Seminario Compiladores

Repaso de la asignatura

Método LL(1)

Sea G la gramática con $V_N = \{D, C, P\}$ y $V_T = \{\texttt{if}, \texttt{cond}, \texttt{id}, \texttt{prim}, \texttt{(}, \texttt{)}, \texttt{;}\}$, siendo P:

$$
\begin{array}{lll}
(1)\ D & \rightarrow C\ \texttt{if cond} \\
(2)\ C & \rightarrow \texttt{id (}\ P\ \texttt{)} \\
(3) & \mid \texttt{prim} \\
(4)\ P & \rightarrow P\ \texttt{;}\ C \\
(5) & \mid C
\end{array}
$$

2. Señala la respuesta correcta:

   a) G es LL(1) porque para todas las reglas $A \rightarrow \alpha$ y $A \rightarrow \beta$ se cumple que $predict(A \rightarrow \alpha) \cap predict(A \rightarrow \beta) = \emptyset$.

   b) G no es LL(1) porque $predict(1) \cap predict(2) = \{\texttt{id}\}$.

   c) G no es LL(1) porque $predict(4) \cap predict(5) = \{\texttt{id}, \texttt{prim}\}$.

(1) $D \rightarrow C$ `if cond`
(2) $C \rightarrow$ `id (` P `)`
(3) $\quad \mid$ `prim`
(4) $P \rightarrow P$ `;` C
(5) $\quad \mid C$

1. Indicar cuál de estas gramáticas es equivalente a la gramática G de la parte de ejercicios y además LL(1). Esta pregunta se refiere a la **gramática del enunciado**.

a)

$D \rightarrow C$ `if cond`
$C \rightarrow$ `id (` P `)`
$\mid$ `prim`
$P \rightarrow C$ `;` P
$\mid \lambda$

b)

$D \rightarrow C$ `if cond`
$C \rightarrow$ `id (` P `)`
$\mid$ `prim`
$P \rightarrow C\ P''$
$P'' \rightarrow \lambda$
$\mid P'$
$P' \rightarrow$ `;` $C\ P''$

c)

$D \rightarrow C$ `if cond`
$C \rightarrow$ `id (` P `)`
$\mid$ `prim`
$P \rightarrow C$
$\mid C\ P'$
$P' \rightarrow$ `;` C
$\mid$ `;` $C\ P'$

Razonar si esta gramática puede ser LL(1).

Método LR(1)

(1) $D \rightarrow C$ `if cond`
(2) $C \rightarrow$ `id ( P )`
(3) $\quad\quad |$ `prim`
(4) $P \rightarrow P$ `;` C
(5) $\quad\quad | C$

2. Completar la colección colección LR(1) siguiente calculando los conjuntos I_0, I_1 e I_2 según las transiciones que se indican:

$I_0 = \{\ \}$

$I_1 = \text{GOTO}(I_0, \texttt{id}) = \{\ \}$

$I_2 = \text{GOTO}(I_1, () = \{\ \}$

$I_3 = \text{GOTO}(I_2, P) = \{\ [\ C \rightarrow \texttt{id (} P \cdot \texttt{) , if }\ ]$
$[\ P \rightarrow P \cdot \texttt{ ; } C \texttt{ ,)/; }\ ]\}$

$I_4 = \text{GOTO}(I_3,)) = \{\ [\ C \rightarrow \texttt{id (} P \texttt{) } \cdot \texttt{ , if }\ ]\}$

$I_5 = \text{GOTO}(I_0, D) = \{\ [\ D' \rightarrow\ D \cdot \texttt{,\$}\ ]\ \}$

$I_6 = \text{GOTO}(I_0, C) = \{\ [\ D \rightarrow C \cdot \texttt{ if cond, \$}\ ]\}$

$I_7 = \text{GOTO}(I_0, \texttt{prim}) = \{\ [\ C \rightarrow \texttt{prim} \cdot \texttt{ , if }\ ]\}$

$I_8 = \text{GOTO}(I_6, \texttt{if}) = \{\ [\ D \rightarrow C \texttt{ if} \cdot \texttt{ cond , \$}\ ]\}$

