Tema 4. Análisis sintáctico descendente

1. Gramáticas LL(1).
2. Cálculo de Primeros y Siguientes.
3. Tabla y análisis sintáctico LL(1).
4. Transformaciones de gramáticas.
5. A.S. Descendente Recursivo

1. Gramáticas LL(1)

Primeros y Siguientes

PRIMEROS: $(N \cup \Sigma)^* \longrightarrow P(\Sigma \cup \{\varepsilon\})$

$$\text{PRIM}(\alpha) = \{ x \in \Sigma \mid \alpha \Rightarrow^* x\ \beta\} \cup \{\varepsilon \mid \alpha \Rightarrow^* \varepsilon \}$$
$$\alpha, \beta \in (N \cup \Sigma)^*$$

SIGUIENTES: $N \longrightarrow P(\Sigma \cup \{\$\})$

$$\text{SIG}(A) = \{ x \in \Sigma \mid S \Rightarrow^* \alpha A x \beta\} \cup \{ \$ \mid S \Rightarrow^* \alpha A\}$$
$$\alpha, \beta \in (N \cup \Sigma)^*;\ A \in N$$

Condición LL(1)

Una gramática independiente del contexto es **LL(1)**, si para cualquier par de producciones
$(A \rightarrow \alpha$ y $A \rightarrow \beta)$ se cumple la condición:

$$\mathbf{Prim}\ (\alpha\ \mathbf{Sig}\ (A)) \cap \mathbf{Prim}\ (\beta\ \mathbf{Sig}\ (A)) = \varnothing$$

Proposición 1:

Si una gramática es LL(1) entonces no es ambigua.

Proposición 2:

Si una gramática es LL(1) entonces no es recursiva a izquierdas.

2. Cálculo de Primeros y Siguientes

Función Primeros

Función **Primeros** *(x ∈ (N ∪ Σ)*): Conjunto de (Σ ∪ {ε})*

Dada *G = (N, Σ, P, S);* ***con*** *PRIM: Conjunto de (Σ ∪ {ε})*

PRIM **:=** $\varnothing$;

Si $x \in (\Sigma \cup \{\varepsilon\})$ **ent** *PRIM* := {x} → Terminal

Si $x \in N$ **ent** → Auxiliar

Para toda $(x \rightarrow \alpha) \in P$ **hacer** *PRIM* := *PRIM* ∪ Primeros(α)

Si $x = x_1 x_2 \ldots x_m \wedge m > 1$ **ent** → Cadena

i := 1

Mientras $(i < m) \wedge (\varepsilon \in \text{Primeros}(x_i))$ **hacer**

PRIM := *PRIM* ∪ (Primeros(x_i) - {ε });

i := i + 1;

PRIM := *PRIM* ∪ (Primeros(x_i));

Devolver *PRIM*

$S \rightarrow Bb \mid Dc$

$B \rightarrow aB \mid \varepsilon$

$D \rightarrow dD \mid \varepsilon$

Prim(S) = Prim(Bb) U Prim(Dc)

Prim(Bb) = Prim(aB b) U Prim(ε b) = {a, b}

Prim(Dc) = Prim(dD c) U Prim(ε c) = {d, c}

Prim(S) = {a, b} U {d, c} = {a, b, c, d}

Recalcular tras añadir las producciones:

$S \rightarrow Sf$	PRIM(S)={a, b, c, d}
$S \rightarrow BD$	PRIM(S)={a, b, c, d, f, ε }

Función Siguientes

Función Siguientes *(A $\in N$: Conjunto de $(\Sigma \cup \{\$\})$*

Dada $G = (N, \Sigma, P, S)$;

