Universidad Nacional Autónoma de México

Facultad de Ciencias

Tarea 02

Compiladores

Bonilla Reyes Dafne
Castañón Maldonado Carlos Emilio
García Ponce José Camilo

Profesora: Lourdes del Carmen González Huesca

Ayudante: Fausto David Hernández Jasso
Ayudante: Juan Alfonso Garduño Solís
Ayudante: Juan Pablo Yamamoto Zazueta

1. **(1.5pts)** Considera la siguiente gramática:

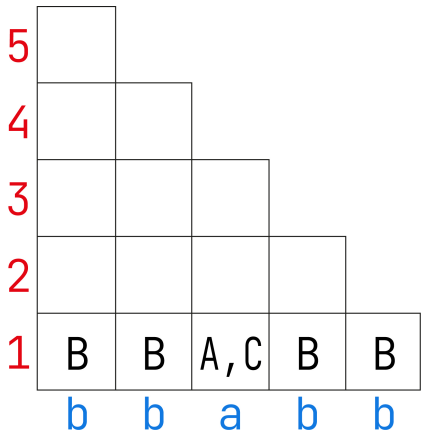
$$S \to AB|BC$$
$$A \to BA|a$$
$$B \to CC|b$$
$$C \to AB|a$$

Ocupa el algoritmo ***CYK*** para verificar si las cadenas *bbabb* y *aba* se pueden generar con la gramática.

Para poder utilizar el algoritmo CYK primero debemos asegurarnos de que la gramática está en la forma normal de Chomsky. En este caso ya lo está, por lo que podemos realizar el algoritmo. Para ambas cadenas seguiremos los mismos pasos:

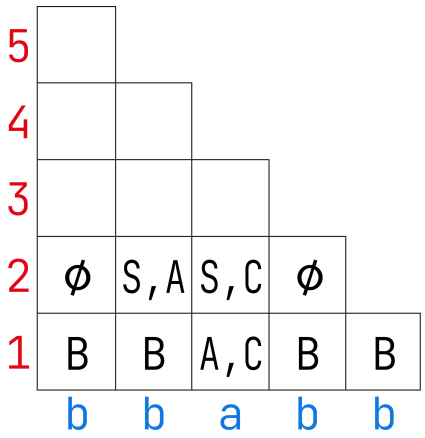
- `bbabb:`
  - (a) Escribimos la tabla y revisamos el primer nivel viendo que reglas derivan a cadenas de tamaño 1.

    El cuadro será el siguiente:



    Notemos que para el primer nivel analizamos que reglas de derivación tienen a los símbolos terminales a y b.
  - (b) Revisamos el segundo nivel viendo que reglas derivan a cadenas de tamaño 2.

    El cuadro quedará entonces de la siguiente forma:



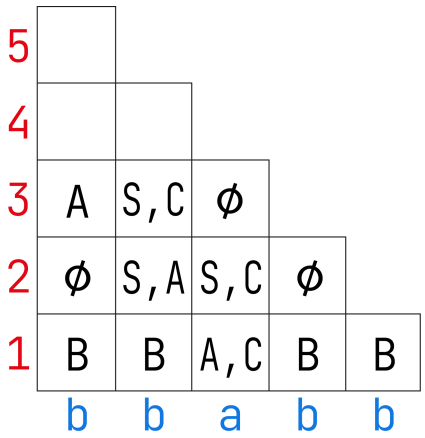
    Notemos que para el segundo nivel analizamos a las subcadenas: *bb*, *ba* y *ab*, revisando el "producto cartesiano" y viendo si el resultado de este producto está en una regla de derivación. Por ejemplo, para la subcadena *bb* hacemos:

$$B \times B = BB \Rightarrow \varnothing$$

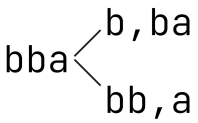
Veamos que *bb* no se encuentra en las reglas de derivación, por lo que en la tabla ponemos al conjunto vacío en ese espacio. Repetimos el mismo proceso para el resto de la subcadenas.

