UNIDAD I: LENGUAJES Y GRAMÁTICAS

Capítulo 1. Introducción.

Capítulo 2. Lenguajes Formales

Capítulo 3. Gramáticas Formales.

3.1. Concepto de gramática formal

Producciones. Gramática Formal. Lenguaje asociado a una gramática. Recursividad. Gramáticas equivalentes.

3.2. Tipos de gramáticas

Jerarquía de Chomsky. Gramáticas de tipos 0, 1, 2, y 3.

3.3. Árboles de derivación

Representación arborescente de derivaciones. Ambigüedad.

3.1 Concepto de gramática formal

- Hemos visto que lenguajes son subconjuntos de las palabras sobre un alfabeto ($L \subseteq W(\Sigma)$).
- ¿Cómo se puede definir las palabras que pertenecen a un determinado lenguaje?
  - Enumerando el conjunto de palabras
    $\Rightarrow$ ¿algunos lenguajes son infinitos?
  - Descripción informal de las palabras:
    $\Rightarrow$ a veces complicado y demasiado impreciso
  - Descripción formal, basada en los operadores de palabras:
    $\Rightarrow$ no suelen ser suficientes para especificar todos los lenguajes

Necesitamos un formalismo para definir los lenguajes (las palabras que pertenecen a un lenguaje).

$\Rightarrow$ una posibilidad - Gramáticas formales:

- Una gramática describe de forma inequívoca la estructura de las palabras de un lenguaje
- Proporcionan un mecanismo para generar todas las palabras que pertenecen a un determinado lenguaje (también se llaman *gramáticas generadoras*)

3.1.1 Producciones

Definición (Producción):
Sea Σ un alfabeto. Una *producción*, o *regla de producción*, o *regla de derivación*, o *regla de escritura*, es un par ordenado (x,y), también representado por x::=y, tal que $x \in \Sigma^+$ e $y \in \Sigma^*$.
Se llama x la *cabeza* o *parte izquierda* de la producción, e y su *cuerpo* o *parte derecha*.

Definición (Derivación directa):
Sea Σ un alfabeto, x::=y una producción sobre ∑, y $v,w \in \Sigma^*$. ´
w es una derivación directa de v, o v produce directamente w, o v se reduce de forma directa a w, representado por $v \rightarrow w$, si:

$\exists\, z, u \in \Sigma^*: v=zxu \wedge w=zyu$

Ejemplos:

Σ={ab}; a::=bb

- v=aabbaa
  - w_1=bbabbaa
  - w_2=aabbaa **¿Para que w_i se verifica $v \rightarrow w_i$?**
  - w_3=bbbbbbaa
  - w_4=aabbab
- v=a
  - w=bb : $v \rightarrow w$

  ⇒ si x::=y, entonces se verifica $x \rightarrow y$

Nota: Aparte de "::=" se usara también "→" como símbolo de la operación de producción. El contexto determina si "→" se refiere a una producción o una derivación.

Definición (Derivación):
Sea Σ un alfabeto, y P un conjunto de producciones sobre ∑. Sean v y w dos palabras de Σ^*. Se dice que w *deriva* de v, o v *produce* w, o v se *reduce* a w, y se presenta por $v \rightarrow^* w$ si:

1. v=w o
2. existe una secuencia finita de derivaciones directas
   $u_0 \rightarrow u_1$
   $u_1 \rightarrow u_2$
   …
   $u_{n-1} \rightarrow u_n$
   tales que $u_0=v$ y $u_n=w$.

La longitud de una derivación $v \rightarrow^* w$ es el número de derivaciones directas en la secuencia.

El operador $\rightarrow+$ se refiere a una derivación con longitud >0.

