Parsing con gramáticas LL(k)

Clase 24

IIC 2223

Prof. Cristian Riveros

Definiciones de prefijos (recordatorio)

Definiciones

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

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

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

Definición de $\texttt{first}_k$ y $\texttt{follow}_k$ (recordatorio)

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

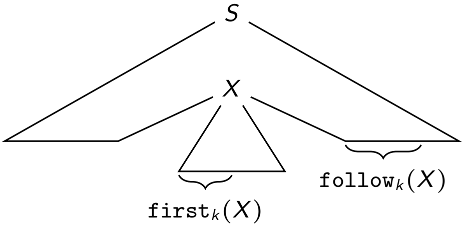
Definiciones

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

$$\texttt{first}_k(\gamma) = \{u|_k \mid \gamma \overset{\star}{\Rightarrow} u\}$$

Se define la función $\texttt{follow}_k : V \to 2^{\Sigma_{\#}^{\leq k}}$ como:

$$\texttt{follow}_k(X) = \{w \mid S \overset{\star}{\Rightarrow} \alpha X \beta \text{ y } w \in \texttt{first}_k(\beta\#)\}$$



Definición gramáticas LL(k) (recordatorio)

Definición

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

- $S \underset{\mathrm{lm}}{\overset{\star}{\Rightarrow}} uY\beta \underset{\mathrm{lm}}{\Rightarrow} u\gamma_1\beta \underset{\mathrm{lm}}{\overset{\star}{\Rightarrow}} uv_1$
- $S \underset{\mathrm{lm}}{\overset{\star}{\Rightarrow}} uY\beta \underset{\mathrm{lm}}{\Rightarrow} u\gamma_2\beta \underset{\mathrm{lm}}{\overset{\star}{\Rightarrow}} uv_2$ y
- $v_1|_k = v_2|_k$

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 $Y \to \gamma_1, Y \to \gamma_2 \in P$ y para todo $S \underset{\mathrm{lm}}{\overset{\star}{\Rightarrow}} uY\beta$, se tiene que:

$$\mathtt{first}_k(\gamma_1\beta) \cap \mathtt{first}_k(\gamma_2\beta) = \varnothing$$

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 $Y \to \gamma_1, Y \to \gamma_2 \in P$ se tiene que:

$$\texttt{first}_k(\gamma_1) \odot_k \texttt{follow}_k(Y) \cap \texttt{first}_k(\gamma_2) \odot_k \texttt{follow}_k(Y) = \varnothing$$

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?

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

Contra-ejemplo

$$\begin{array}{rrcl} \mathcal{G}: & S & \rightarrow & aXaa \mid bXba \\ & X & \rightarrow & b \mid \epsilon \end{array}$$

Recordatorio: $\mathcal{G}$ es LL(k) si para todas dos reglas distintas $Y \to \gamma_1, Y \to \gamma_2 \in P$ y para todo $S \overset{\star}{\underset{\text{lm}}{\Rightarrow}} uY\beta$, se tiene que:

$$\texttt{first}_k(\gamma_1\beta) \cap \texttt{first}_k(\gamma_2\beta) \ = \ \varnothing$$

- Si $S \overset{\star}{\underset{\text{lm}}{\Rightarrow}} aXaa$, entonces $\texttt{first}_2(baa) \cap \texttt{first}_2(aa) = \varnothing$.
- Si $S \overset{\star}{\underset{\text{lm}}{\Rightarrow}} bXba$, entonces $\texttt{first}_2(bba) \cap \texttt{first}_2(ba) = \varnothing$.

Por lo tanto, $\mathcal{G}$ es LL(2).

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

Contra-ejemplo

$$
\begin{array}{llllll}
\mathcal{G}: & S & \rightarrow & aXaa & \mid & bXba \\
 & X & \rightarrow & b & \mid & \epsilon
\end{array}
$$

Recordatorio: $\mathcal{G}$ 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:

$$\texttt{first}_k(\gamma_1) \odot_k \texttt{follow}_k(Y) \;\cap\; \texttt{first}_k(\gamma_2) \odot_k \texttt{follow}_k(Y) \;=\; \varnothing$$

Si vemos $X \rightarrow b$ y $X \rightarrow \epsilon$:

$$
\begin{aligned}
\texttt{first}_2(b) \odot_2 \texttt{follow}_2(X) &\cap \texttt{first}_2(\epsilon) \odot_2 \texttt{follow}_2(X) \\
&= \{b\} \odot_2 \{aa, ba\} \cap \{\epsilon\} \odot_2 \{aa, ba\} \\
&= \{ba, bb\} \cap \{aa, ba\} \\
&= \{ba\}
\end{aligned}
$$

... y $\mathcal{G}$ no es LL(2) fuerte.