Método

Para todo $A \in N$ hacer $Sig[A] = \phi$

$Sig[S] := \{\$\}$; /* S es el símbolo inicial */

Mientras cambie algún $Sig[X]$ $(X \in N)$ hacer

  Para_toda $(B \rightarrow \alpha\, A\, \beta) \in P$ hacer

    Si $\beta \Rightarrow^* \varepsilon$ /* $\varepsilon \in$ Primeros(β) */

     ent $Sig[A] := Sig[A] \cup (Primeros(\beta) - \{\varepsilon\}) \cup Sig[B]$

     sino $Sig[A] := Sig[A] \cup Primeros(\beta)$

Fin

Ejercicio 1

Calcular SIG para todos los símbolos no-terminales

$E \rightarrow T\ E'$

$E' \rightarrow \varepsilon \mid +\ T\ E'$

$T \rightarrow F\ T'$

$T' \rightarrow \varepsilon \mid *\ F\ T'$

$F \rightarrow num \mid (\ E\ ) \mid id$

PRIM(E')={ +, ε } → SIG(T)

PRIM(T')={ *, ε } → SIG(F)

SIG(E) = { $,

SIG(E') = {

SIG(T) = { +,

SIG(T') = {

SIG(F) = { *,

Ejercicio 1

Calcular SIG para todos los símbolos no-terminales

$E \rightarrow T\ E'$

$E' \rightarrow \varepsilon \mid +\ T\ E'$

$T \rightarrow F\ T'$

$T' \rightarrow \varepsilon \mid *\ F\ T'$

$F \rightarrow num \mid (\ E\ ) \mid id$

$PRIM(E')=\{ +, \varepsilon \} \rightarrow SIG(T)$

$PRIM(T')=\{ *, \varepsilon \} \rightarrow SIG(F)$

$SIG(E) = \{ \$,) \}$

$SIG(E') = \{ \$,) \}$

$SIG(T) = \{ +, \$,) \}$

$SIG(T') = \{ +, \$,) \}$

$SIG(F) = \{ *, +, \$,) \}$

3. Tabla y análisis LL(1)

Tabla de Análisis LL(1)

Algoritmo *Construcción de la T.A. LL(1)*

Entrada $G = (N, \Sigma, P, S)$;

Salida TA: $(N \cup \Sigma \cup \{\$\}) \times (\Sigma \cup \{\$\}) \longrightarrow \{(r: A \rightarrow \beta), sacar, aceptar, error\}$

Método

Inicializar TA con la acción "error";

Para toda $(r: A \rightarrow \beta) \in P$ hacer

para todo a $\in$ PRIMEROS (β SIGUIENTES(A)) hacer

TA [A, a] := $(r: A \rightarrow \beta)$;

Para todo a $\in \Sigma$ hacer TA [a, a] := sacar;

TA [\$, \$] := aceptar;

Fin

Análisis Sintáctico Descendente

Algoritmo A.S.D*: basado en la T.A. LL(1)*

Entrada $\omega \in \Sigma^*$; *TA, para una* $G = (N, \Sigma, P, S)$;

Salida *Si* $\omega \in L(G)$ *entonces* χ *else error( )*

Método

apilar (S); sim = yylex(); $\chi := \varepsilon$; fin = falso;

Repetir

Caso TA [cima,sim] sea

"(r: $A \rightarrow \beta$)": desapilar; apilar(β); $\chi := \chi \cdot r$;

"sacar": desapilar; sim := yylex();

"aceptar": fin := verdad;

"error": yyerror( );

Fin;

Hasta fin

Fin

4. Transformación de gramáticas

Factorización

Una gramática es factorizable cuando existen más de una producción de un mismo no terminal cuya parte derecha comienza por un mismo prefijo.

Factorización:

$A \rightarrow \alpha\, \beta_1 \mid \alpha\, \beta_2 \mid \ldots\ldots \mid \alpha\, \beta_n \mid \gamma$

Es equivalente a:

$A \rightarrow \alpha\ A' \mid \gamma$

$A' \rightarrow \beta_1 \mid \beta_2 \mid \ldots\ldots \mid \beta_n$

donde A' es un nuevo no terminal.

Eliminación recursión a izq.

Una GIC es recursiva a izquierdas sii $\exists\, A \in N$: $A \Rightarrow^* A\,\alpha$.

