

Parsing con gramáticas LL(k)

Clase 24

IIC 2223

Prof. Cristian Riveros

Definiciones de prefijos (recordatorio)

Definiciones

$$w|_k = \begin{cases} a_1 \dots a_n & \text{si } n \leq k \\ a_1 \dots a_k & \text{si } k < n \end{cases} \quad L|_k = \{w|_k \mid w \in L\}$$

$$u \odot_k v = (u \cdot v)|_k \quad L_1 \odot_k L_2 = \{w_1 \odot_k w_2 \mid w_1 \in L_1 \text{ y } w_2 \in L_2\}$$

Los operadores $|_k$ y \odot_k “miran” hasta un prefijo k .

Definición de first_k y follow_k (recordatorio)

Sea $\mathcal{G} = (V, \Sigma, P, S)$ una gramática libre de contexto y $k \geq 1$.

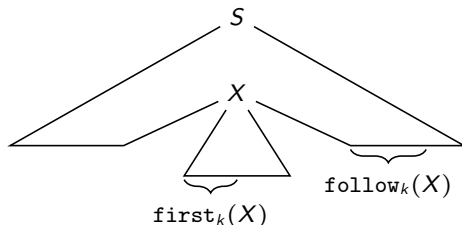
Definiciones

Se define la función $\text{first}_k : (V \cup \Sigma)^* \rightarrow 2^{\Sigma^{\leq k}}$ tal que, para $\gamma \in (V \cup \Sigma)^*$:

$$\text{first}_k(\gamma) = \{u|_k \mid \gamma \xRightarrow{*} u\}$$

Se define la función $\text{follow}_k : V \rightarrow 2^{\Sigma^{\leq k}_{\#}}$ como:

$$\text{follow}_k(X) = \{w \mid S \xRightarrow{*} \alpha X \beta \text{ y } w \in \text{first}_k(\beta\#)\}$$



Definición gramáticas $LL(k)$ (recordatorio)

Definición

$\mathcal{G} = (V, \Sigma, P, S)$ es una gramática $LL(k)$ si para todas derivaciones:

$$\blacksquare S \xRightarrow[\text{lm}]{*} uY\beta \Rightarrow_{\text{lm}} u\gamma_1\beta \xRightarrow[\text{lm}]{*} uv_1$$

$$\blacksquare S \xRightarrow[\text{lm}]{*} uY\beta \Rightarrow_{\text{lm}} u\gamma_2\beta \xRightarrow[\text{lm}]{*} uv_2 \quad \text{y}$$

$$\blacksquare v_1|_k = v_2|_k$$

entonces se cumple que $\gamma_1 = \gamma_2$.

Teorema

\mathcal{G} es una gramática $LL(k)$ si, y solo si, para todas dos reglas distintas

$Y \rightarrow \gamma_1, Y \rightarrow \gamma_2 \in P$ y para todo $S \xRightarrow[\text{lm}]{*} uY\beta$, se tiene que:

$$\text{first}_k(\gamma_1\beta) \cap \text{first}_k(\gamma_2\beta) = \emptyset$$

Gramáticas $LL(k)$ fuerte (recordatorio)

Sea $\mathcal{G} = (V, \Sigma, P, S)$ una gramática libre de contexto y $k \geq 1$.

Definición

\mathcal{G} es una gramática $LL(k)$ **fuerte** si para todas dos reglas distintas $Y \rightarrow \gamma_1, Y \rightarrow \gamma_2 \in P$ se tiene que:

$$\text{first}_k(\gamma_1) \odot_k \text{follow}_k(Y) \cap \text{first}_k(\gamma_2) \odot_k \text{follow}_k(Y) = \emptyset$$

Teorema

Una gramática \mathcal{G} es $LL(1)$ si, y solo si, \mathcal{G} es $LL(1)$ **fuerte**.

¿si \mathcal{G} es $LL(k)$, entonces es $LL(k)$ fuerte?