Outline

Algunas consideraciones

Parsing de LL(k)

Outline

Algunas consideraciones

Parsing de LL(k)

Problema con gramáticas LL(k)

Definición LL(k)

- $S \overset{\star}{\underset{\text{lm}}{\Rightarrow}} uY\beta \underset{\text{lm}}{\Rightarrow} u\gamma_1\beta \overset{\star}{\underset{\text{lm}}{\Rightarrow}} uv_1$
- $S \overset{\star}{\underset{\text{lm}}{\Rightarrow}} uY\beta \underset{\text{lm}}{\Rightarrow} u\gamma_2\beta \overset{\star}{\underset{\text{lm}}{\Rightarrow}} uv_2$ y
- $v_1|_k = v_2|_k$

entonces $\gamma_1 = \gamma_2$.

Considere la siguiente gramática

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

¿es esta gramática del tipo LL(1)?

Problemas de factorización

Definición LL(k)

- $S \overset{\star}{\underset{lm}{\Rightarrow}} uY\beta \underset{lm}{\Rightarrow} u\gamma_1\beta \overset{\star}{\underset{lm}{\Rightarrow}} uv_1$
- $S \overset{\star}{\underset{lm}{\Rightarrow}} uY\beta \underset{lm}{\Rightarrow} u\gamma_2\beta \overset{\star}{\underset{lm}{\Rightarrow}} uv_2$ y
- $v_1|_k = v_2|_k$

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:

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

Otro problema con gramáticas LL(k)

Definición LL(k)

- $S \overset{\star}{\underset{\text{lm}}{\Rightarrow}} uY\beta \underset{\text{lm}}{\Rightarrow} u\gamma_1\beta \overset{\star}{\underset{\text{lm}}{\Rightarrow}} uv_1$
- $S \overset{\star}{\underset{\text{lm}}{\Rightarrow}} uY\beta \underset{\text{lm}}{\Rightarrow} u\gamma_2\beta \overset{\star}{\underset{\text{lm}}{\Rightarrow}} uv_2$ y
- $v_1|_k = v_2|_k$

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 (recordatorio)

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

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

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 $k \geq 1$.

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

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 $X \to X\beta \in P$ y $X \to w \in P$.

Como $\mathcal{G}$ es reducida, entonces existe una derivación $S \underset{\text{lm}}{\overset{\star}{\Rightarrow}} uX\gamma$.

$$S \underset{\text{lm}}{\overset{\star}{\Rightarrow}} uX\gamma \underset{\text{lm}}{\Rightarrow} \overset{n\text{-veces}}{\cdots} \underset{\text{lm}}{\Rightarrow} uX\beta^n\gamma$$

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

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

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

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

$\rightarrow\leftarrow$

Eliminación de recursión por la izquierda (recordatorio)

Eliminación de recursión inmediata

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

$$X \;\to\; X\alpha_1 \mid \cdots \mid X\alpha_m \mid \beta_1 \mid \cdots \mid \beta_n$$

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

$$\begin{array}{rcl} X & \to & \beta_1 X' \mid \cdots \mid \beta_n X' \\ X' & \to & \alpha_1 X' \mid \cdots \mid \alpha_m X' \mid \epsilon \end{array}$$

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

Eliminación de recursión NO-inmediata

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

$$\text{si } \; X_i \to X_j\alpha, \; \text{ entonces } \; i < j.$$

Eliminación de recursión por la izquierda (recordatorio)

Ejemplo

$$\begin{array}{rcl} E & \rightarrow & E+T \mid T \\ T & \rightarrow & T*F \mid F \\ F & \rightarrow & (E) \mid \mathsf{n} \end{array}$$

Eliminando la **recursión inmediata** de E:

$$\begin{array}{rcl} E & \rightarrow & TE' \\ E' & \rightarrow & +TE' \mid \epsilon \\ T & \rightarrow & T*F \mid F \\ F & \rightarrow & (E) \mid \mathsf{n} \end{array}$$

Eliminando la **recursión inmediata** de T:

$$\begin{array}{rcl} E & \rightarrow & TE' \\ E' & \rightarrow & +TE' \mid \epsilon \\ T & \rightarrow & FT' \\ T' & \rightarrow & *FT' \mid \epsilon \\ F & \rightarrow & (E) \mid \mathsf{n} \end{array}$$