(c) Revisamos el tercer nivel viendo que reglas derivan a cadenas de tamaño 3.

El cuadro quedará entonces de la siguiente forma:



Notemos que para el tercer nivel analizamos a las subcadenas: *bba, bab* y *abb*, aplicando el mismo método, solo que esta vez también veremos cuáles son las derivaciones por cada subcadena. Por ejemplo, para la subcadena *bba* tenemos:



Con esto, obtenemos a las posibles 2 subcadenas de *bba* y ahora, revisamos nuevamente que reglas de producción que las derivan, esto es:

- *b:* B
- *ba:* S, A
- *bb:* $\varnothing$
- *a:* A, C

Entonces, a partir de esto hacemos los "productos cartesianos" de cada subcadena:

- Para *b, ba:*

$$B \times S, A = BS, BA \Rightarrow A$$

- Para *bb, a:*

$$\varnothing \times A, C = \varnothing \Rightarrow \varnothing$$

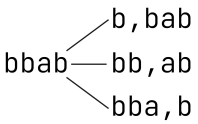
Por lo tanto, para esta subcadena *bba* obtenemos A, y la agregamos a la tabla. Repetimos el mismo proceso para el resto de la subcadenas.

(d) Revisamos el cuarto nivel viendo que reglas derivan a cadenas de tamaño 4.

El cuadro quedará entonces de la siguiente forma:

5					
4	S,C	ϕ			
3	A	S,C	ϕ		
2	ϕ	S,A	S,C	ϕ	
1	B	B	A,C	B	B
	b	b	a	b	b

Notemos que para el cuarto nivel analizamos a las subcadenas: *bbab* y *babb*, aplicando el mismo método mencionado en el inciso anterior. Por ejemplo, para la subcadena *bbab* tenemos:



Con esto, obtenemos a las posibles 3 subcadenas de *bbab* y ahora, revisamos nuevamente que reglas de producción que las derivan, esto es:

- *b:* B
- *bab:* S, C
- *bb:* ∅
- *ab:* S, C
- *bba:* A

Entonces, a partir de esto hacemos los "productos cartesianos" de cada subcadena:

- Para *b, bab:*

$$B \times S, C = BS, BC \Rightarrow S$$

- Para *bb, ab:*

$$\varnothing \times S, C = \varnothing \Rightarrow \varnothing$$

- Para *bba, a:*

$$A \times B = AB \Rightarrow C$$

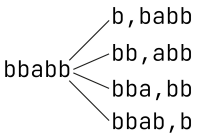
Por lo tanto, para esta subcadena *bbab* obtenemos *S,C*, y la agregamos a la tabla. Repetimos el mismo proceso para la otra subcadena.

(e) Revisamos el quinto nivel viendo que reglas derivan a cadenas de tamaño 5.

El cuadro quedará entonces de la siguiente forma:

5	ϕ				
4	S,C	ϕ			
3	A	S,C	ϕ		
2	ϕ	S,A	S,C	ϕ	
1	B	B	A,C	B	B
	b	b	a	b	b

Notemos que para el quinto nivel analizamos a la cadena: *bbabb*, aplicando el mismo método mencionado en el inciso anterior. Por lo que obtenemos que:



Con esto, obtenemos a las posibles 4 subcadenas de *bbabb* y ahora, revisamos nuevamente que reglas de producción que las derivan, esto es:

- *b:* B
- *babb:* $\varnothing$
- *bb:* $\varnothing$
- *abb:* $\varnothing$
- *bba:* A
- *bbab:* S, C

Entonces, a partir de esto hacemos los "productos cartesianos" de cada subcadena:

