
PARSERS PREDICTIVOS

— Ing. Max Cerna —

Agenda

1. Predictive Top-Down Parsers
2. First
3. Follow
4. Tabla LL(1)

Predictive Top-Down Parsers

PREDICTIVE TOP-DOWN PARSERS

- Analizadores predictivos de arriba hacia abajo
- Similares a RDP pero el analizador puede "predecir" qué producción usar
 - Mirando los siguientes tokens
 - Sin retroceso (backtrack)
- Los analizadores predictivos aceptan gramáticas LL(k)
 - L significa escaneo de entrada "de izquierda a derecha"
 - L significa "derivación (más a la/por) izquierda"
 - k significa "predecir basado en k tokens de anticipación"
 - En la práctica, se utiliza LL(1)

PREDICTIVE TOP-DOWN PARSERS

- Por ejemplo, recordemos la gramática

$E \rightarrow T + E$

| T

$T \rightarrow int$

| $int * T$

| (E)

- Difícil de predecir porque
 - Para T dos producciones comienzan con int
 - Para E no está claro cómo predecir
- Necesitamos factorizar la gramática por/(a la) izquierda

Ejemplo de factorización a la izquierda

Factorizar los prefijos comunes de las producciones

$E \rightarrow T X$ // factor común T, lo de la derecha genera nueva producción

$X \rightarrow + E \mid \epsilon$ // nueva producción

$T \rightarrow int Y \mid (E)$ // factor int, derecha genera nueva producción

$Y \rightarrow * T \mid \epsilon$ // nueva producción

PREDICTIVE TOP-DOWN PARSERS

Elija la alternativa que factoriza correctamente la gramática dada

```
EXPR → if BOOL then { EXPR }  
      | if BOOL then { EXPR } else { EXPR }  
      | ...  
BOOL  → true | false
```

1.

```
EXPR → if true then { EXPR }  
      | if false then { EXPR }  
      | if true then { EXPR } else { EXPR }  
      | if false then { EXPR } else { EXPR }  
      | ...
```

2.

```
EXPR → if BOOL EXPR'  
      | ...  
EXPR' → then { EXPR }  
      | then { EXPR } else { EXPR }  
BOOL  → true | false
```

3.

```
EXPR → EXPR' | EXPR' else { EXPR }  
EXPR' → if BOOL then { EXPR }  
      | ...  
BOOL  → true | false
```

4.

```
EXPR → if BOOL then { EXPR } EXPR'  
      | ...  
EXPR' → else { EXPR } | ε  
BOOL  → true | false
```

PREDICTIVE TOP-DOWN PARSERS

Gramática factorizada

$$E \rightarrow TX$$

$$T \rightarrow (E) | int Y$$

$$X \rightarrow +E | \varepsilon$$

$$Y \rightarrow *T | \varepsilon$$

Tabla LL(1)

siguiente token de entrada

proxima produccion a utilizar


no terminal

	int	*	+	()	\$
E	TX			TX		
X			+ E		ε	ε
T	int Y			(E)		
Y		* T	ε		ε	ε

PREDICTIVE TOP-DOWN PARSERS

Considere la entrada $[E, \text{int}]$

- “Cuando el no terminal actual es E y la siguiente entrada es int , usar la producción $E \rightarrow TX$ ”
- Esto puede generar un int en la primera posición



	int	*	+	()	\$
E	TX			TX		
X			+ E		ϵ	ϵ
T	int Y			(E)		
Y		* T	ϵ		ϵ	ϵ

PREDICTIVE TOP-DOWN PARSERS

Considere la entrada $[Y, +]$

- “Cuando el no terminal actual es Y y la siguiente entrada es $+$, eliminar Y ”
- Y puede ir seguido de $+$ solo si $Y \rightarrow \epsilon$


	int	*	+	()	\$
E	TX			TX		
X			$+E$		ϵ	ϵ
T	$\text{int } Y$			(E)		
Y		$*T$	ϵ		ϵ	ϵ



PREDICTIVE TOP-DOWN PARSERS

Considere la entrada $[E, *]$

- “No hay forma de derivar una cadena que comience con $*$ desde el no terminal E ”



	int	*	+	()	\$
E	T X			T X		
X			+ E		ϵ	ϵ
T	int Y			(E)		
Y		* T	ϵ		ϵ	ϵ

PREDICTIVE TOP-DOWN PARSERS

- Método similar al descenso recursivo, excepto
 - Para el S no terminal más a la izquierda
 - Miramos el siguiente token de entrada a
 - Y elige la producción que se muestra en $[S, a]$
- Una pila registra la frontera del árbol de análisis
 - No terminales que aún no se han ampliado
 - Terminales que aún tienen que coincidir con la entrada
 - Parte superior de la pila = terminal pendiente más a la izquierda o no terminal
- Rechazar al llegar al estado de error
- Aceptar al final de la entrada y pila vacía

