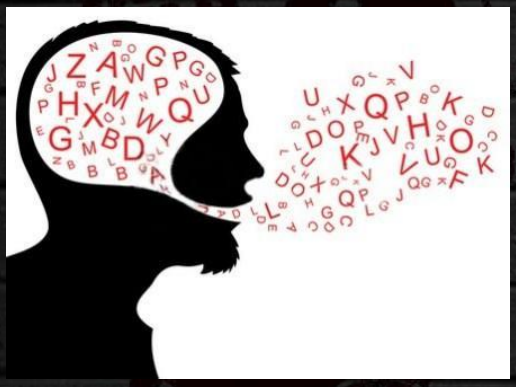
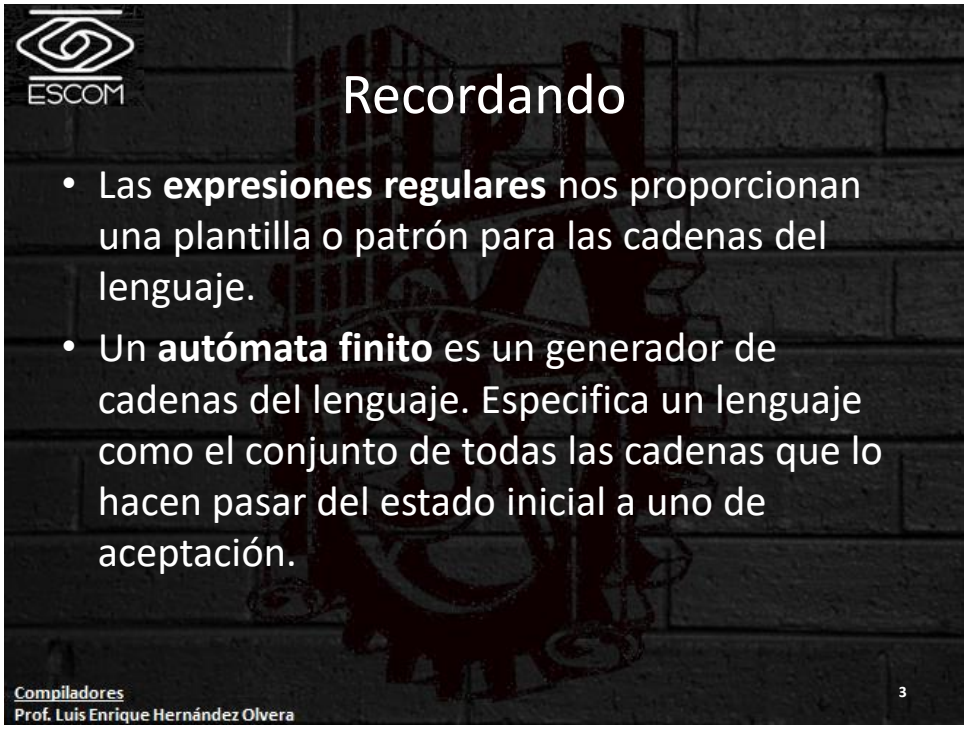
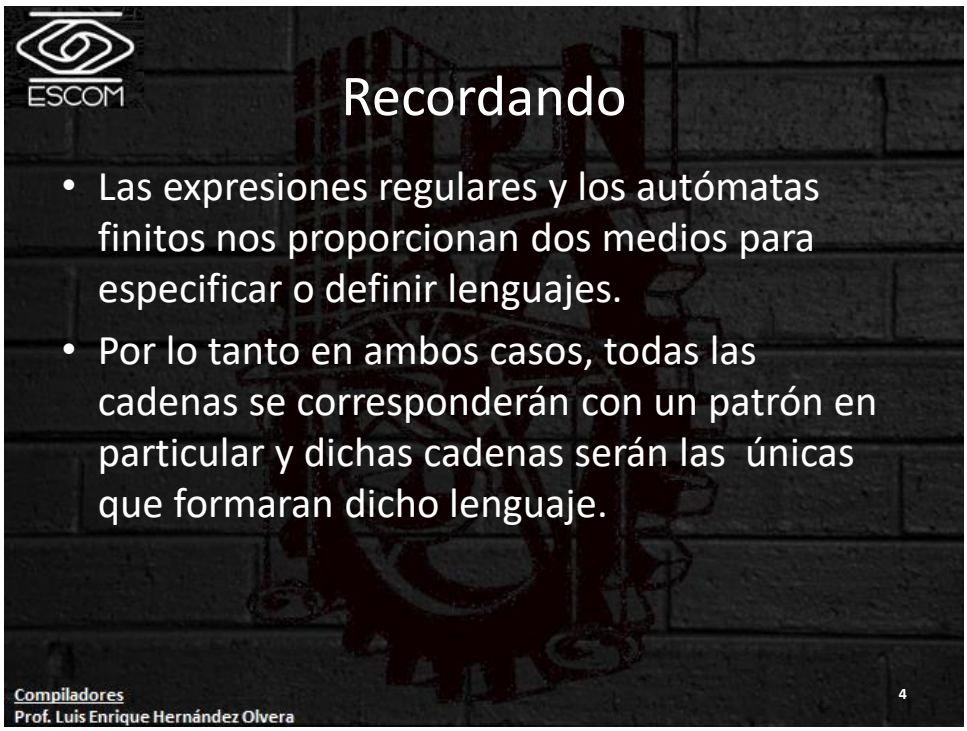
Gramáticas

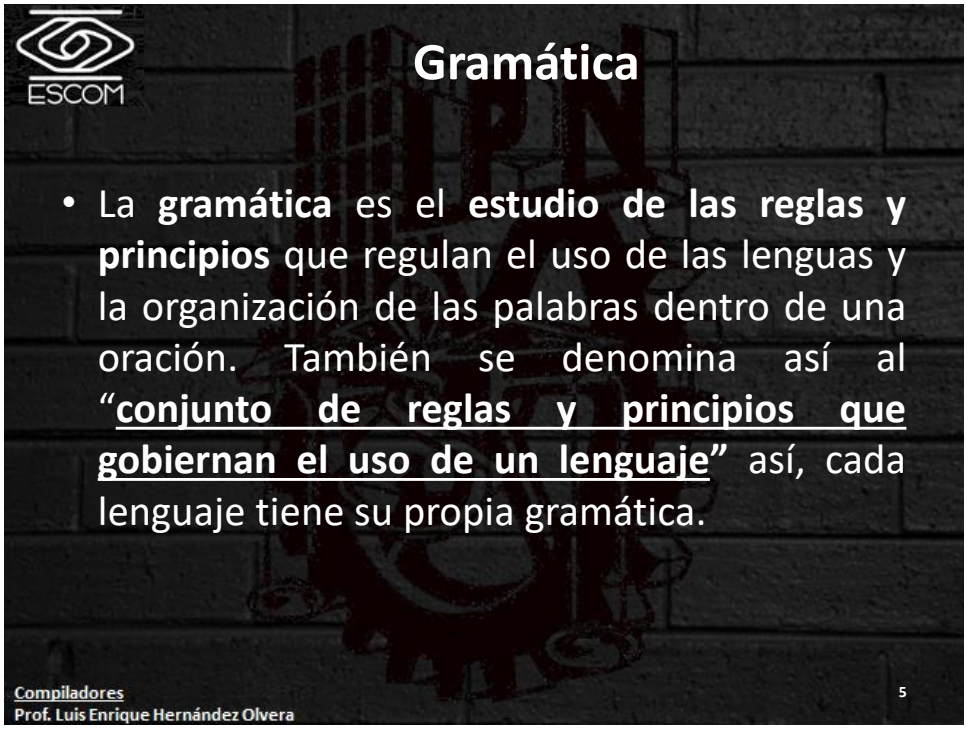


Contenido

- Definición de gramática.
- Elementos de una gramática.
- Clasificación de gramáticas.
- Gramáticas independientes del contexto.
- Derivaciones y arboles de derivación.
- Gramáticas ambiguas.
- Eliminación de la recursividad por la izquierda.

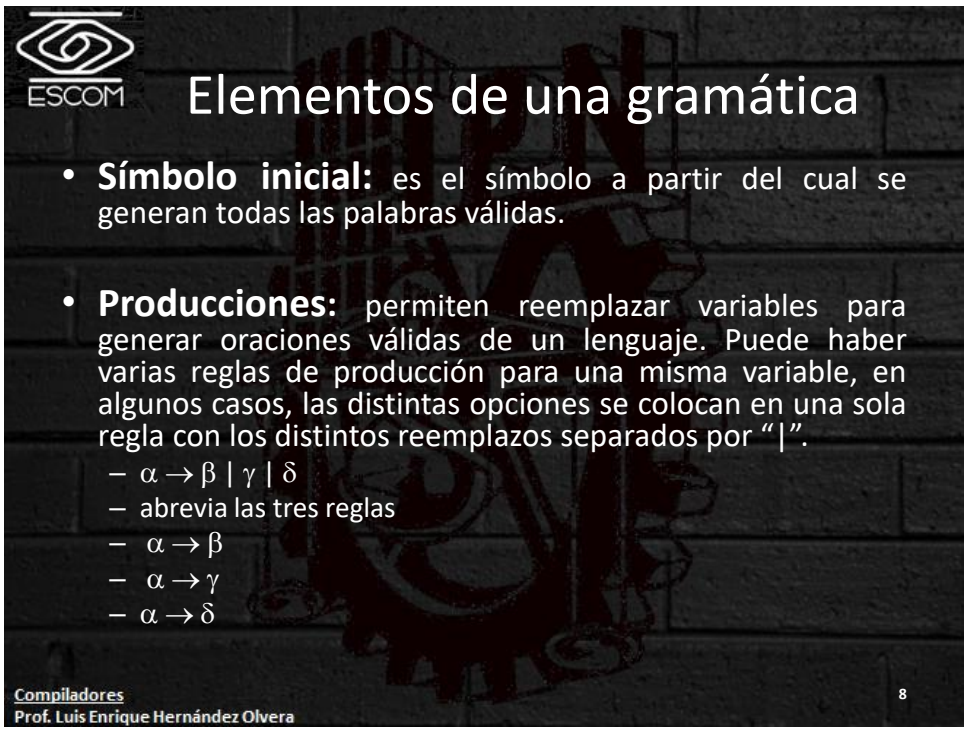

ESCOM
Recordando
• Las **expresiones regulares** nos proporcionan una plantilla o patrón para las cadenas del lenguaje.
• Un **autómata finito** es un generador de cadenas del lenguaje. Especifica un lenguaje como el conjunto de todas las cadenas que lo hacen pasar del estado inicial a uno de aceptación.
Compiladores
Prof. Luis Enrique Hernández Olvera
3

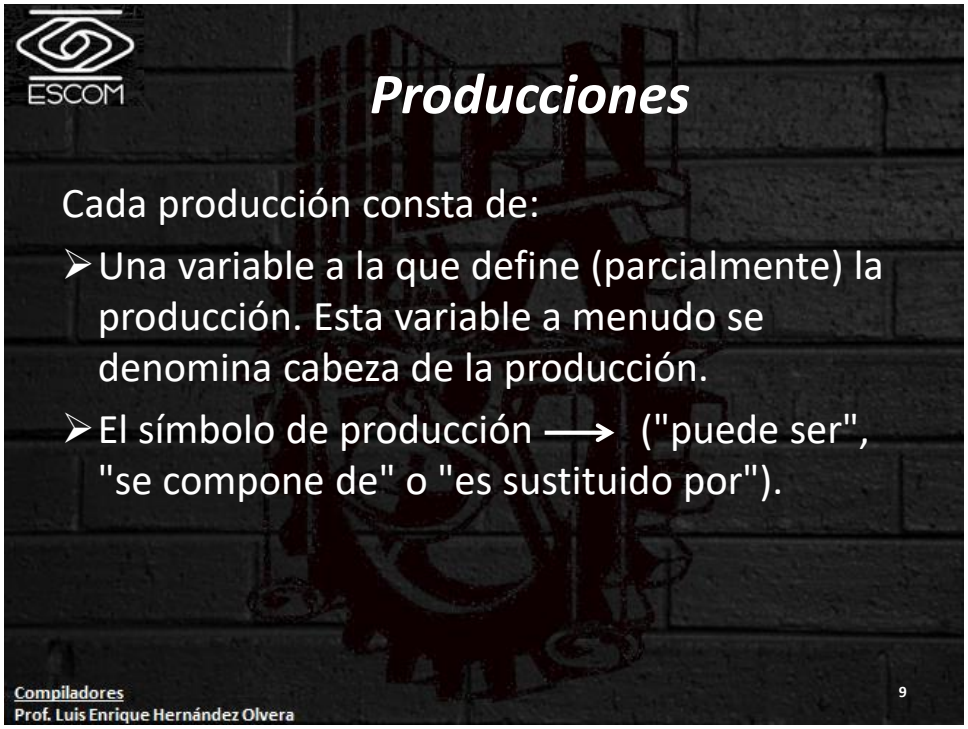

ESCOM
Recordando
• Las expresiones regulares y los autómatas finitos nos proporcionan dos medios para especificar o definir lenguajes.
• Por lo tanto en ambos casos, todas las cadenas se corresponderán con un patrón en particular y dichas cadenas serán las únicas que formaran dicho lenguaje.
Compiladores
Prof. Luis Enrique Hernández Olvera
4