- Para *b, babb:*

$$B \times \varnothing = \varnothing \Rightarrow \varnothing$$

- Para *bb, abb:*

$$\varnothing \times \varnothing = \varnothing \Rightarrow \varnothing$$

- Para *bba, bb:*

$$A \times \varnothing = \varnothing \Rightarrow \varnothing$$

- Para *bbab, b:*

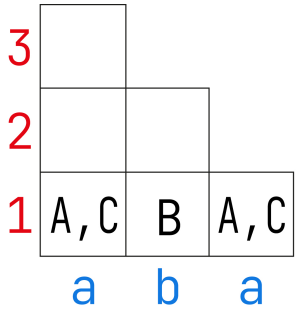
$$S, C \times B = SB, CB \Rightarrow \varnothing$$

Por lo tanto, podemos decir que la cadena *bbabb* no pertenece a la gramática.

- `aba:`

(a) Escribimos la tabla y revisamos el primer nivel viendo que reglas derivan a cadenas de tamaño 1.

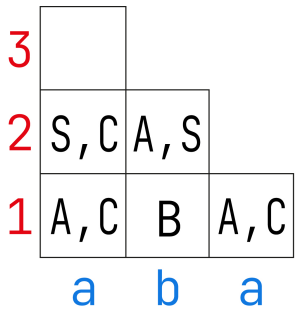
El cuadro será el siguiente:



Notemos que para el primer nivel analizamos que reglas de derivación tienen a los símbolos terminales a y b.

(b) Revisamos el segundo nivel viendo que reglas derivan a cadenas de tamaño 2.

El cuadro quedará entonces de la siguiente forma:



Notemos que para el segundo nivel analizamos a las subcadenas: ab y ba, revisando el "producto cartesiano" y viendo si el resultado de este producto está en una regla de derivación. Esto es:

- Para la subcadena ab hacemos:

$$A, C \times B = AB, CB \Rightarrow S, C$$

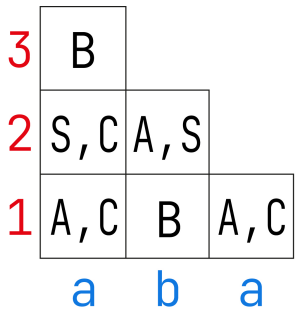
- Para la subcadena ba hacemos:

$$B \times A, C = BA, BC \Rightarrow A, S$$

Ahora llenamos la tabla con los valores obtenidos.

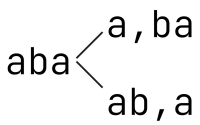
(c) Revisamos el tercer nivel viendo que reglas derivan a cadenas de tamaño 3.

El cuadro quedará entonces de la siguiente forma:



Notemos que para el tercer nivel analizamos a la cadena: aba, aplicando el mismo método, solo que esta vez también veremos cuáles son las derivaciones de la cadena.

Entonces tenemos:



Con esto, obtenemos a las posibles 2 subcadenas de *aba* y ahora, revisamos nuevamente que reglas de producción que las derivan, esto es:

- *a:* A, C
- *ba:* A, S
- *ab:* S, C

Entonces, a partir de esto hacemos los "productos cartesianos" de cada subcadena:

- Para *a, ba:*

$$A, C \times A, S = AA, AS, CA, CS \Rightarrow \varnothing$$

- Para *ab, a:*

$$S, C \times A, C = SA, SC, CA, CC \Rightarrow B$$

Por lo tanto, podemos decir que la cadena *aba* no pertenece a la gramática.

2. **(1 pt)** ¿Qué lenguaje genera la gramática siguiente?

Elimina la recursión izquierda para la siguiente gramática:

$$S \to a\ A\ b|a\ A\ A|a\ B|b\ b\ A \qquad A \to a\ A\ b|a\ b \qquad B \to b\ B\ a|b\ a$$

Para ver el lenguaje que genera la gramática, primero veamos unas cosas:

Notemos que $A \to a\ A\ b \mid a\ b$ genera este lenguaje $a^n b^n$ con n un natural mayor a 0. De manera similar $B \to b\ B\ a \mid b\ a$ genera este lenguaje $b^m a^m$ con m un natural mayor a 0.

