Universidad Rafael Landívar

Facultad de ingeniería

Ingeniería en informática y sistemas

Compiladores

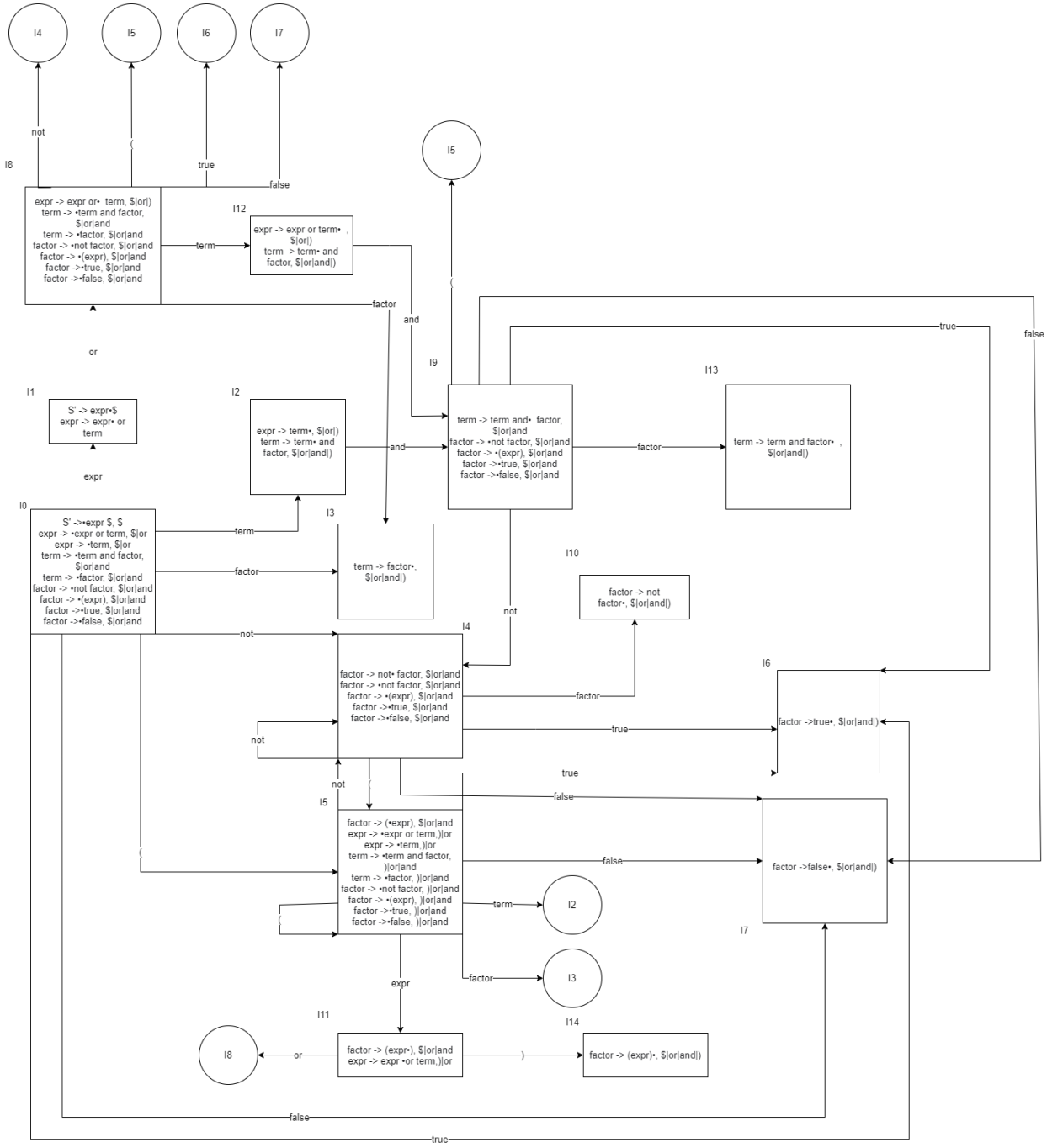
Lab 2: Lex y YACC

Juan Sebastian Mejía Hernández

1172819

Tabla a mano LALR1

(Al momento de reducir y realizar las nuevas conexiones algunas conexiones me fueron imposibles entonces use círculos de diagrama de flujo para evitar líneas inentendibles)



Comparación de tablas

Tabla LALR1

	or	and	not	(	)	TRUE	FALSE	$	expr	term	factor
0			S4	S5		S6	S7	S1	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			

R1	expr -> expr or term
R2	expr -> term
R3	term -> term and factor
R4	term -> factor
R5	factor -> not factor
R6	factor -> (expr)
R7	factor -> true
R8	factor -> false

