Gramáticas Libres de Contexto

Contenido

Gramáticas Libres de Contexto

- Hemos visto que muchos lenguajes no son regulares. Por lo que necesitamos una clase más grande de lenguages
- Las Gramáticas Libres de Contexto (*Context-Free Languages*) o CFL's jugaron un papel central en lenguaje natural desde los 50's y en los compiladores desde los 60's
- Las Gramáticas Libres de Contexto forman la base de la sintáxis BNF
- Son actualmente importantes para XML y sus DTD's (*document type definition*)

Gramáticas Libres de Contexto

- Vamos a ver los CFG's, los lenguajes que generan, los árboles de parseo, el *pushdown automata* y las propiedades de cerradura de los CFL's.
- **Ejemplo:** Considere $L_{pal} = \{w \in \Sigma^* : w = w^R\}$. Por ejemplo, *oso* $\in L_{pal}$, *anitalavalatina* $\in L_{pal}$,
- Sea $\Sigma = \{0, 1\}$ y supongamos que L_{pal} es regular.
- Sea n dada por el *pumping lemma*. Entonces $0^n10^n \in L_{pal}$. Al leer 0^n el FA debe de entrar a un ciclo. Si quitamos el ciclo entonces llegamos a una contradicción.

Palíndromes

- Definamos L_{pal} de forma inductiva.
- *Base*: $\epsilon, 0$ y 1 son palíndromes.
- *Inducción*: Si w es un palíndrome, también $0w0$ y $1w1$.
- Ninguna otra cosa es palíndrome.

Palíndromes

- Las CFG's son un mecanismo formal para la definición como la de palíndrome.

  1. $P \rightarrow \epsilon$
  2. $P \rightarrow 0$
  3. $P \rightarrow 1$
  4. $P \rightarrow 0P0$
  5. $P \rightarrow 1P1$

  donde 0 y 1 son símbolos terminales.

Gramáticas Libre de Contexto

- P es una variable o símbolo no terminal o categoría sintáctica.
- P es en esta gramática también el símbolo inicial.
- $1-5$ son producciones o reglas. La variable definida (parcialmente) en la producción también se llama la *cabeza* de la producción y la cadena de cero, 1 o más símbolos terminales o variables a la derecha de la producción se llama el *cuerpo* de la producción.

Definición formal de CFGs

Una grámatica libre de contexto se define con $G = (V, T, P, S)$ donde:

- V es un conjunto de *variables*
- T es un conjunto de *terminales*
- P es un conjunto finito de *producciones* de la forma $A \rightarrow \alpha$, donde A es una variables y $\alpha \in (V \cup T)^*$
- S es una variable designada llamada el *símbolo inicial*

Ejemplos

- **Ejemplo:** $G_{pal} = (\{P\}, \{0,1\}, A, P)$, donde $A = \{P \rightarrow \epsilon, P \rightarrow 0, P \rightarrow 1, P \rightarrow 0P0, P \rightarrow 1P1\}$.
- Muchas veces se agrupan las producciones con la misma cabeza, e.g., $A = \{P \rightarrow \epsilon|0|1|0P0|1P1\}$.
- **Ejemplo:** Expresiones regulares sobre $\{0,1\}$ se pueden definir por la gramática: $G_{regex} = (\{E\}, \{0,1\}, A, E)$, donde $A = \{E \rightarrow 0, E \rightarrow 1, E \rightarrow E.E, E \rightarrow E+E, E \rightarrow E^*, E \rightarrow (E)\}$.

Ejemplo

- Expresiones en un lenguaje de programación donde los operadores son $+$ y $*$, y los argumentos son identificadores (*strings*) que empiezan con a o b en $L((a+b)(a+b+0+1)^*)$.
- Las expresiones se definen por la gramática: $G = (\{E, I\}, T, P, E)$ donde $T = \{+, *.(,), a, b, 0, 1\}$ y P es el siguiente conjunto de producciones:

$1) E \rightarrow I$	$2) E \rightarrow E + E$
$3) E \rightarrow E * E$	$4) E \rightarrow (E)$
$5) I \rightarrow a$	$6) I \rightarrow b$
$7) I \rightarrow Ia$	$8) I \rightarrow Ib$
$9) I \rightarrow I0$	$10) I \rightarrow I1$

Derivaciones usando gramáticas

- *Inferencia recursiva* usando las producciones del cuerpo a la cabeza para reconocer si una cadena está en el lenguaje definido por la gramática. Ejemplo de una inferencia recursiva de la cadena: $a * (a + b00)$.

