Universidad Nacional Autónoma de México

Facultad de Ciencias

Tarea 4

Presenta

Valeria Camacho Hernández
322007273

Asignatura

Autómatas y Lenguajes Formales 2026-1

Profesor

Enrique F. Soto-Astorga

Ayudante

Laura Itzel Rodríguez Dimayuga

Fecha

Noviembre 13 del 2025

Tarea 4

1. **(1 pt)** Construye la Gramática Libre de Contexto para el siguiente lenguaje formal:

$$\mathcal{L}_1 = \{ww^r \mid w \in \{a, b\}^+\}$$

Respuesta: Para la gramática, recordemos que hay una jerarquía de operaciones: primero los identificadores x y y, luego el producto $*$, le siguen los paréntesis ( ) y al final la suma +. Entonces, sea $\mathcal{G}_1 = (V, \Sigma, P, S)$, donde $V = \{S_0, S, A, B\}$ son los símbolos no terminales, $\Sigma = \{a, b\}$ son los símbolos terminales, S_0 es el símbolo inicial, y P son las reglas de producción:

$$\begin{aligned} S_0 &\Rightarrow S \\ S &\Rightarrow aSa \mid bSb \mid aa \mid bb \end{aligned}$$

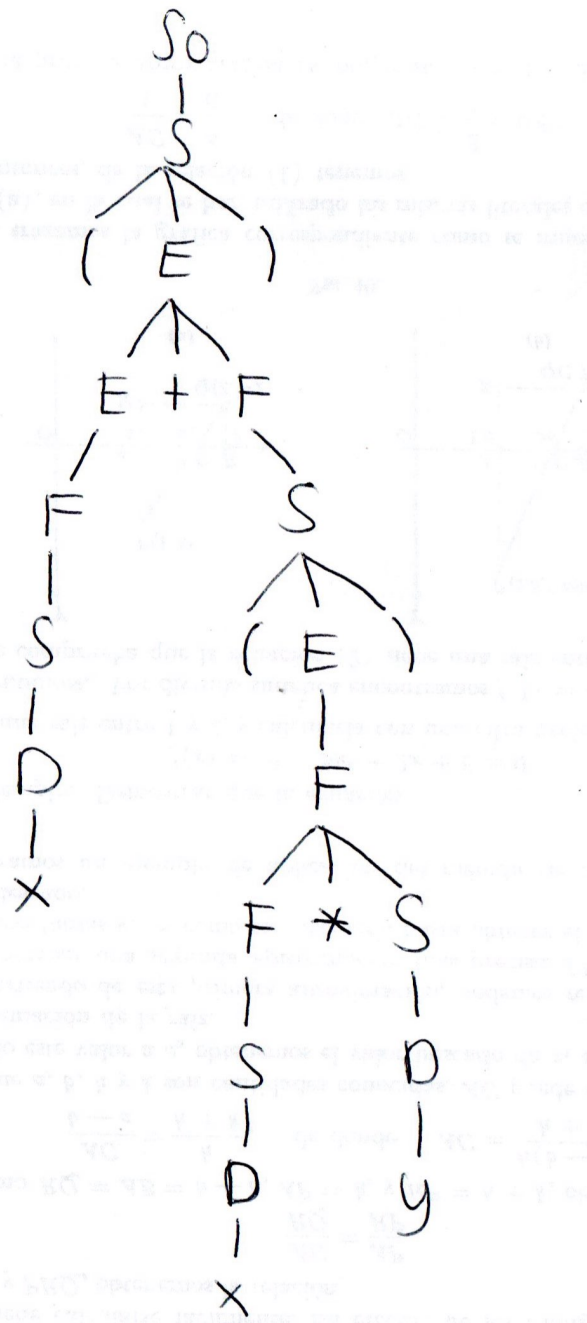
De manera que las primeras dos opciones en S son una llamada recursiva para agregar un símbolo al principio y agregar el mismo al final. Luego, las otras dos opciones son los casos base que generan las cadenas más pequeñas posibles del lenguaje. Así, cuando acaba la recursión, generamos cadenas con cantidad igual de un mismo símbolo, está balanceado.

2. **(1 pt)** Construye la Gramática Libre de Contexto para el lenguaje formal que capture las **expresiones algebraicas** que se construyen con identificadores **x** y **y**, operaciones infijas $*$ y +, y paréntesis izquierdos y derechos, donde puede haber anidación de operaciones, por ejemplo $(x + (x * y))$ y da el árbol de derivación que produce la expresión anterior.

Respuesta: Para la gramática, sea $\mathcal{G}_2 = (V, \Sigma, P, S)$, donde $V = \{S_0, S, D, E, F\}$ son los símbolos no terminales, $\Sigma = \{x, y,), (\}$ son los símbolos terminales, S_0 es el símbolo inicial, y P son las reglas de producción:

$$\begin{aligned} S_0 &\Rightarrow S \\ S &\Rightarrow D \mid (E) \\ D &\Rightarrow x \mid y \\ E &\Rightarrow E + F \mid F \\ F &\Rightarrow F * S \mid S \end{aligned}$$

Para el árbol de derivación:



3. **(1 pt)** Convierte la siguiente Grámatica Libre de Contexto en su correspondiente Forma Normal de Chomsky.

$$S \to aSa \mid bSb \mid a \mid b \mid \epsilon$$

Respuesta: Vamos a seguir los pasos dados en esta tarea.