Ejemplos:

1. La longitud de $u \rightarrow^* u$ es 0.
2. ∑={C,N,0,1,2}
   P={ N::=CN, N::=C, C::=0, C::=1, C::=2}
   - N→CN
     CN→CC
     CC→0C ⇒ $N \rightarrow^* 00$ con longitud 4
     0C→00
   - **¿{x|x∈Σ^* y CCC→*x y x es "irreducible"}?**
   - **¿{x|x∈Σ^* y N→*x y x es "irreducible"}?**

Propiedades de la derivación
1. Reflexiva: $\forall x \in \Sigma^*: x \rightarrow^* x$
2. Transitiva: $\forall x,y,z \in \Sigma^*$: si $x \rightarrow^* y$ e $y \rightarrow^* z$, entonces $x \rightarrow^* z$.
3. Simétrica: no

3.1.2 Gramática formal

Motivación

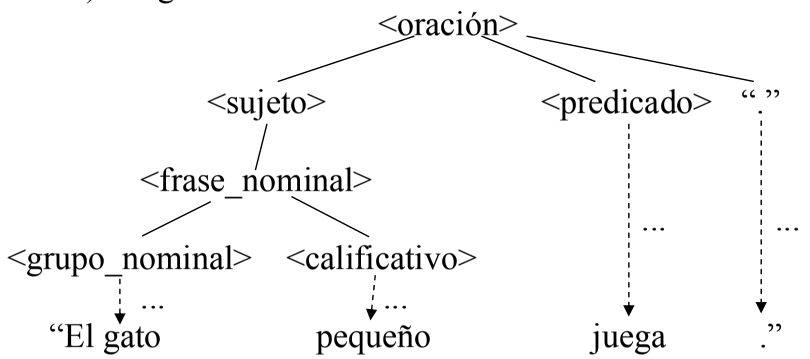
Ejemplo Castellano:
La sintaxis del castellano se define mediante reglas:

1. Un *oración* consta de *sujeto* y *predicado* y termina con un punto.
2. Un *sujeto* es una frase nominal.
3. Una *frase nominal* es un *grupo nominal* seguido de un *calificativo* (que puede faltar)
4. ...

Podemos usar producciones para representar estas reglas:

<oración>::=<sujeto> <predicado>.
<sujeto>::=<frase_nominal>
<frase_nominal>::=<grupo_nominal> <calificativo>
<frase_nominal>::=<grupo_nominal>
...

Podemos analizar si una palabra del lenguaje (en castellano son frases) es "gramaticalmente correcta":



La idea consiste en definir un lenguaje mediante reglas de producción.

Ejemplo lenguaje formal 1

$\Sigma=\{S,A,a,b\}$
$P=\{S::=Ab,\ A::=aAb,\ A::=\lambda\}$

¿Cuáles son todas las posibles derivaciones (irreducibles) desde S?

$S\rightarrow Ab\rightarrow b$
$S\rightarrow Ab\rightarrow aAbb\rightarrow abb$
$S\rightarrow Ab\rightarrow aAbb\rightarrow aaAbbb\rightarrow aabbb$
...
$\{x|S\rightarrow^* x \text{ y } x \text{ es "irreducible"}\}=\{a^n b^{n+1}|n>=0\}$

Ejemplo lenguaje formal 2

$\Sigma=\{S,a,b\}$
$P=\{S::=\lambda,\ S::=aSb\}$
¿ $\{x|S\rightarrow^* x$ y x es "irreducible"$\}$?

Se puede considerar que los elementos de Σ tienen distinta funcionalidad:

S y A – variables ("símbolos no terminales")
a y b – "símbolos terminales"
S – símbolo no terminal de partida ("axioma")

Ejemplo lenguaje formal 3

Definición de la instrucción "if" de un lenguaje de programación.

Definición (Gramática formal):
Se denomina *gramática formal* a una cuádrupla:

$G=(\Sigma_N, \Sigma_T, S, P)$ donde:

- Σ_N es un alfabeto de símbolos no terminales
- Σ_T es un alfabeto de símbolos terminales
- $\Sigma_N \cap \Sigma_T = \varnothing$; llamaremos Σ al conjunto $\Sigma_N \cup \Sigma_T$
- $S \in \Sigma_N$ es el axioma o símbolo inicial
- P es un conjunto de producciones u::=v, tal que:
  - $u \in \Sigma^+$ y $v \in \Sigma^*$
  - $u=xAy$, $x,y \in \Sigma^*$, $A \in \Sigma_N$