$I_9 = \text{GOTO}(I_8, \texttt{cond}) = \{\ [\ D \rightarrow C \texttt{ if cond} \cdot \texttt{ , \$}\ ]\}$

$I_{10} = \text{GOTO}(I_2, C) = \{\ [\ P \rightarrow C \cdot \texttt{ ,)/; }\ ]\}$

$I_{11} = \text{GOTO}(I_2, \texttt{id}) = \{\ [\ C \rightarrow \texttt{id} \cdot \texttt{ (} P \texttt{) ,)/; }\ ]\}$

$I_{12} = \text{GOTO}(I_2, \texttt{prim}) = \{\ [\ C \rightarrow \texttt{prim} \cdot \texttt{ ,)/; }\ ]\}$

$I_{13} = \text{GOTO}(I_3, ;) = \{\ [\ P \rightarrow P \texttt{ ; } \cdot\ C \texttt{ ,)/; }\ ]$
$[\ C \rightarrow \cdot \texttt{ id (} P \texttt{) ,)/; }\ ]$
$[\ C \rightarrow \cdot \texttt{ prim ,)/; }\ ]\}$

$I_{14} = \text{GOTO}(I_{11}, () = \{\ [\ C \rightarrow \texttt{id (} \cdot\ P \texttt{) ,)/; }\ ]$
$[\ P \rightarrow \cdot\ P \texttt{ ; } C\texttt{,)/; }\ ]$
$[\ P \rightarrow \cdot\ C\texttt{,)/; }\ ]$
$[\ C \rightarrow \cdot \texttt{ id (} P \texttt{),)/; }\ ]$
$[\ C \rightarrow \cdot \texttt{ prim,)/; }\ ]\}$

$I_{15} = \text{GOTO}(I_{13}, C) = \{\ [\ P \rightarrow P \texttt{ ; } C \cdot \texttt{ ,)/; }\ ]\}$

$I_{16} = \text{GOTO}(I_{14}, P) = \{\ [\ C \rightarrow \texttt{id (} P \cdot \texttt{) ,)/; }\ ]$
$[\ P \rightarrow P \cdot \texttt{ ; } C\texttt{,)/; }\ ]\}$

$I_{17} = \text{GOTO}(I_{16},)) = \{\ [\ C \rightarrow \texttt{id (} P \texttt{) } \cdot \texttt{ ,)/; }\ ]\}$

3. Pensar qué conjuntos podrían unirse para formar el autómata LALR correspondiente a G.

(1) $D \rightarrow C$ `if cond`
(2) $C \rightarrow$ `id (` P `)`
(3) $\quad\quad |$ `prim`
(4) $P \rightarrow P$ `;` C
(5) $\quad\quad | C$

4. Indica la opción correcta:

   a) En este caso la colección LALR coincide con la LR-Canónica, puesto que no se pueden fusionar estados.

   b) La colección LALR se obtiene a partir de la colección LR(1) uniendo los estados 1-11, 2-14, 3-16, 4-17 y 7-12.

   c) Ninguna de las anteriores opciones es correcta.

5. Indica la opción correcta:

   a) Si G es LALR entonces también es SLR, puesto que en este caso la tabla SLR sería idéntica a la LALR.

   b) El estado I_3 indica un conflicto despaza/reduce y, por tanto, la gramática no puede ser LR-Canónica. Al no ser LR-Canónica, tampoco puede ser SLR ni LALR.

   c) La casilla $T[17, \texttt{if}]$ de la tabla LR-canónica y la casilla $T[4-17, \texttt{if}]$ de la tabla LALR contendrán ambas la acción $r2$, puesto que $\texttt{if} \in SIGUIENTE(C)$.

Lenguajes de programación y gramáticas

18. Podemos asegurar que un lenguaje de programación en el que se requiera la declaración de variables previa a su uso es:

    a) *libre de contexto.*

    b) *sensible al contexto.*

    c) de ambos tipos.