Cadenas	Cabeza	Del Leng. de:	Cadenas usadas
(i) a	I	5	$-$
(ii) b	I	6	$-$
(iii) $b0$	I	9	(ii)
(iv) $b00$	I	9	(iii)
(v) a	E	1	(i)
(vi) $b00$	E	1	(iv)
(vii) $a + b00$	E	2	(v),(vi)
(viii) $(a + b00)$	E	4	(vii)
(ix) $a * (a + b00)$	E	3	(v),(viii)

Derivaciones

- Derivaciones usando las producciones de la cabeza al cuerpo. Con esto derivamos cadenas que pertenecen a la gramática.
- Para esto introducimos un nuevo símbolo: $\Rightarrow$
- Sea $G = (V, T, P, S)$ una CFG, $A \in V, \{\alpha, \beta\} \subset (V \cup T)^*$ y $A \rightarrow \gamma \in P$.
- Entonces, escribimos: $\alpha A \beta \Rightarrow_G \alpha\gamma\beta$ o si se sobre-entiende G: $\alpha A \beta \Rightarrow \alpha\gamma\beta$ y decimos que $\alpha A \beta$ *deriva* $\alpha\gamma\beta$.

Cerrada de $\Rightarrow$

- Definimos $\overset{*}{\Rightarrow}$ como la cerradura reflexiva y transitiva de $\Rightarrow$. Lo que quiere decir es que usamos uno a más pasos de derivación.
- Ideas:
  - *Base*: Sea $\alpha \in (V \cup T)^*$, entronces $\alpha \overset{*}{\Rightarrow} \alpha$ (osea que cada cadena se deriva a sí misma).
  - *Inducción*: Si $\alpha \overset{*}{\Rightarrow} \beta$, y $\beta \Rightarrow \gamma$, entonces $\alpha \overset{*}{\Rightarrow} \gamma$

Ejemplo

- La derivación de $a * (a + b00)$ a partir de E en la gramática anterior sería:
  $E \Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow a * (E) \Rightarrow a * (E + E) \Rightarrow a * (I + E) \Rightarrow a * (a + E) \Rightarrow a * (a + I) \Rightarrow a * (a + I0) \Rightarrow a * (a + I00) \Rightarrow a * (a + b00)$
- Podemos abreviar y simplemente poner:
  $E \stackrel{*}{\Rightarrow} a * (a + b00)$

Derivación e Inferencia

- La derivación y la inferencia recursiva son equivalentes, osea que si podemos inferir que una cadena de símbolos terminales w está en el lenguaje de una variable A entonces $A \stackrel{*}{\Rightarrow} w$ y al revés.
- **Nota 1**: en cada paso podemos tener varios reglas de las cuales escoger, e.g.: $I * E \Rightarrow a * E \Rightarrow a * (E)$ o $I * E \Rightarrow I * (E) \Rightarrow a * (E)$
- **Nota 2**: no todas las opciones nos llevan a derivaciones exitosas de una cadena en particular, por ejemplo: $E \Rightarrow E + E$ no nos lleva a la derivación de $a * (a + b00)$.

Derivaciones más a la izquierda y más a la derecha

Para restringir el número de opciones para derivar una cadena.

- Derivación más a la izquierda (*leftmost derivation*): $\Rightarrow_{lm}$ siempre reemplaza la variable más a la izquierda por uno de los cuerpos de sus producciones.
- Derivación más a la derecha (*rightmost derivation*): $\Rightarrow_{rm}$ siempre reemplaza la variable más a la derecha por uno de los cuerpos de sus producciones.

Ejemplo

La derivación anterior la podemos hacer como derivación más a la izquierda:

$E \Rightarrow_{lm} E * E \Rightarrow_{lm} I * E \Rightarrow_{lm} a * E \Rightarrow_{lm} a * (E) \Rightarrow_{lm}$
$a * (E + E) \Rightarrow_{lm} a * (I + E) \Rightarrow_{lm} a * (a + E) \Rightarrow_{lm}$
$a * (a + I) \Rightarrow_{lm} a * (a + I0) \Rightarrow_{lm} a * (a + I00) \Rightarrow_{lm} a * (a + b00)$
o simplemente $E \stackrel{*}{\Rightarrow}_{lm} a * (a + b00)$

Ejemplo (cont.)

Por otro lado, también la podemos hacer más a la derecha:

$E \Rightarrow_{rm} E*E \Rightarrow_{rm} E*(E) \Rightarrow_{rm} E*(E+E) \Rightarrow_{rm} E*(E+I) \Rightarrow_{rm} E*(E+I0) \Rightarrow_{rm} E*(E+I00) \Rightarrow_{rm} E*(E+b00) \Rightarrow_{rm} E*(I+b00) \Rightarrow_{rm} E*(a+b00) \Rightarrow_{rm} I*(a+b00) \Rightarrow_{rm} a*(a+b00)$
o simplemente $E \overset{*}{\Rightarrow}_{rm} a*(a+b00)$.

Equivalencias

- Cualquier derivación tiene una derivación equivalente más a la izquierda y una más a la derecha.
- Si w es una cadena de símbolos terminales y A es una variable, $A \stackrel{*}{\Rightarrow} w$ si y solo si $A \Rightarrow^*_{lm} w$ y si y solo si $A \Rightarrow^*_{rm} w$.

