5. Lenguajes y Gramáticas Independientes del Contexto

5.1. Gramáticas Independientes del Contexto

Fernando Rosa Velardo

Traducción y adaptación de transparencias de Ananth Kalyanaraman
(http://www.eecs.wsu.edu/~ananth/)

Hay lenguajes no regulares

- ¿Y qué pasa con los lenguajes que no son regulares?

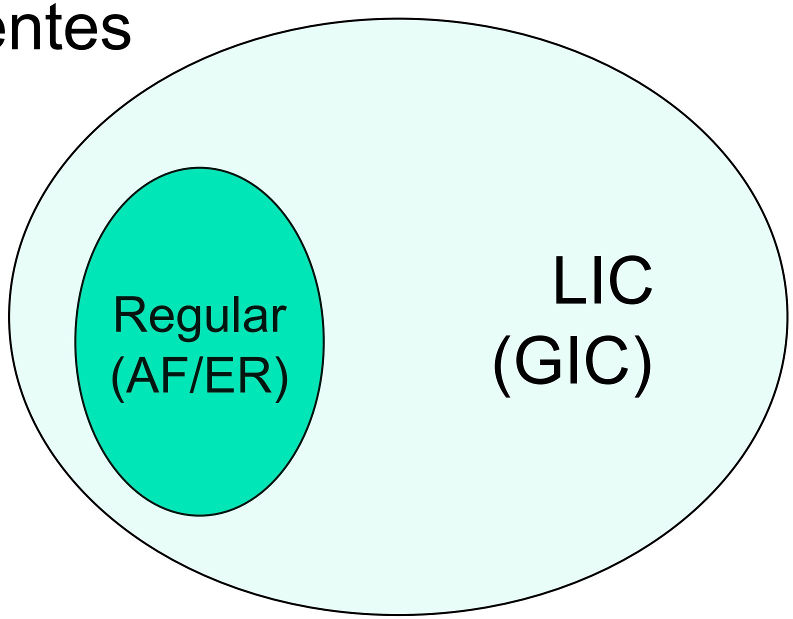
- ¿Existe algún reconocedor para ellos?
  - es decir, algo que acepte (o rechace) palabras que pertenecen (o que no pertenecen) al lenguaje

Lenguajes Independientes del Contexto (LIC)

- Clase de lenguajes que contiene a los lenguajes regulares

- Dados por una notación recursiva llamada "Gramáticas Independientes del Contexto" (GIC)

- Aplicaciones:
  - Árboles sintácticos
  - Compiladores
  - XML



Ejemplo

- Palíndromo: palabra que se lee igual hacia adelante que hacia atrás
  - reconocer, radar, oso, 010010010
- Sea L = { w | w palíndromo binario}
- ¿L regular?
  - No.
  - <u>Demostración:</u>
    - Tomamos $w=0^N10^N$ donde N es la constante del lema de bombeo
    - Sea w=xyz, con $|xy|\leq N$ y $y\neq\varepsilon$
    - ==> $y=0^l$ con $l>0$
    - ==> $xz=0^{N-l}10^N$ *NO* pertenece a L
    - ==> Contradicción

Pero el lenguaje de palíndromos…

es un LIC, porque admite una definición recursiva (en forma de GIC)

- Podemos construir una *"gramática"* de la siguiente forma:

Producciones

1. $A \rightarrow \varepsilon$
2. $A \rightarrow 0$
3. $A \rightarrow 1$
4. $A \rightarrow 0A0$
5. $A \rightarrow 1A1$

Terminales

Variable o no terminal

Lo mismo que:
$A \rightarrow 0A0 \mid 1A1 \mid 0 \mid 1 \mid \varepsilon$

¿Y cómo funciona una gramática?

¿Cómo funciona la GIC de los palíndromos?