Ejemplos:

1. $G_1=(\{S,A\}, \{a,b\},S,P)$
   $P=\{S::=Ab, A::=aAb, A::=\lambda\}$
2. $G=(\Sigma_N, \Sigma_T, S, P)$ no es una gramática si
   - $S \notin \Sigma_N$
   - $u::=v \in P$ con $u \in \Sigma_T^*$
   - $\exists$ a: $a \in \Sigma_T \wedge a \in \Sigma_N$

Definición (gramática en BNF):
Una gramática se encuentra representada en la *forma Backus-Naur* (BNF), si todas las producciones tienen la forma u::=v y todas las producciones con la misma cabeza:

u::=v , u::=w, ...
se representan por:
u::=v | w | ...

3.1.3 Lenguaje asociado a una gramática

Definición (forma sentencial):
Sea una gramática $G=(\Sigma_N, \Sigma_T, S, P)$. Una *forma sentencial* de G es una secuencia de símbolos $x \in \Sigma^*$ tal que $S \rightarrow^* x$.

Ejemplos:
$G=(\{S,A\}, \{a,b\},S,P)$
$P=\{S::=Ab, A::=aAb|\lambda\}$

- Formas sentenciales: S, Ab, aaAbbb, aabbb, b
- No lo son: a, aaAbb

Definición (sentencia):
Una *sentencia* de G es una forma sentencial x tal que $x \in \Sigma_T^*$.

Nota: Usaremos también el término *palabra*.

Definición (lenguaje reconocido):
Dada una gramática $G=(\Sigma_N, \Sigma_T, S, P)$, el *lenguaje reconocido o generado* por G está formado por todas las sentencias (palabras) de G:

$L(G)=\{x | S \rightarrow^* x \text{ y } x \in \Sigma_T^*\}$

Ejemplos:

1. $G_1=(\{S,A\}, \{a,b\},S,P)$ con $P=\{S::=Ab, A::=aAb|\lambda\}$
   $\Rightarrow L(G_1)=\{a^n b^{n+1} | n \geq 0\}$
2. $G_2=(\{S,A\}, \{a,b\},S,P)$ con $P=\{S::=Ab, A::=aAb\}$
   $\Rightarrow$ **¿$L(G_2)$?**
3. $G_3=(\{S,A\}, \{a,b\},S,P)$ con $P=\{S::=Ab, A::=aAb|aS\}$
   $\Rightarrow$ **¿$L(G_3)$?**
4. $G_4=(\{S,A\}, \{a,b\},S,P)$ con $P=\{S::=Ab, A::=ab|a\}$
   $\Rightarrow$ **¿$L(G_4)$?**
5. $G_5=(\{A,B,C,D,E,S\},\{a\},S,P)$ con $P=\{S::=ABaC, Ba::=aaB, BC::=DC|E, aD::=Da, AD::=AB, aE::=Ea, AE::=\lambda\}$
   $\Rightarrow$ **¿$L(G_5)$?**
6. $G_6=(\{S,A\}, \{a,b\},S,P)$ con $P=\{S::=Ab, A::=Aab|a\}$
   $\Rightarrow$ **¿$L(G_6)$?**

3.1.4 Recursividad

Definición (gramática recursiva):

- Una gramática se llama *recursiva en U*, $U \in \Sigma_N$, si existen derivaciones $U \rightarrow^+ xUy$ con $x,y \in \Sigma^*$.
- Si para todas las derivaciones recursivas en U se verifica $x=\lambda$, se dice que la gramática es *recursiva a izquierdas en U*.
- Si para todas las derivaciones recursivas en U se verifica $y=\lambda$, se dice que la gramática es *recursiva a derechas en U*.

Ejemplos:

1. G=({S,A), {a,b},S,P) con P={S::=Ab|Sa, A::=aAb|λ}
   ⇒ recursiva en A y S; recursiva a izquierdas en S
2. G=({S,A), {a,b},S,P) con P={S::=aAb, A::=Sb|λ}
   ⇒ recursiva en A y S; no recursiva a izquierdas en S

Corolarios:

- Lenguajes infinitos sólo pueden describirse mediante gramáticas recursivas.
- Existen gramáticas recursivas que generan lenguajes finitos.