El Lenguaje de la Gramática

- Si $G(V, T, P, S)$ es una CFG, entonces el lenguaje de G es: $L(G) = \{w \in T^* : S \overset{*}{\Rightarrow}_G w\}$, osea el conjunto de cadenas sobre T^* derivadas del símbolo inicial.
- Si G es una CFG al $L(G)$ se llama *lenguaje libre de contexto*. Por ejemplo, $L(G_{pal})$ es un lenguaje libre de contexto.
- **Teorema:** $L(G_{pal}) = \{w \in \{0, 1\}^* : w = w^R\}$
- **Prueba:** ($\Rightarrow$) Suponemos $w = w^R$. Mostramos por inducción en $|w|$ que $w \in L(G_{pal})$.

Prueba:

- *Base*: $|w| = 0$ or $|w| = 1$. Entonces w es $\epsilon, 0$ o 1. Como $P \rightarrow \epsilon, P \rightarrow 1$ y $P \rightarrow 0$ son producciones, concluimos que $P \stackrel{*}{\Rightarrow}_G w$ en todos los casos base.
- *Inducción*: Suponemos $|w| \geq 2$. Como $w = w^R$, tenemos que $w = 0x0$ o $w = 1x1$ y que $x = x^R$.
- Si $w = 0x0$ sabemos de la hipótesis inductiva que $P \stackrel{*}{\Rightarrow} x$, entonces $P \Rightarrow 0P0 \stackrel{*}{\Rightarrow} 0x0 = w$, entonces $w \in L(G_{pal})$.
- El caso para $w = 1x1$ es similar.

Palíndrome

- $(\Leftarrow)$: Supopnemos que $w \in L(G_{pal})$ y tenemos que mostrar que $w = w^R$.
- Como $w \in L(G_{pal})$, tenemos que $P \stackrel{*}{\Rightarrow} w$. Lo que hacemos es inducción sobre la longitud de $\stackrel{*}{\Rightarrow}$.
- *Base*: La derivación de $\stackrel{*}{\Rightarrow}$ se hace en un solo paso, por lo que w debe de ser ϵ, 0 o 1, todos palíndromes.

Palíndrome

- *Inducción*: Sea $n \geq 1$ y suponemos que la derivación toma $n+1$ pasos y que el enunciado es verdadero para n pasos. Osea, si $P \stackrel{*}{\Rightarrow} x$ en n pasos, x es palíndrome. Por lo que debemos de tener para $n+1$ pasos:
- $w = 0x0 \stackrel{*}{\Leftarrow} 0P0 \Leftarrow P$ o $w = 1x1 \stackrel{*}{\Leftarrow} 1P1 \Leftarrow P$ donde la segunda derivación toma n pasos.
- Por la hipótesis inductiva, x es un palíndrome, por lo que se completa la prueba.

Sentential Forms

- Sea $G = (V, T, P, S)$ una CFG y $\alpha \in (V \cup T)^*$. Si $S \overset{*}{\Rightarrow} \alpha$ decimos que α está en forma de sentencia (*sentential form*)
- Si $S \Rightarrow_{lm} \alpha$ decimos que α es una forma de sentencia izquierda (*left-sentencial form*), y si $S \Rightarrow_{rm} \alpha$ decimos que α es una forma de sentencia derecha (*right-sentencial form*).
- $L(G)$ son las formas de sentencia que estan en T^*.

Ejemplo

- Si tomamos la gramática del lenguaje sencillo que definimos anteriormente, $E * (I + E)$ es una forma de sentencia ya que: $E \Rightarrow E * E \Rightarrow E * (E) \Rightarrow E * (E + E) \Rightarrow E * (I + E)$.
- Esta derivación no es ni más a la izquierda ni más a la derecha.
- Por otro lado: $a * E$ es una forma de sentencia izquierda, ya que: $E \Rightarrow_{lm} E * E \Rightarrow_{lm} I * E \Rightarrow_{lm} a * E$ y $E * (E + E)$ es una forma de sentencia derecha, ya que: $E \Rightarrow_{rm} E * E \Rightarrow_{rm} E * (E) \Rightarrow_{rm} E * (E + E)$

Ejemplos

- Defina un CFG que acepte el siguiente lenguaje: $\{0^n 1^n | n \geq 1\}$ (cadenas de 1 o más 0's seguido del mismo número de 1's) (C)
- Defina un CFG que acepte cadenas con el doble de 0's que de 1's (T)
- Generar la gramática para $0^* 1 (0 + 1)^*$ y hacer las derivaciones más a la izquierda y más a la derecha de:
  - 00101 (C)
  - 00011 (T)
- Gramática de paréntesis balanceados (C)
- Gramática de paréntesis redondos y cuadrados balanceados (T)

Árboles de Parseo

- Si $w \in L(G)$ para alguna CFG, entonces w tiene un árbol de parseo (*parse tree*), el cual nos da la estructura (sintáctica) de w. w puede ser un programa, un *query* en SQL, un documento en XML, etc.
- Los árboles de parseo son una representación atlernativa de las derivaciones e inferencias recursivas.
- Pueden existir varios árboles de parseo para la misma cadena.
- Idealmente nos gustaría que existiera solo uno, i.e., que el lenguaje fuera no ambigüo. Desafortunadamente no siempre se puede quitar la ambigüedad.