Lenguajes de programación y autómatas

1. La **sintaxis de los lenguajes de programación**:

   a) Se analiza con *autómatas de pila*, puesto que suelen ser generados por *gramáticas libres de contexto.*

   b) Se analiza con *autómatas linealmente acotados*, puesto que suelen ser generados por *gramáticas sensibles al contexto.*

   c) Se analiza con *autómatas de pila*, aunque suelen ser *sensibles al contexto.*

La tabla de símbolos

7. La **tabla de símbolos**:

   a) Almacena información útil acerca de los identificadores de un lenguaje de programación, siendo útil exclusivamente en la etapa de análisis del proceso de compilación.

   b) Es una estructura de datos que se usa a lo largo de todo el proceso de compilación y que sirve, entre otras cosas, para poder controlar las restricciones contextuales de un lenguaje de programación.

   c) Es una estructura de datos donde se almacena el código intermedio generado por el compilador.

Máquina abstracta

19. Una *máquina abstracta*:

   a) traduce el código intermedio a código máquina, que posteriormente será ejecutado.

   b) es un intérprete para un lenguaje de alto nivel.

   c) puede considerarse como la implementación software de una máquina.

Análisis léxico

3. Elige la frase correcta acerca del **análisis de léxico**:

   a) Suele ser una función a la que llama el *analizador sintáctico* cada vez que necesita un token, aunque podría no realizarse de forma explícita y dejar el reconocimiento de palabras como parte del análisis sintáctico.

   b) Suele generar un fichero explícito de tokens que constituye la entrada del *analizador sintáctico*.

   c) Se realiza mediante la simulación de *autómatas de pila*, que proporcionan la potencia suficiente para las tareas de E/S.

Propiedades de las gramáticas

42. La siguiente gramática:

$$
\begin{array}{lcl}
S & \rightarrow & S\ A \\
 & | & A \\
A & \rightarrow & id\ =\ L\ ; \\
L & \rightarrow & id \\
 & | & L\ =\ L
\end{array}
$$

   a) Es propia.
   b) No es propia, pues es recursiva por la izquierda.
   c) No es propia, pues es ambigua.

43. La gramática anterior:

   a) Es LL(1) y SLR(1).
   b) No es LL(1) aunque sí LR(1).
   c) No es ni LL(1) ni SLR(1).

Colección LR(0)

38. Considera la siguiente gramática:

$S \to A\ (\ S\ )\ B \mid \lambda$
$A \to S \mid S\ B \mid x \mid \lambda$
$B \to S\ B \mid y$

¿Cuál es el conjunto de ítems I_0 de la colección LR(0) de la gramática de la pregunta anterior?

a) $\{\ [\ S' \to \cdot S\ ]\ ,\ [\ S \to \cdot A\ (\ S\ )\ B\ ]\ ,\ [\ S \to \cdot\ ]\ ,\ [\ A \to \cdot S\ ]\ ,\ [\ A \to\ \cdot S\ B\ ]\ ,\ [\ A \to \cdot x\ ]\ ,\ [\ A \to \cdot\ ]\ ,$
$[\ B \to\ \cdot S\ B\ ]\ ,\ [\ B \to \cdot\ y\ ]\ \}$

b) $\{\ [\ S' \to \cdot S\ ]\ ,\ [\ S \to \cdot A\ (\ S\ )\ B\ ]\ ,\ [\ S \to \cdot\ ]\ ,\ [\ A \to \cdot S\ ]\ ,\ [\ A \to\ \cdot S\ B\ ]\ ,\ \ [\ A \to \cdot x\ ]\ ,\ [\ A \to \cdot\ ]\ \}$

c) $\{\ [\ S' \to \cdot S\ ]\ ,\ [\ S \to \cdot A\ (\ S\ )\ B\ ]\ ,\ [\ A \to \cdot S\ ]\ ,\ [\ A \to\ \cdot S\ B\ ]\ ,\ [\ A \to \cdot x\ ]\ \}$

39. ¿Qué tipo de conflictos se producen en el conjunto I_0 de la pregunta anterior?

a) Reduce-reduce.

b) Desplaza-reduce.

c) Desplaza-reduce y reduce-reduce.