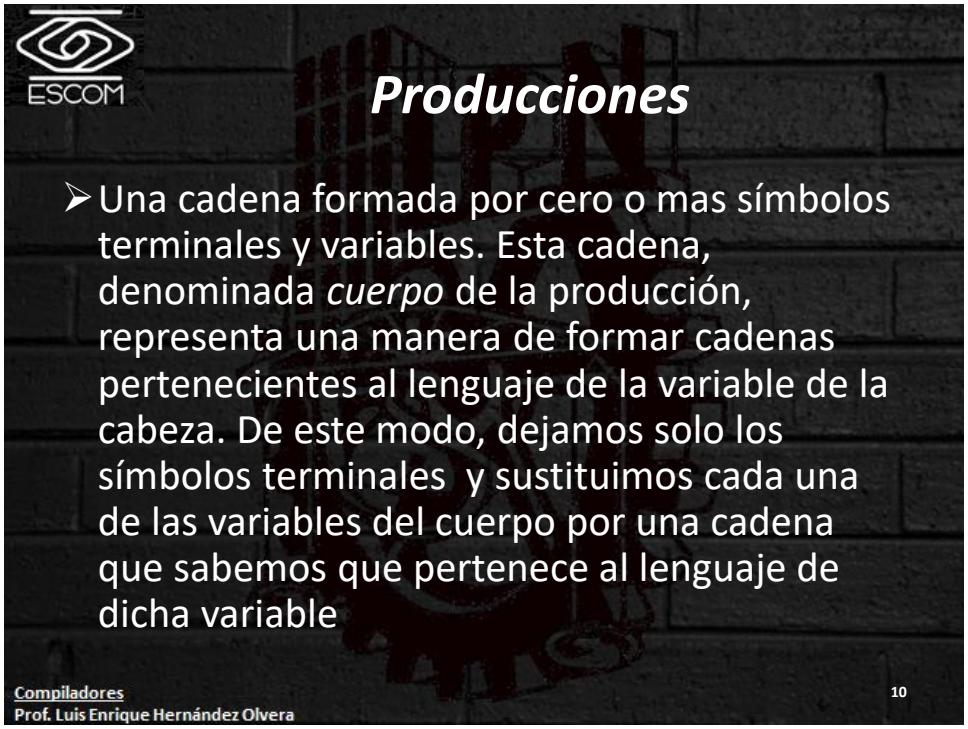

ESCOM
Gramática
• La **gramática** es el **estudio de las reglas y principios** que regulan el uso de las lenguas y la organización de las palabras dentro de una oración. También se denomina así al "**conjunto de reglas y principios que gobiernan el uso de un lenguaje**" así, cada lenguaje tiene su propia gramática.
Compiladores
Prof. Luis Enrique Hernández Olvera
5


ESCOM
Gramática
Una Gramática G se representa con una cuádrupla:
G=(N,Σ,S,P)
Donde:
• N es una colección finita de no terminales.
• Σ es un alfabeto (Conjunto de terminales).
• S es un no terminal llamado **Símbolo inicial.**
• P es una colección finita de **reglas de sustitución** llamadas **producciones**.
Compiladores
Prof. Luis Enrique Hernández Olvera
6


ESCOM
Elementos de una gramática
• Símbolos terminales: son elementos del alfabeto que no se pueden transformar, por eso se llaman terminales. Normalmente se denotan por letras minúsculas.
• Variables o símbolos no terminales: son elementos auxiliares que permiten poner restricciones sintácticas a un lenguaje. Las variables sí se pueden transformar, utilizando las reglas, en una cadena de variables y/o terminales. Por lo general se denotan por letras mayúsculas o por la notación <variable>.
Compiladores
Prof. Luis Enrique Hernández Olvera
7


ESCOM
Elementos de una gramática
• Símbolo inicial: es el símbolo a partir del cual se generan todas las palabras válidas.
• Producciones: permiten reemplazar variables para generar oraciones válidas de un lenguaje. Puede haber varias reglas de producción para una misma variable, en algunos casos, las distintas opciones se colocan en una sola regla con los distintos reemplazos separados por "|".
– α → β | γ | δ
– abrevia las tres reglas
– α → β
– α → γ
– α → δ
Compiladores
Prof. Luis Enrique Hernández Olvera
8


ESCOM
Producciones
Cada producción consta de:
Una variable a la que define (parcialmente) la producción. Esta variable a menudo se denomina cabeza de la producción.
El símbolo de producción ⟶ ("puede ser", "se compone de" o "es sustituido por").
Compiladores
Prof. Luis Enrique Hernández Olvera
9


ESCOM
Producciones
Una cadena formada por cero o mas símbolos terminales y variables. Esta cadena, denominada *cuerpo* de la producción, representa una manera de formar cadenas pertenecientes al lenguaje de la variable de la cabeza. De este modo, dejamos solo los símbolos terminales y sustituimos cada una de las variables del cuerpo por una cadena que sabemos que pertenece al lenguaje de dicha variable
Compiladores
Prof. Luis Enrique Hernández Olvera
10

Ejemplo 1: Gramática

- **G = (N, Σ, S, P)**
  - **N** = {*S, A, B*}
  - **Σ** = {*a, b, c*}
  - **P**: $S \rightarrow AccA$ $A \rightarrow BA \mid \lambda$ $B \rightarrow a \mid b \mid c$

w1 = abcc $\in$ **L(G)**
w2 = acb $\notin$ **L(G)**

Ejemplo 1: Gramática

- **G = (N, Σ, S, P)**
  - **N** = {*S, A, B*}
  - **Σ** = {*a, b, c*}
  - **P**: $S \rightarrow AccA$ $A \rightarrow BA \mid \lambda$ $B \rightarrow a \mid b \mid c$

w1 = abcc $\in$ **L(G)**
w2 = acb $\notin$ **L(G)**

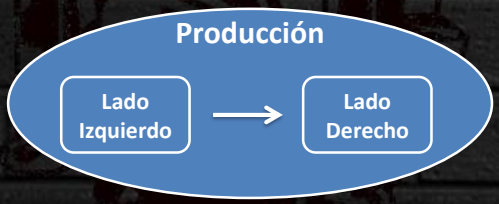
- $S \Rightarrow AccA \Rightarrow BAccA \Rightarrow aAccA \Rightarrow aBAccA \Rightarrow abAccA \Rightarrow ab\lambda ccA \Rightarrow abcc\lambda \Rightarrow abcc$

Lenguaje generado por una gramática

- El lenguaje *L(G)* generado por una gramática *G* es el conjunto de todas las **cadenas** que puede generar *G*.
- Una **cadena** pertenece a L(G) si :
  - Está compuesta de símbolos terminales
  - La cadena puede derivarse del símbolo inicial S aplicando las reglas de producción de la gramática.

Gramáticas Formales

- Una gramática formal consta de un conjunto finito de *símbolos terminales* (las palabras en un lenguaje formal), un conjunto finito de *símbolos no terminales*, un conjunto de *reglas de producción* con un lado izquierdo y otro derecho, y un *símbolo inicial (No terminal "S")*.



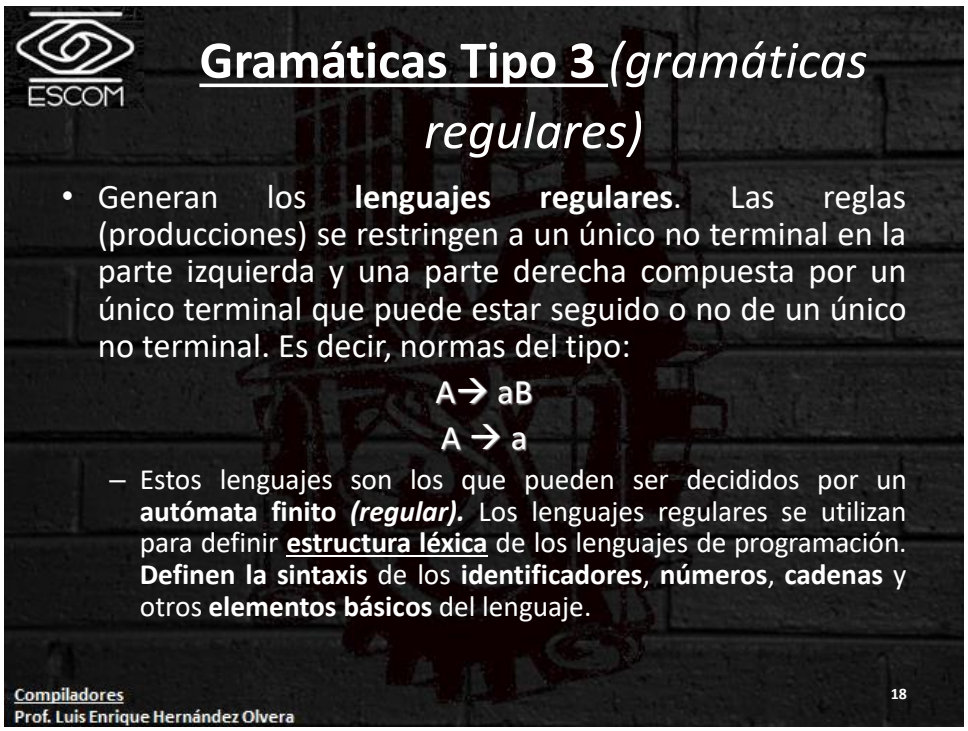
Gramáticas Formales

- Las reglas se aplican sustituyendo la parte de la izquierda por la parte de la derecha. Una derivación es una secuencia de aplicaciones de reglas.
- Cada gramática define el lenguaje formal de todas las sentencias que están formadas exclusivamente por los símbolos terminales a los que se puede llegar mediante derivación a partir del símbolo inicial.