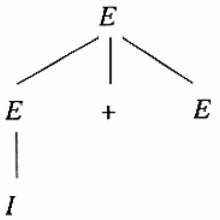
Construyendo Árboles de Parseo

Sea $G = (V, T, P, S)$ una CFG. Un árbol es un árbol de parseo de G si:

1. Cada nodo interior está etiquetado con una variable en V
2. Cada hoja está etiquetada con un símbolo en $V \cup T \cup \{\epsilon\}$. Cada hoja etiquetada con ϵ es el único hijo de su padre.
3. Si un nodo interior tiene etiqueta A y sus hijos (de izquierda a derecha) tienen etiquetas: $X_1, X_2, \ldots, X_k$, entonces: $A \rightarrow X_1, X_2, \ldots, X_k \in P$

Ejemplo

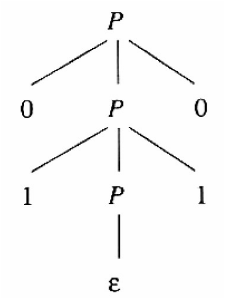
En la gramática: $E \rightarrow I$, $E \rightarrow E + E$, $E \rightarrow E * E$, $E \rightarrow (E)$, ..., el siguiente es un árbol de parseo:



Este árbol muestra la derivación: $E \stackrel{*}{\Rightarrow} I + E$.

Ejemplo

En la gramática: $P \rightarrow \epsilon$, $P \rightarrow 0$, $P \rightarrow 1$, $P \rightarrow 0p0$, $P \rightarrow 0p0$, el siguiente es un árbol de parseo:



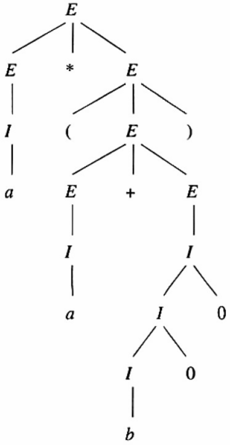
Este árbol muestra la derivación: $P \stackrel{*}{\Rightarrow} 0110$.

El producto de un árbol de parseo

- El producto (*yield*) de un árbol de parseo es la cadena de hojas de izquierda a derecha.
- Son en especial relevantes los árboles de parseo que:
  1. El producto es una cadena terminal
  2. La raíz esté etiquetada con el símbolo inicial
- El conjunto de productos de estos árboles de parseo nos definen el lenguaje de la gramática.

Ejemplo

Considere el siguiente árbol de parseo:



cuyo producto es: $a * (a + b00)$ (podemos comparar este árbol con la derivación que hicimos antes).

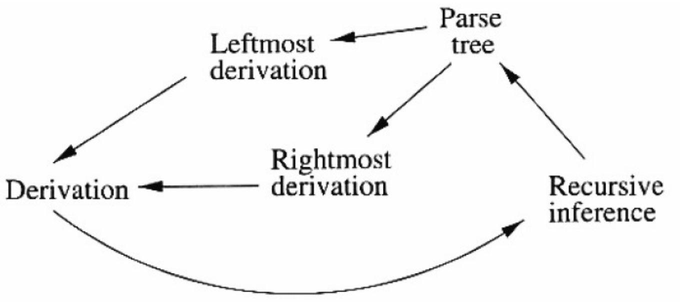
Equivalencias

Sea $G = (V, T, P, S)$ una CFG y $A \in V$. Vamos a demostrar que lo siguiente es equivalente:

- Podemos determinar por inferencia recursiva que w esté en el lenguaje de A
- $A \stackrel{*}{\Rightarrow} w$
- $A \Rightarrow^*_{lm} w$ y $A \Rightarrow^*_{rm} w$
- Existe un árbol de parseo de G con raíz A que produce w

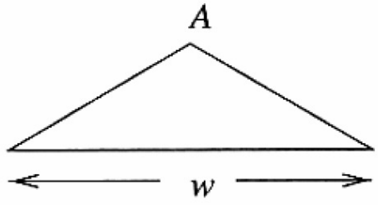
Equivalencias

El siguiente es el plan a seguir para probar las equivalencias:



De inferencias a árboles

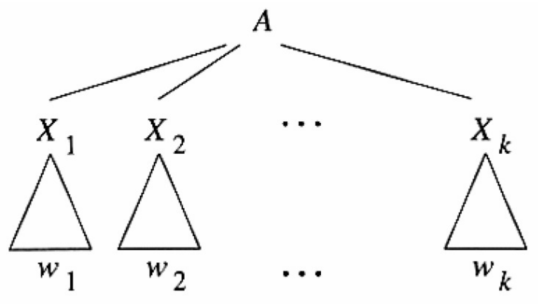
- **Teorema:** Sea $G = (V, T, P, S)$ una CFG y supongan que podemos mostrar que w está en el lenguaje de una variable A. Entonces existe un árbol de parseo para G que produce w.
- **Prueba:** la hacemos por inducción en la longitud de la inferencia
- *Base*: Un paso. Debemos de usar la producción $A \rightarrow w$. El árbol de parseo es entonces:



De inferencias a árboles

- *Inducción*: w es inferido en $n+1$ pasos. Suponemos que el último paso se baso en la producción: $A \rightarrow X_1, X_2, \ldots, X_k$, donde $X_i \in V \cup T$.
- Descomponemos w en: $w_1 w_2 \ldots w_k$, donde $w_i = X_i$ cuando $X_i \in T$, y cuando $X_i \in V$, entonces w_i fué previamente inferida en X_i en a los más n pasos.
- Por la hipótesis de inferencia existen i árboles de parseo con raíz X_i que producen w_i. Entonces el siguiente en una árbol de parseo de G con raíz en A que produce w:

De inferencias a árboles

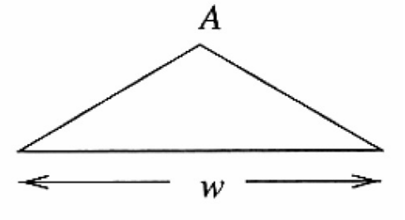


De árboles a derivaciones

- Mostraremos como construir una derivación más a la izquierda de un árbol de parseo.
- **Ejemplo:** De una gramática podemos tener la siguiente derivación: $E \Rightarrow I \Rightarrow Ib \Rightarrow ab$.
- Entonces, para cualquier α y β existe uan derivación $\alpha E \beta \Rightarrow \alpha I \beta \Rightarrow \alpha I b \beta \Rightarrow \alpha a b \beta$.
- Por ejemplo, supongamos que tenemos la derivación: $E \Rightarrow E + E \Rightarrow E + (E)$
- Entonces podemos escoger $\alpha =$ "$E + ($" y $\beta =$ "$)$" y seguir con la derivación como: $E + (E) \Rightarrow E + (I) \Rightarrow E + (Ib) \Rightarrow E + (ab)$
- Por esto es que las CFG se llaman libres de contexto (substitución de cadenas por variables, independientes del contexto).

De árboles a derivaciones

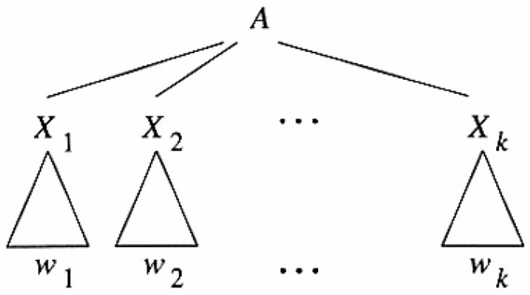
- **Teorema:** Sea $G = (V, T, P, S)$ una CFG y supongamos que existe un árbol de parseo cuya raíz tiene etiqueta A y que produce w. Entonces $A \Rightarrow^*_{lm} w$ en G.
- **Prueba:** la hacemos por inducción en la altura del árbol.
- *Base*: La altura es 1, y el árbol debe de verse asi:



- Por lo tanto $A \rightarrow w \in P$ y $A \Rightarrow_{lm} w$.

De árboles a derivaciones

Inducción: Altura es $n+1$. El árbol debe de verse así:



De árboles a derivaciones

Entonces $w = w_1 w_2 \ldots w_k$ donde:

1. Si $X_i \in T$, entonces $w_i = X_i$
2. Si $X_i \in V$, entonces debe de ser la raíz de un subárbol que nos da w_i, $X_i \overset{*}{\Rightarrow}_{lm} w_i$ en G por la hipótesis inductiva

De árboles a derivaciones

- Ahora mostramos $A \Rightarrow_{lm}^{*} w_1 w_2 \ldots w_i X_{i+1} X_{i+2} \ldots X_k$, obtener una derivación más a la izquierda. Probamos sobre i:
- *Base*: Sea $i = 0$. Sabemos que: $A \Rightarrow_{lm} X_1 X_2 \ldots X_k$
- *Inducción*: Hacemos la hipótesis inductiva: $A \Rightarrow_{lm}^{*} w_1 w_2 \ldots w_{i-1} X_i X_{i+1} \ldots X_k$
- Caso 1: Si $X_i \in T$, no hacemos nada ya que $X_i = w_i$ que nos da: $A \Rightarrow_{lm}^{*} w_1 w_2 \ldots w_i X_{i+1} \ldots X_k$

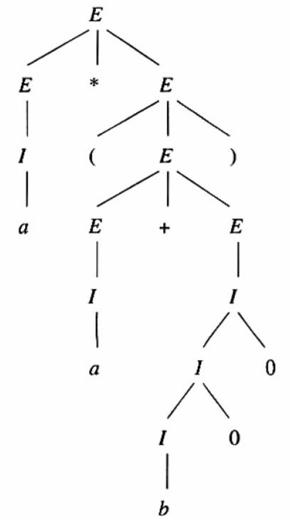
De árboles a derivaciones

- Case 2: $X_i \in V$. Por la hipótesis inductiva existe una derivación $X_i \Rightarrow_{lm} \alpha_1 \Rightarrow_{lm} \alpha_2 \ldots \Rightarrow_{lm} w_i$. Por la propiedad libre de contexto de las revivaciones podemos proceder como:

$$
\begin{aligned}
A \Rightarrow^*_{lm} \quad & w_1 w_2 \ldots w_{i-1} X_i X_{i+1} \ldots X_k \Rightarrow^*_{lm} \\
& w_1 w_2 \ldots w_{i-1} \alpha_1 X_{i+1} \ldots X_k \Rightarrow^*_{lm} \\
& w_1 w_2 \ldots w_{i-1} \alpha_2 X_{i+1} \ldots X_k \Rightarrow^*_{lm} \\
& \ldots \\
& w_1 w_2 \ldots w_{i-1} w_i X_{i+1} \ldots X_k \Rightarrow^*_{lm}
\end{aligned}
$$

Ejemplo

Construyamos la derivación más a la izquierda del árbol:



Ejemplo (cont.)