Prefijo viable e ítems válidos

6. Dada la siguiente gramática:

$$
\begin{array}{lcl}
S & \rightarrow & S\ A \\
 & | & A \\
A & \rightarrow & id\ =\ L\ ; \\
L & \rightarrow & id \\
 & | & L\ =\ L
\end{array}
$$

elegir, de entre los siguientes, el conjunto de *items válidos* para el *prefijo viable* $id =$

a) $\{[A \rightarrow id\ = \cdot L\ ;], [L \rightarrow \cdot id], [L \rightarrow \cdot L\ =\ L]\}$

b) $\{[S \rightarrow A \cdot]\}$

c) $\{[A \rightarrow id\ =\ L \cdot ;], [L \rightarrow L \cdot =\ L]\}$

Conflictos LALR si la gramática es LR-canónica

3. Si a partir del automáta de una gramática LR-Canónica construimos el automáta de la gramática LALR, al construir la tabla...

   a) ...pueden aparecer conflictos shift/reduce y reduce/reduce.

   b) ...sólo pueden aparecer conflictos reduce/reduce.

   c) ...sólo pueden aparecer conflictos shift/reduce.

Ambigüedad

4. Una *gramática ambigua*

   a) no puede ser LL ni LR.

   b) puede ser LL pero no LR.

   c) puede ser LR pero no LL.

Recursión por la izquierda

8. Una **gramática recursiva por la izquierda:**

   a) No puede ser *LL*, ni *LR*.

   b) No puede ser *LL*, aunque sí *SLR*.

   c) No puede ser *SLR*, aunque sí *LR-Canónica*.

Lambda reglas

9. Las **gramáticas LL** y **LR**:

   a) Pueden tener λ-*reglas* y ser *ambiguas*.

   b) Pueden tener λ-*reglas* pero no pueden ser *ambiguas*.

   c) No pueden tener λ-*reglas* ni ser *ambiguas*.

Gramática if / if-else

13. Dada la siguiente gramática:

$$
\begin{array}{lcl}
\textit{sent} & \rightarrow & \textbf{if } \textit{expr} \textbf{ then } \textit{sent} \\
 & | & \textbf{if } \textit{expr} \textbf{ then } \textit{sent} \textbf{ else } \textit{sent} \\
 & | & S \\
\textit{expr} & \rightarrow & E
\end{array}
$$

a) es posible encontrar una gramática equivalente no ambigua y LR.

b) es posible encontrar una gramática equivalente no ambigua y LL.

c) no es posible encontrar una gramática equivalente no ambigua.

Relación entre el tamaño de las tablas

25. Una tabla LALR:

   a) Tiene un tamaño intermedio entre una SLR y una LR-Canónica para la misma gramática.

   b) Tiene necesariamente que tener un tamaño menor que una LR-Canónica para la misma gramática.

   c) Tiene necesariamente el mismo tamaño que una tabla SLR para la misma gramática.

Reducciones

37. Sea G la gramática con las producciones:

$S \rightarrow 0\ S\ 1 \mid 0\ 1$

Un analizador ascendente predictivo realizaría la siguiente secuencia de reducciones para reconocer la sentencia '000111':

a) r2 r1 r1

b) r1 r2 r1

c) r2 r2 r1

Conflictos (1)

11. Supongamos que se realiza un análisis SLR de la gramática siguiente

$$L \rightarrow L \odot L \mid L \oslash L \mid \Delta$$

y uno de los conjuntos de items es el siguiente:

$I_j = \{ [L \rightarrow L \odot L \bullet], [L \rightarrow L \bullet \odot L], [L \rightarrow L \bullet \oslash L] \}$

de manera que, en la tabla de análisis, la fila correspondiente al estado j quedaría así:

Estado	Acción				Ir-a
	$\odot$	$\oslash$	Δ	\$	L
		...			
j	r1/di	r1/dk		r1	
		...			

Si $\oslash$ y $\odot$ son asociativos por la izquierda, y $\oslash$ tiene menor precedencia que $\odot$, para eliminar los conflictos:

a) Debemos elegir las reducciones en ambos los casos.

b) Debemos elegir el desplazamiento en las casilla $[j, \odot]$ y la reducción en $[j, \oslash]$.

c) Debemos elegir la reducción en la casilla $[j, \odot]$ y desplazamiento en $[j, \oslash]$.