Por lo tanto, si juntamos esto y las reglas de S, tenemos que el lenguaje de la gramática es:

$$(a(a^n b^n)b) + (a(a^m b^m)(a^p b^p)) + (a(b^q a^q)) + (bb(a^k b^k))$$

Con n, m, p, q y k naturales mayores a 0. La gramática no tiene recursión izquierda, esto debido a que no tiene reglas de la forma $C \to^+ C\alpha$ (la definición de que una gramática tenga recursión izquierda), ya que el primer símbolo del lado derecho de todas las reglas son símbolos terminales por lo cual la gramática ya está en una forma sin recursión izquierda.

3. Considera la siguiente gramática:

$S \to a$ **(1)** $\quad S \to$ **if** b **then** S **(2)** $\quad S \to$ **if** b **then** S **else** S **(3)**

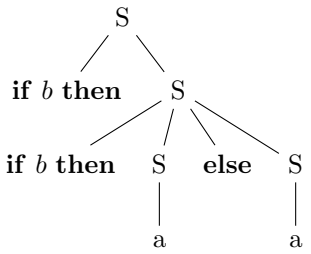
(a) **(1 pt)** Proporciona una derivación y un *parse tree* para la cadena :

if b **then if** b **then** a **else** a

Derivación:

$$\begin{aligned} S &\to \textbf{if } b \textbf{ then } S \quad \text{Por } \textbf{(2)} \\ &\to \textbf{if } b \textbf{ then if } b \textbf{ then } S \textbf{ else } S \quad \text{Por } \textbf{(3)} \\ &\to \textbf{if } b \textbf{ then if } b \textbf{ then } a \textbf{ else } S \quad \text{Por } \textbf{(1)} \\ &\to \textbf{if } b \textbf{ then if } b \textbf{ then } a \textbf{ else } a \quad \text{Por } \textbf{(1)} \end{aligned}$$

parse tree:



Notemos como es que tanto en las derivaciones como en el *parse tree* fue posible llegar a la cadena final que queríamos.

(b) **(1.5 pt)** Demuestra que esta gramática es ambigua.

Para demostrar que una gramática es ambigua, necesitamos encontrar al menos una cadena que pueda ser derivada de dos maneras diferentes.

Para eso usaremos a la cadena:

if b **then if** b **then** a **else** a

Derivación 1:

$$\begin{aligned} S &\to \textbf{if } b \textbf{ then } S \quad (\text{Por regla } \textbf{(2)}) \\ &\to \textbf{if } b \textbf{ then if } b \textbf{ then } S \textbf{ else } S \quad (\text{Por regla } \textbf{(3)}) \\ &\to \textbf{if } b \textbf{ then if } b \textbf{ then } a \textbf{ else } S \quad (\text{Por regla } \textbf{(1)}) \\ &\to \textbf{if } b \textbf{ then if } b \textbf{ then } a \textbf{ else } a \quad (\text{Por regla } \textbf{(1)}) \end{aligned}$$

Derivación 2:

$$\begin{aligned} S &\to \textbf{if } b \textbf{ then } S \textbf{ else } S \quad (\text{Por regla } \textbf{(3)}) \\ &\to \textbf{if } b \textbf{ then if } b \textbf{ then } S \textbf{ else } S \quad (\text{Por regla } \textbf{(2)}) \\ &\to \textbf{if } b \textbf{ then if } b \textbf{ then } a \textbf{ else } S \quad (\text{Por regla } \textbf{(1)}) \\ &\to \textbf{if } b \textbf{ then if } b \textbf{ then } a \textbf{ else } a \quad (\text{Por regla } \textbf{(1)}) \end{aligned}$$

Esto nos da fe que en efecto, la presente gramática es ambigua.