Eliminación de la recursión directa a izquierdas:

$A \rightarrow A\,\alpha_1 \mid A\,\alpha_2 \mid \ldots. \mid A\,\alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots.. \mid \beta_p$

donde los β_i no comienzan por A , $\forall$i.

Es equivalente a:

$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots.. \mid \beta_p A'$

$A' \rightarrow \alpha_1\ A' \mid \alpha_2\ A' \mid \ldots \mid \alpha_n\ A' \mid \varepsilon$

donde A′ es un nuevo no terminal.

Eliminación recursión a izq.

Algoritmo Eliminación de recursión indirecta a izquierdas

Entrada *G, GIC recursiva a izquierdas sin ciclos ni* A→ e

Método

Ordenar los no terminales $A_{1,}$ A_2 ....A_n

Para i:=1 Hasta n do

Para k:=1 Hasta i -1 Hacer

Si existe $A_i \rightarrow A_k\gamma$ sustituir por $A_i \rightarrow \gamma_1 , \gamma_2 , \ldots \gamma_p$

(donde $A_k \rightarrow \gamma_1 , \gamma_2 , \ldots \gamma_p$)

Eliminar recursión inmediata en $A_i \rightarrow A_i\ \alpha$

Fin para

Fin para

S→ Aa | b B→ab A→SB →

S → Aa | b B→ab

A→bBA'

A'→aBA' | ε

5. A. S. Descendente Recursivo

A. Sintáctico Descendente Recursivo

A → a B c D | Bc   B → b | f | ε   D → d

```
void A () {
   switch(sim){
      case 'a':  empareja('a'); B(); empareja('c'); D();   break ;
      case 'b': case 'f';  case 'c': B() ; empareja('c');  break ;
      default: yyerror();
} }

void B ( ) {
   switch (sim) {
      case 'b'  : empareja ('b') ;  break ;
      case 'f'  : empareja('f') ;  break ;
      case 'c'  :  { } ; break ;
      default:  yyerror () ;
} }

void D ( ) {
   switch (sim) {
      case 'd'  : empareja ('d') ;  break ;
      default:  yyerror () ;
} }
```

```
int main(char x) {
    sim = yylex();
    A () ;
}
```

```
void empareja (char x) {
  if (x == sim) sim=yylex();
  else yyerror();
}
```

Ejercicio 2

Dada la siguiente gramática:

S -> A

A -> B **%** A | B C

B -> D | D ***** B

D -> **x** | **(** C **)**

C -> **+x** | **-x**

a) Obtener la tabla de análisis LL(1). ¿Es una gramática LL(1)? ¿Por qué?

b) Obtener una gramática equivalente LL(1).

	x	**(**	**+**	**-**
S	A, 1	A, 1		
A	B%A, 2 BC, 3	B%A, 2 BC, 3		
B	D, 4 D*B, 5	D, 4 D*B, 5		
D	x, 6	(C), 7		
C			+x, 8	-x, 9

No es LL(1) porque la tabla de análisis LL(1) contiene entradas múltiples para A y B.

S -> A
A -> B A'
A' -> **%** A | C
B -> D B'
B' -> ***** B | ε
D -> **x** | **(** C **)**
C -> **+ x** | **- x**

Ejercicio 3

$S \rightarrow B\ A$

$A \rightarrow \%\ B\ A \quad | \quad \varepsilon$

$B \rightarrow D\ C$

$C \rightarrow \&\ D\ C \quad | \quad \varepsilon$

$D \rightarrow (\ S\ ) \quad | \quad b$

a) Demostrar que la gramática es LL(1) y construir su tabla de análisis LL(1).

b) Realizar la traza de análisis LL(1) para la cadena "((b))".