LL(1) Parsing Algorithm

```
initialize stack = <S $> and next
repeat
  case stack of
    <X, rest> : if T[X,*next] = Y1...Yn
                  then stack ← <Y1...Yn, rest>;
                  else error ();

    <t, rest>  : if t == *next ++
                  then stack ← <rest>;
                  else error ();

until stack == < >
```

LL(1) Parsing Algorithm

initialize stack = $\langle S \$ \rangle$ and next

marca el fondo de la pila

repeat

case stack of

$\langle X, \text{rest} \rangle$: if $T[X, *next] = Y_1 \dots Y_n$
then stack $\leftarrow \langle Y_1 \dots Y_n, \text{rest} \rangle$;
else error ();

Para X no terminal en la parte superior de la pila, búsqueda de producción

$\langle t, \text{rest} \rangle$: if $t == *next ++$
then stack $\leftarrow \langle \text{rest} \rangle$;
else error ();

Para la terminal t en la parte superior de la pila, verifique que t coincida con el siguiente token de entrada.

Pop X , push la producción a la pila. Ten en cuenta que el símbolo más a la izquierda de la producción está en la parte superior de la pila.

until stack == $\langle \rangle$

Ejemplo LL(1) Parsing

	int	*	+	()	\$
E	T X			T X		
X			+ E		ϵ	ϵ
T	int Y			(E)		
Y		* T	ϵ		ϵ	ϵ

Stack	Entrada	Acción
E\$	int * int\$	T X
T X \$	int * int\$	int Y
int Y X \$	int * int\$	terminal
Y X \$	* int \$	* T
* T X \$	* int \$	terminal
T X \$	int \$	int Y
int Y X \$	int \$	terminal
Y X \$	\$	ϵ
X \$	\$	ϵ
\$	\$	ACCEPT

PREDICTIVE TOP-DOWN PARSERS

Considere la siguiente tabla y gramática. ¿Cual es el siguiente estado si el stack actualmente contiene $E'\$$ y la entrada contiene

else { if false then { false } } \$?

	if	then	else	{	}	true	false	\$
E	if B then { E } E'				ϵ	B	B	ϵ
E'			else { E }		ϵ			ϵ
B						true	false	

$E \rightarrow \text{if } B \text{ then } \{E\} E' \mid B \mid \epsilon$

$E' \rightarrow \text{else } \{E\} \mid \epsilon$

$B \rightarrow \text{true} \mid \text{false}$

La Intuición para la construcción de tablas de análisis

- Considere el no terminal A , la producción $A \rightarrow \alpha$ y el token t
- Agregar $T[A, t] = \alpha$ en dos casos
 1. Si $A \rightarrow \alpha \rightarrow^* t\beta$
 - α puede derivar t en la primera posición
 - Decimos que $t \in \text{First}(\alpha)$
 2. Si $A \rightarrow \alpha \rightarrow^* \epsilon$ y $S \rightarrow^* \gamma A t \delta$
 - Útil si la pila tiene A , la entrada es t y A no puede derivar t
 - En este caso, la única opción es deshacerse de A (derivando ϵ)
 - Lo anterior solo puede funcionar si t puede seguir a A en al menos una derivación
 - Decimos $t \in \text{Follow}(A)$

First

First

Definición:

$$\text{First}(X) = \{t \mid X \rightarrow t\alpha\} \cup \{\varepsilon \mid X \rightarrow \varepsilon\}$$

Algoritmo:

1) $\text{First}(t) = \{t\}$

2) $\varepsilon \in \text{First}(X)$

a) si $X \rightarrow \varepsilon$

b) si $X \rightarrow A_1 \dots A_n$ y $\varepsilon \in \text{First}(A_i)$ para todo $1 \leq i \leq n$

3) $\text{First}(\alpha) \subseteq \text{First}(X)$

a) si $X \rightarrow \alpha$

b) si $X \rightarrow A_1 \dots A_n \alpha$ y $\varepsilon \in \text{First}(A_i)$ para todo $1 \leq i \leq n$