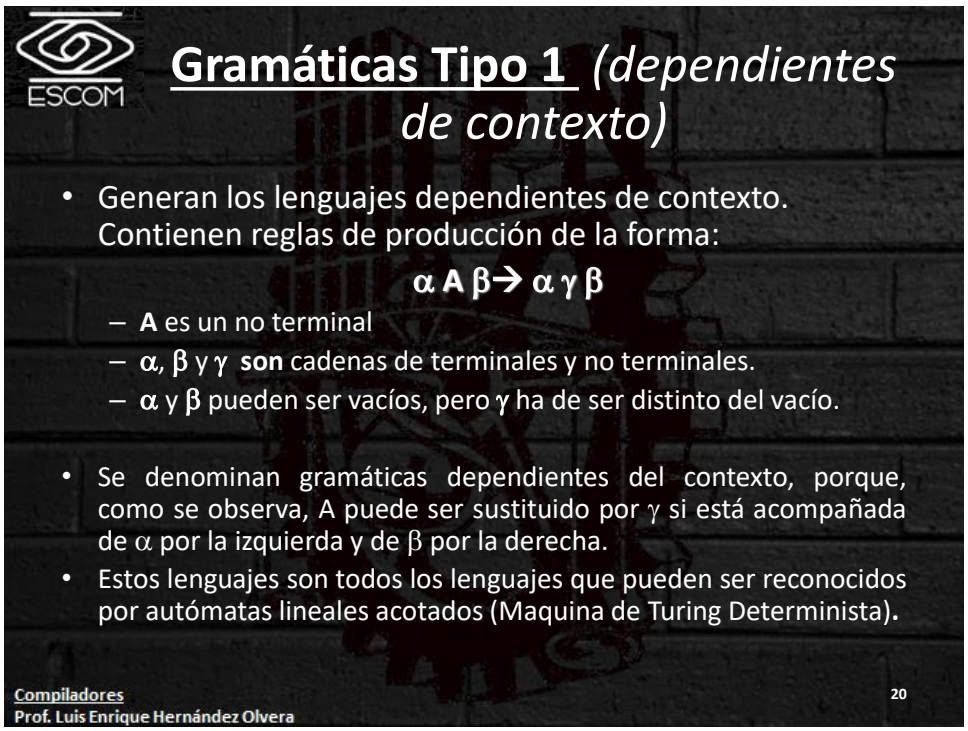
Clasificación de Gramáticas

- Los cuatro tipos son:


ESCOM
Clasificación de Gramáticas
• En función de la forma de sus producciones, se puede caracterizar qué tan compleja es una gramática formal.
– Gramáticas Tipo 0 (sin restricciones)
– Gramáticas Tipo 1 (dependientes de contexto)
– Gramáticas Tipo 2(independientes o libres de contexto)
– Gramáticas Tipo 3 (gramáticas regulares)
Compiladores
Prof. Luis Enrique Hernández Olvera
17


ESCOM
Gramáticas Tipo 3 (gramáticas regulares)
• Generan los lenguajes regulares. Las reglas (producciones) se restringen a un único no terminal en la parte izquierda y una parte derecha compuesta por un único terminal que puede estar seguido o no de un único no terminal. Es decir, normas del tipo:
A→ aB
A → a
– Estos lenguajes son los que pueden ser decididos por un autómata finito (regular). Los lenguajes regulares se utilizan para definir estructura léxica de los lenguajes de programación. Definen la sintaxis de los identificadores, números, cadenas y otros elementos básicos del lenguaje.
Compiladores
Prof. Luis Enrique Hernández Olvera
18


ESCOM
Gramáticas Tipo 2(independientes o libres de contexto)
• Generan los lenguajes libres de contexto. Están definidas por reglas de la forma:
A → γ
– A es un no terminal
– γ es una cadena de terminales y no terminales.
• Se denominan independientes de contexto porque A puede sustituirse por γ independientemente de las cadenas por las que esté acompañada.
• Estos lenguajes son todos los lenguajes que pueden ser reconocidos por los autómatas de pila.
• Los lenguajes independientes de contexto constituyen la base teórica para la sintaxis de la mayoría de los lenguajes de programación. Definen la sintaxis de las declaraciones, las proposiciones, las expresiones, etc.(i.e. la estructura
Compiladores
Prof. Luis Enrique Hernández Olvera
19

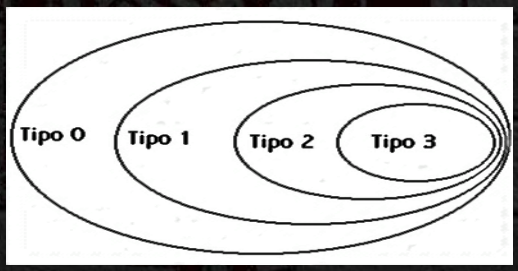

ESCOM
Gramáticas Tipo 1 (dependientes de contexto)
• Generan los lenguajes dependientes de contexto. Contienen reglas de producción de la forma:
α A β → α γ β
– A es un no terminal
– α, β y γ son cadenas de terminales y no terminales.
– α y β pueden ser vacíos, pero γ ha de ser distinto del vacío.
• Se denominan gramáticas dependientes del contexto, porque, como se observa, A puede ser sustituido por γ si está acompañada de α por la izquierda y de β por la derecha.
• Estos lenguajes son todos los lenguajes que pueden ser reconocidos por autómatas lineales acotados (Maquina de Turing Determinista).
Compiladores
Prof. Luis Enrique Hernández Olvera
20

Gramáticas Tipo 0 *(sin restricciones)*

- Incluyen todas las gramáticas formales.
- El más general, al que pertenece la semántica de los lenguajes naturales y artificiales.
- A estos lenguajes no se les impone restricción alguna.
- Estos lenguajes son todos los lenguajes que pueden ser reconocidos por una *máquina de Turing.*

Clasificación de Gramáticas

- Todo lenguaje de tipo 3 es de tipo 2, todo lenguaje de tipo 2 es de tipo 1, y todo lenguaje de tipo 1 es de tipo 0.
- Se dice que un lenguaje es de tipo k [k = 0, k = 1, k = 2, k = 3] cuando existe una gramática de tipo k que genera ese lenguaje.



Clasificación de Gramáticas

- Para clasificar una gramática hemos de analizar una a una todas sus reglas de producción obteniendo el tipo de cada una de ellas. La clasificación de la gramática será la correspondiente al tipo de la producción de menor clasificación

Gramática	Lenguaje	Reglas de Producción	Si $\mu \rightarrow \varphi$, relación entre $\lvert\mu\rvert$ y $\lvert\varphi\rvert$	Solución
Tipo-0	Recursivas	Sin restricciones		Máquinas de Turing
Tipo-1	Dependiente de contexto	$\alpha A\beta \rightarrow \alpha\gamma\beta$	$\lvert\mu\rvert \leq \lvert\varphi\rvert$	Autómatas lineales acotados
Tipo-2	Independiente de contexto	$A \rightarrow \gamma$	$\lvert\mu\rvert = 1$	Autómatas de pila
Tipo-3	Regular	$A \rightarrow aB$ $A \rightarrow a$	$\lvert\mu\rvert = 1$	Autómatas finitos, regulares

Gramática independiente del contexto (GIC)

Una "Gramática independiente del contexto" (GIC) es una cuádrupla:

$$G=(N,\Sigma,S,P)$$

¡RECUERDA! El orden de los argumentos cambia según la referencia.

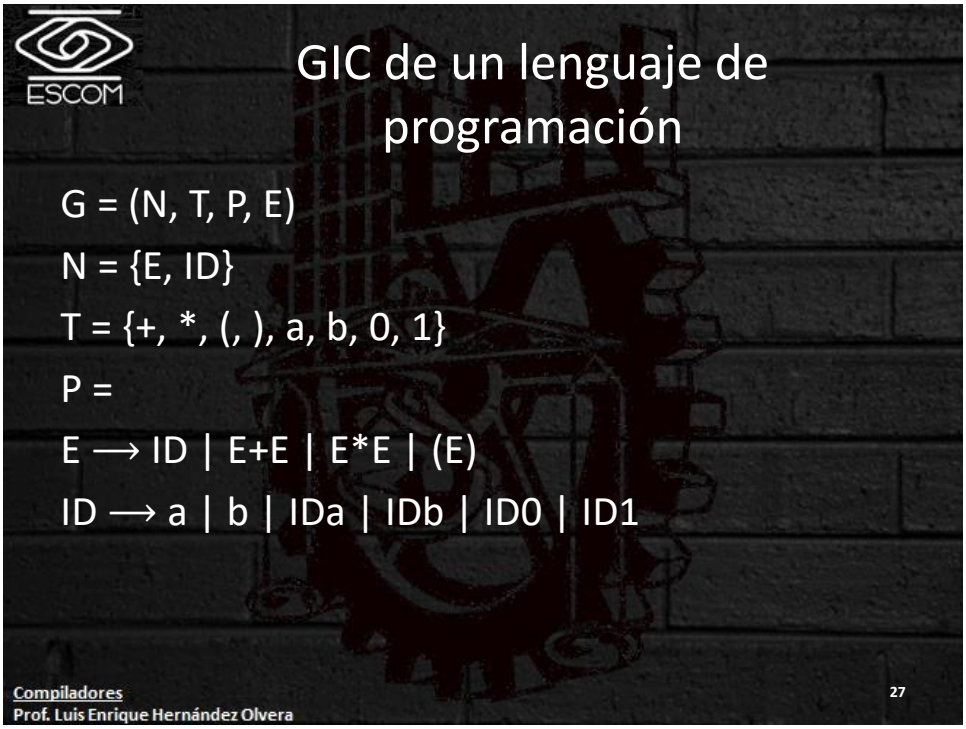
Donde:

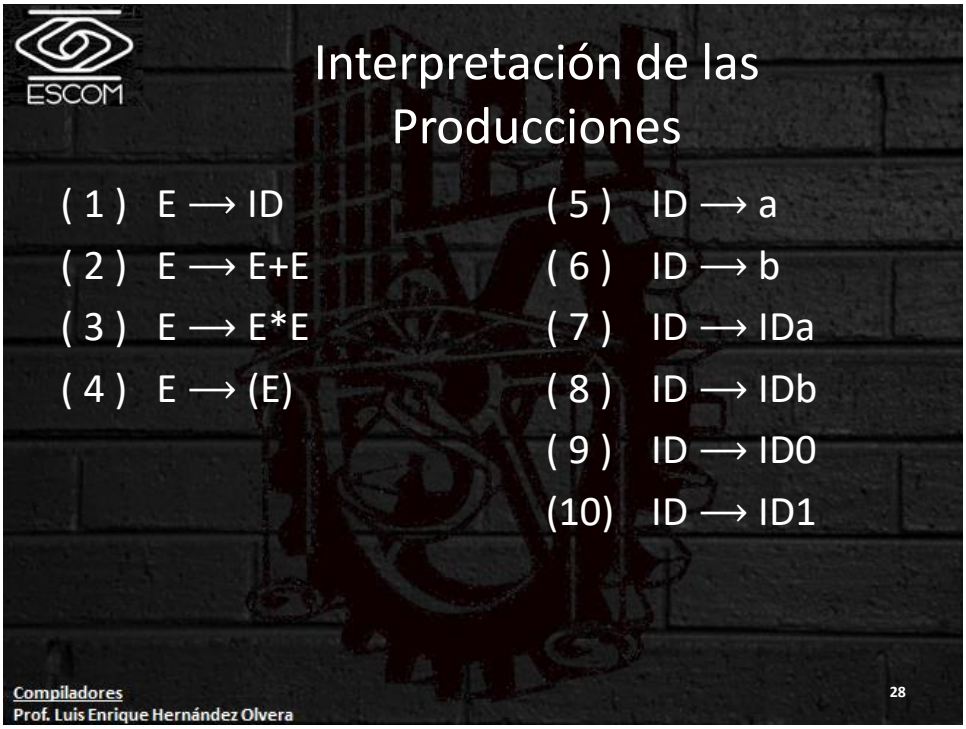
- N es una colección finita de no terminales.
- Σ es un alfabeto (Conjunto de terminales).
- S es un no terminal llamado **Símbolo inicial.**
- P es una colección finita de **reglas de sustitución** llamadas **producciones**.

