Ciencias computacionales

Propedéutico: Teoría de Autómatas y Lenguajes Formales

Gramáticas libres de contexto y lenguajes

Contents

1 Gramáticas libres de contexto **2**
- 1.1 Un ejemplo informal . . . 2
- 1.2 Definición de las gramáticas libres de contexto . . . 2
- 1.3 Derivaciones utilizando gramáticas . . . 3
- 1.4 Derivaciones por la izquierda y por la derecha . . . 4
- 1.5 El lenguaje de una gramática . . . 4
- 1.6 Formas enunciativas . . . 5
- 1.7 Ejercicios . . . 5

2 Análisis sintáctico de árboles **6**
- 2.1 Construcción de analizadores sintácticos de árboles . . . 6
- 2.2 El producto de un analizador sintáctico de árboles . . . 7
- 2.3 Inferencia, derivaciones y analizadores sintácticos de árboles . . . 7
- 2.4 De inferencias a árboles . . . 7
- 2.5 De árboles a derivaciones . . . 9
- 2.6 De derivaciones a inferencias recursivas . . . 10

3 Aplicaciones de las gramáticas libres de contexto **10**
- 3.1 Analizadores sintácticos . . . 11
- 3.2 Lenguajes de marcado . . . 11

4 Ambigüedad en gramáticas y lenguajes **11**
- 4.1 Gramáticas ambigüas . . . 11
- 4.2 Eliminación de la ambigüedad en las gramáticas . . . 11
- 4.3 Derivaciones por la izquierda como una forma de expresar ambigüedad . . . 12
- 4.4 Ambigüedad inherente . . . 13
- 4.5 Ejercicios . . . 13

1 Gramáticas libres de contexto

Hemos visto que muchos lenguajes no son regulares. Por lo que necesitamos una clase mas grande de lenguajes. 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*).

Una gramática libre de contexto es una notación para describir lenguajes. Son más capaces que los autómatas finitos o las expresiones regulares, pero no pueden definir todos los posibles lenguajes. Son útiles para estructuras anidadas, e.g. paréntesis en lenguajes de programación.

1.1 Un ejemplo informal

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 1 *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.

Existe una definición recursiva para cuando una cadena de 0's y 1's están en L_{pal}. Se comienza con el caso base donde $\epsilon, 0$ y 1 son palíndromes. Después se utiliza la definición de una cadena palíndroma: el primero y último símbolo deben de ser iguales. Entonces, el paso inductivo es Si w es un palíndrome, también $0w0$ y $1w1$.

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

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

donde 0 y 1 son símbolos terminales.

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. Los símbolos terminales son símbolos del alfabeto del lenguaje que se está definiendo. Las variables o símbolos no terminales son un conjunto finito de otros símbolos, cada uno representa un lenguaje. Existe además el símbolo de inicio = la variable cuyo lenguaje es el que se está definiendo.

1.2 Definición de las gramáticas libres de contexto

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 $A \to \alpha$, donde A es una variables y $\alpha \in (V \cup T)^*$

- S es una variable designada llamada el *símbolo inicial*

Ejemplo 2 *La cadena de numeros binarios se pueden expresar como una grámatica libre de contexto:* $G_{pal} = (\{P\}, \{0,1\}, A, P)$*, donde* $A = \{P \to \epsilon, P \to 0, P \to 1, P \to 0P0, P \to 1P1\}$.

Muchas veces se agrupan las producciones con la misma cabeza, e.g., $A = \{P \to \epsilon|0|1|0P0|1P1\}$. De la misma forma se pueden expresar expresiones regulares sobre {0,1} de la siguiente manera: $G_{regex} = (\{E\}, \{0,1\}, A, E)$, donde $A = \{E \to 0, E \to 1, E \to E.E, E \to E + E, E \to E^*, E \to (E)\}$.

Las expresiones en un lenguaje de programación donde los operadores son + y $*$, y los argumentos son identificadores (*strings*) que empiezan con a o b en $L((a+b)(a+b+0+1)^*)$.