3.1.5 Gramáticas equivalentes

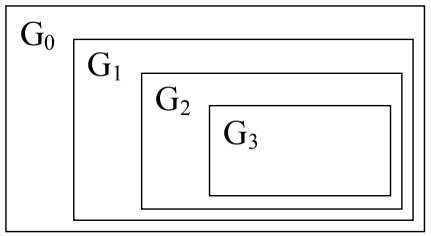
Definición (Gramáticas equivalentes):
Dos gramáticas G_1 y G_2 son *equivalentes* si $L(G_1)=L(G_2)$.
(se escribe $G_1 \equiv G_2$).

Ejemplos:

1. G_1=({S},{a,b},S,{S::=aSb|ab})
   G_2=({A,B,C,E},{a,b},A,P) con
   P={A::=aABC|abC, CB::=BC, bB::=bb, bC::=b, E::=b}
   ⇒ G_1 y G_2 son equivalentes.

3.2 Tipos de gramáticas

3.2.1 Jerarquía de Chomsky



Todas las gramáticas posibles:

$G_3 \subset G_2 \subset G_1 \subset G_0$

$L(G_0)$="lenguajes recursivamente enumerables" o "sin restricciones"
$L(G_1)$="lenguajes sensibles al contexto"
$L(G_2)$="lenguajes independientes del contexto"
$L(G_3)$="lenguajes regulares"

3.2.2 Gramáticas de tipo 0 (gramáticas sin restricciones)

Definición (Gramáticas de tipo 0):
Las *gramáticas de tipo 0* son las definidas anteriormente, cuyas reglas de producción tiene la forma:

$xAy::=v$ con $x,y,v \in \Sigma^*$, $A \in \Sigma_N$

Definición (gramática de estructura de frases):
Una *gramática de estructura de frases* es aquella cuyas producciones tienen la forma:

$xAy::=xvy$ con $x,y,v \in \Sigma^*$, $A \in \Sigma_N$

Nota: v puede ser λ ⇒ la cramática puede tener reglas *compresoras* y *derivaciones decrecientes*.

Propiedades de gramáticas de tipo 0

1. Todo lenguaje descrito por una gramática del tipo 0, puede ser generado por una gramática de estructuras de frases.

Ejemplos:

1. De las siguientes producciones, indicar cuales cumplen el criterio de gramáticas de "estructura de frases" y cuales no.
   - aABcD::=aABD; aAd::=ad; aABd::=aACDEfAd aABE::=aABeE; CB::=BC
2. ¿Cómo convertir CB::=BC en estructura de frases?
   $\Rightarrow$CB::=XB; XB::=XY; XY::=BY; BY::=BC

3.2.3 Gramáticas de tipo 1 (gramáticas sensibles al contexto)

Definición (Gramáticas de tipo 1):

Las *gramáticas de tipo 1* son aquellas cuyas producciones tienen una de las dos formas siguientes:

1. $xAy::=xvy$ con $x,y \in \Sigma^*$, $A \in \Sigma_N$, $v \in \Sigma^+$
2. $S::=\lambda$, siendo $S \in \Sigma_N$ el axioma

Nota:

- Las producciones mantienen el contexto pero no pueden ser compresoras (con la excepción de $S::=\lambda$):
  $\Rightarrow S \rightarrow u_1 \rightarrow u_2 \rightarrow ... \rightarrow u_n$ con $1 \leq |u_1| \leq |u_2| \leq ... \leq |u_n|$
  (si no se utiliza la regla $S::=\lambda$)
- La regla $S::=\lambda$ permite generar la palabra vacía.

Ejemplos:

¿Cuáles de las siguientes reglas son de tipo 1?