Conflictos (2)

$$
\begin{array}{lcl}
S & \rightarrow & S\ A \\
 & | & A \\
A & \rightarrow & id\ =\ L\ ; \\
L & \rightarrow & id \\
 & | & L\ =\ L
\end{array}
$$

44. Si en la colección LR(0) de la gramática anterior obtenemos el estado

$I_{10} \equiv \{\ L \rightarrow L = L\bullet\ ,\ L \rightarrow L\bullet = L\}$

y la siguiente tabla:

ESTADO	**accion**				**ir_a**		
	id	=	;	$	**S**	**A**	**L**
0	*d*3				1	2	
1	*d*3			*aceptar*		4	
2	*r*2			*r*2			
3		*d*5					
4	*r*1			*r*1			
5	*d*7						6
6		*d*9	*d*8				
7		*r*4	*r*4				
8	*r*3			*r*3			
9	*d*7						10
10		*d*9/*r*5	*r*5				

Para conseguir que el operador = sea asociativo por la derecha, en la casilla [10,=] debemos elegir la acción:

a) r5

b) d9

c) Ninguna valdría.

Conflictos (3)

4. Considerar la siguiente gramática G que genera *expresiones regulares*:

$$
\begin{array}{rcl}
R & \rightarrow & R \; '|' \; R \\
 & | & R \; R \\
 & | & R \; '*' \\
 & | & a \\
 & | & b
\end{array}
$$

Suponemos que, una vez calculada la colección LR(1) correspondiente a dicha gramática, el conjunto I_5 es el siguiente:

$$
\begin{array}{rl}
I_5 = \{ & [R \rightarrow R \cdot |R, \quad a/b/|/*/\$] \\
 & [R \rightarrow RR\cdot, \quad a/b/|/*/\$] \\
 & [R \rightarrow R \cdot R, \quad a/b/|/*/\$] \\
 & [R \rightarrow R \cdot *, \quad a/b/|/*/\$] \\
 & [R \rightarrow \cdot R|R, \quad a/b/|/*/\$] \\
 & [R \rightarrow \cdot RR, \quad a/b/|/*/\$] \\
 & [R \rightarrow \cdot R*, \quad a/b/|/*/\$] \\
 & [R \rightarrow \cdot a, \quad a/b/|/*/\$] \\
 & [R \rightarrow \cdot b, \quad a/b/|/*/\$] \; \}
\end{array}
$$

Y la tabla LR-Canónica correspondiente al estado 5 es la siguiente:

Estado	Acción					Ir-a
	$*$	a	b	|	\$	R
0						
1						
2						
3						
4						
5	r2/d6	r2/d2	r2/d3	r2/d4	r2	5
6						
7						

Resolver los conflictos existentes en las casillas de la tabla $[5, *]$, $[5, a]$ y $[5, |]$, eliminando las acciones correspondientes a las entradas múltiples con el objeto de dotar a cada operador de la precedencia y asociatividad usual en expresiones regulares.

Flex

Dado el siguiente fragmento de un fichero *flex*:

```
letra [a-zA-Z]
digito[0-9]
%%
[ \n\t] ;
"=" return EQ;
"+" return MAS;
"-" return MENOS;
"*" return POR;
"/" return DIV;
";"  return PUNTOYCOMA;
({letra}|"_")({letra}|{digito}|"_")*   { return ID;}
[^a-zA-Z0-9;*+=/-]*     print ("Error carácter no reconocido %s", yytext);
```

indica cual sería la primera salida por pantalla del analizador léxico teniendo en cuenta la siguiente cadena

```
?__id10 = var_1 + c;
```

a) `Error carácter no reconocido ?_`

b) `Error carácter no reconocido ?__`

c) `Error carácter no reconocido ?`

Bison

Dado el siguiente fragmento de un fichero con formato Bison:

```
%{
int cs=0;
%}
%token  INICIO FIN CAB COL
%%
inicial : { cs++; } INICIO cabecera cola FIN {if (cs) printf("%d\n",$3);else printf("%d\n",$4);};
cabecera : CAB    {$$=2017;cs--;}
           |      {$$=2018;cs--;};
cola : COL         {$$=2019;cs++;}
           |       {$$=2020;cs++;};
```