(c) **(1.5 pt)** Describe una propuesta detallada para eliminar la ambigüedad.
Como primer paso cambiaremos la siguiente producción:
La producción $S \to a$ se modificará de la siguiente forma:

$$S \to S_1 | S_2$$

Ahora, dividiremos a S en S_1 y S_2, en donde queremos que S_1 maneje las sentencias condicionales y S_2 maneje las sentencias normales.
Quedando de esta forma:

$$S \to S_1 | S_2$$

Con esto las nuevas producciones quedarán de esta forma:

$$S_1 \to \text{if } b \text{ then } S | \text{if } b \text{ then } S_2 \text{ else } S_1$$

$$S_2 \to \text{if } b \text{ then } S_2 \text{ else } S_2 | a$$

Con esto tendremos que nuestra nueva propuesta de gramática es:

$$S \to S_1 | S_2$$

$$S_1 \to \text{if } b \text{ then } S | \text{if } b \text{ then } S_2 \text{ else } S_1$$

$$S_2 \to \text{if } b \text{ then } S_2 \text{ else } S_2 | a$$

También se puede eliminar la ambigüedad usando paréntesis:
$S \to a$ $S \to \text{if } b \text{ then } (S)$ $S \to \text{if } b \text{ then } (S) \text{ else } (S)$

4. **(2 pts)** Sea G la siguiente gramática:

$$S \to aS \mid aSbS \mid c$$

y sea G_1 otra gramática, también libre de contexto con reglas:

$$S_1 \to T \mid U \qquad T \to aTbT \mid c \qquad U \to aS_1 \mid aTbU$$

Demuestra que G es ambigua y G_1 no lo es.
¿Será que ambas gramáticas generan el mismo lenguaje? Justifica tu respuesta.

Demostración. Demostrar que G es ambigua y G_1 no lo es.

(a) Primero veamos que G es ambigua:

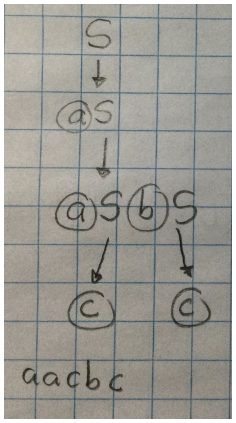
Tengamos la cadena $w = aacbc$
Ahora veamos que existen dos árboles de sintaxis para esta cadena con la gramática G

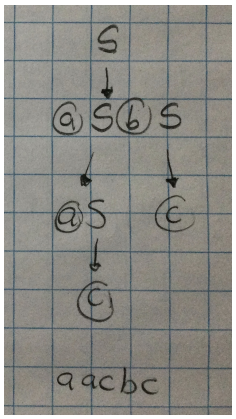
- El primero
- El segundo

$\therefore$ G es una gramática ambigua, ya que tenemos 2 árboles de sintaxis distintos para la misma cadena.

(b) Ahora veamos que G_1 no es ambigua:
Para esto veremos que una cadena cualquiera del lenguaje generado por G_1 solo tiene un árbol de sintaxis/derivación.
Notemos que las reglas de producción están creadas de cierta forma (son similares a las de la gramática G pero con ciertos cambios para evitar ambigüedad).
Revisemos cada regla para ver si podemos encontrar algunas que generen ambigüedad:

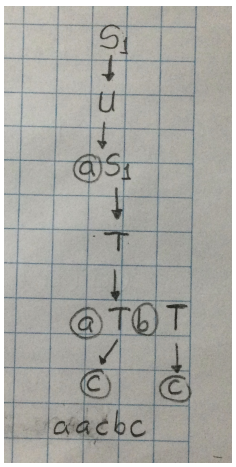
- Primero la regla $S_1 \to T \mid U$, aquí no podemos generar ambigüedad, ya que solo escogemos una regla y en la decisión no hay ambigüedad, ya que T y U generan cosas distintas.





