

Parsing: cómputo de First y Follow

Clase 28

IIC2223 / IIC2224

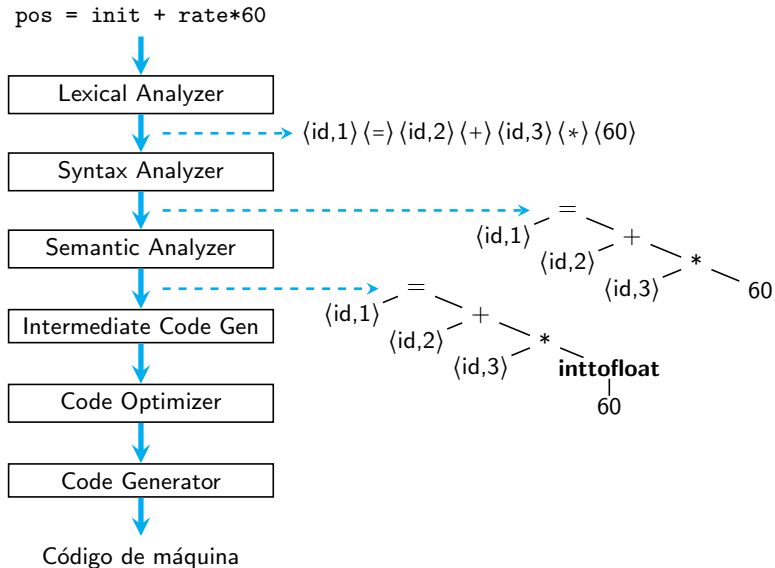
Amaranta Salas

Sintaxis y semántica de un lenguaje (recordatorio)

Definición

1. La **sintaxis** de un lenguaje es un conjunto de reglas que describen los programas válidos que tienen significado.
2. La **semántica** de un lenguaje define el significado de un programa correcto según la sintaxis.

La estructura de un compilador (recordatorio)



Verificación de sintaxis

En este proceso se busca:

- verificar la sintaxis de un programa.
- entregar la estructura de un programa (árbol de parsing).

Consta de tres etapas:

1. Análisis léxico (**Lexer**). ✓
2. Análisis sintáctico (**Parser**).
3. Análisis semántico.

Ahora veremos como hacer el **Parser**.

Análisis sintáctico (Parser)

Informalmente

“Dado una secuencia de tokens w' y una gramática \mathcal{G} construir un árbol de derivación (parsing) de \mathcal{G} para w .”

Con el **árbol de derivación** habremos:

- verificado la sintaxis.
- obtenido la estructura.

Análisis sintáctico (Parser)

Ejemplo de gramática

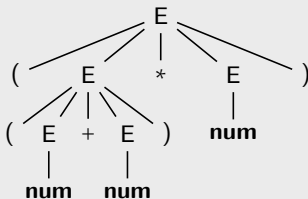
$$E \rightarrow (E + E) \mid (E * E) \mid \text{num}$$

Para un input $w = ((43 + 56) * 27)$:

- Convertimos w en una secuencia de **tokens**:

$$w' = ((\text{num} + \text{num}) * \text{num})$$

- Construimos un árbol de **parsing** para w' :



Análisis sintáctico (Parser)

Problema de parsing

Dado una palabra w y dado una gramática \mathcal{G} ,
generar un árbol de parsing \mathcal{T} de \mathcal{G} para w .

¿ya sabemos resolver este problema?

El algoritmo CKY nos permite hacer esto, pero ...

- impracticable para grandes inputs.
- multiples pasadas sobre el input.

Deseamos hacer parsing en **tiempo lineal** en el tamaño del input.

Autómatas apiladores al rescate (recordatorio)

Para una gramática $\mathcal{G} = (V, \Sigma, P, S)$

podemos construir un PDA alternativo \mathcal{D} que acepta $\mathcal{L}(\mathcal{G})$:

$$\mathcal{D} = (V \cup \Sigma \cup \{q_0, q_f\}, \Sigma, \Delta, q_0, \{q_f\})$$

La relación de transición Δ se define como:

$$\begin{aligned} \Delta = & \{ (q_0, \epsilon, S \cdot q_f) \} && \cup \\ & \{ (X, \epsilon, \gamma) \mid X \rightarrow \gamma \in P \} && \cup \text{ (Expandir)} \\ & \{ (a, a, \epsilon) \mid a \in \Sigma \} && \text{ (Reducir)} \end{aligned}$$

¿cómo usamos este PDA para hacer parsing?