Dichas 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:

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

De esta forma compacta, se pueden expresar una gran cantidad de lenguajes.

Ejercicio 1 *Construya la grámatica para la expresión regular:* 00^*11^*

Ejercicio 2 *Encuentre una gramática que genere el siguiente lenguaje:* $L(G) = \{a^n b^m c^m d^{2n} \mid n \geq 0, m \geq 0\}$

Ejercicio 3 *¿Qué lenguaje es construido por la siguiente gramática?:*

1. $S \to abScB \mid \epsilon$
2. $B \to bB \mid b$

1.3 Derivaciones utilizando gramáticas

Se pueden aplicar las producciones de un CFG para inferir que algunas cadenas se encuentran en el lenguaje de una cierta variable. Existen dos enfoques para esto. El primer enfoque convencional es utilizar las reglas del cuerpo mediante inferencia recursiva. La cual consta de utilizar las producciones del cuerpo a la cabeza para reconocrer 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)

El segundo enfoque es empezar en la cabeza y terminar en el cuerpo. Se comienza con un símbolo de inicio, y reemplazando repetidamente alguna variables por sus producciones.

Ejemplo 3 *Sea el siguiente lenguaje representado con las siguiente reglas:*

- $S \to 01$

- $S \to 0S1$

Una posible derivación es: $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000111$
Otra posible derivación es: $0S1 \Rightarrow 0011$

El operador $\Rightarrow^*$ significa "cero o más pasos de derivación". Formalmente, se define $\overset{*}{\Rightarrow}$ como la cerradura reflexiva y transitiva de $\Rightarrow$. Lo que quiere decir es que usamos uno a mas pasos de derivación. Utilizando inducción.

- *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 4 *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: $E \overset{*}{\Rightarrow} a * (a + b00)$

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 $A \overset{*}{\Rightarrow} w$ 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 $a * (a + b00)$.

1.4 Derivaciones por la izquierda y por la derecha

Debido a las múltiples formas de derivar un lenguaje, se utilizan dos técnicas 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 re-emplaza la variable más a la derecha por uno de los cuerpos de sus producciones.

Ejemplo 5 *La derivación de la gramática de paréntesis balanceados:*

$$S \to SS|(S)|()$$

Puede ser derivada a la izquierda y/o derecha.

- *Derivación a la izquierda:* $S \Rightarrow_{lm} SS \Rightarrow_{lm} (S)S \Rightarrow_{lm} (())S \Rightarrow_{lm} (())()$. *Entonces,* $S \Rightarrow^*_{lm} (())()$.
- *Derivación a la derecha:* $S \Rightarrow_{rm} SS \Rightarrow_{rm} S() \Rightarrow_{rm} (S)() \Rightarrow_{rm} (())$. *Entonces,* $S \Rightarrow^*_{rm} (())()$

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, $A \overset{*}{\Rightarrow} w$ si y solo si $A \Rightarrow^*_{lm} w$ y si y solo si $A \Rightarrow^*_{rm} w$.

1.5 El lenguaje de una gramática

Si $G(V, T, P, S)$ es una CFG, entonces el lenguaje de G es: $L(G) = \{w \in T^* : S \Rightarrow^*_G w\}$, 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 1 $L(G_{pal}) = \{w \in \{0, 1\}^* : w = w^R\}$

Para probar el Teorema 1 se utiliza inducción: ($\Rightarrow$) Suponemos $w = w^R$. Mostramos por inducción en $|w|$ que $w \in L(G_{pal})$.

- *Base*: $|w| = 0$ or $|w| = 1$. Entonces w es $\epsilon, 0$ o 1. Como $P \to \epsilon, P \to 1$ y $P \to 0$ son producciones, concluimos que $P \Rightarrow_G^* w$ 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.

Para la segunda parte de la prueba se continua con la inducción: ($\Leftarrow$): Asumimos 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 $\epsilon, 0$ o 1, todos palíndromes.
- *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.