- Luego la regla $T \to aTbT \mid c$, para decidir entre $aTbT$ y c depende de la cadena que se esté formando con la gramática, por lo tanto, no hay ambigüedad, ya que solo una opción es correcta para la cadena.
- Y por último la regla $U \to aS_1 \mid aTbU$, notemos que al decidir cuál elegir no hay debido a que si elegimos $aTbU$ al momento de elegir la nueva U solo puede ser aS_1 o $aTbU$ y de esta manera estamos restringiendo algo lo que podemos formar (restringiendo a comparación de G, ya que en este caso no podemos generar una c luego de la b si elegimos a la U, para eso tenemos que usar a T), logrando que no exista ambigüedad que se encontraba antes en G (antes había varias formas de hacer la subcadena "cbc").

Ahora veamos que la cadena $w = aacbc$ solo tiene un árbol de sintaxis y que en cada nivel del árbol la regla por usar es clara y no tenemos ambigüedad u opciones sobre cuál podría servir, y esto se cumple para cualquier cadena del lenguaje.



(c) Por último, veamos que ambas gramáticas generan el mismo lenguaje:

El lenguaje generado por G y G_1 son el mismo, esto debido a que G_1 es una transformación de la gramática G.

Ahora veamos la "demostración"de esto:

Supongamos una cadena del lenguaje generado por G, y veamos que puede ser generada por G_1 (esta en el lenguaje generado por G_1). Si la cadena es generada por la regla $S \to aS$ entonces se puede generar con las reglas $S_1 \to U$ y $U \to aS$.

Luego, si la cadena es generada por la regla $S \to aSbS$ entonces se puede generar con las reglas $S_1 \to U$ y $U \to aTbU$ o con las reglas $S_1 \to T$ y $T \to aTbT$ dependiendo de si lo que se derive de la segunda S de $aSbS$ contiene a c (entonces con T) o no (entonces con U). Y por último, si la cadena es generada por la regla $S \to c$ entonces se puede generar con la regla $S \to T$ y $T \to c$.

$\therefore$ podemos generar con G_1 todas las cadenas del lenguaje de G.

Ahora solo falta el regreso, supongamos una cadena del lenguaje generado por G_1, y veamos que puede ser generada por G (esta en el lenguaje generado por G). Si la cadena es generada por la regla $T \to aTbT$ entonces se puede generar con la regla $S \to aSbS$. Luego, si la cadena es generada por la regla $T \to c$ entonces se puede generar con la regla $S \to c$.

Después, si la cadena es generada por la regla $U \to aS_1$ entonces se puede generar con la regla $S \to aS$. Luego, si la cadena es generada por la regla $U \to aTbU$ entonces se puede generar con la regla $S \to aSbS$. Y por último, si la cadena es generada por la regla $S \to U$ lo podemos generar con lo que vimos en las reglas de U, de manera similar para la regla $S \to T$, ya que arriba vimos que podemos generar todo lo que generan las reglas de T y U.

$\therefore$ podemos generar con G todas las cadenas del lenguaje de G_1.

Y concluimos que ambas gramáticas generan el mismo lenguaje, solo que G es una versión con ambigüedad de G_1, ya que existen varias formas para formar a algunas cadenas, en cambio, en G_1 solo hay una forma.

Espero que estas explicaciones sean suficiente explicación para mostrar que G_1 no es ambigua y ambas generan el mismo lenguaje.

□

5. **(1.5pts)** Considera la siguiente gramática:

$$\begin{aligned} \mathbb{C}^2 &\to V | V\ op\ V \\ V &\to (elem \oplus elem\ ,\ elem \oplus elem) | id \\ elem &\to num | id \\ op &\to + | - | * \end{aligned}$$

Da una tabla con las funciones `First` y `Follow` de cada una de las variables de la gramática.

	$\mathbb{C}^2$	V	elem	op
`FIRST`	{`FIRST`(V)}	{(id}	{id num}	{+ - *}
`FOLLOW`	{}	{`FIRST`(op)}	{$\oplus$,)}	{`FIRST`(V)}