Problema: muchas alternativas para **expandir**.

¿cómo elegir la siguiente producción para expandir?

$$X \rightarrow \alpha \mid \beta$$

¿cómo elegir entre α o β ?

Queremos elegir la **próxima producción** $X \rightarrow \gamma$ de tal manera que, si existe una derivación para el input, entonces $X \rightarrow \gamma$ **es parte de esa derivación**:

$$\text{si } S \xRightarrow[\text{lm}]{*} uX\gamma' \xRightarrow[\text{lm}]{*} uv, \text{ entonces } \gamma\gamma' \xRightarrow[\text{lm}]{*} v$$

Necesitamos **mirar las siguientes letras en v** y ver si pueden ser producidas por α o β .

...para esto ocuparemos first y follow.

Outline

Prefijos

First y Follow

Calcular First

Calcular Follow

Outline

Prefijos

First y Follow

Calcular First

Calcular Follow

Definiciones de prefijos

Definiciones

Sea Σ un alfabeto finito. Para un $k \geq 0$, se define:

$$\Sigma^{\leq k} = \bigcup_{i=0}^k \Sigma^i$$

$$\Sigma_{\#}^{\leq k} = \Sigma^{\leq k} \cup (\Sigma^{\leq k-1} \cdot \{\#\})$$

Ejemplos

Para $\Sigma = \{a, b\}$:

- $\Sigma^{\leq 2} = \{\epsilon, a, b, aa, ab, ba, bb\}$
- $\Sigma_{\#}^{\leq 2} = \{\epsilon, a, b, aa, ab, ba, bb\} \cup \{\#, a\#, b\#\}$

Símbolo $\#$ representará un EOF (End of File),
marcando el **fin de una palabra**.

Definiciones de prefijos

Definiciones

Sea Σ un alfabeto finito. Para un $k \geq 0$, se define:

$$\Sigma^{\leq k} = \bigcup_{i=0}^k \Sigma^i$$

$$\Sigma_{\#}^{\leq k} = \Sigma^{\leq k} \cup (\Sigma^{\leq k-1} \cdot \{\#\})$$

Para una palabra $w = a_1 a_2 \dots a_n \in \Sigma^*$ se define el **k -prefijo** de w como:

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

Definimos la **k -concatenación** \odot_k entre strings $u, v \in \Sigma$ como:

$$u \odot_k v = (u \cdot v)|_k$$

Definiciones de prefijos

Definiciones

Para una palabra $w = a_1 a_2 \dots a_n \in \Sigma^*$ se define el **k -prefijo** de w como:

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

Definimos la **k -concatenación** \odot_k entre strings $u, v \in \Sigma$ como:

$$u \odot_k v = (u \cdot v)|_k$$

Ejemplos

- $(abaa)|_2 = ab$ $(ab)|_2 = ab$ $(a)|_2 = a$ $(\epsilon)|_2 = \epsilon$
- $a \odot_2 baa = (abaa)|_2 = ab$
- $bba \odot_2 a = (bbaa)|_2 = bb$
- $b \odot_2 \epsilon = (b)|_2 = b$

Definiciones de prefijos

Definiciones

Para una palabra $w = a_1 a_2 \dots a_n \in \Sigma^*$ se define el **k -prefijo** de w como:

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

Definimos la **k -concatenación** \odot_k entre strings $u, v \in \Sigma$ como:

$$u \odot_k v = (u \cdot v)|_k$$

Extendemos estas operaciones para lenguajes $L, L_1, L_2 \subseteq \Sigma^*$ como:

$$L|_k = \{w|_k \mid w \in L\}$$

$$L_1 \odot_k L_2 = \{w_1 \odot_k w_2 \mid w_1 \in L_1 \text{ y } w_2 \in L_2\}$$

Definiciones de prefijos

Definiciones

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

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

Ejemplos

- $((ab)^*)|_3 = \{\epsilon, ab, aba\}$
- $(a)^* \odot_3 (ab)^* = \{\epsilon, a, aa, aaa, ab, aba, aab\}$