Outline

Algunas consideraciones

Parsing de $LL(k)$

Outline

Algunas consideraciones

Parsing de $LL(k)$

Problema con gramáticas $LL(k)$

Definición $LL(k)$

$$\left. \begin{array}{l} \blacksquare S \xRightarrow[\text{lm}]{*} uY\beta \Rightarrow_{\text{lm}} u\gamma_1\beta \xRightarrow[\text{lm}]{*} uv_1 \\ \blacksquare S \xRightarrow[\text{lm}]{*} uY\beta \Rightarrow_{\text{lm}} u\gamma_2\beta \xRightarrow[\text{lm}]{*} uv_2 \quad y \\ \blacksquare v_1|_k = v_2|_k \end{array} \right\} \text{ entonces } \gamma_1 = \gamma_2.$$

Considere la siguiente gramática

$$\begin{array}{lcl} S & \rightarrow & Xa \mid Xb \\ X & \rightarrow & c \end{array}$$

¿es esta gramática del tipo $LL(1)$?

Problemas de factorización

$$\left. \begin{array}{l} \text{Definición LL}(k) \\ \begin{array}{l} \blacksquare S \xRightarrow[\text{lm}]{*} uY\beta \Rightarrow_{\text{lm}} u\gamma_1\beta \xRightarrow[\text{lm}]{*} uv_1 \\ \blacksquare S \xRightarrow[\text{lm}]{*} uY\beta \Rightarrow_{\text{lm}} u\gamma_2\beta \xRightarrow[\text{lm}]{*} uv_2 \text{ y} \\ \blacksquare v_1|_k = v_2|_k \end{array} \end{array} \right\} \text{ entonces } \gamma_1 = \gamma_2.$$

Solución (factorización)

En general, si tenemos una regla:

$$X \rightarrow \gamma\alpha_1 \mid \gamma\alpha_2$$

siempre podemos “**factorizar**” la regla manteniendo la semántica, como:

$$\begin{array}{l} X \rightarrow \gamma X' \\ X' \rightarrow \alpha_1 \mid \alpha_2 \end{array}$$

Otro problema con gramáticas $LL(k)$

$$\left. \begin{array}{l} \text{Definición } LL(k) \\ \begin{array}{l} \blacksquare S \xRightarrow[\text{lm}]{*} uY\beta \Rightarrow_{\text{lm}} u\gamma_1\beta \xRightarrow[\text{lm}]{*} uv_1 \\ \blacksquare S \xRightarrow[\text{lm}]{*} uY\beta \Rightarrow_{\text{lm}} u\gamma_2\beta \xRightarrow[\text{lm}]{*} uv_2 \quad y \\ \blacksquare v_1|_k = v_2|_k \end{array} \end{array} \right\} \text{ entonces } \gamma_1 = \gamma_2.$$

Considere la siguiente gramática:

$$E \rightarrow E * E \mid n$$

¿es esta gramática del tipo $LL(1)$? ¿ $LL(k)$?

... ¿cuál es el problema con esta gramática?

Problema con recursión por la izquierda

Definición

Una gramática \mathcal{G} se dice **recursiva por la izquierda** si existe $X \in V$ tal que:

$$X \xRightarrow{+} X\gamma \quad \text{para algún } \gamma \in (V \cup \Sigma)^*$$

Teorema

Si $\mathcal{G} = (V, \Sigma, P, S)$ es una gramática reducida y recursiva por la izquierda, entonces \mathcal{G} NO es $LL(k)$ para todo $k \geq 1$.

Problema con recursión por la izquierda

Teorema

Si $\mathcal{G} = (V, \Sigma, P, S)$ es una gramática reducida y recursiva por la izquierda, entonces \mathcal{G} NO es $LL(k)$ para todo $k \geq 1$.

Demostración

Por simplicidad, suponga que $X \rightarrow X\beta \in P$ y $X \rightarrow w \in P$.