1.6 Formas enunciativas

Sea $G = (V, T, P, S)$ una CFG y $\alpha \in (V \cup T)^*$. Si $S \stackrel{*}{\Rightarrow} \alpha$ 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^*.

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: $E \Rightarrow_{rm} E * E \Rightarrow_{rm} E * (E) \Rightarrow_{rm} E * (E + E)$.

1.7 Ejercicios

Ejercicio 4 *Diseñe las gramáticas libres de contexto de los siguientes lenguajes:*

1. *El conjunto* $\{0^n 1^n | n \geq 1\}$*, esto es, el conjunto de todas las cadenas de uno o más ceros seguidos por un numero igual de unos.*
2. *El conjunto de todas las cadenas de as y bs que no tienen la forma de ww, es decir, no son iguales a cadenas repetidas.*

Ejercicio 5 *La siguiente gramática genera el lenguaje de la expresión regular:* $0^*1(0 + 1)^*$

- $S \to A1B$
- $A \to 0A|\epsilon$
- $B \to 0B|1B|\epsilon$

De las derivaciones izquierda y derecha de las siguientes cadenas:

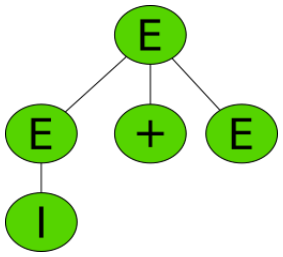

Figure 1: Árbol de expresiones.

1. *00101*
2. *1001*
3. *00011*

Ejercicio 6 *Considere el CFG G definido por las producciones:*

$$S \to aS|Sb|a|b$$

a *Pruebe por inducción en la longitud de cadena que ninguna cadena en $L(G)$ tiene ba como subcadena.*

b *Describa $L(G)$ de manera informal. Justifique su respuesta usando a*

2 Análisis sintáctico de árboles

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.

2.1 Construcción de analizadores sintácticos de árboles

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: $A \to X_1, X_2, \ldots, X_k \in P$.

En la gramática: $E \to I$, $E \to E + E$, $E \to E * E$, $E \to (E)$, ..., la Figura 1 muestra el árbol de parseo. Dicho árbol muestra la derivación $E \Rightarrow^* I + E$

Ejemplo 6 *En la gramática:*

- $P \to \epsilon|0|1$
- $P \to 0P0$

  El árbol se muestra en la Figura 2. Dicho árbol muestra la derivación $P \Rightarrow^* 0110$.

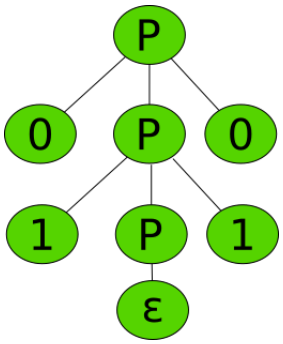

Figure 2: Representación de la CFG del Ejemplo 6 en forma de árbol.

2.2 El producto de un analizador sintáctico de árboles

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.

2.3 Inferencia, derivaciones y analizadores sintácticos de árboles

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

La Figura 3 muestra el plan a seguir para probar las equivalencias.

2.4 De inferencias a árboles

Para llevar de la inferencia recursiva a un árbol de parseo se utiliza el siguiente Teorema.

Teorema 2 *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.*

La prueba es por inducción en la longitud de la inferencia.