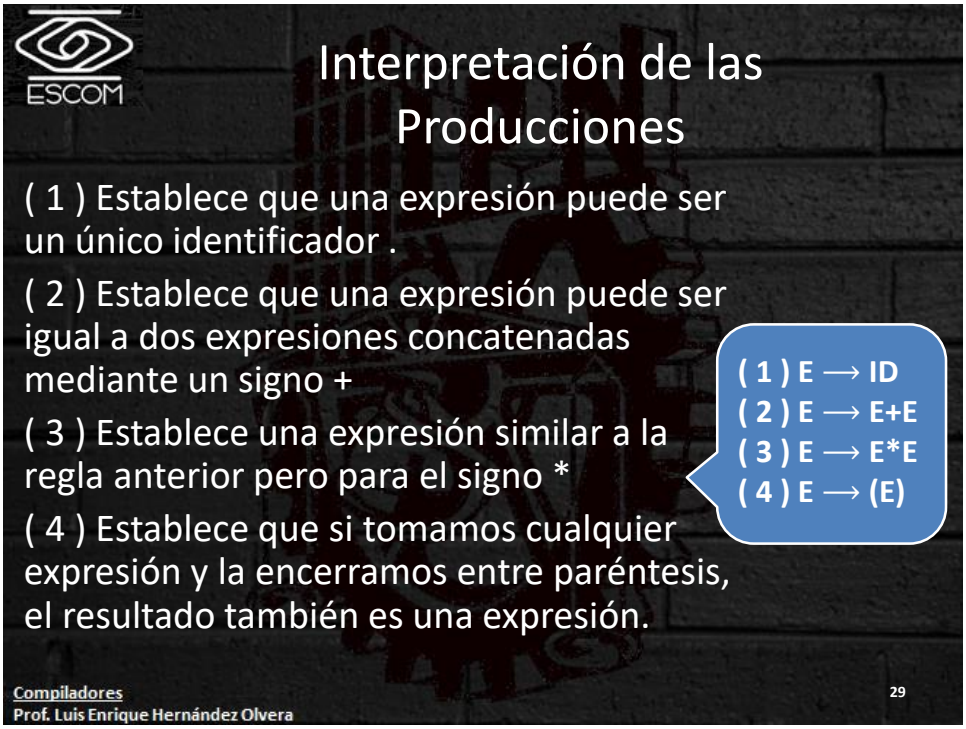
El lenguaje generado por la GIC G se denota por L(G) y se llama lenguaje independiente del contexto (LIC).

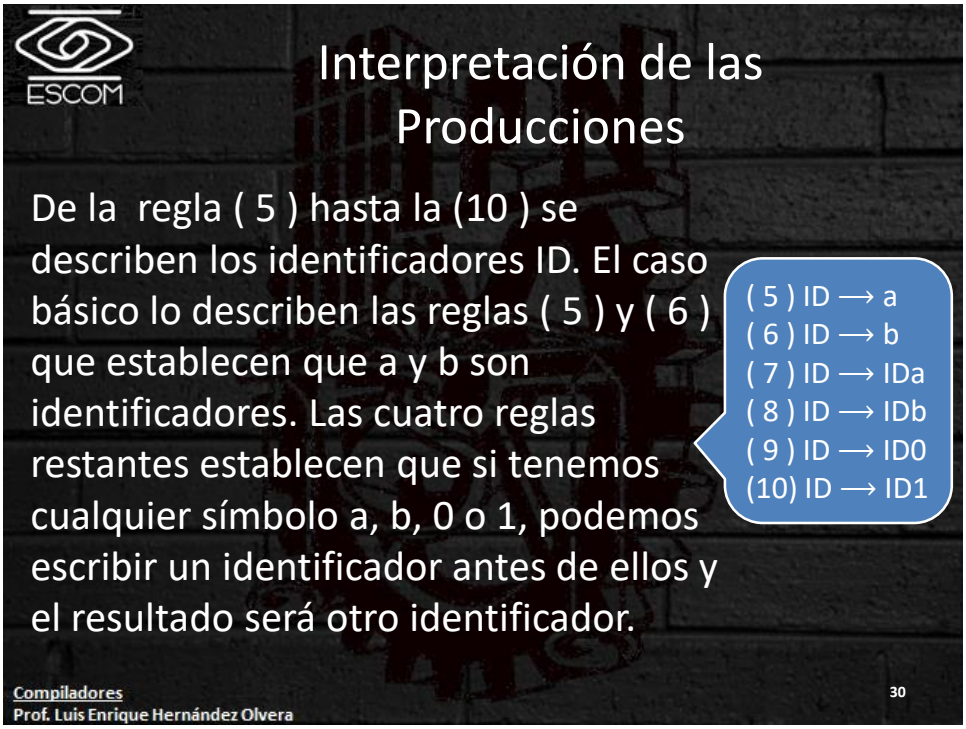
GIC de un lenguaje de programación

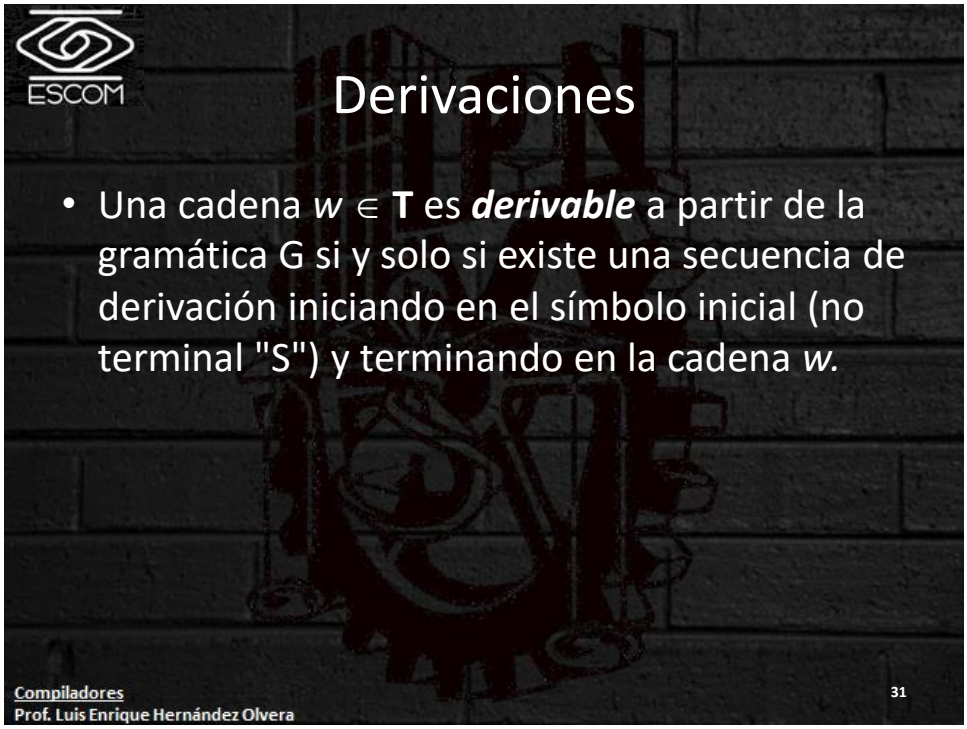
- Para un ejemplo simplificado nos limitaremos a los operadores de + y *, los identificadores serán letras seguidas por cero o más letras y dígitos (a|b)(a|b|0|1)*.
- Se necesitaran 2 variables ("E" e "ID").
  - E representa las expresiones (el lenguaje de las expresiones que se van a definir).
  - ID representa los identificadores.

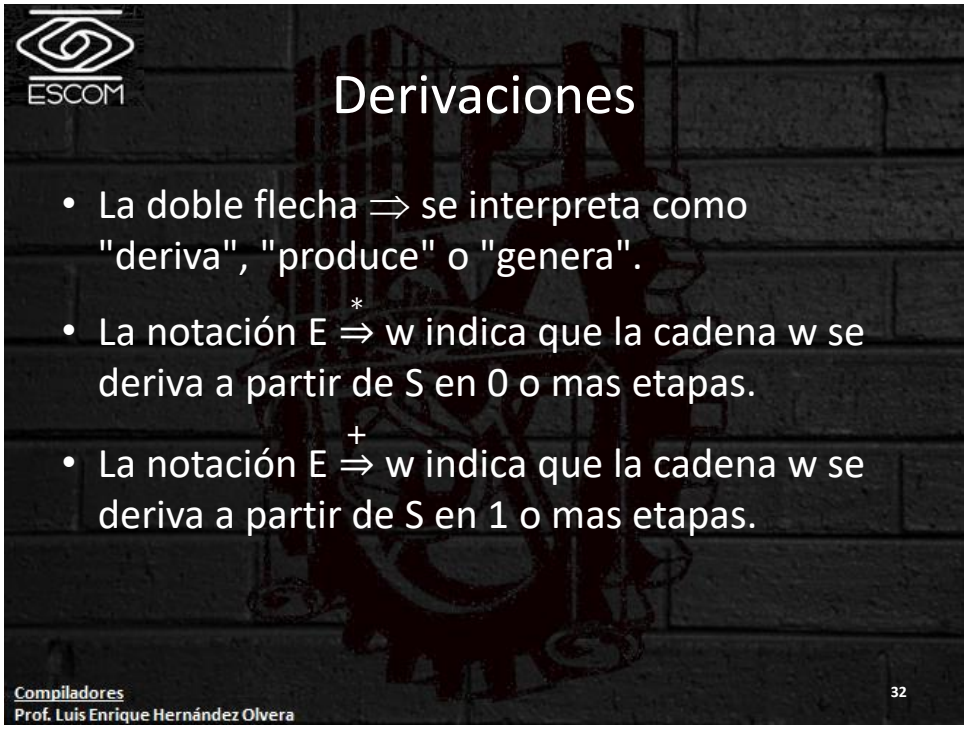

ESCOM
GIC de un lenguaje de programación
G = (N, T, P, E)
N = {E, ID}
T = {+, *, (,), a, b, 0, 1}
P =
E ⟶ ID | E+E | E*E | (E)
ID ⟶ a | b | IDa | IDb | ID0 | ID1
Compiladores
Prof. Luis Enrique Hernández Olvera
27

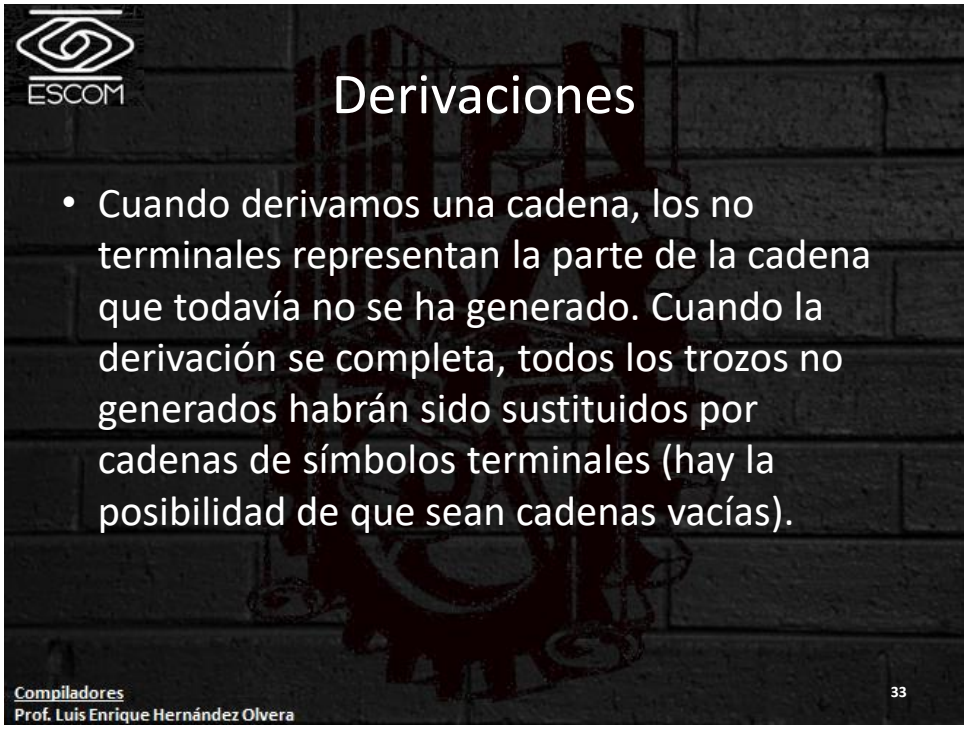

ESCOM
Interpretación de las Producciones
(1) E ⟶ ID
(2) E ⟶ E+E
(3) E ⟶ E*E
(4) E ⟶ (E)
(5) ID ⟶ a
(6) ID ⟶ b
(7) ID ⟶ IDa
(8) ID ⟶ IDb
(9) ID ⟶ ID0
(10) ID ⟶ ID1
Compiladores
Prof. Luis Enrique Hernández Olvera
28