La salida del programa en C generado a partir de él, ante la entrada `INICIO FIN` será:

a) 2017 o 2019

b) 2018

c) 2020

Flex y Bison

Supongamos una especificación de **bison** con la siguiente declaración de tipos:

```
%union {
        char * cadena;
        double numero;
}
%token <cadena> ID
%token <numero> FLT
```

¿qué instrucciones serían correctas en la acción asociada a FLT en un analizador léxico implementado con Flex?

a) `{yylval.numero = atoi(yytext); return FLT; }`

b) `{yylval.cadena = strdup(yytext); return FLT; }`

c) `{yylval.numero = atof(yytext); return FLT; }`

LL y LR (1)

16. Si una gramática contiene (entre otras) las siguientes reglas:

$$\begin{aligned} A &\rightarrow a\ x \\ &\mid \lambda \\ B &\rightarrow A\ a\ y \end{aligned}$$

a) puede ser LL(1).

b) puede ser SLR(1).

c) no puede ser LL(1).

LL y LR (2)

34. Considérese la siguiente gramática:

$S \rightarrow X\ S\ a \mid b$
$X \rightarrow \lambda \mid z$

¿Cuál es la respuesta correcta?

a) La gramática es recursiva por la izquierda, y por tanto no puede ser LL(1).

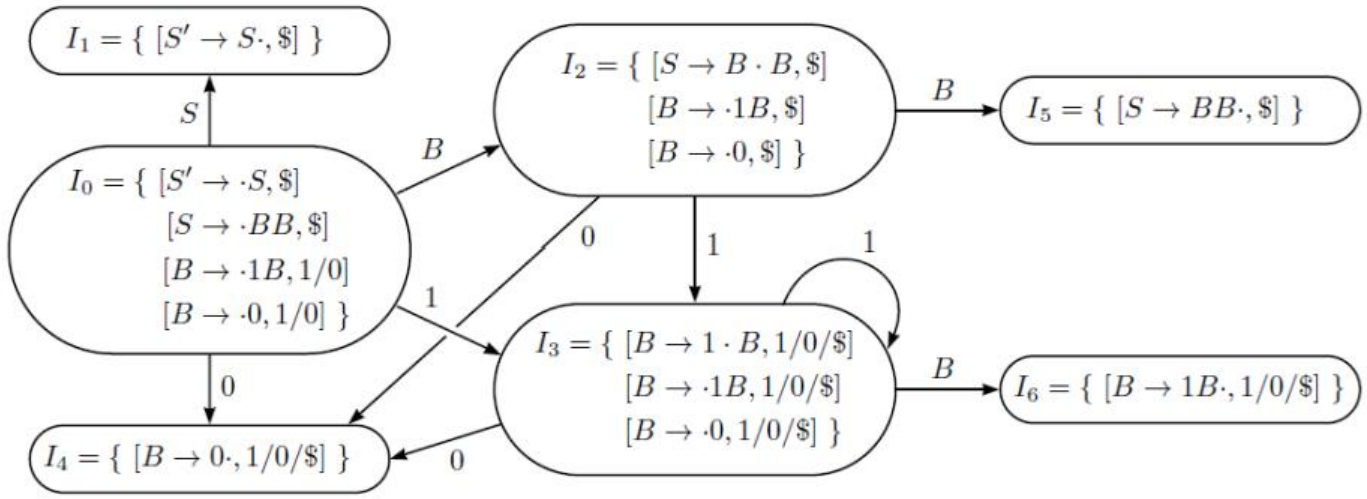
b) La gramática es SLR y LALR.

c) La gramática es LR(1).

Razonando con colección LALR

26. Supongamos que construimos un analizador LALR de la siguiente gramática:
$S \to BB$
$B \to 1B$
$B \to 0$

y obtenemos los siguientes conjuntos de ítems:



Indicar la respuesta correcta:

a) Sólo se puede afirmar que la gramática es LALR.

b) Sólo se puede afirmar que la gramática es LR(1).

c) Se puede afirmar que la gramática es SLR y LR(1).

27. Continuando con el analizador LALR del ejemplo anterior, ¿en qué estado se encontraría el analizador al terminar de procesar la subcadena de entrada 011?

a) I_3

b) I_5

c) I_6