- *Base*: Un paso. Debemos de usar la producción $A \to w$. Ver Figura 4.
- *Inducción*: w es inferido en $n + 1$ pasos. Suponemos que el último paso se baso en la producción: $A \to X_1, X_2, \ldots, X_k$, 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. Ver Figura 5

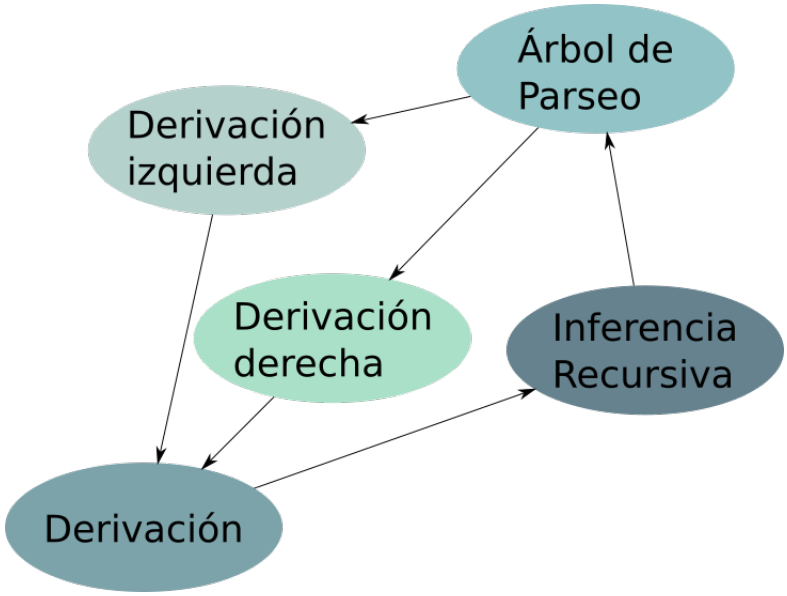

Figure 3: Plan a seguir para demostrar equivalencia entre diferentes derivaciones e inferencias.

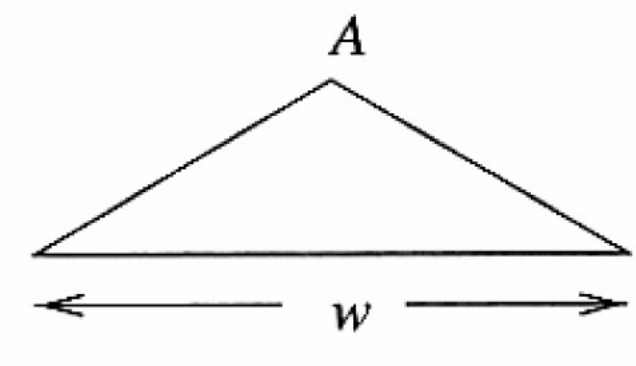

Figure 4: Caso base.

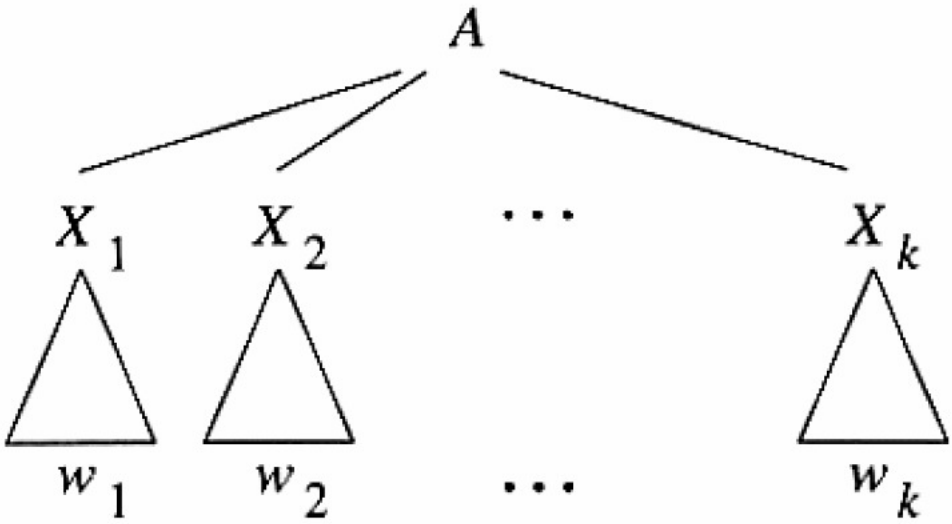

Figure 5: Inducción.