Como \mathcal{G} es reducida, entonces existe una derivación $S \xRightarrow[\text{lm}]{*} uX\gamma$.

$$S \xRightarrow[\text{lm}]{*} uX\gamma \Rightarrow_{\text{lm}}^{n\text{-veces}} uX\beta^n\gamma$$

Por **contradicción**, suponga que \mathcal{G} es $LL(k)$. Por lo tanto:

$$\text{first}_k(X\beta^{n+1}\gamma) \cap \text{first}_k(w\beta^n\gamma) = \emptyset$$

Suponga que $\beta \xRightarrow{*} v \in \Sigma^*$ y $\gamma \xRightarrow{*} v' \in \Sigma^*$. Con $n = k$, tendremos que:

$$(wv^k v')|_k \in \text{first}_k(X\beta^{k+1}\gamma) \cap \text{first}_k(w\beta^k\gamma) \quad \rightarrow \leftarrow$$

¿qué podemos hacer si \mathcal{G} es recursiva por la izquierda?

Recursión inmediata por la izquierda

Suponga que existe $X \in V$ tal que:

$$X \rightarrow X\alpha_1 \mid \dots \mid X\alpha_m \mid \beta_1 \mid \dots \mid \beta_n$$

¿cómo podemos **eliminar** la recursión inmediata por la izquierda?

Considere la misma gramática pero **cambiando** las reglas de X por:

$$\begin{aligned} X &\rightarrow \beta_1 X' \mid \dots \mid \beta_n X' \\ X' &\rightarrow \alpha_1 X' \mid \dots \mid \alpha_m X' \mid \epsilon \end{aligned}$$

¿es la nueva gramática **recursiva inmediata por la izquierda** en X ?

Recursión inmediata por la izquierda

Idea de eliminación de recursión inmediata

$$X \rightarrow X\alpha_1 \mid \dots \mid X\alpha_m \mid \beta_1 \mid \dots \mid \beta_n$$

$$\begin{aligned} X &\rightarrow \beta_1 X' \mid \dots \mid \beta_n X' \\ X' &\rightarrow \alpha_1 X' \mid \dots \mid \alpha_m X' \mid \epsilon \end{aligned}$$

Ejemplo de eliminar la **recursión inmediata**

$S \rightarrow Xa \mid b$	$S \rightarrow Xa \mid b$
$X \rightarrow Xc \mid d$	$X \rightarrow dX'$
	$X' \rightarrow cX' \mid \epsilon$

Recursión inmediata por la izquierda

Teorema

Sea \mathcal{G} una gramática tal que que existe $X \in V$:

$$X \rightarrow X\alpha_1 \mid \cdots \mid X\alpha_m \mid \beta_1 \mid \cdots \mid \beta_n$$

Sea \mathcal{G}' la misma gramática \mathcal{G} pero cambiando las reglas de X por:

$$\begin{aligned} X &\rightarrow \beta_1 X' \mid \cdots \mid \beta_n X' \\ X' &\rightarrow \alpha_1 X' \mid \cdots \mid \alpha_m X' \mid \epsilon \end{aligned}$$

Entonces $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$.

Demostración

Una derivación por la izquierda de X en \mathcal{G} :

$$X \Rightarrow_{\text{lm}} X\alpha_{i_1} \Rightarrow_{\text{lm}} X\alpha_{i_2}\alpha_{i_1} \Rightarrow_{\text{lm}} \cdots \Rightarrow_{\text{lm}} X\alpha_{i_p}\alpha_{i_{p-1}}\cdots\alpha_{i_1} \Rightarrow_{\text{lm}} \beta_j\alpha_{i_p}\alpha_{i_{p-1}}\cdots\alpha_{i_1}$$

