) generado con el YACC/LEX con los estados en común del analizador generado manualmente.

	or	and	not	(	)	true	false	$	/n	error	bexpr	bterm	bfactor	error	lines
0								R4							GOTO 1
1			S3	S8		S4	S5	S2	S7	S6	GOTO 10	GOTO 11	GOTO 12	GOTO 9	
2								ACC							
3			S3	S8		S4	S5						GOTO 13		
4								R12							
5								R13							
6								R5							
7								R2							
8			S3	S8		S4	S5				GOTO 14	GOTO 11	GOTO 12		
9									S15						
10	S16								S17						
11		S18						R7							
12								R9							
13								R10							
14	S16				S19										
15								R3							
16			S3	S8		S4	S5					GOTO 20	GOTO 12		
17								R1							
18			S3	S8		S4	S5						GOTO 21		
19								R11							
20		S18						R6							
21								R8							

Entrada

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

Salida

$$false\ (0)$$