Una palabra pertenece al lenguaje (es decir, es aceptada) si puede ser generada por la GIC

G:
$A \rightarrow 0A0 \mid 1A1 \mid 0 \mid 1 \mid \varepsilon$

- Ejemplo: 01110
- G puede generar la palabra como sigue:
  - A=> 0A0
    => 01A10
    => 01110

Gramática independiente del Contexto: Definición

- Una Gramática Independiente del Contexto (GIC) es G=(V,T,P,S), donde:
  - V: conjunto de variables o no terminales
  - T: conjunto de terminales (alfabeto)
  - P: conjunto de *producciones* $A \rightarrow \alpha$
    - donde A es una variable, y
    - α es una cadena de variables y terminales
  - S: variable inicial

GIC de los palíndromos

<u>G:</u>
$A \rightarrow 0A0 \mid 1A1 \mid 0 \mid 1 \mid \varepsilon$

- $G=(\{A\},\{0,1\},P,A)$
  - $P=\{ A\rightarrow 0A0 \;,\; A\rightarrow 1A1 \;,\; A \rightarrow 0 \;,\; A\rightarrow 1 \;,\; A\rightarrow \varepsilon \}$

Derivaciones

- $\alpha A \beta \Rightarrow \alpha\ \gamma\ \beta$ si existe una producción $A \rightarrow \gamma$
  - Ejemplo: 0A1=>00A11 (con $A \rightarrow 0A1$)
- $\alpha \Rightarrow \beta$ es un paso de derivación.
- $\alpha \Rightarrow^* \beta$ si desde α se pueden dar 0 o más pasos de derivación para llegar a β
  - 0A1=>*0A1 (en cero pasos)
  - 0A1=>*000A111 (en dos pasos)
- Transitividad:

  si A =>*B y B =>*C entonces A =>* C

Lenguajes Independientes del Contexto

- El lenguaje de la GIC G=(V,T,P,S) es
  - L(G) = { w en T* | S =>* w }
- Un lenguaje L es Independiente del Contexto si existe una GIC G tal que L=L(G).

Ejemplo #2

- $L = \{0^m 1^n \mid m \geq n\}$ es un LIC
- Gramática para L:

G:
$S \rightarrow 0S1 \mid A$
$A \rightarrow 0A \mid \varepsilon$

5. Lenguajes y Gramáticas Independientes del Contexto

5.2. Lenguajes Independientes del Contexto

Fernando Rosa Velardo

Traducción y adaptación de transparencias de Ananth Kalyanaraman (http://www.eecs.wsu.edu/~ananth/)

Corrección de una GIC

G:
$A \rightarrow 0A0 \mid 1A1 \mid 0 \mid 1 \mid \varepsilon$

- Teorema:

  w en L(G) $\Leftrightarrow$ w palíndromo.

- Demostración:
  - Por inducción
    - => Sobre la longitud de la derivación
    - <= Sobre la longitud de la palabra

Corrección de una GIC: Ejemplo #2

G:
$S \to 0S1 \mid A$
$A \to 0A \mid \varepsilon$

- $L = \{0^m 1^n \mid m \geq n\}$
- Teorema:

  $$w \text{ en } L(G) \Leftrightarrow w = 0^m 1^n \text{ con } m \geq n$$

- Demostración:
  - Probamos:
    - $S \Rightarrow^* w \Leftrightarrow w = 0^m 1^n$ con $m \geq n$
    - $A \Rightarrow^* w \Leftrightarrow w = 0^n$ con $n \geq 0$

Derivaciones más a la izquierda/derecha

G:
$E \rightarrow E+E \mid E*E \mid (E) \mid F$
$F \rightarrow aF \mid bF \mid 0F \mid 1F \mid \varepsilon$

Derivamos la cadena a*(ab+10)

E =>* a*(ab+10)

Derivación más a la izquierda:

Siempre usamos la variable más a la izquierda

- E
- =>$_{lm}$ E * E
- =>$_{lm}$ F * E
- =>$_{lm}$ a * E
- =>$_{lm}$ a * (E)
- =>$_{lm}$ a * (E + E)
- =>$_{lm}$ a * (F + E)
- =>$_{lm}$ a * (aF + E)
- =>$_{lm}$ a * (abF + E)
- =>$_{lm}$ a * (ab + E)
- =>$_{lm}$ a * (ab + F)
- =>$_{lm}$ a * (ab + 1F)
- =>$_{lm}$ a * (ab + 10F)
- =>$_{lm}$ a * (ab + 10)

Derivación más a la derecha:

Siempre usamos la variable más a la derecha

- E
- =>$_{rm}$ E * E
- =>$_{rm}$ E * (E)
- =>$_{rm}$ E * (E + E)
- =>$_{rm}$ E * (E + F)
- =>$_{rm}$ E * (E + 1F)
- =>$_{rm}$ E * (E + 10F)
- =>$_{rm}$ E * (E + 10)
- =>$_{rm}$ E * (F + 10)
- =>$_{rm}$ E * (aF + 10)
- =>$_{rm}$ E * (abF + 0)
- =>$_{rm}$ E * (ab + 10)
- =>$_{rm}$ F * (ab + 10)
- =>$_{rm}$ aF * (ab + 10)
- =>$_{rm}$ a * (ab + 10)

Derivaciones más a la izquierda y más a la derecha

1) Toda palabra generada por una GIC puede ser generada por derivaciones más a la izquierda y más a la derecha. ¿Cierto?