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

Algunas propiedades de k -prefijos

Definiciones

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

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

Propiedades

Para todo $k \geq 1$ y $L_1, L_2, L_3 \subseteq \Sigma^*$:

1. $L_1 \odot_k (L_2 \odot_k L_3) = (L_1 \odot_k L_2) \odot_k L_3$
2. $L_1 \odot_k \{\epsilon\} = \{\epsilon\} \odot_k L_1 = L_1|_k$
3. $(L_1 L_2)|_k = L_1|_k \odot_k L_2|_k$
4. $L_1 \odot_k (L_2 \cup L_3) = (L_1 \odot_k L_2) \cup (L_1 \odot_k L_3)$

Demostración: ejercicio.

Outline

Prefijos

First y Follow

Calcular First

Calcular Follow

Definición de first_k y follow_k

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 $\gamma \in (V \cup \Sigma)^*$:

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

Ejemplos

$$E \rightarrow (E + E) \mid (E * E) \mid \mathbf{n}$$

- $\text{first}_1(E) = \{ (, \mathbf{n} \}$
- $\text{first}_2(E) = \{ \mathbf{n}, (\mathbf{n}, (\}$
- $\text{first}_3(E) = \{ \mathbf{n}, (\mathbf{n}+, (\mathbf{n}*, ((\mathbf{n}, (((\}$

Definición de first_k y follow_k

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 de first_k y follow_k

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

Definiciones

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

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

Ejemplos

$$E \rightarrow (E + E) \mid (E * E) \mid n$$

■ $\text{follow}_1(E) = \{ \#, +, *,) \}$

■ $\text{follow}_2(E) = \{ \#,)\#,)) ,)+ ,)*, +(, *(, +n , *n \}$

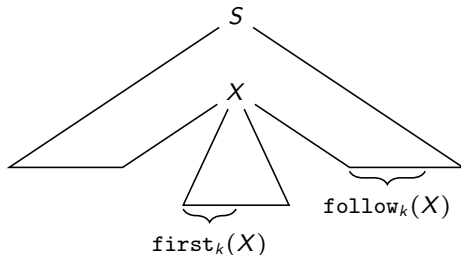
Definición de first_k y follow_k

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

Definiciones

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

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



Outline

Prefijos

First y Follow

Calcular First

Calcular Follow

Propiedades de first_k

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

Proposición

Para $X_1, \dots, X_n \in V \cup \Sigma$:


$$\text{first}_k(X_1 \dots X_n) = \text{first}_k(X_1) \odot_k \dots \odot_k \text{first}_k(X_n)$$

Demostración

Defina $\mathcal{L}(X) = \{w \mid X \xRightarrow{*} w\}$ y $\mathcal{L}(\gamma) = \{w \mid \gamma \xRightarrow{*} w\}$.

Notar que $\text{first}_k(\gamma) = \mathcal{L}(\gamma)|_k$.

Por lo tanto, tenemos que:

$$\begin{aligned}\text{first}_k(X_1 \dots X_n) &= \mathcal{L}(X_1 \dots X_n)|_k \\ &= (\mathcal{L}(X_1) \cdot \mathcal{L}(X_2) \cdot \dots \cdot \mathcal{L}(X_n))|_k \\ &= \mathcal{L}(X_1)|_k \odot_k \mathcal{L}(X_2)|_k \odot_k \dots \odot_k \mathcal{L}(X_n)|_k \\ &= \text{first}_k(X_1) \odot_k \dots \odot_k \text{first}_k(X_n)\end{aligned}$$


Propiedades de first_k

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

Proposición

Para $X_1, \dots, X_n \in V \cup \Sigma$:

$$\text{first}_k(X_1 \dots X_n) = \text{first}_k(X_1) \odot_k \dots \odot_k \text{first}_k(X_n)$$

En particular, tenemos que:

$$\text{first}_k(X) = \bigcup_{X \rightarrow X_1 \dots X_n \in P} \text{first}_k(X_1) \odot_k \dots \odot_k \text{first}_k(X_n)$$