Tabla YACC

	or	and	not	(	)	TRUE	FALSE	$	\n	expr	term	factor	lines
0													1
1			S3	S7		S4	S5	S2	S6	8	9	10	
2								Accept					
3			S3	S7		S4	S5					11	
4						R10							
5							R11						
6									R2				
7			S3	S7		S4	S5			12	9	10	
8	S13								S14				
9		S15						R5					
10								R7					
11								R8					
12	S13				S16								
13			S3	S7		S4	S5				17	10	
14								R1					
15			S3	S7		S4	S5					18	
16								R9					
17		S15						R4					
18								R6					

Comparación

Las principales dos diferencias es que en el analizador de yacc, se genera un estado adicional llamado líneas y se espera el carácter \n al final de la cadena, eso añade un nivel no de complejidad sino de extensión a la gramática, también tomando en cuenta que lines tiene varias reglas.

Después de eso si observamos el estado 0 del creado con el estado 1 de yacc se puede ver que son los mismos. De ahí podemos sacar varias conclusiones:

-Los estados 6 creado a mano y 4 yacc son la reducción de true.

-Al igual que los estados 7 a mano y 5 yacc para la reducción de false.

-El estado 1 a mano y estado 2 son los estados de aceptación

-Los demás estados que se puede notar un comportamiento exacto igual son: (A mano izquierda y yacc derecha)

- 4 y 3
- 5 y 12
- 8 y 13
- 9 y 15

Son los estados que hacen shift en not,(,true y false

-Entonces se puede ver el analizador generado en yacc es legitimo y funcionara como se espera cuando se hace una corrida a mano del analizador. Los estados son parecidos y las únicas diferencias son diferencias esperadas por realizar el código con una gramática ampliada de manera diferente.

Output Yacc

Después de realizar Bison -v se obtiene:

state 0

0 $accept: . lines $end

$default reduce using rule 3 (lines)

lines go to state 1

state 1

0 $accept: lines . $end

1 lines: lines . expr '\n'

2 | lines . '\n'

$end   shift, and go to state 2

NOT   shift, and go to state 3

TRUE   shift, and go to state 4

FALSE shift, and go to state 5

'\n'   shift, and go to state 6

'('   shift, and go to state 7

expr   go to state 8

term   go to state 9

factor go to state 10

state 2

0 $accept: lines $end .

$default accept

state 3

8 factor: NOT . factor

NOT shift, and go to state 3

TRUE shift, and go to state 4

FALSE shift, and go to state 5

'(' shift, and go to state 7

factor go to state 11

state 4

10 factor: TRUE .

$default reduce using rule 10 (factor)

state 5

11 factor: FALSE .

$default reduce using rule 11 (factor)

state 6

2 lines: lines '\n' .

$default reduce using rule 2 (lines)

state 7

9 factor: '(' . expr ')'

NOT shift, and go to state 3

TRUE shift, and go to state 4

FALSE shift, and go to state 5

'(' shift, and go to state 7

```
    expr    go to state 12
    term    go to state 9
    factor  go to state 10

state 8

    1 lines: lines expr . '\n'
    4 expr: expr . OR term
    OR    shift, and go to state 13
    '\n'  shift, and go to state 14

state 9

    5 expr: term .
    6 term: term . AND factor
    AND  shift, and go to state 15
    $default  reduce using rule 5 (expr)

state 10

    7 term: factor .
    $default  reduce using rule 7 (term)

state 11

    8 factor: NOT factor .
    $default  reduce using rule 8 (factor)

state 12

    4 expr: expr . OR term
    9 factor: '(' expr . ')'
```

```
        OR   shift, and go to state 13
        ')'  shift, and go to state 16

state 13

   4 expr: expr OR . term

   NOT    shift, and go to state 3
   TRUE   shift, and go to state 4
   FALSE  shift, and go to state 5
   '('    shift, and go to state 7

   term    go to state 17
   factor  go to state 10


state 14

   1 lines: lines expr '\n' .

   $default  reduce using rule 1 (lines)


state 15

        6 term: term AND . factor
        NOT    shift, and go to state 3
        TRUE   shift, and go to state 4
        FALSE  shift, and go to state 5
        '('    shift, and go to state 7
```

factor go to state 18

state 16

9 factor: '(' expr ')' .

$default reduce using rule 9 (factor)

state 17

4 expr: expr OR term .

6 term: term . AND factor

AND shift, and go to state 15

$default reduce using rule 4 (expr)

state 18

6 term: term AND factor .

$default reduce using rule 6 (term)