Cierto – fácil de demostrar

2) Para cada derivación más a la izquierda existe una derivación más a la derecha equivalente. ¿Cierto?

Cierto – lo veremos con árboles de derivación

3) ¿Existen palabras con más de una derivación más a la izquierda (o a la derecha)?

Puede ser – depende de la gramática

Aplicaciones de los LIC y GIC

- Los lenguajes de programación son LIC
- Los compiladores usan “parsers” para el análisis sintáctico
- Los parsers se expresan como GIC
  1. Paréntesis equilibrados
  2. If-then-else
  3. Llaves en C { … }
  4. *begin-end* de Pascal

- YACC (<u>Y</u>et <u>A</u>nother <u>C</u>ompiler-<u>C</u>ompiler)

Ejemplo #3

- Paréntesis equilibrados
- ()(((())))((())).…
- ¿GIC?

G:
$S \rightarrow (S) \mid SS \mid \varepsilon$

Expresiones aritméticas

- Sumas y productos de variables y números
- Variables son palabras sobre {a,b,0,1} que empiezan por a o b
- Los números los escribimos en binario
- Permitimos paréntesis

$$
\begin{aligned}
E &\rightarrow E+E \mid E*E \mid (E) \mid V \mid N \\
V &\rightarrow aF \mid bF \\
F &\rightarrow aF \mid bF \mid 0F \mid 1F \mid \varepsilon \\
N &\rightarrow 0 \mid 1 \mid 0N \mid 1N
\end{aligned}
$$

5. Lenguajes y Gramáticas Independientes del Contexto

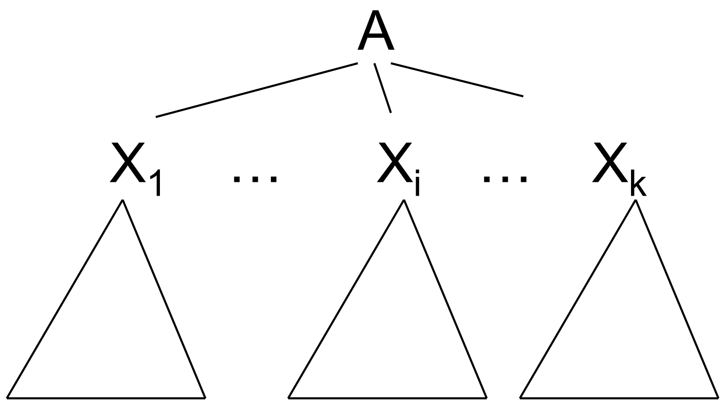
5.3. Árboles de Derivación

Fernando Rosa Velardo

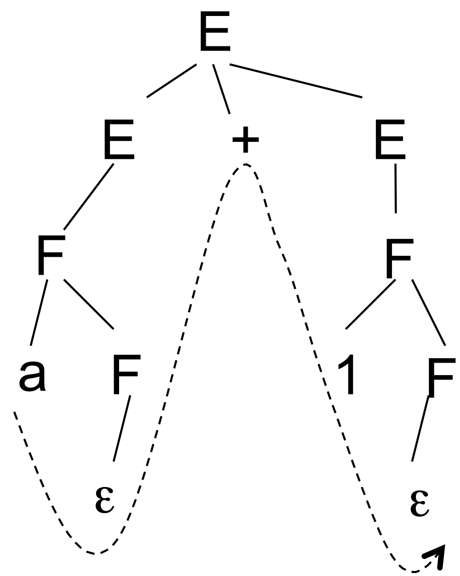
Traducción y adaptación de transparencias de Ananth Kalyanaraman
(http://www.eecs.wsu.edu/~ananth/)

Árboles de derivación

- Representación alternativa de las derivaciones:
  - Cada nodo interno está etiquetado por una variable de V
  - Cada hoja está etiquetada por un símbolo terminal
  - Si un nodo interno A tiene hijos $X_1, X_2, \ldots X_k$ (de izquierda a derecha) entonces hay una producción $A \rightarrow X_1 X_2 \ldots X_k$