First

Trabajemos con la gramática factorizada

$$E \rightarrow T X$$

$$T \rightarrow (E) \mid \text{int } Y$$

$$X \rightarrow + E \mid \varepsilon$$

$$Y \rightarrow * T \mid \varepsilon$$

Ejercicio 1 - First

$$S \rightarrow A B C$$

$$A \rightarrow x A$$

$$| \varepsilon$$

$$B \rightarrow y B$$

$$| \varepsilon$$

$$C \rightarrow z$$

Ejercicio 2 - First

$S \rightarrow a A$

$| b B$

$A \rightarrow c A$

$| d$

$B \rightarrow \epsilon$

Follow

Follow

Definición:

$$\text{Follow}(X) = \{t \mid S \rightarrow^* \beta X t \delta\}$$

Razonamiento

- Si $X \rightarrow AB$ entonces $\text{First}(B) \subseteq \text{Follow}(A)$ y $\text{Follow}(X) \subseteq \text{Follow}(B)$
 - También si $B \rightarrow^* \epsilon$ entonces $\text{Follow}(X) \subseteq \text{Follow}(A)$
- Si s es el símbolo inicial entonces $\$ \in \text{Follow}$

Follow

Algoritmo:

1. $\$ \in \text{Follow}(S)$
2. $\text{First}(\beta) - \{\varepsilon\} \subseteq \text{Follow}(X)$
 - a. Para cada producción $A \rightarrow \alpha X \beta$
3. $\text{Follow}(A) \subseteq \text{Follow}(X)$
 - a. Para cada producción $A \rightarrow \alpha X \beta$ donde $\varepsilon \in \text{First}(\beta)$

Ejercicio 1 - Follow

$$S \rightarrow X Y$$

$$X \rightarrow a X \mid \varepsilon$$

$$Y \rightarrow b \mid \varepsilon$$

Ejercicio 2 - Follow

$$S \rightarrow A B$$

$$A \rightarrow a A \mid \varepsilon$$

$$B \rightarrow b B \mid c$$

Tabla LL(1)

Tabla LL(1)

Construir una tabla de análisis T para CFG

Para cada producción $A \rightarrow \alpha$ en G debemos:

- Para cada terminal $t \in \text{First}(\alpha)$ colocamos $T[A, t] = \alpha$
- Si $\epsilon \in \text{First}(\alpha)$, para cada $t \in \text{Follow}(A)$ colocamos $T[A, t] = \alpha$
- Si $\epsilon \in \text{First}(\alpha)$ y $\$ \in \text{Follow}(A)$ colocamos $T[A, \$] = \alpha$

Tabla LL(1)

Trabajemos con la gramática factorizada

$$E \rightarrow T X$$

$$T \rightarrow (E) \mid int Y$$

$$X \rightarrow + E \mid \varepsilon$$

$$Y \rightarrow * T \mid \varepsilon$$

Tabla LL(1)

No Terminal	First	Follow
E	{int, (}	{ \$,) }
X	{ +, ϵ }	{ \$,) }
T	{ int, (}	{ +, \$,) }
Y	{ *, ϵ }	{ +, \$,) }

Ejemplo LL(1) Parsing

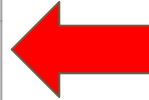
$$E \rightarrow T X$$

$$T \rightarrow (E) \mid int Y$$

$$X \rightarrow + E \mid \varepsilon$$

$$Y \rightarrow * T \mid \varepsilon$$

	int	*	+	()	\$
E	T X			T X		
X						
T						
Y						



No Terminal	First	Follow
E	{int, (}	{ \$,) }
X	{ +, ε }	{ \$,) }
T	{ int, (}	{ +, \$,) }
Y	{ *, ε }	{ +, \$,) }

Ejemplo LL(1) Parsing

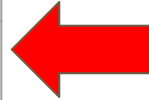
$$E \rightarrow T X$$

$$T \rightarrow (E) \mid int Y$$

$$X \rightarrow + E \mid \varepsilon$$

$$Y \rightarrow * T \mid \varepsilon$$

	int	*	+	()	\$
E	TX			TX		
X			+ E			
T						
Y						



No Terminal	First	Follow
E	{int, (}	{ \$,) }
X	{ +, ε }	{ \$,) }
T	{ int, (}	{ +, \$,) }
Y	{ *, ε }	{ +, \$,) }

Ejemplo LL(1) Parsing

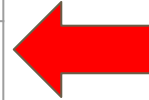
$$E \rightarrow T X$$

$$T \rightarrow (E) \mid \text{int } Y$$

$$X \rightarrow + E \mid \varepsilon$$

$$Y \rightarrow * T \mid \varepsilon$$

	int	*	+	()	\$
E	T X			T X		
X			+ E			
T	int Y					
Y						



No Terminal	First	Follow
E	{int, (}	{ \$,) }
X	{ +, ε }	{ \$,) }
T	{ int, (}	{ +, \$,) }
Y	{ *, ε }	{ +, \$,) }