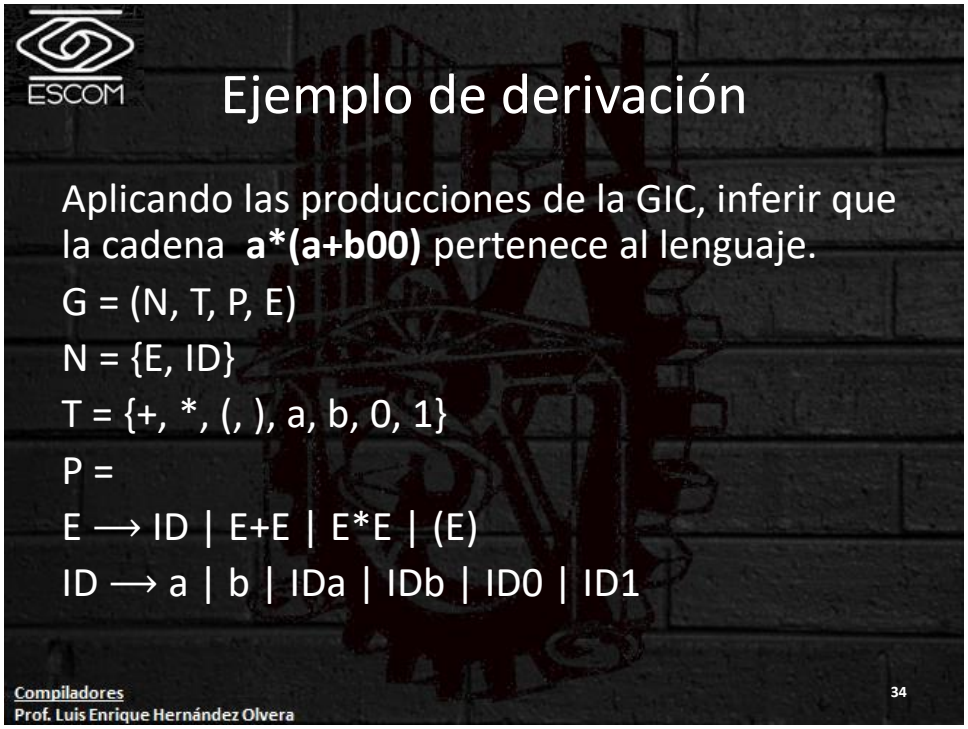

ESCOM
Interpretación de las Producciones
(1) Establece que una expresión puede ser un único identificador .
(2) Establece que una expresión puede ser igual a dos expresiones concatenadas mediante un signo +
(3) Establece una expresión similar a la regla anterior pero para el signo *
(4) Establece que si tomamos cualquier expresión y la encerramos entre paréntesis, el resultado también es una expresión.
(1) E ⟶ ID
(2) E ⟶ E+E
(3) E ⟶ E*E
(4) E ⟶ (E)
Compiladores
Prof. Luis Enrique Hernández Olvera
29

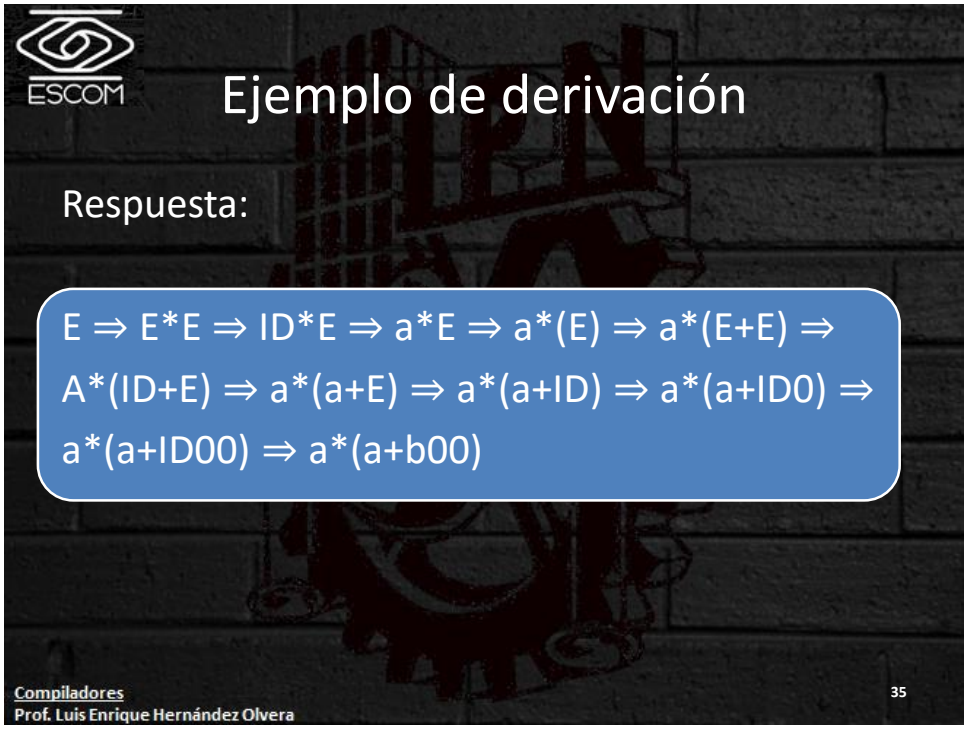

ESCOM
Interpretación de las Producciones
De la regla (5) hasta la (10) se describen los identificadores ID. El caso básico lo describen las reglas (5) y (6) que establecen que a y b son identificadores. Las cuatro reglas restantes establecen que si tenemos cualquier símbolo a, b, 0 o 1, podemos escribir un identificador antes de ellos y el resultado será otro identificador.
(5) ID ⟶ a
(6) ID ⟶ b
(7) ID ⟶ IDa
(8) ID ⟶ IDb
(9) ID ⟶ ID0
(10) ID ⟶ ID1
Compiladores
Prof. Luis Enrique Hernández Olvera
30

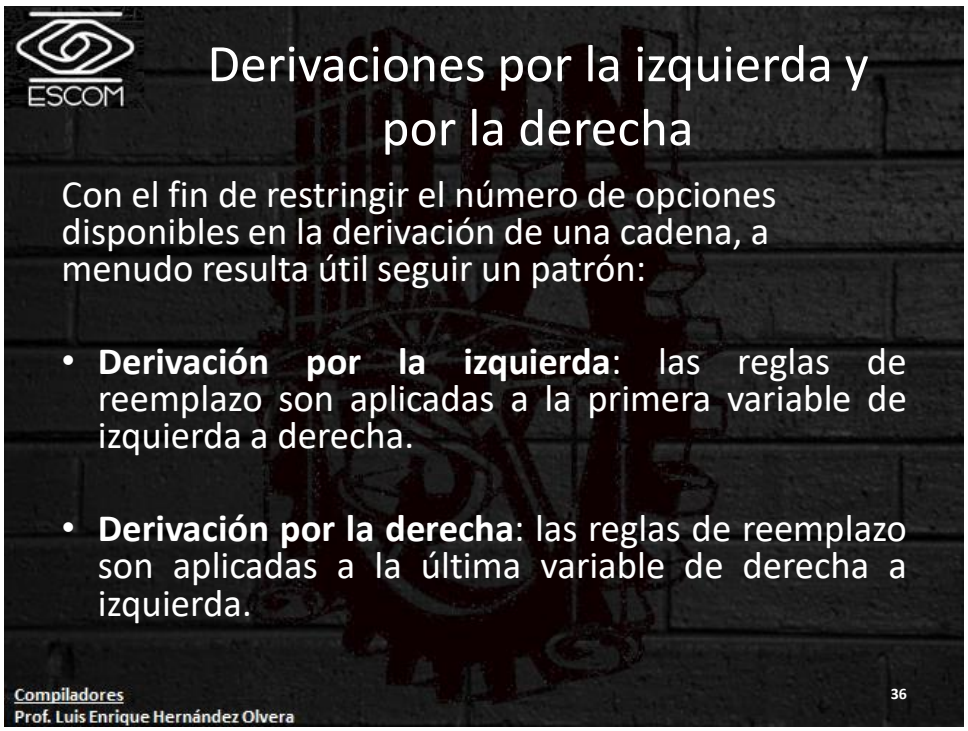

ESCOM
Derivaciones
• Una cadena $w \in \mathbf{T}$ es ***derivable*** a partir de la gramática G si y solo si existe una secuencia de derivación iniciando en el símbolo inicial (no terminal "S") y terminando en la cadena *w.*
Compiladores
Prof. Luis Enrique Hernández Olvera
31


ESCOM
Derivaciones
• La doble flecha $\Rightarrow$ se interpreta como "deriva", "produce" o "genera".
• La notación $E \overset{*}{\Rightarrow} w$ indica que la cadena w se deriva a partir de S en 0 o mas etapas.
• La notación $E \overset{+}{\Rightarrow} w$ indica que la cadena w se deriva a partir de S en 1 o mas etapas.
Compiladores
Prof. Luis Enrique Hernández Olvera
32


ESCOM
Derivaciones
• Cuando derivamos una cadena, los no terminales representan la parte de la cadena que todavía no se ha generado. Cuando la derivación se completa, todos los trozos no generados habrán sido sustituidos por cadenas de símbolos terminales (hay la posibilidad de que sean cadenas vacías).
Compiladores
Prof. Luis Enrique Hernández Olvera
33


ESCOM
Ejemplo de derivación
Aplicando las producciones de la GIC, inferir que la cadena a*(a+b00) pertenece al lenguaje.
G = (N, T, P, E)
N = {E, ID}
T = {+, *, (,), a, b, 0, 1}
P =
E ⟶ ID | E+E | E*E | (E)
ID ⟶ a | b | IDa | IDb | ID0 | ID1
Compiladores
Prof. Luis Enrique Hernández Olvera
34

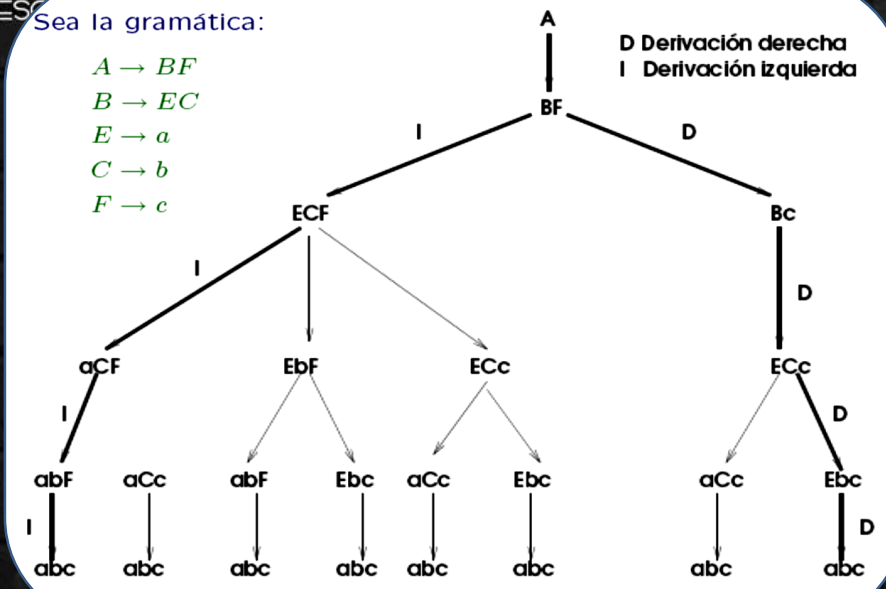

ESCOM
Ejemplo de derivación
Respuesta:
E ⇒ E*E ⇒ ID*E ⇒ a*E ⇒ a*(E) ⇒ a*(E+E) ⇒
A*(ID+E) ⇒ a*(a+E) ⇒ a*(a+ID) ⇒ a*(a+ID0) ⇒
a*(a+ID00) ⇒ a*(a+b00)
Compiladores
Prof. Luis Enrique Hernández Olvera
35


ESCOM
Derivaciones por la izquierda y por la derecha
Con el fin de restringir el número de opciones disponibles en la derivación de una cadena, a menudo resulta útil seguir un patrón:
• Derivación por la izquierda: las reglas de reemplazo son aplicadas a la primera variable de izquierda a derecha.
• Derivación por la derecha: las reglas de reemplazo son aplicadas a la última variable de derecha a izquierda.
Compiladores
Prof. Luis Enrique Hernández Olvera
36

Sea la gramática:

$A \rightarrow BF$
$B \rightarrow EC$
$E \rightarrow a$
$C \rightarrow b$
$F \rightarrow c$

D Derivación derecha
I Derivación izquierda



ESCOM

Ejemplo de derivación

Aplicando las producciones de la GIC, inferir que la cadena **a*(a+b00)** pertenece al lenguaje.

