EXAMEN DE COMPILADORES (2° Grado en Informática, final julio-2023)

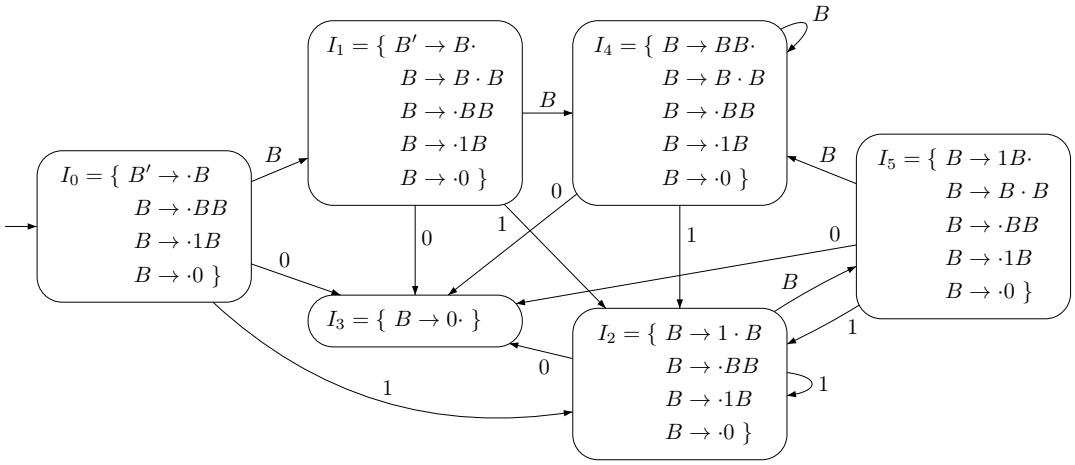
Facultad Informática Universidad Murcia

Apellidos, nombre: **DNI:**

Instrucciones: Este enunciado y todos los folios usados deben entregarse al salir **Grupo:**

Parte I: PREGUNTAS TIPO TEST. 40 %. Cada dos respuestas incorrectas anulan una correcta.

1. A continuación se muestra el autómata de la colección LR(0) de la gramática $B \to BB \mid 1B \mid 0$. Supongamos que se quiere emplear la concatenación de símbolos con asociatividad izquierda. Con entradas válidas, algunas de las transiciones del autómata no se usarían tras resolver los conflictos. Indica la respuesta correcta.



   a) No se usaría la transición de I_0 a I_3.

   b) Desde I_5 no se usarían las transiciones que llegan a I_2 e I_3.

   c) No se usaría la transición que llega a I_2 desde él mismo.

2. Dada la colección LR(0) anterior, considera la entrada 10\$. Marca la respuesta correcta:

   a) Al terminar de reconocer la entrada se encontraría en el estado I_1.

   b) Al terminar de reconocer la entrada se encontraría en el estado I_3.

   c) El autómata no usaría en ningún caso el estado I_5.

3. Supongamos que se tiene una gramática G que es LALR. Indica la respuesta correcta:

   a) La tabla SLR de G puede tener más acciones de desplazamiento que la LALR.

   b) La tabla SLR de G puede tener menos estados que la LALR.

   c) La tabla SLR de G puede tener conflictos reduce/reduce.

4. Supongamos que tenemos una gramática G con las reglas $P \to P \wedge P \mid P \vee P \mid$ `true` $\mid$ `false`, siendo $V_T = \{\wedge, \vee,$ `true`,`false`$\}$ y $V_N = \{P\}$. Supongamos ahora que realizamos transformaciones en la gramática para realizar un análisis LL(1), obteniendo una gramática G' equivalente que es LL(1). Indica la respuesta correcta:

   a) Se puede definir una DDS S-atribuida sobre G' para evaluar el valor lógico de las entradas válidas.

   b) No se puede definir una DDS S-atribuida sobre G', pero sí sobre G.

   c) Es necesaria una DDS L-atribuida sobre G' para evaluar el valor lógico de las entradas válidas.