Una derivación por la derecha de X en \mathcal{G}' **equivalente**:

$$X \Rightarrow_{\text{rm}} \beta_j X' \Rightarrow_{\text{rm}} \beta_j \alpha_{i_p} X' \Rightarrow_{\text{rm}} \cdots \Rightarrow_{\text{rm}} \beta_j \alpha_{i_p} \cdots \alpha_{i_2} \alpha_{i_1} X' \Rightarrow_{\text{rm}} \beta_j \alpha_{i_p} \alpha_{i_{p-1}} \cdots \alpha_{i_1} \blacksquare$$

Recursión por la izquierda no-inmediata

Considere la siguiente gramática **recursiva por la izquierda**:

$$\begin{aligned} S &\rightarrow Xa \mid b \\ X &\rightarrow Yc \\ Y &\rightarrow Xd \mid e \end{aligned}$$

¿cómo eliminamos la recursión por la izquierda **no-inmediata**?

Estrategia

Dado $V = \{X_1, \dots, X_n\}$, removemos la recursión inductivamente en n tal que, en cada paso i de la inducción, se cumplirá que para todo $i, j \leq n$:

si $X_i \rightarrow X_j \alpha$, entonces $i < j$.

Recursión por la izquierda no-inmediata

input : Gramática $\mathcal{G} = (V, \Sigma, P, S)$ y $V = \{X_1, \dots, X_n\}$

output: Gramática \mathcal{G}' sin recursión por la izquierda

Function EliminarRecursión (\mathcal{G})

$P' := P$

for $i = 1$ **to** n **do**

for $j = 1$ **to** $i - 1$ **do**

foreach $X_i \rightarrow X_j \gamma \in P'$ **do**

foreach $X_j \rightarrow \alpha \in P'$ **do**

$P' := P' \cup \{X_i \rightarrow \alpha \gamma\}$

$P' := P' - \{X_i \rightarrow X_j \gamma\}$

 Remove recursión inmediata para X_i en P' (si existe)

$V' := \{X_1, \dots, X_n\} \cup \{X'_1, \dots, X'_n\}$

return (V', Σ, P', S)

Ejercicio: demuestre la correctitud del algoritmo.

Recursión por la izquierda no-inmediata

Ejemplo

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid n$$

Eliminando la **recursión inmediata** de E :

$$E \rightarrow TE'$$

$$E' \rightarrow +TE' \mid \epsilon$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid n$$

Eliminando la **recursión inmediata** de T :

$$E \rightarrow TE'$$

$$E' \rightarrow +TE' \mid \epsilon$$

$$T \rightarrow FT'$$

$$T' \rightarrow *FT' \mid \epsilon$$

$$F \rightarrow (E) \mid n$$

Problema con recursión por la izquierda (conclusión)

Definición

Una gramática \mathcal{G} se dice **recursiva por la izquierda** si existe $X \in V$ tal que:

$$X \Rightarrow^+ X\gamma \quad \text{para algún } \gamma \in (V \cup \Sigma)^*$$

Teorema

Si $\mathcal{G} = (V, \Sigma, P, S)$ es una gramática reducida y recursiva por la izquierda, entonces \mathcal{G} NO es $LL(k)$ para todo $k \geq 1$.

Conclusión

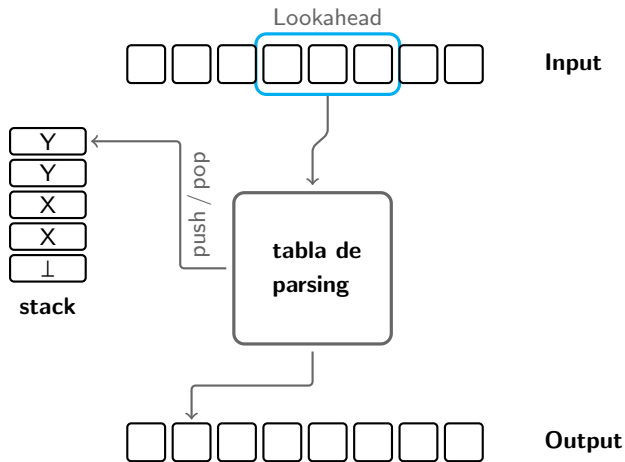
Es **posible eliminar** la recursividad por la izquierda, pero esto **NO asegura** que el resultado sea una gramática $LL(k)$ para algún k .