- AB::=aCdB; aA::=aCBbb; aBA::=a; ABC::=AC; AaBF::=AaCF

3.2.4 Gramáticas de tipo 2 (gramáticas independientes del contexto)

Definición (Gramáticas de tipo 2):

Las *gramáticas de tipo 2* son aquellas cuyas producciones tienen la forma:

$A::=v$ con $v \in \Sigma^*$, $A \in \Sigma_N$

Nota:

$A::=\lambda$ es una producción de tipo 2.

Sin embargo, se puede demostrar que todo lenguaje generado por una gramática de tipo 2 puede ser generado por una gramática cuyas producciones tienen la forma:

1. $A::=v$ con $v \in \Sigma^+$, $A \in \Sigma_N$
2. $S::=\lambda$ siendo S el axioma de la gramática

(se demuestra en el capítulo 10)

Ejemplos:

¿Cuáles de las siguientes reglas son de tipo 2?

- AA::=b; A::=aaB; aBA::=a; a::=AC; A::=λ

3.2.5 Gramáticas de tipo 3 (gramáticas regulares)

Definición (Gramáticas regulares normalizados):

- *Gramáticas lineales por la izquierda*: todas aquellas cuyas producciones tienen una de las siguientes formas:
  A::=a
  A::=Va (S,A,V$\in\Sigma_N$, a$\in\Sigma_T$, y S el axioma)
  S::=λ
- *Gramáticas lineales por la derecha*: todas aquellas cuyas producciones tienen una de las siguientes formas:
  A::=a
  A::=aV (S,A,V$\in\Sigma_N$, a$\in\Sigma_T$, y S el axioma)
  S::=λ
- Una gramática es *regular (o de tipo 3)* si es lineal por la derecha o lineal por la izquierda.

Ejemplos:

1. ¿Cuáles de las siguientes reglas son de tipo 3?
   - aA::=aaV; A::=abV; a::=Ba;
     a::=C; A::=λ
2. G=({A,B,C},{a,b},A,P) con P={A::=aB, B::=Cb, ...}
   $\Rightarrow$ No es lineal por la derecha ni lineal por la izquierda, pero sí es una *gramática lineal*.

Nota:
Hay definiciones más generales para gramáticas lineales por la derecha (izquierda) [Linz, pp. 91-...]:
Una gramática se denomina lineal por la derecha si sus producciones tienen las siguientes formas:

1. A::=xB
2. A::=x , donde A,B$\in\Sigma_N$ y x$\in\Sigma_T^*$.

(en clase usaremos las definiciones dadas anteriormente)

Ejemplos:

1. Lenguaje representado por G_1=({S,A,B},{0,1},S,P) con P={S::=0A|1, A::=1A|1B, B::=0S}
   $\Rightarrow$S$\rightarrow$1
   S$\rightarrow$0A$\rightarrow$01B$\rightarrow$010S$\rightarrow$0101
   S$\rightarrow$0A$\rightarrow$01A$\rightarrow$011B$\rightarrow$0110S$\rightarrow$01101
   S$\rightarrow$0A$\rightarrow$01A$\rightarrow$011B$\rightarrow$0110S$\rightarrow$01100A$\rightarrow$011001B
   $\rightarrow$0110010S$\rightarrow$01100101
   S$\rightarrow$0A$\rightarrow$*01...1A$\rightarrow$01...1B$\rightarrow$01...1B$\rightarrow$01...10S$\rightarrow$
   01...101
   S$\rightarrow$0A$\rightarrow$*01...1A$\rightarrow$01...1B$\rightarrow$01...1B$\rightarrow$01...10S$\rightarrow$
   01...100A$\rightarrow$*01...1001...1A$\rightarrow$*01...1001...1B$\rightarrow$
   01...1001...10S$\rightarrow$01...1001...101
   L(G)={$(01^i0)^j1|i\geq1, j\geq0$}
2. Lenguaje representado por G_2=({S,B},{0,1},S,P) con P={S::=B1|1, B::=S0}
3. Lenguaje representado por G_3=({S,B},{0,1},S,P) con P={S::=1B|1, B::=0S}