- Sopongamos que construimos inductivamente la derivación más a la izquierda: $E \Rightarrow_{lm} I \Rightarrow_{lm} a$ que corresponde con la rama izquierda del árbol, y la derivación más a la izquierda:
- $E \Rightarrow_{lm} (E) \Rightarrow_{lm} (E + E) \Rightarrow_{lm} (I + E) \Rightarrow_{lm} (a + E) \Rightarrow_{lm} (a * I) \Rightarrow_{lm} (a + I0) \Rightarrow_{lm} (a + I00) \Rightarrow_{lm} (a + b00)$ correspondiendo a la rama derecha del árbol.

Ejemplo (cont.)

- Para las derivaciones correspondientes del árbol completo empezamos con $E \Rightarrow_{lm} E * E$ y expandimos la primera E con la primera derivación y la segunda E con la segunda derivación:
- $E \Rightarrow_{lm} E * E \Rightarrow_{lm} I * E \Rightarrow_{lm} a * E \Rightarrow_{lm} a * (E) \Rightarrow_{lm} a * (E + E) \Rightarrow_{lm} a * (I + E) \Rightarrow_{lm} a * (a + E) \Rightarrow_{lm} a*(a*I) \Rightarrow_{lm} a*(a+I0) \Rightarrow_{lm} a*(a+I00) \Rightarrow_{lm} a*(a+b00)$
- De forma similar podemos convertir un árbol en una derivación más a la derecha. Osea, si existe un árbol de parseo con raíz etiquetada con la variable A que produce $w \in T^*$, entonces existe: $A \Rightarrow^*_{rm} w$.

De derivaciones a inferencias recursivas

- Supongamos que $A \Rightarrow X_1 X_2 \ldots X_k \stackrel{*}{\Rightarrow} w$, entonces: $w = w_1 w_2 \ldots w_k$ donde $X_i \stackrel{*}{\Rightarrow} w_i$.
- El factor w_i se puede extraer de $A \stackrel{*}{\Rightarrow} w$ viendo únicamente a la expansión de X_i.
- Por ejemplo: $E \Rightarrow a * b + a$ y

$$E \Rightarrow \underbrace{E}_{X_1} \underbrace{*}_{X_2} \underbrace{E}_{X_3} \underbrace{+}_{X_4} \underbrace{E}_{X_5}$$

De derivaciones a inferencias recursivas

- Tenemos que: $E \Rightarrow E * E \Rightarrow E * E + E \Rightarrow I * E + E \Rightarrow I * I + E \Rightarrow I * I + I \Rightarrow a * I + I \Rightarrow a * b + I \Rightarrow a * b + a$
- Viendo solo a la expansión de $X_3 = E$ podemos extraer: $E \Rightarrow I \Rightarrow b$
- **Teorema:** Sea $G = (V, T, P, S)$ una CFG. Suponga $A \overset{*}{\Rightarrow}_G w$ y que w es una cadena de símbolos terminales. Entonces podemos inferir que w está en el lenguaje de la variable A.
- **Prueba:** la hacemos por inducción en la longitud de la derivación $A \overset{*}{\Rightarrow}_G w$

De derivaciones a inferencias recursivas

- *Base*: Un paso. Si $A \Rightarrow_G w$ entonces debe de existir una producción $A \rightarrow w$ en P. Por lo que podemos inferir que w está en el lenguaje de A.
- *Inducción*: Suponemos $A \stackrel{*}{\Rightarrow}_G w$ en $n+1$ pasos. Escribimos la derivación como: $A \Rightarrow_G X_1 X_2 \ldots X_k \stackrel{*}{\Rightarrow}_G w$

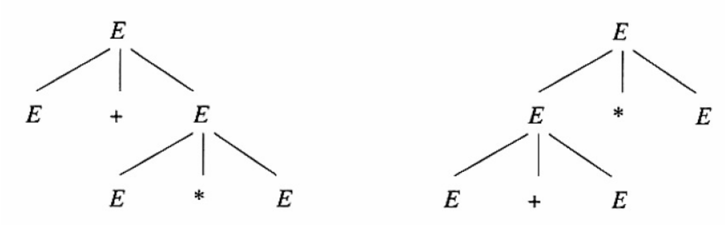
De derivaciones a inferencias recursivas

- Como vimos, podemos partir w como $w_1 w_2 \ldots w_k$ donde $X_i \stackrel{*}{\Rightarrow}_G w_i$. Además $X_i \stackrel{*}{\Rightarrow}_G w_i$ puede usar a lo más n pasos.
- Ahora tenemos una producción $A \rightarrow X_1 X_2 \ldots X_k$ y sabemos por la hipótesis inductiva que podemos inferir que w_i está en el lenguaje de X_i.
- Por lo que podemos inferir que $w_1 w_2 \ldots w_k$ está en el lenguaje de A.