Outline

Algunas consideraciones

Parsing de $LL(k)$

Parsing de gramáticas $LL(k)$



Parsing de gramáticas $LL(k)$

Sea Σ un alfabeto finito.

Definiciones

Se definen los siguientes conjuntos de palabra:

- $\dot{\Sigma} = \Sigma^* \times \Sigma^*$
- $\dot{\Sigma}^{\leq k} = \{ (u, v) \in \dot{\Sigma} \mid |uv| \leq k \}$
- $\dot{\Sigma}_{\#}^{\leq k} = \{ (u, v) \in \dot{\Sigma} \mid |uv| \leq k \} \cup \{ (u, v\#) \mid (u, v) \in \dot{\Sigma} \mid |uv| < k \}$

Notación

- En vez de $(u, v) \in \dot{\Sigma}_{\#}^{\leq k}$, escribiremos $u.v \in \dot{\Sigma}_{\#}^{\leq k}$.
- El par $\epsilon.\epsilon$ lo denotaremos solamente por ϵ .

Transductor apilador con k -lookahead

Definición

Un transductor apilador con k -lookahead (k -PDT) es una tupla:

$$\mathcal{T} = (Q, \Sigma, \Omega, \Delta, q_0, F)$$

- Q es un conjunto finito de estados.
- Σ es el alfabeto de input.
- Ω es el alfabeto de output.
- $\Delta \subseteq Q^+ \times \dot{\Sigma}_{\#}^{\leq k} \times (\Omega \cup \{\epsilon\}) \times Q^*$ es la **relación de transición**.
- $q_0 \in Q$ es un conjunto de estados iniciales.
- $F \subseteq Q$ es el conjunto de estados finales.

Configuración de un k -PDT

Sea $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, q_0, F)$ un k -PDT.

Definición

Una **configuración** de \mathcal{T} es una tupla:

$$(q_1 \dots q_k, w, o) \in (Q^+, \Sigma^* \cdot \{\#\}, \Omega^*)$$

- $q_1 \dots q_k$ es el contenido del stack con q_1 el tope del stack.
- w es el contenido del input.
- o es el contenido del output.

Decimos que una configuración:

- $(q_0, w\#, \epsilon)$ es **inicial**.
- $(q_f, \#, o)$ es **final** si $q_f \in F$.

Ejecución de un k -PDT

Sea $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, q_0, F)$ un k -PDT.

Definición

Se define la relación $\vdash_{\mathcal{T}}$ de **siguiente-paso** entre configuraciones de \mathcal{T} :

$$(\gamma_1, w_1, o_1) \vdash_{\mathcal{T}} (\gamma_2, w_2, o_2)$$

si, y solo si, existe $(\alpha, u.v, a, \beta) \in \Delta$, $\gamma \in \Gamma^*$ y $w \in \Sigma^* \cdot \{\#\}$ tal que:

- **Stack:** $\gamma_1 = \alpha \cdot \gamma$ y $\gamma_2 = \beta \cdot \gamma$
- **Look-ahead:** $w_1 = u \cdot v \cdot w$ y $w_2 = v \cdot w$
- **Output:** $o_2 = o_1 \cdot a$

Se define $\vdash_{\mathcal{T}}^*$ como la clausura **refleja** y **transitiva** de $\vdash_{\mathcal{T}}$.

Función definida por un k -PDT

Sea $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, q_0, F)$ un k -PDT, $w \in \Sigma^*$ y $o \in \Omega^*$.