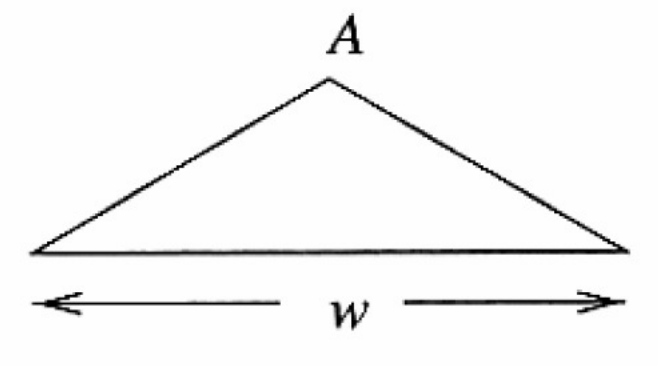

Figure 6: Caso base.

2.5 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 una derivación $\alpha E\beta \Rightarrow \alpha I\beta \Rightarrow \alpha Ib\beta \Rightarrow \alpha ab\beta$. 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). A continuación se presenta un teorema para llevar de árboles a derivaciones.

Teorema 3 *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.*

La prueba se hace por inducción el la altura del árbol.

- *Base*: La altura es 1. Por lo tanto $A \to w \in P$ y $A \Rightarrow_{lm} w$.
- *Inducción*: Altura es $n + 1$. Si $X_i \in T$, entonces $w_i = X_i$. Si $X_i \in V$, entonces debe de ser la raíz de un subárbol que nos da w_i, $X_i \Rightarrow^*_{lm} w_i$ en G por la hipótesis inductiva

Ahora mostramos $A \Rightarrow^*_{lm} w_1w_2 \dots w_iX_{i+1}X_{i+2} \dots X_k$, obtener una derivación más a la izquierda. Probamos sobre i:

- *Base*: Sea $i = 0$. Sabemos que: $A \Rightarrow_{lm} X_1X_2 \dots X_k$. Ver Figura 6.
- *Inducción*: Hacemos la hipótesis inductiva: $A \Rightarrow^*_{lm} w_1w_2 \dots w_{i-1}X_iX_{i+1} \dots X_k$. Ver Figura 7
- Caso 1: Si $X_i \in T$, no hacemos nada ya que $X_i = w_i$ que nos da: $A \Rightarrow^*_{lm} w_1w_2 \dots w_iX_{i+1} \dots X_k$
- Case 2: $X_i \in V$. Por la hipótesis inductiva existe una derivación $X_i \Rightarrow_{lm} \alpha_1 \Rightarrow_{lm} \alpha_2 \dots \Rightarrow_{lm} w_i$. Por la propiedad libre de contexto de las revivaciones podemos proceder como:

$$\begin{aligned} A \Rightarrow^*_{lm} \quad & w_1w_2 \dots w_{i-1}X_iX_{i+1} \dots X_k \Rightarrow^*_{lm} \\ & w_1w_2 \dots w_{i-1}\alpha_1X_{i+1} \dots X_k \Rightarrow^*_{lm} \\ & w_1w_2 \dots w_{i-1}\alpha_2X_{i+1} \dots X_k \Rightarrow^*_{lm} \\ & \dots \\ & w_1w_2 \dots w_{i-1}w_iX_{i+1} \dots X_k \Rightarrow^*_{lm} \end{aligned}$$