Lenguaje	Gramática			Producciones	
Recursivamente enumerable (sin estricciones)	Tipo0	General		$xAy::=v$	con $x,y,v\in\Sigma^*$ y $A\in\Sigma_N$
		Estructura de frases		$xAy::=xvy$	con $x,y,v\in\Sigma^*$ y $A\in\Sigma_N$
Sensibles al contexto	Tipo1			$xAy::=xvy$ $S::=\lambda$	con $x,y\in\Sigma^*$, $v\in\Sigma^+$ y $A\in\Sigma_N$ S axioma
Independientes del contexto	Tipo2	General		$A::=v$	con $v\in\Sigma^*$ y $A\in\Sigma_N$
		Normalizado		$A::=v$ $S::=\lambda$	con $v\in\Sigma^+$ y $A\in\Sigma_N$ S axioma
Regulares	Tipo3	Lineal por la izquierda	General	$A::=x$ $A::=Bx$	con $x\in\Sigma_T^*$ y $A,B\in\Sigma_N$
			Normalizado	$A::=a$ $A::=Ba$ $S::=\lambda$	con $a\in\Sigma_T$ y $A,B\in\Sigma_N$ S axioma
		Lineal por la derecha	General	$A::=x$ $A::=xB$	con $x\in\Sigma_T^*$ y $A,B\in\Sigma_N$
			Normalizado	$A::=a$ $A::=aB$ $S::=\lambda$	con $a\in\Sigma_T$ y $A,B\in\Sigma_N$ S axioma

3.3 Árboles de derivación

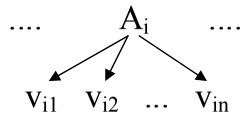
3.3.1 Representación arborescente de derivaciones

Definición (Árbol de derivación):

Sea $G=(\Sigma_N,\Sigma_T, S, P)$ una gramática de estructura de frases y w una forma sentencial de G, que se obtiene mediante una derivación: $u_0=S\rightarrow u_1\rightarrow\ldots\rightarrow u_n=w$

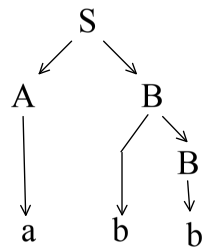
El árbol de derivación de w correspondiente a dicha derivación cumple las siguientes propiedades:

1. La raíz del árbol es S
2. Cada derivación directa se representa mediante un conjunto de ramas que salen de un nodo dado de la siguiente forma:
   Si $u_i\rightarrow u_{i+1}$ es una derivación directa en la secuencia, tal que:
   $u_i= x_iA_iy_i$ con $A_i\in\Sigma_N$, $x_i,y_i\in\Sigma^*$
   $u_{i+1}= x_iv_iy_i$ con $v_i\in\Sigma^*$ y $v_i=v_{i1}\ldots v_{in}$
   entonces



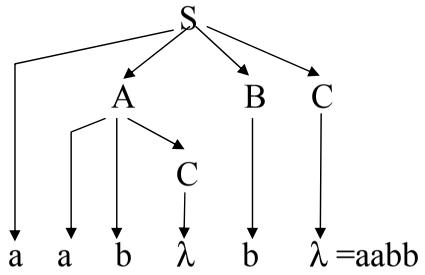
Ejemplos:

1. $G_1=$ ({a,b} {A,B,S}, S, { S::=AB, A::= aA | a, B::= bB|b})
   y la derivación $S \rightarrow AB \rightarrow aB \rightarrow abB \rightarrow abb$
   $\Rightarrow$



2. $G_2=(\{A,B,C\},\{a,b\},A,P)$ con

P={A::=aABC|abC, bCB::=bB, bB::=bb, bC::=b}
S→aABC→aabCBC→aabBC→aabbC→aabb
⇒



3. Si tenemos una gramática en forma general de tipo 0, entonces ¿es posible representa las derivaciones en forma de árbol?

Notas: sobre árboles de derivación

- Todos los nodos interiores corresponden a símbolos no terminales.
- Si un nodo λ es hijo de un nodo padre, este padre no puede tener otros hijos.
- El *resultado* de un árbol de derivación es el conjunto de nodos hoja de izquierda a derecha y corresponde a una forma sentencial (se eliminan los símbolos λ sobrantes).
- Si todas las hojas del árbol son símbolos terminales o λ, entonces el resultado es una sentencia (o palabra).
- Un árbol de derivación cuya raíz no es el axioma de la gramática es un *árbol parcial*.