5. Supongamos que implementamos con Flex un analizador léxico de la gramática anterior, sustituyendo $\wedge$ por `&&` y $\vee$ por `||`, y añadimos el símbolo terminal ; para separar expresiones lógicas:

```
%{
#include <stdio.h>
#include "sintactico.tab.h"
int inicio_error = 0;
void msg_error();
int contador_errores = 0;
%}
espacio             [ \n\r\t]
%x error_lex
%%
```

```
{espacio}+                {  }
"&&"                      { return AND; }
"||"                      { return OR;  }
";"                       { return PYC; }
"true"                    { yylval.ent = 1; return VAL; }
"false"                   { yylval.ent = 0; return VAL; }
.                         { inicio_error = yylineno; yymore(); BEGIN(error_lex); }
<error_lex>{espacio}|";" { yyless(yyleng-1); msg_error(); BEGIN(0); }
<error_lex>.              { yymore(); }
<error_lex><<EOF>>        { msg_error(); return 0; }
%%
void msg_error() {
    printf("Error lexico en linea %d: %s\n", inicio_error, yytext);
    contador_errores++;
}
```

donde `yyless(n)` devuelve a la entrada los caracteres de `yytext` desde n en adelante. Indica la respuesta verdadera suponiendo que la entrada a este analizador es `true && (false true || true);`

a) El contador de errores termina tomando el valor 1.

b) El contador de errores termina tomando el valor 2.

c) El analizador léxico devuelve siete tokens.

6. Supongamos la siguiente implementación de un analizador sintáctico con Bison para la gramática de expresiones lógicas, que se combina con el analizador léxico anterior:

```
%{
#include <stdio.h>
extern int yylineno;
extern int yylex();
extern int yyparse();
extern int contador_errores;
void yyerror(const char *msg);
%}
%union {
    int ent;
}
%token AND OR PYC
%token <ent> VAL
%type <ent> predicado
%left OR
%left AND
%%
entrada   : predicado PYC           { if (contador_errores == 0)
                                          printf("Valor: %d\n", $1);
                                    }
          ;
predicado : predicado AND predicado { $$ = $1 && $3; }
          | predicado OR predicado  { $$ = $1 || $3; }
          | VAL                     { $$ = $1; }
          ;
%%
void yyerror(const char *msg) {
    printf("Error sintactico en linea %d: %s\n", yylineno, msg);
}
int main(int argc, char **argv) {
    yyparse();
}
```

Supongamos la entrada `true && false || true;` . Indica la respuesta verdadera:

a) Primero se evaluaría el operador `&&` y luego el operador `||`.

b) Primero se evaluaría el operador `||` y luego el operador `&&`.

c) Habría un conflicto desplaza/reduce sin resolver.

7. Si se construye un analizador LALR de la siguiente gramática para representar operaciones de conjuntos:

$$\begin{array}{ll} C & \rightarrow C \cup C \mid C \cap C \mid (C) \mid \{L\} \\ L & \rightarrow L\,,L \mid id \mid \lambda \end{array}$$

se obtiene una tabla que contiene, entre otras, las siguientes filas:

Estado	Acción								Ir-a	
	(	)	,	id	∩	∪	{	}	$	C	L
9		r1			r1/d5	r1/d4			r1		
10		r2			r2/d5	r2/d4			r2		
14			r5/d13					r5			

siendo los conjuntos de ítems de los estados indicados los siguientes:

$I_9 = \{ [C \to C \cup C \bullet , \$/\cap/\cup] , [C \to C \bullet \cup C , \$/\cap/\cup] , [C \to C \bullet \cap C , \$/\cap/\cup] \}$

$I_{10} = \{ [C \to C \cap C \bullet , \$/\cap/\cup] , [C \to C \bullet \cap C , \$/\cap/\cup] , [C \to C \bullet \cup C , \$/\cap/\cup] \}$