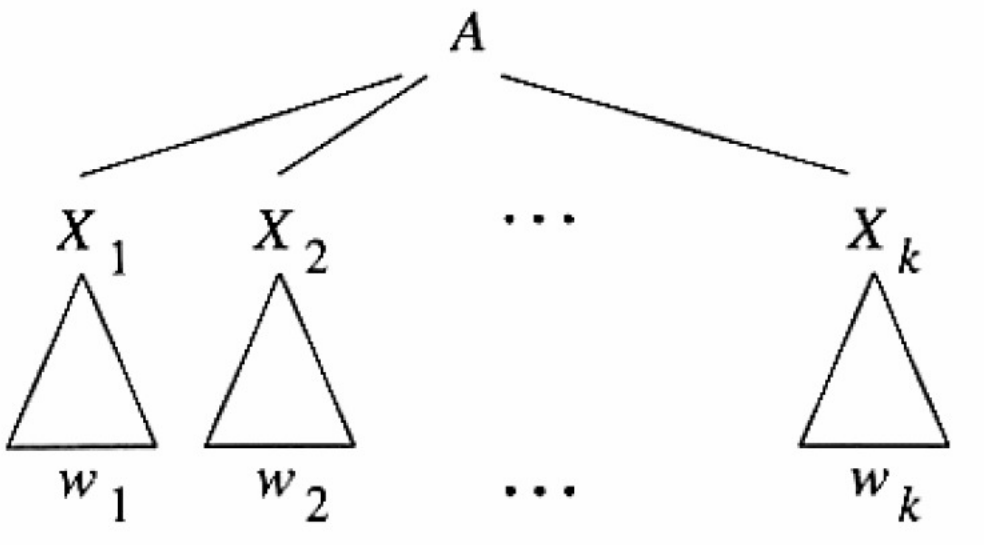

Figure 7: Inducción.

2.6 De derivaciones a inferencias recursivas

Supongamos que $A \Rightarrow X_1X_2 \dots X_k \stackrel{*}{\Rightarrow} w$, entonces: $w = w_1w_2 \dots w_k$ 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

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

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 4 *Sea $G = (V, T, P, S)$ una CFG. Suponga $A \stackrel{*}{\Rightarrow}_G w$ y que w es una cadena de símbolos terminales. Entonces podemos inferir que w está en el lenguaje de la variable A.*

La prueba es por inducción y se hace sobre la longitud de la derivación $A \stackrel{*}{\Rightarrow}_G w$.

- *Base*: Un paso. Si $A \Rightarrow_G w$ entonces debe de existir una producción $A \to w$ 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: $A \Rightarrow_G X_1X_2 \dots X_k \stackrel{*}{\Rightarrow}_G w$

Como vimos, podemos partir w como $w_1w_2 \dots w_k$ 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 produccuón $A \to X_1X_2 \dots X_k$ 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 $w_1w_2 \dots w_k$ está en el lenguaje de A.

3 Aplicaciones de las gramáticas libres de contexto

Las gramáticas libre de contexto fueron concebidas por N. Chomsky como una forma de describir lenguajes naturales. Esto no se ha cumplido. Sin embargo, las CFGs permiten describir otro tipo de lenguajes. Presentamos a continuación algunos ejemplos del uso de las mismas.

3.1 Analizadores sintácticos

Un analizador sintáctico (o parser) es una de las partes de un compilador que transforma su entrada en un árbol de derivación. Se puede describir un lenguaje de programación mediante reglas de producción. Por ejemplo una gramática que describe las posibles secuencias de **if** y **else** es: $S \to \epsilon|SS|iS|iSeS$. Las secuencias $ieie$,iie pueden ser leídas por la gramáticas. Una secuencia ilegal es la siguiente ei.

3.2 Lenguajes de marcado

Un lenguaje de marcado o lenguaje de marcas es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación. El lenguaje de marcado con mayor difusión es HTML. Dicho lenguaje tiene dos funciones principales: crear enlaces entre documentos y describir el formato de documentos.

Por otro lado, el lenguaje XML de un documento no tiene como fin describir el formato de los documentos, sino describir la semántica del texto. Por ejemplo para decir la dirección de una calle, em XML se escribe de la siguiente forma: $< ADDR > 12Sta.Rita < /ADDR >$. Este texto adquiere el significado de dirección.