$S \to B\ A$

$A \to \%\ B\ A \quad | \quad \varepsilon$

$B \to D\ C$

$C \to \&\ D\ C \quad | \quad \varepsilon$

$D \to (\ S\ ) \quad | \quad b$

SIG(S) = { $,) }

SIG(B) = { %, $,) }

SIG(D) = { &, %, $,) }

SIGC) = { %, $,) }

SIG(A) = { $,) }

PRIM(B A Sig(S)) = { (, b }

PRIM(D C Sig(B)) = { (, b }

PRIM((S) Sig(D)) = { (}

PRIM(b Sig(D)) = { b }

PRIM(% B A Sig(A)) = { % }

PRIM(ε Sig(A)) = sig(A) = { $,) }

PRIM(& D C Sig(C)) = { & }

PRIM(ε Sig(C)) = sig(C) = { %, $,) }

	(	)	%	&	b	$
S	S → B A				S → B A	
B	B → D C				B → D C	
D	D → (S)				D → **b**	
C		C → ε	C → ε	C → **&** D C		C → ε
A		A → ε	A → % B A			A → ε
(	pop					
)		pop				
%			pop			
&				pop		
b					pop	
$						Aceptar

PILA	CAD. ENTRADA	CADENA SALIDA
S $	((b)) $	
B A $	((b)) $	1
D C A $	((b)) $	1-4
(S) C A $	((b)) $	1-4-7
S) C A $	(b)) $	1-4-7
B A) C A $	(b)) $	1-4-7-1
D C A) C A $	(b)) $	1-4-7-1-4
(S) C A) C A $	(b)) $	1-4-7-1-4-7
S) C A) C A $	b)) $	1-4-7-1-4-7
B A) C A) C A $	b)) $	1-4-7-1-4-7-1
D C A) C A) C A $	b)) $	1-4-7-1-4-7-1-4
b C A) C A) C A $	b)) $	1-4-7-1-4-7-1-4-8
C A) C A) C A $	)) $	1-4-7-1-4-7-1-4-8
A) C A) C A $	)) $	1-4-7-1-4-7-1-4-8-6
) C A) C A $	)) $	1-4-7-1-4-7-1-4-8-6-3
C A) C A $	) $	1-4-7-1-4-7-1-4-8-6-3
A) C A $	) $	1-4-7-1-4-7-1-4-8-6-3-6
) C A $	) $	1-4-7-1-4-7-1-4-8-6-3-6-3
C A $	$	1-4-7-1-4-7-1-4-8-6-3-6-3
A $	$	1-4-7-1-4-7-1-4-8-6-3-6-3-6
$	$	1-4-7-1-4-7-1-4-8-6-3-6-3-6-3

Ejercicio 4

Dada la gramática

S -> A B C

A -> (S) | x S y | z

B -> x B | ε

C -> z C | ε

a) Construye la tabla de análisis LL(1)

b) ¿Es una gramática LL(1)? ¿Por qué?

c) Realiza la traza para la cadena ω= (zx)

SIG(B)={z, $,), y}

SIG(C)={$,), y}

	(	)	x	y	z	$
S	ABC, 1		ABC,1		ABC,1	
A	(S), 2		xSy, 3		z, 4	
B		ε, 6	xB, 5	ε, 6	ε, 6	ε, 6
C		ε, 8		ε, 8	zC, 7	ε, 8

(S$, (zx)$,) |- (ABC, (zx)$, 1) |-

((S)BC$, (zx)$, 1-2) |- (S)BC$, zx)$, 1-2) |-

(ABC)BC$, zx)$, 1-2-1) |- (zBC)BC$, zx)$, 1-2-1-4) |-

(BC)BC$, x)$, 1-2-1-4) |- (xBC)BC$, x)$, 1-2-1-4-5) |-

(BC)BC$,)$, 1-2-1-4-5) |- (C)BC$,)$, 1-2-1-4-5-6) |-

( )BC$,)$, 1-2-1-4-5-6-8) |- (BC$, $, 1-2-1-4-5-6-8) |-

(C$, $, 1-2-1-4-5-6-8-6) |- ($, $, 1-2-1-4-5-6-8-6-8) |- ACEPTAR