Propiedades de first_k

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

En particular, tenemos que:

$$\text{first}_k(X) = \bigcup_{X \rightarrow X_1 \dots X_n \in P} \text{first}_k(X_1) \odot_k \dots \odot_k \text{first}_k(X_n)$$

Defina el siguiente **programa recursivo** para todo $X \in V \cup \Sigma$:

$$\begin{aligned} \text{first}_k^0(X) &:= \bigcup_{X \rightarrow w \in P} w|_k \\ \text{first}_k^i(X) &:= \bigcup_{X \rightarrow X_1 \dots X_n \in P} \text{first}_k^{i-1}(X_1) \odot_k \dots \odot_k \text{first}_k^{i-1}(X_n) \end{aligned}$$

Es fácil ver que:

- $\text{first}_k^{i-1}(X) \subseteq \text{first}_k^i(X)$ para todo $i > 1$.
- Como $\text{first}_k(X) \subseteq \Sigma^{\leq k}$, entonces para algún $i \leq k \cdot |\Sigma|^k \cdot |V|$ tendremos:

$$\text{first}_k^j(X) = \text{first}_k^{j+1}(X) \text{ para todo } j \geq i$$

¿cómo calcular first_k ?

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

Defina el siguiente **programa recursivo** para todo $X \in V \cup \Sigma$:

$$\begin{aligned}\text{first}_k^0(X) &:= \bigcup_{X \rightarrow w \in P} w|_k \\ \text{first}_k^i(X) &:= \bigcup_{X \rightarrow X_1 \dots X_n \in P} \text{first}_k^{i-1}(X_1) \odot_k \dots \odot_k \text{first}_k^{i-1}(X_n)\end{aligned}$$

Teorema

Sea i^* el menor número tal que $\text{first}_k^{i^*}(X) = \text{first}_k^{i^*+1}(X)$ para todo $X \in V$. Entonces para todo $X \in V$:

$$\text{first}_k^{i^*}(X) = \text{first}_k(X)$$

Demostración (idea)

- \subseteq Por inducción, demostrar que $\text{first}_k^i(X) \subseteq \text{first}_k(X)$.
- \supseteq Por inducción, si $X \xRightarrow{*} w$, entonces $w|_k \in \text{first}_k^i(X)$ para algún i .

¿cómo calcular first_k ?

input : Gramática $\mathcal{G} = (V, \Sigma, P, S)$ y $k \geq 1$

output: Todos los conjuntos $\text{first}_k(X)$ para todo $X \in V \cup \Sigma$.

Function CalcularFIRST (\mathcal{G}, k)

foreach $a \in \Sigma$ **do**

└ $\text{first}_k^0(a) := \{a\}$

foreach $X \in V$ **do**

└ $\text{first}_k^0(X) := \bigcup_{X \rightarrow w \in P} w|_k$

$i := 0$

repeat

└ $i := i + 1$

foreach $a \in \Sigma$ **do**

└ $\text{first}_k^i(a) := \{a\}$

foreach $X \in V$ **do**

└ $\text{first}_k^i(X) := \bigcup_{X \rightarrow X_1 \dots X_n \in P} \text{first}_k^{i-1}(X_1) \odot_k \dots \odot_k \text{first}_k^{i-1}(X_n)$

until $\text{first}_k^i(X) = \text{first}_k^{i-1}(X)$ para todo $X \in V \cup \Sigma$

return $\{\text{first}_k(X)\}_{X \in V \cup \Sigma}$

Outline

Prefijos

First y Follow

Calcular First

Calcular Follow

Propiedades de follow_k

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

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

Si consideramos $X \neq S$:

$$\begin{aligned} \text{follow}_k(X) &= \bigcup_{S \xRightarrow{*} \alpha X \beta} \text{first}_k(\beta\#) \\ &= \bigcup_{S \xRightarrow{*} \alpha Y \beta \Rightarrow \alpha \alpha' X \beta' \beta} \text{first}_k(\beta' \beta\#) \\ &= \bigcup_{Y \rightarrow \alpha' X \beta'} \bigcup_{S \xRightarrow{*} \alpha Y \beta} \text{first}_k(\beta' \beta\#) \\ &= \bigcup_{Y \rightarrow \alpha' X \beta'} \bigcup_{S \xRightarrow{*} \alpha Y \beta} \text{first}_k(\beta') \odot_k \text{first}_k(\beta\#) \\ &= \bigcup_{Y \rightarrow \alpha' X \beta'} \text{first}_k(\beta') \odot_k \bigcup_{S \xRightarrow{*} \alpha Y \beta} \text{first}_k(\beta\#) \\ &= \bigcup_{Y \rightarrow \alpha' X \beta'} \text{first}_k(\beta') \odot_k \text{follow}_k(Y) \end{aligned}$$