4 Ambigüedad en gramáticas y lenguajes

No todas las gramáticas proveen estructuras únicas. Cuando esto ocurre, a veces es posible rediseñar la gramática para crear una estructura única, pero no todo el tiempo. En ese caso se tiene gramáticas ambigüas. Por ejemplo en la gramática:
$E \to I$
$E \to E + E$
$E \to E * E$
$E \to (E)$
...
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.

4.1 Gramáticas ambigüas

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*.Por ejemplo, la cadena $a + a * a$ tiene dos árboles de parseo.

4.2 Eliminación de la ambigüedad en las gramáticas

Aunque es posible eliminar la ambigüedad en las gramática, no existe un algoritmo para hacerlo. Incluso existen algunos CFLs que solo tienen CFGs ambigüas. Por ejemplo, en la gramática: $E \to I|E + E|E * E|(E)$ y $I \to a|b|Ia|Ib|I0|I1$ 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 $(E + E) + E$.

Una solución es 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 $+$

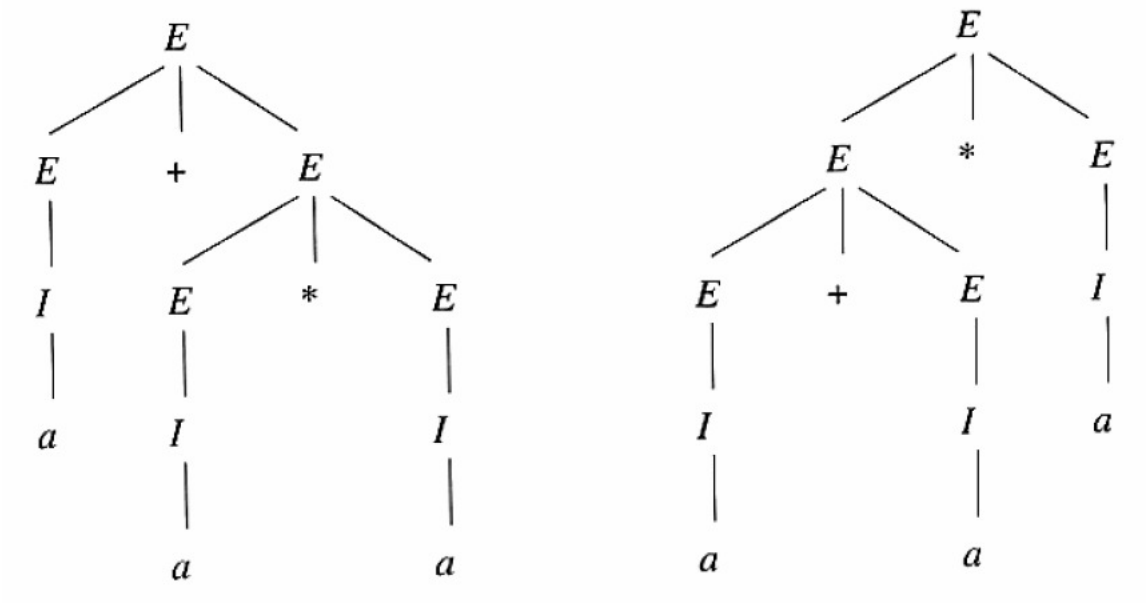

Figure 8: Árboles de $a + a * a$.

La gramática queda:
$I \to a|b|Ia|Ib|I0|I1$
$F \to I|(E)$
$T \to F|T * F$
$E \to T|E + T$

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.

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.

4.3 Derivaciones por la izquierda como una forma de expresar 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:
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
$E \Rightarrow_{lm} E + E * E \Rightarrow_{lm} I + E * E \Rightarrow_{lm} a + E * E \Rightarrow_{lm} a + I * E \Rightarrow_{lm} a + a * E \Rightarrow_{lm} a + a * I \Rightarrow_{lm} a + a * a.$