Ambigüedad en Gramáticas y Lenguajes

En la gramática:
$E \rightarrow I$
$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow (E)$
...
la sentencia $E + E * E$ tiene dos derivaciones:
$E \Rightarrow E + E \Rightarrow E + E * E$ y $E \Rightarrow E * E \Rightarrow E + E * E$ lo cual nos da dos árboles de parseo:

Ambigüedad en Gramáticas y Lenguajes



Si tuvieramos números, e.g., 2,4 y 6, en lugar de las E's nos daría 26 por un lado y 36 por el otro. La existencia de varias derivaciones no es gran problema, sino la existencia de varios árboles de parseo.

Ambigüedad en Gramáticas y Lenguajes

Por ejemplo, en la misma gramática:

$\ldots$

$I \rightarrow a$

$I \rightarrow b$

$I \rightarrow Ia$

$I \rightarrow Ib$

$I \rightarrow I0$

$I \rightarrow I1$

la cadena $a + b$ tiene varias derivaciones:

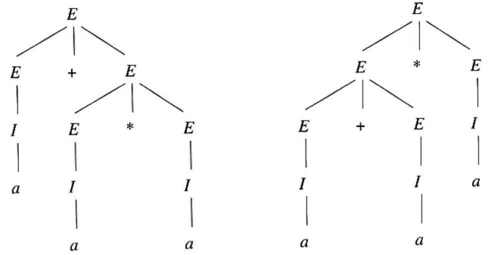
$E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$

$E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow I + b \Rightarrow a + b$

sin embargo, sus árboles de parseo son los mismos y la estructura de $a + b$ no es ambígüa.

Ambigüedad en Gramáticas y Lenguajes

- **Definición**: Sea $G = (V, T, P, S)$ una CFG. Decimos que G es *ambígüa* si existe una cadena en T^* que tenga más de un árbol de parseo.
- Si todas las cadenas en $L(G)$ tienen a lo más un árbol de parseo, se dice que G es *no ambígüa*.
- **Ejemplo**: La cadena $a + a * a$ tiene dos árboles de parseo:



Removiendo la ambigüedad de las gramáticas

- Las buenas noticias: A veces se puede remover la ambigüedad "a mano"
- Las malas noticias: no hay un algoritmo para hacerlo
- Peores noticias: Algunos CFLs solo tienen CFGs ambígüas.
- En la gramática: $E \rightarrow I|E + E|E * E|(E)$ y $I \rightarrow a|b|Ia|Ib|I0|I1$ existen dos problemas:
  1. No hay precedencia entre $*$ y $+$
  2. No existe un agrupamiento en las secuencias de operadores, e.g., $E + E + E$ significa: $E + (E + E)$ o $(E + E) + E$.

Removiendo la ambigüedad de las gramáticas

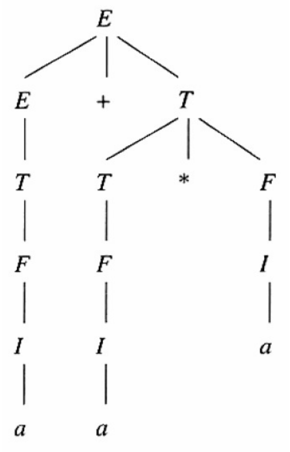
Solución: podemos introducir más variables para forzar un agrupamiento uniforme:

- Un *factor* (F) es una expresión que no puede separarse por un $*$ o $+$
- Un *término* (T) es una expresión que no puede separarse por un $+$
- El resto son expresiones que pueden separarse por $*$ o $+$

Removiendo la ambigüedad de las gramáticas

La gramática queda:
$I \rightarrow a|b|Ia|Ib|I0|I1$
$F \rightarrow I|(E)$
$T \rightarrow F|T * F$
$E \rightarrow T|E + T$
Con esto, el único árbol de parseo de $a + a * a$ es:

Removiendo la ambigüedad de las gramáticas

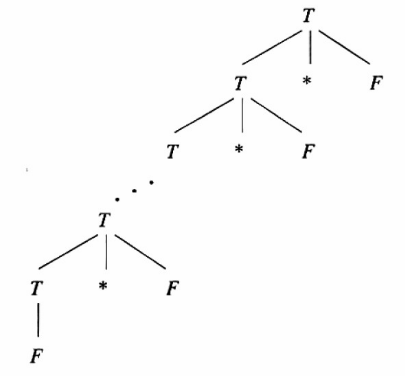


Removiendo la ambigüedad de las gramáticas

Las razones por las cuales la gramática nueva es no ambígüa son:

- Un factor es o un identificador o (E) para una expresión E
- El único árbol de parseo de una secuencia $f_1 * f_2 * \ldots * f_{n-1} * f_n$ de factores es el que da $f_1 * f_2 * \ldots * f_{n-1}$ como término y f_n como factor.