Definiciones

- \mathcal{T} **entrega** o con **input** w si existe una configuración **inicial** $(q_0, w \cdot \#, \epsilon)$ y una configuración **final** $(q_f, \#, o)$ tal que:

$$(q_0, w \cdot \#, \epsilon) \vdash_{\mathcal{T}}^* (q_f, \#, o)$$

- Se define la función $\llbracket \mathcal{T} \rrbracket : \Sigma^* \rightarrow 2^{\Omega^*}$:

$$\llbracket \mathcal{T} \rrbracket(w) = \{o \in \Omega^* \mid \mathcal{T} \text{ entrega } o \text{ con input } w\}$$

En un k -PDT combinamos las ideas de **autómatas apilador**, **transductor** y **k -lookahead** vistas anteriormente.

Determinismo en k -PDT

Sea $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, q_0, F)$ un k -PDT.

Definición

\mathcal{T} es **determinista** si para todo $(\alpha_1, u_1.v_1, a_1, \beta_1), (\alpha_2, u_2.v_2, a_2, \beta_2) \in \Delta$ con $(\alpha_1, u_1.v_1, a_1, \beta_1) \neq (\alpha_2, u_2.v_2, a_2, \beta_2)$ se cumple que:

α_1 NO es prefijo de α_2 o $u_1.v_1$ NO es prefijo de $u_2.v_2$.

“Para cualquier configuración (γ, w, o) existe a lo más una configuración (γ', w', o) tal que $(\gamma, w, o) \vdash_{\mathcal{T}}^ (\gamma', w', o')$.”*

¿cuál es la **ventaja** de un k -PDT determinista?

Parser k -PDT para gramática $LL(k)$ fuerte

Sea $\mathcal{G} = (V, \Sigma, P, S)$ una gramática $LL(k)$ fuerte.

Construcción

Se define el k -PDT para \mathcal{G} :

$$\mathcal{T}[\mathcal{G}] = (V \cup \Sigma \cup \{q_0, q_f\}, \Sigma, \underbrace{P}_{\Omega}, \Delta, q_0, \{q_f\})$$

La relación de transición Δ se define como:

Inicio: $(q_0, \epsilon., \epsilon, S \cdot q_f)$

Reducir: $(a, a., \epsilon, \epsilon)$ para cada $a \in \Sigma$

Expandir: $(X, .u, p, \gamma)$

para cada $p := (X \rightarrow \gamma) \in P$ tal que $u \in \text{first}_k(\gamma) \odot_k \text{follow}_k(X)$

Parser k -PDT para gramática $LL(k)$ fuerte

$$\mathcal{T}[\mathcal{G}] = (V \cup \Sigma \cup \{q_0, q_f\}, \Sigma, \underbrace{P}_{\Omega}, \Delta, q_0, \{q_f\})$$

Inicio: $(q_0, \epsilon., \epsilon, S \cdot q_f)$

Reducir: $(a, a., \epsilon, \epsilon)$ para cada $a \in \Sigma$

Expandir: $(X, .u, p, \gamma)$

para cada $p := (X \rightarrow \gamma) \in P$ tal que $u \in \text{first}_k(\gamma) \odot_k \text{follow}_k(X)$

Propiedades

1. $\mathcal{T}[\mathcal{G}]$ es un k -PDT **determinista** si, y solo si, \mathcal{G} es $LL(k)$ fuerte.
2. si $w \notin \mathcal{L}(\mathcal{G})$ entonces $\llbracket \mathcal{T} \rrbracket(w) = \emptyset$.
3. si $w \in \mathcal{L}(\mathcal{G})$ entonces $\llbracket \mathcal{T} \rrbracket(w) = \{r_1 \dots r_m\}$ es una derivación por la izquierda de \mathcal{G} sobre w .