Más ejemplos (escriba los árboles de derivación):

4. G_3=({S},{0,1},S,P) con P={S::=SS|λ|0S|1S}
   S→SS→0SS→0S1S→00S1S→001S→001
5. G_3 como en 4.
   S→SS→S1S→0S1S→0S1→00S1→001

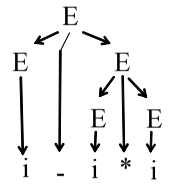
En general se puede afirmar que: **Dada una derivación, existe un único árbol que la representa**.

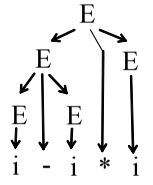
Por el contrario: **Un mismo árbol puede corresponder a varias derivaciones** (ejemplos 5 y 6 anteriores).

Está última afirmación no es válida para gramáticas lineales (que incluyen las gramáticas del tipo 3). **¿Por que?**
Consideramos: G=({S,A},{0,1},S,{S::=0S|1A|λ, A::=1A|1S})

3.3.2 Ambigüedad

Ejemplo:
(Precedencia de operadores)
G_1=({E},{-,*,i, (,)},E,{E::=E-E|E*E|(E)|i})
⇒ x=i+i*i





E→E-E→i-E
→i-E*E→i-i*E→i-i*i

E→E*E→E*i
→E-E*i→i-E*i→i-i*i

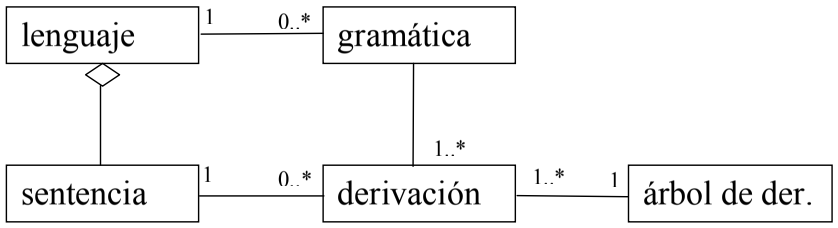
Dos posibles interpretaciones: (i-i)*i y i-(i*i) (ambigüedad)

Definiciones (ambigüedad):

- Sea una gramática G. Una sentencia $x \in L(G)$ es *ambigua* si puede obtenerse por medio de varias derivaciones distintas correspondientes a árboles de derivación diferentes.
- Una gramática es *ambigua* si genera alguna sentencia ambigua.
- Un *lenguaje* es *inherentemente ambiguo* si no existe una gramática que la describe y que no sea ambigua.

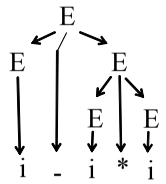
Notas:

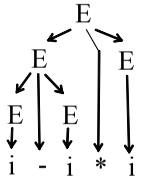
- La ambigüedad es una propiedad indeseada en lenguajes de programación (véase el ejemplo anterior). Cada instrucción debe tener sólo una interpretación.
- No existen algoritmos generales que nos digan si una gramática dada es ambigua (un problema no decidible). Tan sólo se pueden encontrar condiciones suficientes para asegurar que una gramática no es ambigua.
- El hecho de que la gramática que describe un lenguaje sea ambigua no implica necesariamente que el lenguaje sea inherentemente ambiguo
- Existen lenguajes que son inherentemente ambiguos (que sólo se pueden describir, por ejemplo, con gramáticas independientes del contexto ambiguas).



La ambigüedad puede presentar un problema en los lenguajes de programación. Por ejemplo, para calcular el valor de una expresión el árbol de derivación especifica en que orden se resuelven las operaciones (de abajo a arriba).