$I_{14} = \{ [L \to L,L \bullet , \}/,] , [L \to L \bullet , L , \}/,] \}$

Considera que siempre se utiliza asociatividad izquierda, y que $\cap$ tiene más precedencia que $\cup$. Se emplea $M[s, c]$ para indicar la casilla de la tabla en la fila del estado s y la columna del símbolo c. Indica la forma de resolver los conflictos:

a) $M[9,\cap] = r1$; $M[9,\cup] = r1$; $M[10,\cup] = d5$; $M[10,\cap] = r2$; $M[14,,] = r5$

b) $M[9,\cap] = r1$; $M[9,\cup] = r1$; $M[10,\cup] = r2$; $M[10,\cap] = r2$; $M[14,,] = r5$

c) $M[9,\cap] = d5$; $M[9,\cup] = r1$; $M[10,\cup] = r2$; $M[10,\cap] = r2$; $M[14,,] = r5$

8. Dada la siguiente gramática G, con $V_T = \{a, b, x, y\}$ y $V_N = \{S, A, B, C\}$, y P:

$$\begin{array}{lcl} S & \to & a\ A\ B\ C \\ A & \to & x \mid \lambda \\ B & \to & b \\ C & \to & y \mid \lambda \end{array}$$

respecto a la simulación descendente predictiva, señalar la afirmación <u>incorrecta</u>:

a) Si en la pila de análisis aparece $\$\texttt{CBA}$ y en la entrada `b`$, en el siguiente paso de cálculo, la pila contendría $\$\texttt{CB}$ y la entrada `b`$.

b) Si en la pila de análisis aparece $\$\texttt{CBA}$ y en la entrada `yb`$, en el siguiente paso de cálculo, la pila contendría $\$\texttt{CB}$ y la entrada `b`$.

c) Si en la pila de análisis aparece $\$\texttt{CBA}$ y en la entrada `yb`$, en el siguiente paso de cálculo, la pila contendría $\$\texttt{CBA}$ y la entrada `b`$.

9. Dada la siguiente gramática G y su correspondiente tabla LALR:

$$\begin{array}{lcl} (1)\ S & \to & s\ (\ A\ ) \\ (2)\ A & \to & A\ ,\ B \\ (3) & \mid & B \\ (4)\ B & \to & b \end{array}$$

Estado	Acción						Ir-a		
	,	(	)	b	s	$	*A*	*B*	*S*
0					d2				1
1						acepta			
2		d3							
3				d6			4	5	
4	d8		d7						
5	r3		r3						
6	r4		r4						
7						r1			
8				d6				9	
9	r2		r2						

a) G es LL, LALR y LR-Canónica.

b) G es SLR, LALR, LR-Canónica.

c) G es LALR y LR-Canónica, pero no SLR.

10. Dada la gramática G de la pregunta anterior, y con respecto a la simulación LR:

a) Si el contenido de la pila de un analizador LR es `0s2(3A4,8B9` y el de la entrada `,b)`$, la siguiente configuración tendrá en la pila `0s2(3A4` y en la entrada `b)`$.

b) Si el contenido de la pila de un analizador LR es `0s2(3A4,8B9` y el de la entrada `(b)`$, la siguiente configuración tendrá en la pila `0s2(3A4,8B9` y en la entrada `)`$.

c) Si el contenido de la pila de un analizador LR es `0s2(3A4,8b6` y el de la entrada `,b)`$, la siguiente configuración tendrá en la pila `0s2(3A4,8B9` y en la entrada `b)`$.

Parte II: EJERCICIOS. 60 %.
La siguiente gramática G_1, con $V_T = \{i, (,), =\}$, $V_N = \{S, A\}$ y P:

$$\begin{array}{lll} (1) & S & \to i = A \\ (2) & A & \to A = A \\ (3) & A & \to i \\ (4) & A & \to (A) \end{array}$$

sirve para representar sentencias de asignación en donde el operador = es asociativo por la derecha.

1. *(0,5 p.)* Indica si G_1 tiene alguna propiedad (o propiedades) que, de partida, le impidan ser LL(1). En caso de ser así, especificar cuáles.
2. *(1 p.)* Teniendo en cuenta la asociatividad del operador =, realiza transformaciones para obtener una gramática G_1' equivalente a G_1 y que se pueda analizar con el método LL(1).
3. *(0,75 p.)* Calcula los conjuntos PRIMERO, SIGUIENTE y PREDICT de los no terminales de G_1'.
4. *(0,75 p.)* Crea la tabla de análisis LL de G_1' y razona si es LL(1).