Parsing lineal para gramática $LL(k)$ fuerte

Propiedades

1. $\mathcal{T}[\mathcal{G}]$ es un k -PDT **determinista** si, y solo si, \mathcal{G} es $LL(k)$ fuerte.
2. si $w \notin \mathcal{L}(\mathcal{G})$ entonces $\llbracket \mathcal{T} \rrbracket(w) = \emptyset$.
3. si $w \in \mathcal{L}(\mathcal{G})$ entonces
 $\llbracket \mathcal{T} \rrbracket(w) = \{r_1 \dots r_m\}$ es una derivación por la izquierda de \mathcal{G} sobre w .

Algoritmo

Para una gramática $LL(k)$ \mathcal{G} y una palabra $w \in \Sigma^*$:

1. Construya el k -PDT determinista $\mathcal{T}[\mathcal{G}]$ a partir de \mathcal{G} .
2. Ejecute $\mathcal{T}[\mathcal{G}]$ sobre w .

Como $\mathcal{T}[\mathcal{G}]$ es determinista, entonces algoritmo toma **tiempo lineal** en w .

Tabla predictiva para $LL(k)$ fuerte

Sea $\mathcal{G} = (V, \Sigma, P, S)$ una gramática $LL(k)$ fuerte.

Definición

Para cada $u \in \Sigma^k \cup \Sigma^{<k} \cdot \{\#\}$, se define $M[X, u] \in (V \cup \Sigma)^* \cup \{\text{ERROR}\}$:

$$M[X, u] = \begin{cases} \gamma & \text{si } X \rightarrow \gamma \in P \text{ y } u \in \text{first}_k(\gamma) \odot_k \text{follow}_k(X) \\ \text{ERROR} & \text{en otro caso.} \end{cases}$$

Computo de tabla predictiva puede tomar **tiempo exponencial** en $|\mathcal{G}|$ y k .

Caso especial: tabla predictiva para LL(1)

Sea $\mathcal{G} = (V, \Sigma, P, S)$ una gramática LL(1) fuerte.

Definición

Para cada $a \in \Sigma \cup \{\#\}$, se define $M[X, a] \in (V \cup \Sigma)^* \cup \{\text{ERROR}\}$:

$$M[X, a] = \begin{cases} \gamma & \text{si } X \rightarrow \gamma \in P \text{ y } a \in \text{first}_1(\gamma) \\ \gamma & \text{si } X \rightarrow \gamma \in P, \epsilon \in \text{first}_1(\gamma) \text{ y } a \in \text{follow}_1(X) \\ \text{ERROR} & \text{en otro caso.} \end{cases}$$

Este cálculo se puede hacer en tiempo $\mathcal{O}(|V| \cdot |P|)$.

Caso especial: tabla predictiva para LL(1)

Ejemplo de tabla predictiva

$E \rightarrow TE'$	$\text{first}_1(E) = \{ (, \text{id} \}$	$\text{first}_1(E') = \{ +, \epsilon \}$
$E' \rightarrow +TE' \mid \epsilon$	$\text{first}_1(T) = \{ (, \text{id} \}$	$\text{first}_1(T') = \{ *, \epsilon \}$
$T \rightarrow FT'$	$\text{first}_1(F) = \{ (, \text{id} \}$	
$T' \rightarrow *FT' \mid \epsilon$	$\text{follow}_1(E) = \{), \# \}$	$\text{follow}_1(E') = \{), \# \}$
$F \rightarrow (E) \mid \text{id}$	$\text{follow}_1(T) = \{ +,), \# \}$	$\text{follow}_1(T') = \{ +,), \# \}$
	$\text{follow}_1(F) = \{ +, *,), \# \}$	

	id	+	*	()	#
E	<i>TE'</i>	ERROR	ERROR	<i>TE'</i>	ERROR	ERROR
E'	ERROR	<i>+TE'</i>	ERROR	ERROR	ϵ	ϵ
T	<i>FT'</i>	ERROR	ERROR	<i>FT'</i>	ERROR	ERROR
T'	ERROR	ϵ	<i>*FT'</i>	ERROR	ϵ	ϵ
F	id	ERROR	ERROR	<i>(E)</i>	ERROR	ERROR