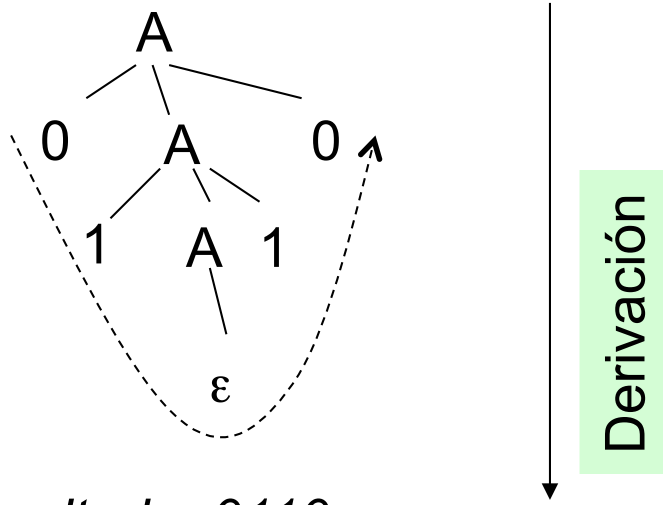


Ejemplos



Resultado: a + 1

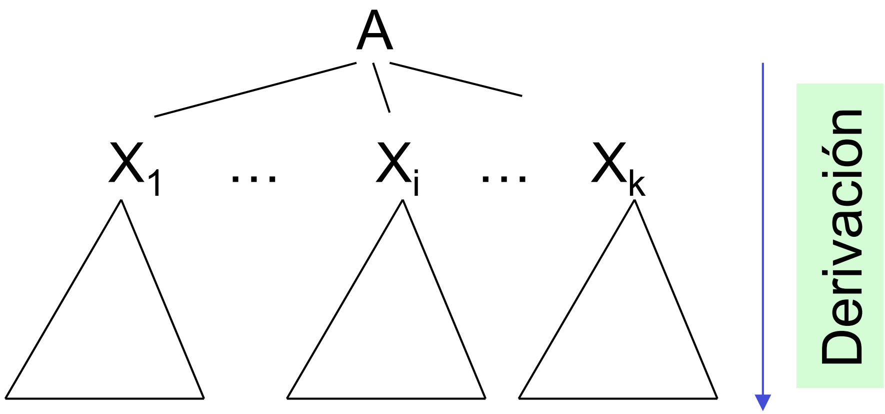
G:
E → E+E | E*E | (E) | F
F → aF | bF | 0F | 1F | ε