Ejemplo LL(1) Parsing

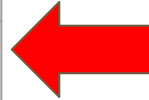
$$E \rightarrow T X$$

$$T \rightarrow (E) \mid int Y$$

$$X \rightarrow + E \mid \varepsilon$$

$$Y \rightarrow * T \mid \varepsilon$$

	int	*	+	()	\$
E	T X			T X		
X			+ E			
T	int Y			(E)		
Y						



No Terminal	First	Follow
E	{int, (}	{ \$,) }
X	{ +, ε }	{ \$,) }
T	{ int, (}	{ +, \$,) }
Y	{ *, ε }	{ +, \$,) }

Ejemplo LL(1) Parsing

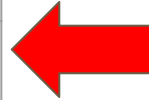
$$E \rightarrow T X$$

$$T \rightarrow (E) \mid int Y$$

$$X \rightarrow + E \mid \varepsilon$$

$$Y \rightarrow * T \mid \varepsilon$$

	int	*	+	()	\$
E	T X			T X		
X			+ E			
T	int Y			(E)		
Y		* T				



No Terminal	First	Follow
E	{int, (}	{ \$,) }
X	{ +, ε }	{ \$,) }
T	{ int, (}	{ +, \$,) }
Y	{ *, ε }	{ +, \$,) }

Ejemplo LL(1) Parsing

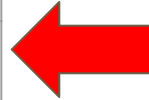
$$E \rightarrow T X$$

$$T \rightarrow (E) \mid \text{int } Y$$

$$X \rightarrow + E \mid \epsilon$$

$$Y \rightarrow * T \mid \epsilon$$

	int	*	+	()	\$
E	T X			T X		
X			+ E		ϵ	ϵ
T	int Y			(E)		
Y		* T				



No Terminal	First	Follow
E	{int, (}	{ \$,) }
X	{ +, ϵ }	{ \$,) }
T	{ int, (}	{ +, \$,) }
Y	{ *, ϵ }	{ +, \$,) }

Ejemplo LL(1) Parsing

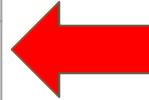
$$E \rightarrow T X$$

$$T \rightarrow (E) \mid \text{int } Y$$

$$X \rightarrow + E \mid \varepsilon$$

$$Y \rightarrow * T \mid \varepsilon$$

	int	*	+	()	\$
E	T X			T X		
X			+ E		ε	ε
T	int Y			(E)		
Y		* T	ε		ε	ε



No Terminal	First	Follow
E	{int, (}	{ \$,) }
X	{ +, ε }	{ \$,) }
T	{ int, (}	{ +, \$,) }
Y	{ *, ε }	{ +, \$,) }

Ejercicio 1 - Tabla LL(1)

$S \rightarrow X Y$

$X \rightarrow a X \mid \varepsilon$

$Y \rightarrow b \mid \varepsilon$

No Terminal	First	Follow
S	{ a, b, ε }	{ \$, b }
X	{ a, ε }	{ b }
Y	{ b, ε }	{ \$, b }

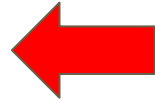
Ejercicio 1 - Tabla LL(1)

$$S \rightarrow X Y$$

$$X \rightarrow a X \mid \varepsilon$$

$$Y \rightarrow b \mid \varepsilon$$

	a	b	\$
S	XY	XY	XY
X	aX	ε	
Y		b/ ε	ε



No Terminal	First	Follow
S	{ a, b, ε }	{ \$, b }
X	{ a, ε }	{ b }
Y	{ b, ε }	{ \$, b }

Ejercicio 1 - Tabla LL(1)

$S \rightarrow A B$

$A \rightarrow a A \mid \epsilon$

$B \rightarrow b B \mid c$

No Terminal	First	Follow
S	{ a, b, c }	{ \$ }
A	{ a, ϵ }	{ b, c }
B	{ b, c }	{ \$ }

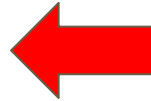
Ejercicio 1 - Tabla LL(1)

$$S \rightarrow A B$$

$$A \rightarrow a A \mid \varepsilon$$

$$B \rightarrow b B \mid c$$

	a	b	c	\$
S	AB			AB
A	aA	ε	ε	
B		bB	c	



No Terminal	First	Follow
S	{ a, b, c }	{ \$ }
A	{ a, ε }	{ b, c }
B	{ b, c }	{ \$ }

Tabla LL(1)

- Si una entrada es definida múltiples veces entonces G no es LL(1)
 - No está factorizada por la izquierda
 - Tiene recursividad por la izquierda
 - Es ambigua
- La mayoría de lenguajes de programación no son LL(1)