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

Teorema 5 *Para 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.*

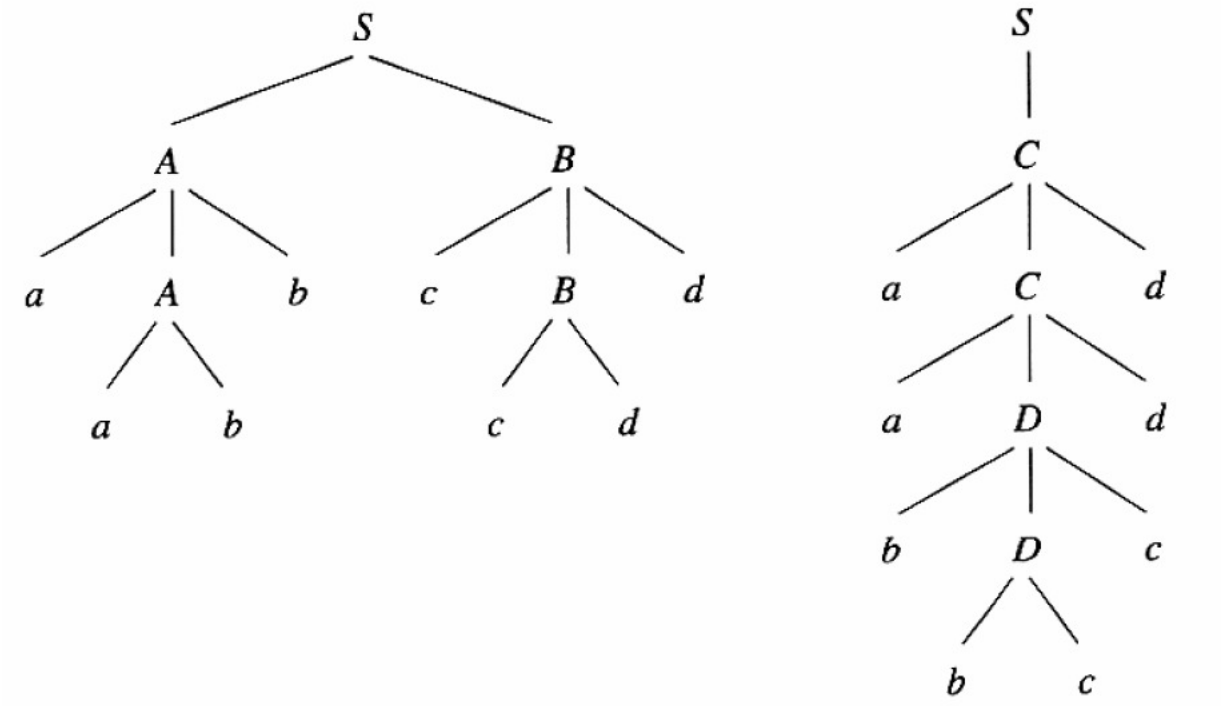

Figure 9: Árboles de *aabbccdd*.

- **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.

4.4 Ambigüedad inherente

Un CFL L es *inherentemente ambígüo* si todas las gramáticas para L son ambígüas. Por 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:
$S \to AB|C$
$A \to aAb|ab$
$B \to cBd|cd$
$C \to aCd|aDd$
$D \to bDc|bc$

La representacion de esta gramática en árboles para: *aabbccdd* se muestra en la Figura 9.

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.

4.5 Ejercicios

Ejercicio 7 *Considere la siguiente gramática:* $S \to aS|aSbS|\epsilon$. *Esta gramática es ambigüa. Mostrar que la cadena aab tiene dos:*

- *Árboles de derivación.*
- *Derivaciones a la izquierda.*
- *Derivaciones a la derecha.*

Ejercicio 8 *Encuentre una gramática no ambigüa del Ejercicio 7.*

Ejercicio 9 *Muestra que la siguiente gramática es ambigüa.*
$S \to A1B$
$A \to 0A|\epsilon$
$B \to 0B|1B|\epsilon$

Ejercicio 10 *Encuentre una gramática para el mismo lenguaje del Ejercicio 10 que sea ambigüa y demuestre su ambigüedad.*