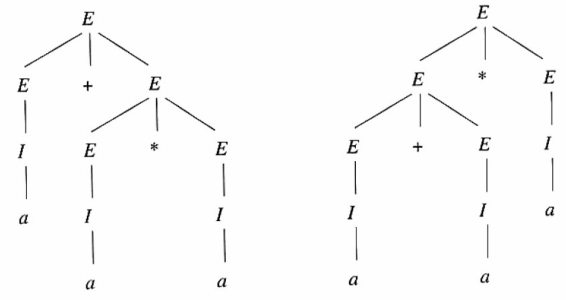
Removiendo la ambigüedad de las gramáticas



- Una expresión es una sequencia de: $t_1 + t_2 + \ldots + t_{n-1} + t_n$ de términos t_i y solo se puede parsear con $t_1 + t_2 + \ldots + t_{n-1}$ como una expresión y con t_n como término.

Derivaciones más a la izquierda y ambigüedad

En gramáticas no ambígüas, las derivaciones más a la izquierda y más a la derecha son únicas. Los dos árboles de derivación de $a + a * a$ son:



Derivaciones más a la izquierda y ambigüedad

Que nos da dos derivaciones:

$E \Rightarrow_{lm} E + E \Rightarrow_{lm} I + E \Rightarrow_{lm} a + E * E \Rightarrow_{lm} a + I * E \Rightarrow_{lm} a + a * E \Rightarrow_{lm} a + a * I \Rightarrow_{lm} a + a * a$

y

$E \Rightarrow_{lm} E + E * E \Rightarrow_{lm} I + E * E \Rightarrow_{lm} a + E * E \Rightarrow_{lm} a + I * E \Rightarrow_{lm} a + a * E \Rightarrow_{lm} a + a * I \Rightarrow_{lm} a + a * a$

Derivaciones y ambigüedad

En general:

- Se tienen un árbol de parseo pero varias derivaciones
- Muchas derivaciones más a la izquierda implican muchos árboles de parseo
- Muchas derivaciones más a la derecha implican muchos árboles de parseo

Derivaciones y ambigüedad

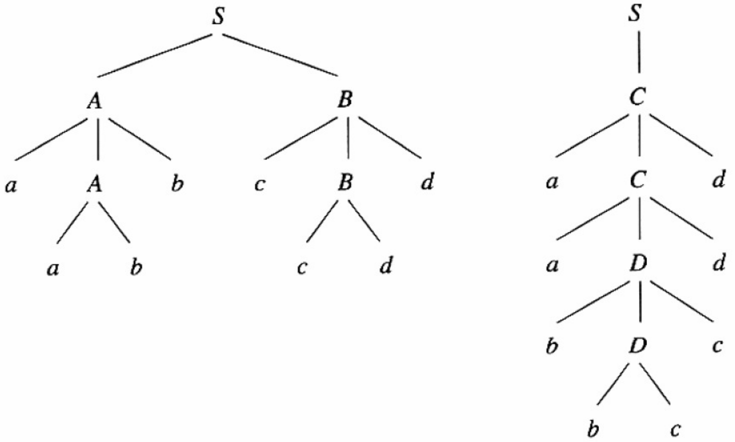
- **Teorema**: For cualquier CFG G, una cadena de terminales w tiene dos árboles de parseo diferentes si y solo si w tiene dos derivaciones más a la izquierda distintas desde el símbolo inicial.
- **Esquema de la prueba**: (Solo si) Si dos árboles de parseo difieren, tienen un nodo con diferentes producciones. Las derivaciones más a la izquierda correspondientes usaran sus derivaciones en estas dos producciones diferentes y por lo tanto serán distintas.
- (Si) Analizando como se construye un árbol de parseo a partir de una derivación más a la izquierda, debe de ser claro que dos derivaciones distintas producen dos árboles de parseo distintos.

Ambigüedad Inherente

- Un CFL L es *inherentemente ambígüo* si todas las gramáticas para L son ambígüas.
- **Ejemplo**: Considere $L = \{a^n b^n c^m d^m : n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n : n \geq 1, m \geq 1\}$
  Una gramática para L es:
  $S \rightarrow AB|C$
  $A \rightarrow aAb|ab$
  $B \rightarrow cBd|cd$
  $C \rightarrow aCd|aDd$
  $D \rightarrow bDc|bc$

Ejemplo

Veamos los árboles para: *aabbccdd*:



Ejemplo (cont.)

- Vemos que existen dos derivaciones más a la izquierda: $S \Rightarrow_{lm} AB \Rightarrow_{lm} aAbB \Rightarrow_{lm} aabbB \Rightarrow_{lm} aabbcBd \Rightarrow_{lm} aabbccdd$
  y
  $S \Rightarrow_{lm} C \Rightarrow_{lm} aCd \Rightarrow_{lm} aabDdd \Rightarrow_{lm} aabbccdd$
- Se puede mostrar que todas las gramáticas para L se comportan como la anterior. L es inherentemente ambígüo.