La gramática G_1 genera dos árboles de derivación distintas para la palabra x=i-i*i:

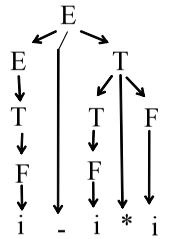




$\Rightarrow$ cada uno daría un valor distinto a la expresión

Se pude modificar la gramática G_1 para obtener una gramática equivalente que no es ambigua (evitando así la doble interpretación):

$G_2=(\{E,T,F\},\{-,*,i,(,)\},E,\{E::=E-T|T,\ T::=T*F|F,\ F::=(E)|i\})$
$\Rightarrow$ solo existe ún árbol para la palabra x=i-i*i



$E \rightarrow E-T \rightarrow i-T \rightarrow i-T*F \rightarrow i-F*F \rightarrow i-i*F \rightarrow i-i*i$

Para dar preferencia a la adicción hay que especificar la palabra como (i-i)*i. **¿Cómo sería el árbol de derivación?**

Para desambiguar la gramática de las expresiones (G_1) se usan dos ideas:

1. agrupación en una dirección (en este caso a la izquierda):
   Cadenas de operandos para un mismo operador se generan de la derecha hasta la izquierda.
   $\Rightarrow$i-i-i-i
   E::=E-T
   (eso implica que se calculan las expresiones de izquierda a derecha)
2. reglas de preferencia (* tiene preferencia sobre -)
   Los operadores se generan en distintos "niveles".
   - Desde el axioma se genera el operador de menor preferencia (este va a ser más alto en el árbol de derivación).
   - Sucesivamente, se generan operadores de mayor preferencia desde "niveles más bajos" (variables a los que sólo se llegan a través de varias derivaciones).
     $\Rightarrow$ Orden: E>T>F
   - Los operandos de una expresión de un nivel de preferencia sólo pueden ser expresiones del mismo nivel o de un nivel más bajo.
     $\Rightarrow$ E::=E-T | T
     T::=T*F | F
   - Para que el operando de una operación sea otra operación con un operador de menor nivel es necesario usar los símbolos de cambio de preferencia "(" y ")".
     $\Rightarrow$ F::=(E)

Ejemplos:

1. Una gramática para el lenguaje:
   $L=\{x|\ x\in \{a,b\}^* \text{ y } n_a(x)= n_b(x)\}$

   - Primer intento:
     $G_1=(\{S\},\{a,b\},S,\{S::=\lambda|aSb|bSa|SS\})$
     Es una gramática ambigua (véase la sentencia abab).
   - Segundo intento:
     $G_2=(\{S\},\{a,b\},S,\{S::=\lambda|aSbS|bSaS\})$
     Siegue siendo ambigua (véase la sentencia abab)

   ¿Es L inherentemente ambiguo? NO.
   $\Rightarrow$ $G_3=(\{S,T,U\},\{a,b\},S,P)$ con
   $P=\{S::=aTbS|bUaS|\lambda,\ T::=aTbT|\lambda,\ U::=bUaU|\lambda\}$
   ¿Cómo es el árbol de derivación para abab?
   (se puede demostrar que G_3 no es ambigua)

2. Un lenguaje inherentemente ambiguo:
   $L=\{a^nb^nc^m|n,m\geq 0\}\cup\{a^nb^mc^m|n,m\geq 0\}$
   Se puede definir L como la unión de dos lenguajes $L_1\cup L_2$.

   $\Rightarrow$Gramática para L_1:
   $G_4=(\{S_1,A\},\{a,b,c\},S_1,\{S_1::=S_1c|A,\ A::=aAb|\lambda\})$
   $\Rightarrow$Gramática para L_2:
   $G_5=(\{S_2,B\},\{a,b,c\},S_2,\{S_2::=aS_2|B,\ B::=bBc|\lambda\})$
   $\Rightarrow$Gramática para L (uniendo G_4 y G_5):
   $G_6=(\{S_2,S_1,S,A,B\},\{a,b,c\},S,P)$ con $P=\{S::=S_1|S_2,$
   $S_2::=aS_2|B,\ B::=bBc|\lambda,\ S_1::=S_1c|A,\ A::=aAb|\lambda\})$

   Consideramos las palabras $a^nb^nc^n$ para cualquier n.
   (se puede demostrar que L es inherentemente ambiguo)