G = (N, T, P, E)

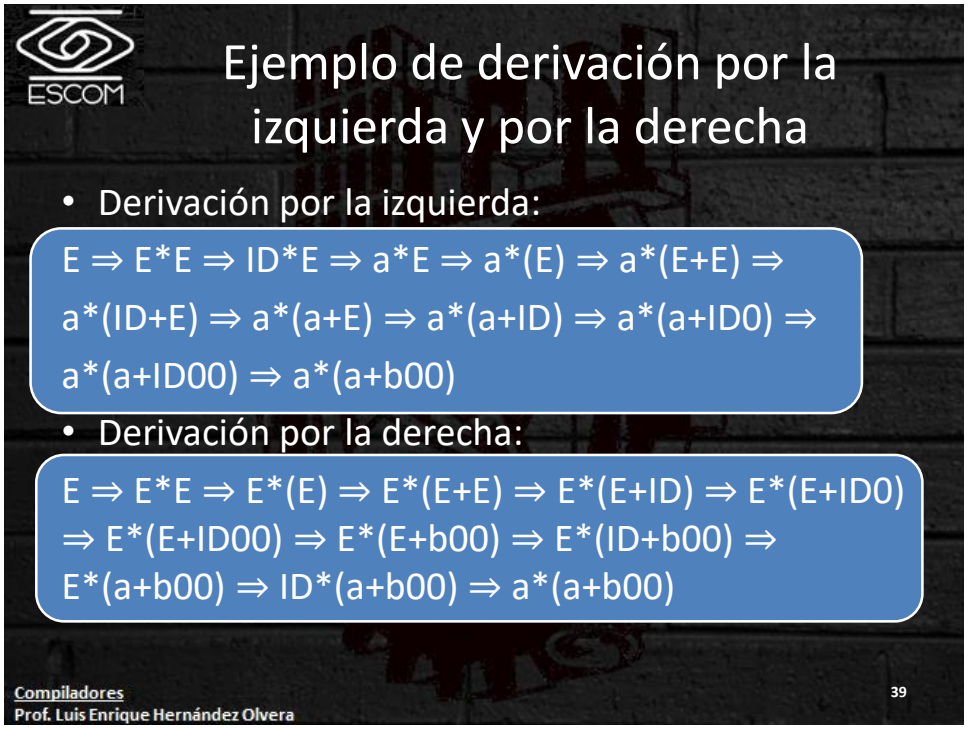
N = {E, ID}

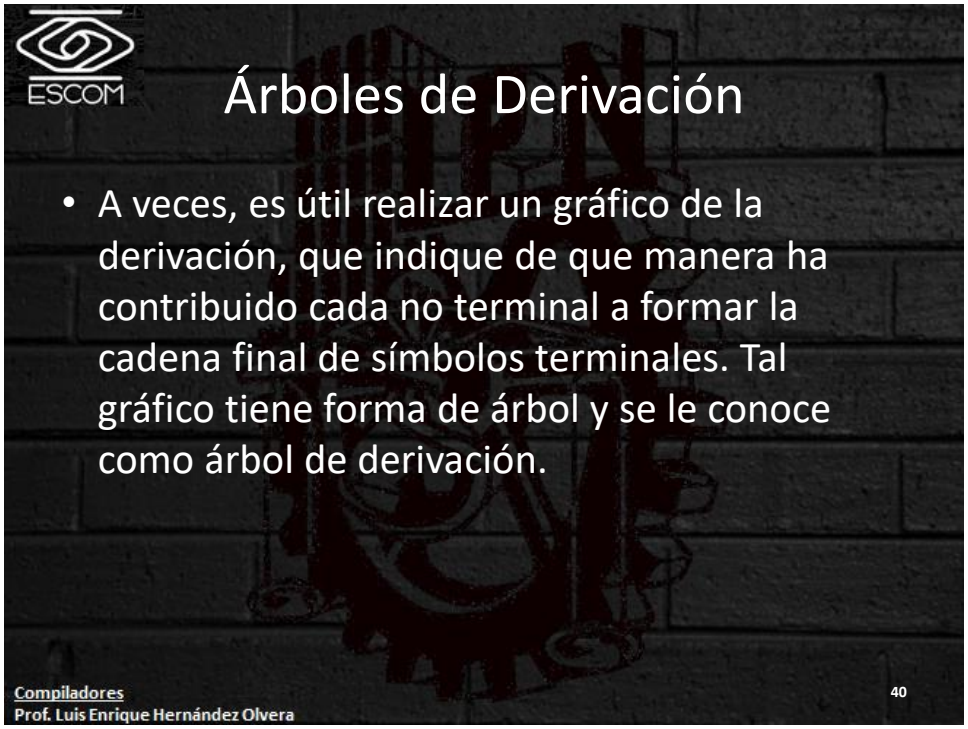
T = {+, *, (,), a, b, 0, 1}

P =

E ⟶ ID | E+E | E*E | (E)

ID ⟶ a | b | IDa | IDb | ID0 | ID1


ESCOM
Ejemplo de derivación por la izquierda y por la derecha
• Derivación por la izquierda:
E ⇒ E*E ⇒ ID*E ⇒ a*E ⇒ a*(E) ⇒ a*(E+E) ⇒ a*(ID+E) ⇒ a*(a+E) ⇒ a*(a+ID) ⇒ a*(a+ID0) ⇒ a*(a+ID00) ⇒ a*(a+b00)
• Derivación por la derecha:
E ⇒ E*E ⇒ E*(E) ⇒ E*(E+E) ⇒ E*(E+ID) ⇒ E*(E+ID0) ⇒ E*(E+ID00) ⇒ E*(E+b00) ⇒ E*(ID+b00) ⇒ E*(a+b00) ⇒ ID*(a+b00) ⇒ a*(a+b00)
Compiladores
Prof. Luis Enrique Hernández Olvera
39


ESCOM
Árboles de Derivación
• A veces, es útil realizar un gráfico de la derivación, que indique de que manera ha contribuido cada no terminal a formar la cadena final de símbolos terminales. Tal gráfico tiene forma de árbol y se le conoce como árbol de derivación.
Compiladores
Prof. Luis Enrique Hernández Olvera
40

Árboles de Derivación

- Los árboles de derivación están estrechamente ligados a las derivaciones y a las inferencias recursivas.
- Una de sus mas importantes aplicaciones es en la ambigüedad en las gramáticas.

Árboles de Derivación

Sea $G = (V,T,P,S)$ una gramática. Los *árboles de derivación* para G son aquellos árboles que cumplen las condiciones siguientes:

1. Cada nodo interior está etiquetado con una variable de V.
2. Cada hoja está etiquetada bien con una variable, un símbolo terminal o ε . Sin embargo, si la hoja está etiquetada con ε , entonces tiene que ser el único hijo de su padre.
3. Si un nodo interior está etiquetado como A y sus hijos están etiquetados como $X1, X2, \ldots, Xk$ respectivamente, comenzando por la izquierda, entonces $A \longrightarrow X1X2 \cdots Xk$ es una producción de P.

Observe que el único caso en que una de las X puede reemplazarse por ε es cuando es la etiqueta del único hijo y $A \longrightarrow \varepsilon$ es una producción de G.

Árboles de Derivación
Ejemplos:

Derivando ID+E de E:

Derivación de

$P \Rightarrow^{*} 0110$

GIC para palíndromos

(1) P ⟶ ε
(2) P ⟶ 0
(3) P ⟶ 1
(4) P ⟶ 0P0
(5) P ⟶ 1P1

Árboles de Derivación

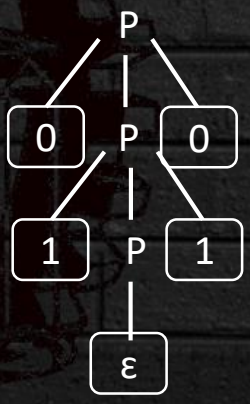
- Si nos fijamos en las hojas de cualquier árbol de derivación y las concatenamos empezando por la izquierda, de arriba hacia abajo, obtenemos una cadena denominada *resultado* del árbol, que siempre es una cadena que se deriva de la variable raíz.

Árboles de Derivación
Ejemplos:

Derivando ID+E de E:

Derivación de

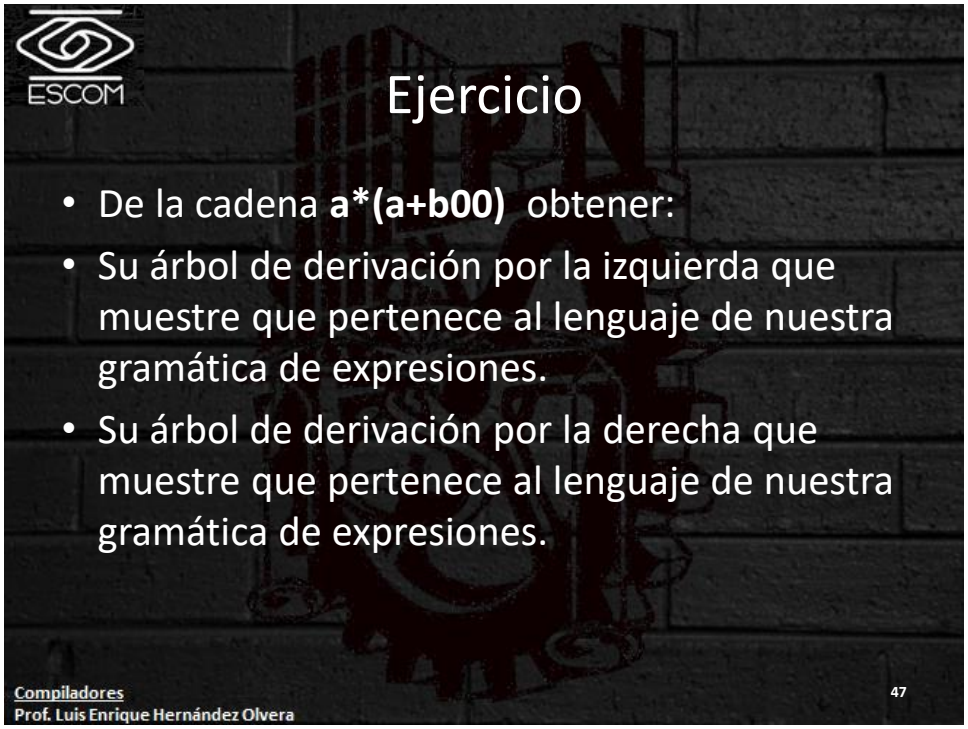
$P \Rightarrow^* 0110$

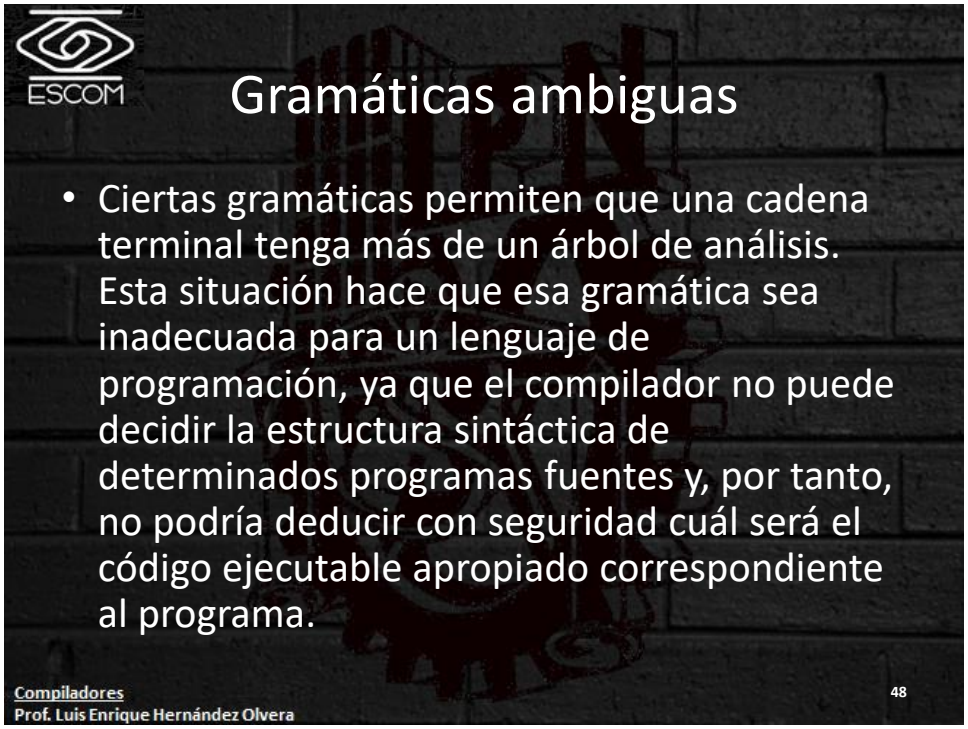


Árboles de Derivación

Los árboles de derivación cuyos resultados son cadenas pertenecientes al lenguaje de la gramática deben de cumplir con:

1. El resultado es una cadena terminal. Es decir, todas las hojas están etiquetadas con un símbolo terminal o con ε .
2. La raíz está etiquetada con el símbolo inicial.