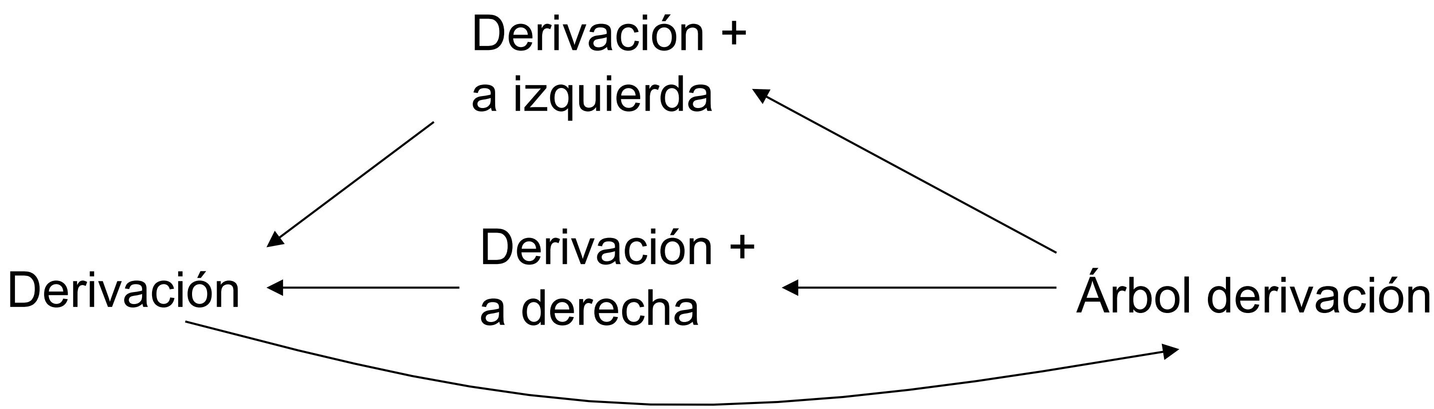
Resultado: 0110

G:
A → 0A0 | 1A1 | 0 | 1 | ε

Árboles de derivación y derivaciones

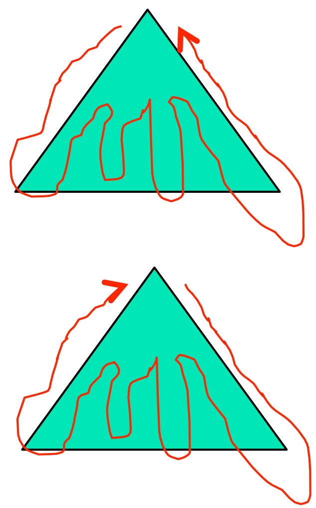


Si A=>* w entonces existe un árbol de derivación cuya raiz es A y cuyo resultado es w



Equivalencia de las distintas representaciones

- Derivación => derivación más a la izquierda/derecha (obvio)
- Árbol de derivación => derivación más a la izquierda/derecha
  - Recorridos en profundidad del árbol
- Derivación => Árbol de derivación



5. Lenguajes y Gramáticas Independientes del Contexto

5.4. Gramáticas ambiguas. Lenguajes inherentemente ambiguos

Fernando Rosa Velardo

Traducción y adaptación de transparencias de Ananth Kalyanaraman (http://www.eecs.wsu.edu/~ananth/)

Gramáticas ambiguas

- Una GIC es *ambigua* si existe una palabra con más de una derivación más a la izquierda

Ejemplo:

$S \rightarrow AS \mid \varepsilon$

$A \rightarrow A1 \mid 0A1 \mid 01$

Palabra: 00111

Derivación+izq #1:

S => AS
=> 0A1S
=>0A11S
=> 00111S
=> 00111

Derivación+izq.#2:

S => AS
=> A1S
=> 0A11S
=> 00111S
=> 00111

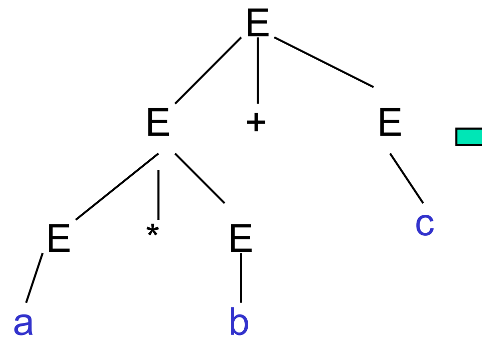
¿Por qué importa la ambigüedad?

$E \rightarrow E + E \mid E * E \mid (E) \mid a \mid b \mid c$

¡¡¡Distintos significados!!!

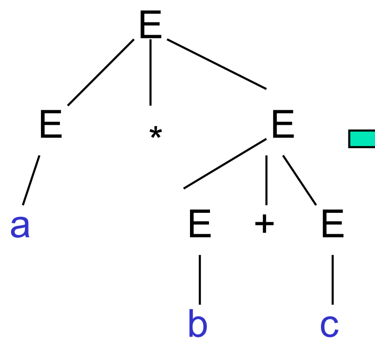
*palabra = a * b + c*

- Derivación+izq. #1:
  - •E => E + E => E * E + E ==>* a * b + c



(a*b)+c

- Derivación+izq. #2
  - •E => E * E => a * E => a * E + E ==>* a * b + c



a*(b+c)

GIC no ambigua para expresiones aritméticas

- Para algunas gramáticas es posible eliminar la ambigüedad
  - Por ejemplo, en la GIC de las expresiones aritméticas, forzando la precedencia de las operaciones

- <u>Precedencia:</u> (), * , +

<u>Versión no ambigua:</u>

$E \rightarrow E + T \mid T$
$T \rightarrow T * F \mid F$
$F \rightarrow I \mid (E)$
$I \rightarrow a \mid b \mid c$

LICs inherentemente ambiguos

- Para algunos lenguajes, es imposible eliminar la ambigüedad

- Un LIC es *inherentemente ambiguo* si toda GIC que lo genera es ambigua

Ejemplo:

- $L = \{ a^n b^n c^m d^m \mid n,m \geq 1\} \cup \{a^n b^m c^m d^n \mid n,m \geq 1\}$
- L es inherentemente ambiguo

Palabra: $a^n b^n c^n d^n$