Propiedades de follow_k

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

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

Si consideramos $X \neq S$:

$$\begin{aligned}\text{follow}_k(X) &= \bigcup_{S \xRightarrow{*} \alpha X \beta} \text{first}_k(\beta\#) \\ &\vdots \\ &= \bigcup_{Y \rightarrow \alpha' X \beta'} \text{first}_k(\beta') \odot_k \text{follow}_k(Y)\end{aligned}$$

Si consideramos $X = S$:

$$\begin{aligned}\text{follow}_k(S) &= \{\#\} \cup \bigcup_{S \xRightarrow{+} \alpha S \beta} \text{first}_k(\beta\#) \\ &= \{\#\} \cup \bigcup_{S \xRightarrow{*} \alpha Y \beta \Rightarrow \alpha \alpha' S \beta' \beta} \text{first}_k(\beta' \beta\#) \\ &= \{\#\} \cup \bigcup_{Y \rightarrow \alpha' S \beta'} \text{first}_k(\beta') \odot_k \text{follow}_k(Y)\end{aligned}$$

Propiedades de follow_k

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

Teorema

$$\text{Para } X \neq S: \quad \text{follow}_k(X) = \bigcup_{Y \rightarrow \alpha X \beta} \text{first}_k(\beta) \odot_k \text{follow}_k(Y)$$

$$\text{follow}_k(S) = \{\#\} \cup \bigcup_{Y \rightarrow \alpha S \beta} \text{first}_k(\beta) \odot_k \text{follow}_k(Y)$$

Defina el siguiente **programa recursivo** para todo $X \in V$:

$$\text{Para } X \neq S: \quad \text{follow}_k^0(X) := \emptyset$$

$$\text{follow}_k^0(S) := \{\#\}$$

$$\text{Para } X \neq S: \quad \text{follow}_k^i(X) := \bigcup_{Y \rightarrow \alpha X \beta} \text{first}_k(\beta) \odot_k \text{follow}_k^{i-1}(Y)$$

$$\text{follow}_k^i(S) := \{\#\} \cup \bigcup_{Y \rightarrow \alpha S \beta} \text{first}_k(\beta) \odot_k \text{follow}_k^{i-1}(Y)$$

¿cómo calcular follow_k ?

Similar al caso de first_k :

- $\text{follow}_k^{i-1}(X) \subseteq \text{follow}_k^i(X)$ para todo $i > 1$.
- Como $\text{follow}_k(X) \subseteq \Sigma^{\leq k}$, entonces para algún $i \leq k \cdot |\Sigma|^k \cdot |V|$:

$$\text{follow}_k^j(X) = \text{follow}_k^{j+1}(X) \quad \text{para todo } j \geq i.$$

Teorema

Sea i^* el menor número tal que $\text{follow}_k^{i^*}(X) = \text{follow}_k^{i^*+1}(X)$ para todo $X \in V$. Entonces para todo $X \in V$:

$$\text{follow}_k^{i^*}(X) = \text{follow}_k(X)$$

Demostración: ejercicio.

...y podemos calcular $\text{follow}_k(X)$
con un algoritmo similar que $\text{first}_k(X)$.

¿cómo calculamos first_k y follow_k eficientemente?

- Algoritmos toman $\mathcal{O}(k \cdot |\Sigma|^k \cdot |V|)$ repeticiones en el peor caso.
- Si $k = 1$, el número de repeticiones será $\mathcal{O}(|\Sigma| \cdot |V|)$ y tiempo del algoritmo será polinomial en $|\mathcal{G}|$ en el peor caso.
- Para $k = 1$ incluso se puede hacer en tiempo $\mathcal{O}(|V| \cdot |P|)$ en total.