ESCOM
Ejercicio
• De la cadena **a*(a+b00)** obtener:
• Su árbol de derivación por la izquierda que muestre que pertenece al lenguaje de nuestra gramática de expresiones.
• Su árbol de derivación por la derecha que muestre que pertenece al lenguaje de nuestra gramática de expresiones.
Compiladores
Prof. Luis Enrique Hernández Olvera
47


ESCOM
Gramáticas ambiguas
• Ciertas gramáticas permiten que una cadena terminal tenga más de un árbol de análisis. Esta situación hace que esa gramática sea inadecuada para un lenguaje de programación, ya que el compilador no puede decidir la estructura sintáctica de determinados programas fuentes y, por tanto, no podría deducir con seguridad cuál será el código ejecutable apropiado correspondiente al programa.
Compiladores
Prof. Luis Enrique Hernández Olvera
48

Gramáticas ambiguas

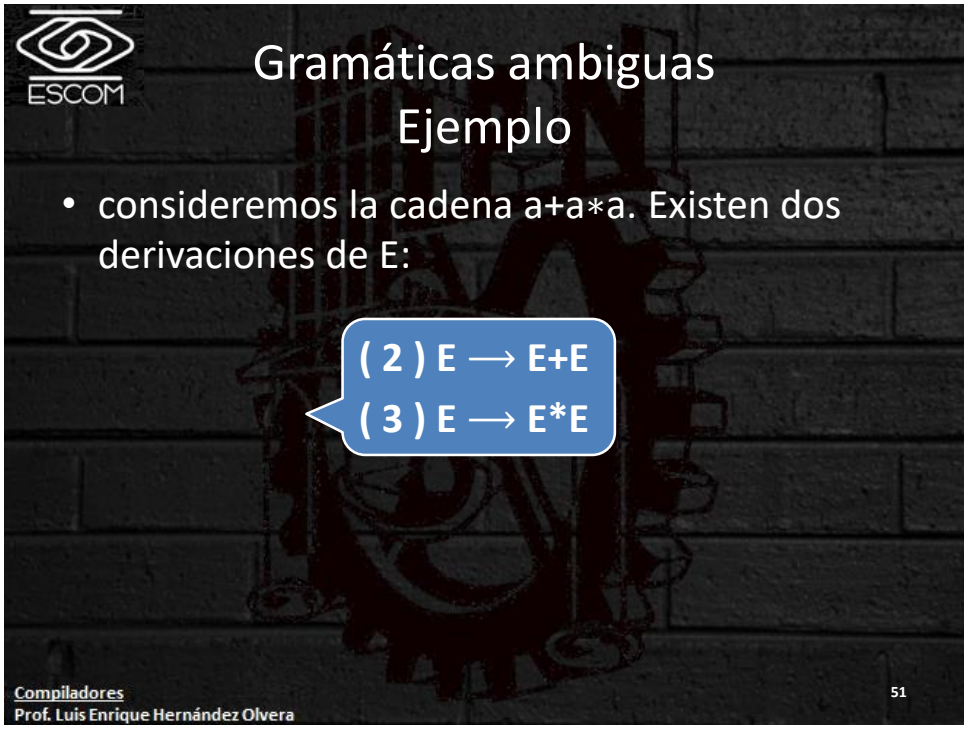
- Usando el ejemplo de gramática de expresiones observemos que las producciones
- E ⟶E*E | E+E

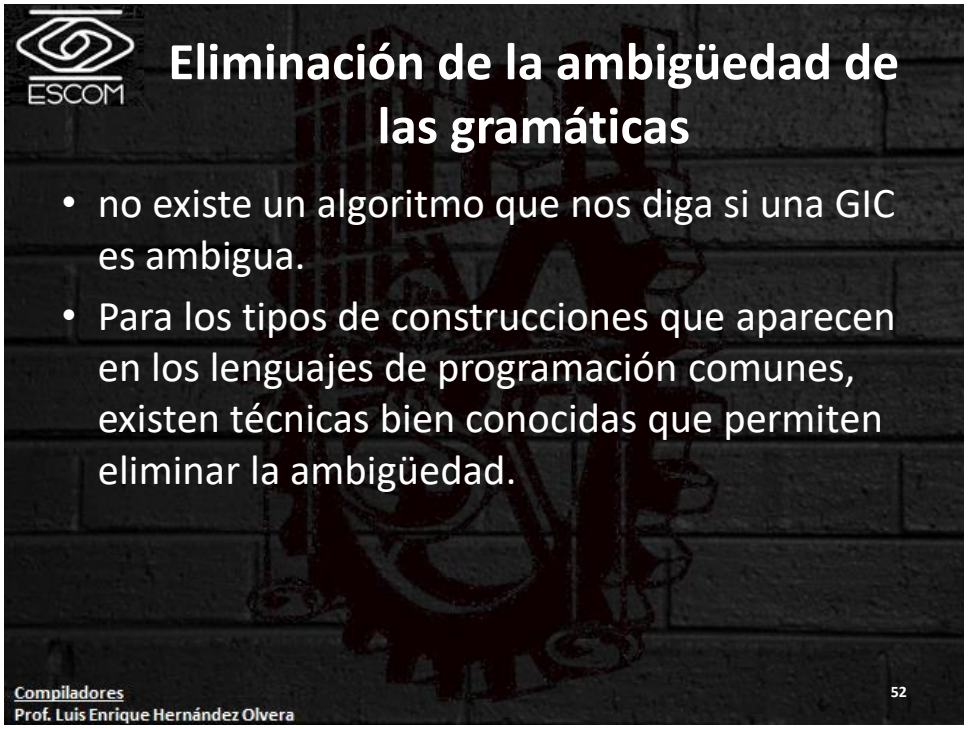
nos permite generar estas expresiones en cualquier orden que elijamos.

Gramáticas ambiguas Ejemplo

- consideremos la forma sentencial E+E∗E. Existen dos derivaciones de E:

(2) E ⟶ E+E
(3) E ⟶ E*E


ESCOM
Gramáticas ambiguas
Ejemplo
• consideremos la cadena a+a∗a. Existen dos derivaciones de E:
(2) E ⟶ E+E
(3) E ⟶ E*E
Compiladores
Prof. Luis Enrique Hernández Olvera
51


ESCOM
Eliminación de la ambigüedad de las gramáticas
• no existe un algoritmo que nos diga si una GIC es ambigua.
• Para los tipos de construcciones que aparecen en los lenguajes de programación comunes, existen técnicas bien conocidas que permiten eliminar la ambigüedad.
Compiladores
Prof. Luis Enrique Hernández Olvera
52

Eliminación de la ambigüedad de las gramáticas

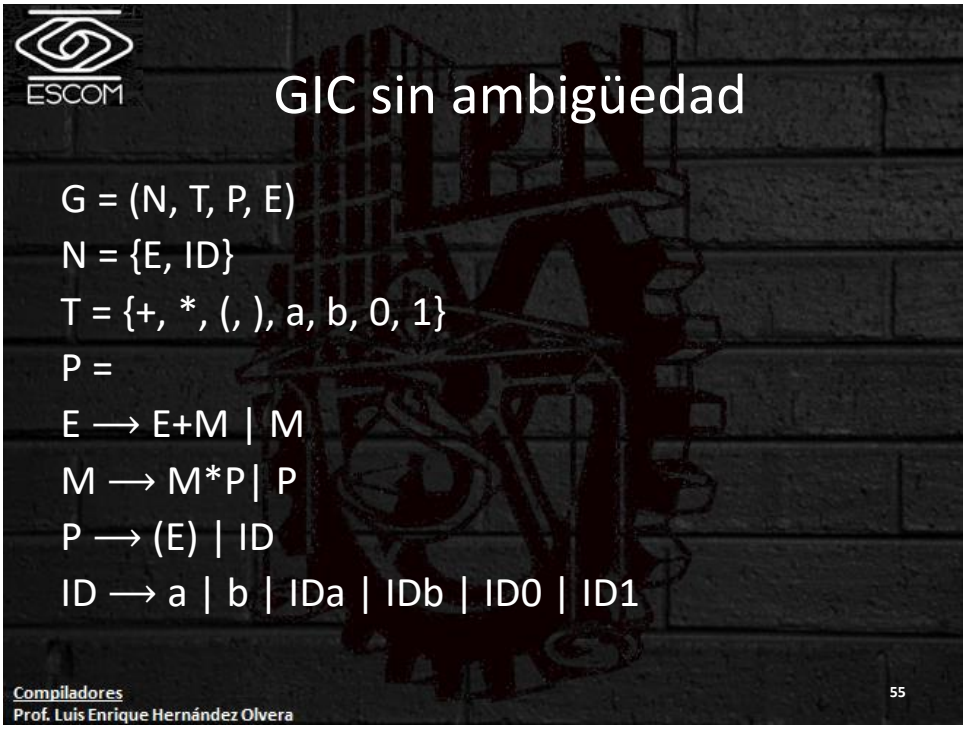
- Analizando el ejemplo anterior vemos claramente que la precedencia de operadores no se respeta.
- La solución al problema de forzar la precedencia se resuelve introduciendo varias variables distintas, cada una de las cuales representa aquellas expresiones que comparten el mismo nivel de "fuerza de acoplamiento".