Dada la siguiente gramática G_2, con $V_T = \{a, b\}$, $V_N = \{S, A, B, C\}$ y P:

$$\begin{array}{lll} (1) & S & \to A\ B \\ (2) & S & \to B \\ (3) & S & \to C\ a \\ (4) & A & \to b\ B\ a \\ (5) & B & \to A \\ (6) & B & \to b \\ (7) & C & \to A \end{array}$$

5. *(1 p.)* Completa la colección LR(1) para la gramática G_2 calculando los conjuntos I_0, I_5 e I_8 según los GOTO que se indican. Calcula también los GOTO que faltan, es decir, que conducen a estados previamente calculados.

$I_0 = \{\ \}$

$I_1 = \text{GOTO}(I_0, S) = \{\ [\ S' \to\ S\cdot\ ,\ \$\ ]\ \}$

$I_2 = \text{GOTO}(I_0, A) = \{\ [\ S \to\ A \cdot B\ ,\ \$\ ]$
$[\ B \to\ A\cdot\ ,\ \$\ ]$
$[\ C \to\ A\cdot\ ,\ a\ ]$
$[\ B \to\ \cdot A\ ,\ \$\ ]$
$[\ B \to\ \cdot b\ ,\ \$\ ]$
$[\ A \to\ \cdot bBa\ ,\ \$\ ]\ \}$

$I_3 = \text{GOTO}(I_0, B) = \{\ [\ S \to\ B\cdot\ ,\ \$\ ]\ \}$

$I_4 = \text{GOTO}(I_0, C) = \{\ [\ S \to\ C \cdot a\ ,\ \$\ ]\ \}$

$I_5 = \text{GOTO}(I_0, b) = \{\ \}$

$I_6 = \text{GOTO}(I_2, B) = \{\ [\ S \to\ AB\cdot\ ,\ \$\ ]\ \}$

$I_7 = \text{GOTO}(I_2, A) = \{\ [\ B \to\ A\cdot\ ,\ \$\ ]\ \}$

$I_8 = \text{GOTO}(I_2, b) = \{\ \}$

$I_9 = \text{GOTO}(I_4, a) = \{\ [\ S \to\ Ca\cdot\ ,\ \$\ ]\ \}$

$I_{10} = \text{GOTO}(I_5, B) = \{\ [\ A \to\ bB \cdot a\ ,\ b/\$/a\ ]\ \}$

$I_{11} = \text{GOTO}(I_5, A) = \{\ [\ B \to\ A\cdot\ ,\ a\ ]\ \}$

$I_{12} = \text{GOTO}(I_5, b) = \{\ [\ A \to\ b \cdot Ba\ ,\ a\ ]$
$[\ B \to\ b\cdot\ ,\ a\ ]$
$[\ B \to\ \cdot A\ ,\ a\ ]$
$[\ B \to\ \cdot b\ ,\ a\ ]$
$[\ A \to\ \cdot bBa\ ,\ a\ ]\ \}$

$I_{13} = \text{GOTO}(I_8, B) = \{\ [\ A \to\ bB \cdot a\ ,\ \$\ ]\ \}$

$I_{14} = \text{GOTO}(I_{10}, a) = \{\ [\ A \to\ bBa\cdot\ ,\ b/\$/a\ ]\ \}$

$I_{15} = \text{GOTO}(I_{12}, B) = \{\ [\ A \to\ bB \cdot a\ ,\ a\ ]\ \}$

$I_{16} = \text{GOTO}(I_{13}, a) = \{\ [\ A \to\ bBa\cdot\ ,\ \$\ ]\ \}$

$I_{17} = \text{GOTO}(I_{15}, a) = \{\ [\ A \to\ bBa\cdot\ ,\ a\ ]\ \}$

6. *(0,75 p.)* Calcula las filas correspondientes a los estados 2, 5 y 14 de la tabla LR-Canónica.

ESTADO	ACCIÓN			IR-A			
	a	b	\$	S	A	B	C
2							
5							
14							

7. *(0,75 p.)* Indica qué conjuntos se agrupan, y su contenido, para obtener la colección LALR.
8. *(0,5 p.)* Justifica si la gramática es LR-Canónica, LALR y/o SLR.