Problema con recursión por la izquierda (conclusión)

Definición (recordatorio)

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

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

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

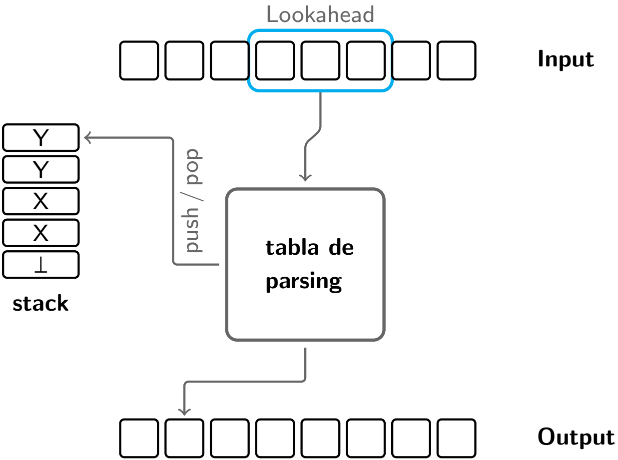
Es **posible eliminar** la recursividad por la izquierda, pero
esto **NO asegura** que el resultado sea una gramática LL(k) para algún k.

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:

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

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:

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

- 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 $[\![\mathcal{T}]\!] : \Sigma^* \to 2^{\Omega^*}$:

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

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:

$$\alpha_1 \text{ NO es prefijo de } \alpha_2 \quad \text{o} \quad u_1 v_1 \text{ NO es prefijo de } u_2 v_2.$$

"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 $\mathcal{G}$:

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

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 $p := (X \to \gamma) \in P$ tal que $u \in \mathtt{first}_k(\gamma) \odot_k \mathtt{follow}_k(X)$

Parser k-PDT para gramática LL(k) fuerte

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

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 $p := (X \to \gamma) \in P$ tal que $u \in \mathtt{first}_k(\gamma) \odot_k \mathtt{follow}_k(X)$

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 $[\![\mathcal{T}]\!](w) = \varnothing$.
3. si $w \in \mathcal{L}(\mathcal{G})$ entonces
   $[\![\mathcal{T}]\!](w) = \{r_1 \dots r_m\}$ 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 $[\![\mathcal{T}]\!](w) = \emptyset$.
3. si $w \in \mathcal{L}(\mathcal{G})$ entonces
   $[\![\mathcal{T}]\!](w) = \{r_1 \dots r_m\}$ 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 $M[X, u] \in (V \cup \Sigma)^* \cup \{\mathtt{ERROR}\}$:

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

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 $M[X, a] \in (V \cup \Sigma)^* \cup \{\texttt{ERROR}\}$:

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

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

Caso especial: tabla predictiva para LL(1)

Ejemplo de tabla predictiva

$$
\begin{array}{lcl}
E & \rightarrow & TE' \\
E' & \rightarrow & +TE' \mid \epsilon \\
T & \rightarrow & FT' \\
T' & \rightarrow & *FT' \mid \epsilon \\
F & \rightarrow & (E) \mid \mathbf{id}
\end{array}
$$

$\mathtt{first}_1(E) = \{(, \mathbf{id}\}$ $\quad$ $\mathtt{first}_1(E') = \{+, \epsilon\}$

$\mathtt{first}_1(T) = \{(, \mathbf{id}\}$ $\quad$ $\mathtt{first}_1(T') = \{*, \epsilon\}$

$\mathtt{first}_1(F) = \{(, \mathbf{id}\}$

$\mathtt{follow}_1(E) = \{), \#\}$ $\quad$ $\mathtt{follow}_1(E') = \{), \#\}$

$\mathtt{follow}_1(T) = \{+,), \#\}$ $\quad$ $\mathtt{follow}_1(T') = \{+,), \#\}$

$\mathtt{follow}_1(F) = \{+, *,), \#\}$

	id	+	*	(	)	#
E	TE'	ERROR	ERROR	TE'	ERROR	ERROR
E'	ERROR	$+TE'$	ERROR	ERROR	ϵ	ϵ
T	FT'	ERROR	ERROR	FT'	ERROR	ERROR
T'	ERROR	ϵ	$*FT'$	ERROR	ϵ	ϵ
F	**id**	ERROR	ERROR	(E)	ERROR	ERROR