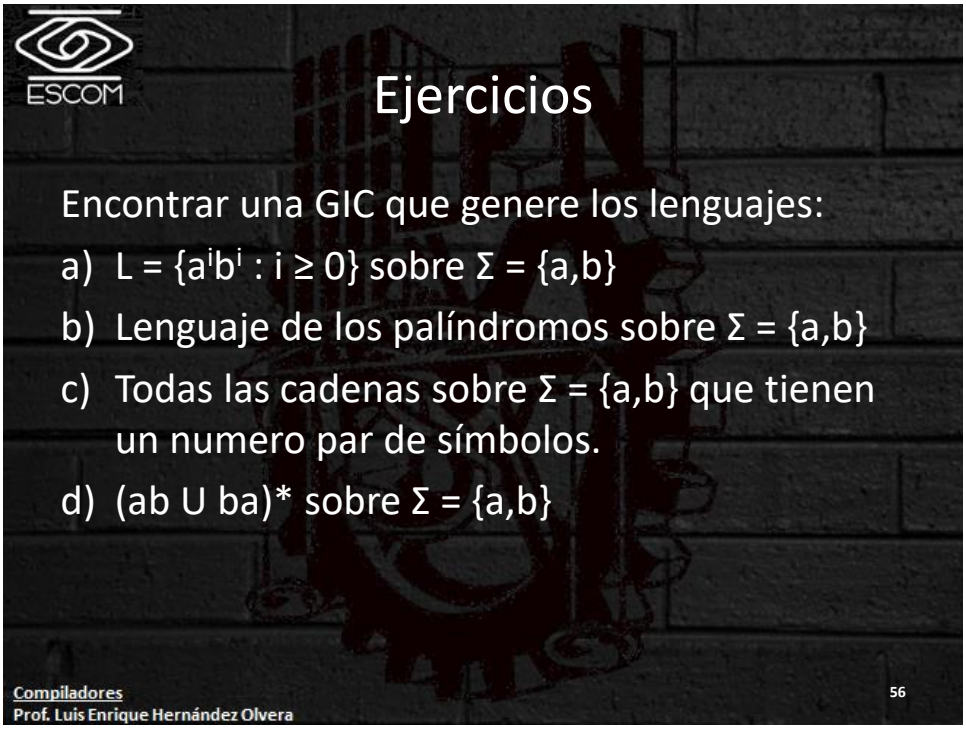
Eliminación de la ambigüedad de las gramáticas

- La gramática resultante para la ambigüedad en nuestra GIC queda de la siguiente forma:

$E \longrightarrow ID \mid E+E \mid E*E$

$E \longrightarrow E+M \mid M$
$M \longrightarrow M*M \mid ID$


ESCOM
GIC sin ambigüedad
G = (N, T, P, E)
N = {E, ID}
T = {+, *, (,), a, b, 0, 1}
P =
E ⟶ E+M | M
M ⟶ M*P| P
P ⟶ (E) | ID
ID ⟶ a | b | IDa | IDb | ID0 | ID1
Compiladores
Prof. Luis Enrique Hernández Olvera
55


ESCOM
Ejercicios
Encontrar una GIC que genere los lenguajes:
a) L = {aⁱbⁱ : i ≥ 0} sobre Σ = {a,b}
b) Lenguaje de los palíndromos sobre Σ = {a,b}
c) Todas las cadenas sobre Σ = {a,b} que tienen un numero par de símbolos.
d) (ab U ba)* sobre Σ = {a,b}
Compiladores
Prof. Luis Enrique Hernández Olvera
56

Ejercicios (Soluciones posibles)

G =(N,Σ,S,P)

a)
Σ = {a,b}
N={S}
P=
S ⟶ aSb
S ⟶ λ

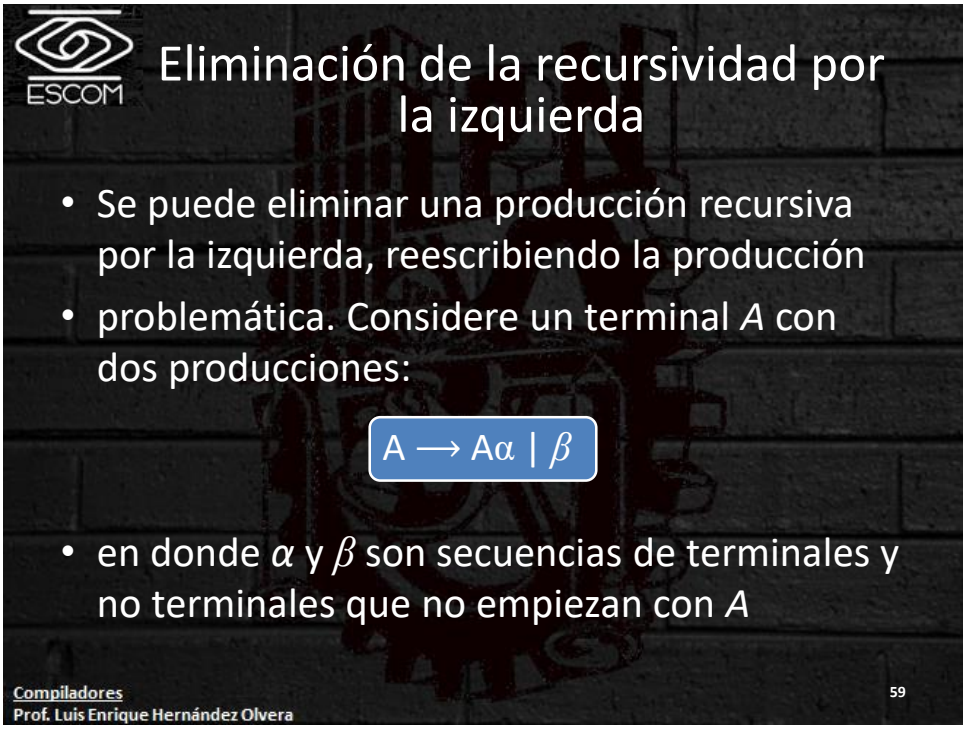
b)
Σ = {a,b}
N={S}
P=
S ⟶ aSa | bSb | a | b | λ

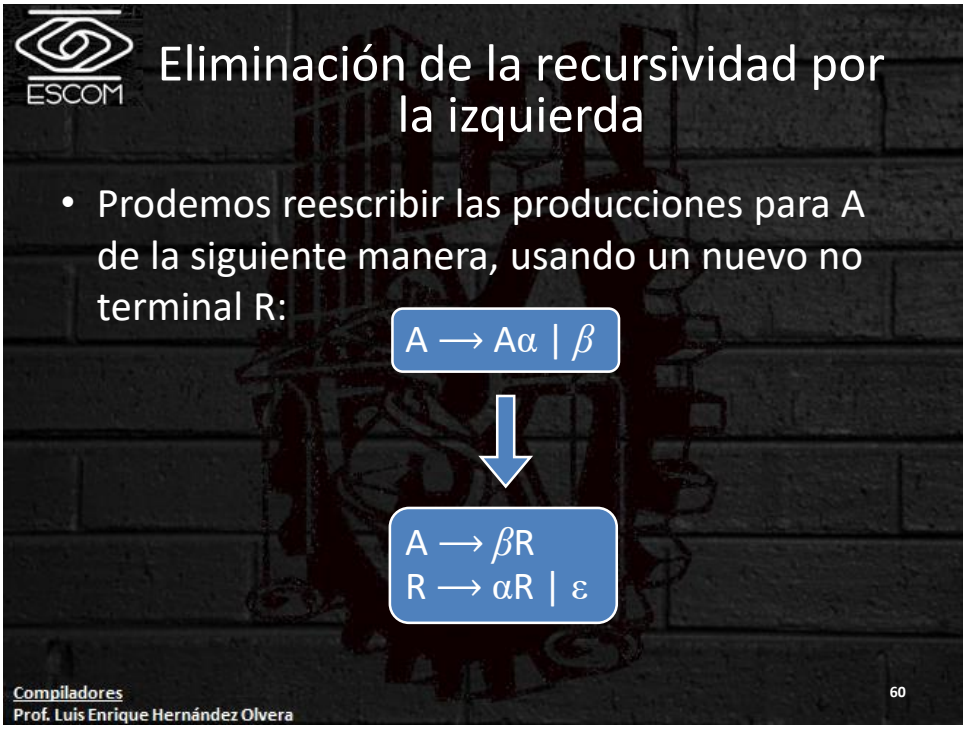
c)
Σ = {a,b}
N={S}
P=
S ⟶ aSa | bSb | aSb | bSa | λ

d)
Σ = {a,b}
N={S}
P=
S ⟶ abS
S ⟶ baS
S ⟶ λ

Eliminación de la recursividad por la izquierda

- Una gramática es *recursiva por la izquierda* si tiene una terminal *A* tal que haya una derivación:
  $-\ A \overset{+}{\Rightarrow} A\alpha$ para cierta cadena α
- Los métodos de análisis sintáctico descendentes no pueden manejar las gramáticas recursivas por la izquierda, por lo que se necesita una transformación para eliminar la recursividad por la izquierda.


ESCOM
Eliminación de la recursividad por la izquierda
• Se puede eliminar una producción recursiva por la izquierda, reescribiendo la producción
• problemática. Considere un terminal A con dos producciones:
A ⟶ Aα | β
• en donde α y β son secuencias de terminales y no terminales que no empiezan con A
Compiladores
Prof. Luis Enrique Hernández Olvera
59


ESCOM
Eliminación de la recursividad por la izquierda
• Prodemos reescribir las producciones para A de la siguiente manera, usando un nuevo no terminal R:
A ⟶ Aα | β
A ⟶ βR
R ⟶ αR | ε
Compiladores
Prof. Luis Enrique Hernández Olvera
60

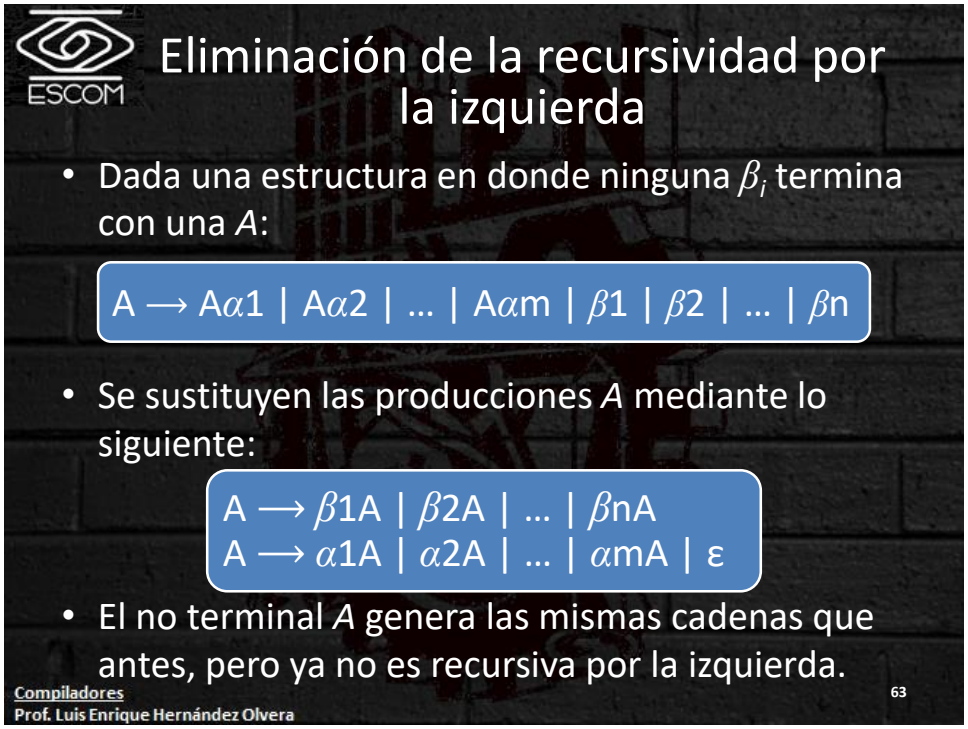
Eliminación de la recursividad por la izquierda

- Nótese que la producción $A \rightarrow A\alpha$ tiene a la misma *A* como el símbolo de más a la izquierda en el lado derecho. La aplicación repetida de esta producción genera una secuencia de α's a la derecha de *A*. Cuando por fin *A* se sustituye por β, tenemos una β seguida por una secuencia de cero o más α's. Este comportamiento se mantiene idéntico en la nueva representación pero sin la recursión a la izquierda.

Eliminación de la recursividad por la izquierda

- La recursividad inmediata por la izquierda puede eliminarse mediante la siguiente técnica, que funciona para cualquier número de producciones *A*.
- En primer lugar, se agrupan las producciones de la siguiente manera:

$$A \longrightarrow A\alpha 1 \mid A\alpha 2 \mid \ldots \mid A\alpha m \mid \beta 1 \mid \beta 2 \mid \ldots \mid \beta n$$


ESCOM
Eliminación de la recursividad por la izquierda
• Dada una estructura en donde ninguna β_i termina con una *A*:
A ⟶ Aα1 | Aα2 | … | Aαm | β1 | β2 | … | βn
• Se sustituyen las producciones *A* mediante lo siguiente:
A ⟶ β1A | β2A | … | βnA
A ⟶ α1A | α2A | … | αmA | ε
• El no terminal *A* genera las mismas cadenas que antes, pero ya no es recursiva por la izquierda.
Compiladores
Prof. Luis Enrique Hernández Olvera
63