Paso 1. Creamos una nueva producción inicial $S_0 \Rightarrow S$ de manera que ahora tenemos:

$$S_0 \Rightarrow S \quad (1)$$
$$S \Rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon \quad (2)$$

Paso 2. Quitamos las producciones vacías. Para ello nos fijamos en las que aparece y como podemos sustituirlas para que no se generen otras cosas sino que se puedan sustituir directamente. Estas son a y b junto con ϵ Entonces tenemos:

$$S_0 \Rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon \quad (1,1)$$
$$S \Rightarrow aSa \mid bSb \mid a \mid b \mid \quad (2,2)$$

Paso 3. Eliminamos las producciones que llevan a un terminal y a un no terminal (mixtas). Entonces vamos a crear producciones unitarias para terminales.

$$U_a \Rightarrow a \quad (3)$$
$$U_b \Rightarrow b \quad (4)$$

Las reemplazamos en (1,1) y (2,1):

$$S_0 \Rightarrow U_aSU_a \mid U_bSU_b \mid U_a \mid U_b \mid \epsilon \quad (1,2)$$
$$S \Rightarrow U_aSU_a \mid U_aU_a \mid U_bSU_b \mid U_bU_b \mid U_a \mid U_b \mid \quad (2,2)$$

Paso 4. Como no cumplen que sean de longitud < 3, entonces creamos nuevas producciones que también deben cumplir con la longitud.

$$V_a \Rightarrow SU_a \quad (5)$$
$$V_b \Rightarrow SU_b \quad (6)$$

Las reemplazamos en (1,2) y (2,2):

$$S_0 \Rightarrow U_aV_a \mid U_bV_b \mid U_a \mid U_b \mid \epsilon \quad (1,3)$$
$$S \Rightarrow U_aV_a \mid U_aU_a \mid U_bV_b \mid U_bU_b \mid U_a \mid U_b \mid \quad (2,3)$$

Ahora las reglas de producción son:

$$
\begin{aligned}
S_0 &\Rightarrow U_aV_a \mid U_bV_b \mid U_a \mid U_b \mid \epsilon && (1,3)\\
S &\Rightarrow U_aV_a \mid U_aU_a \mid U_bV_b \mid U_bU_b \mid U_a \mid U_b \mid && (2,3)\\
U_a &\Rightarrow a && (3)\\
U_b &\Rightarrow b && (4)\\
V_a &\Rightarrow SU_a && (5)\\
V_b &\Rightarrow SU_b && (6)
\end{aligned}
$$

4. **(1.5 pt)** Convierte la siguiente Grámatica Libre de Contexto en su correspondiente Forma Normal de Chomsky.

$$
\begin{aligned}
\rightarrow S &\Rightarrow 0Y\\
S &\Rightarrow XSX\\
X &\Rightarrow Y\\
X &\Rightarrow S\\
Y &\Rightarrow b\\
Y &\Rightarrow \lambda
\end{aligned}
$$

Respuesta: Vamos a seguir los pasos dados en esta tarea.

Paso 1. Creamos una nueva producción inicial $S_0 \Rightarrow S$ de manera que ahora tenemos:

$$
\begin{aligned}
S_0 &\Rightarrow S && (1)\\
S &\Rightarrow 0Y \mid XSX && (2)\\
X &\Rightarrow Y \mid S && (3)\\
Y &\Rightarrow b \mid \lambda && (4)
\end{aligned}
$$

Paso 2. Quitamos las producciones vacías (λ). Para ello nos fijamos en las que aparece y como podemos sustituirlas para que no se generen otras cosas sino que se puedan sustituir directamente. Las variables anulables son: Y (por 4), X (por $X \Rightarrow Y$ y Y anulable).

Entonces reemplazamos esos casos con variables anulables:

- De $S \Rightarrow 0Y$: añadimos $S \Rightarrow 0$
- De $S \Rightarrow XSX$: añadimos $S \Rightarrow SX$, $S \Rightarrow XS$ (ignoramos $S \Rightarrow S$)
- Eliminamos $Y \Rightarrow \lambda$

Ahora tenemos:

$$
\begin{aligned}
S_0 &\Rightarrow S && (1)\\
S &\Rightarrow 0Y \mid 0 \mid XSX \mid SX \mid XS && (2,1)\\
X &\Rightarrow Y \mid S && (3,1)\\
Y &\Rightarrow b && (4,1)
\end{aligned}
$$

Paso 3. Eliminar producciones unitarias. Estas unidades son: (1), (3,1) y $X \Rightarrow S$.

- En (1), de S_0 tenemos producciones no unitarias de S, entonces

$$S_0 \Rightarrow 0Y \mid 0 \mid XSX \mid SX \mid XS$$

- En $X \Rightarrow Y$ tenemos $Y \Rightarrow b$, entonces añadimos $X \Rightarrow b$
- En $X \Rightarrow S$, entonces añadimos producciones de S:

$$X \Rightarrow 0Y \mid 0 \mid XSX \mid SX \mid XS$$

- Eliminamos producciones unitarias originales

Así obtenemos:

$$\begin{aligned} S_0 &\Rightarrow 0Y \mid 0 \mid XSX \mid SX \mid XS \quad (1,2) \\ S &\Rightarrow 0Y \mid 0 \mid XSX \mid SX \mid XS \quad (2,2) \\ X &\Rightarrow b \mid 0Y \mid 0 \mid XSX \mid SX \mid XS \quad (3,2) \\ Y &\Rightarrow b \quad (4,2) \end{aligned}$$

Paso 4. Introducir producciones para terminales

$$U_0 \Rightarrow 0 \quad (5), \quad U_b \Rightarrow b \quad (6)$$

Reemplazamos en reglas mixtas:

$$\begin{aligned} S_0 &\Rightarrow U_0Y \mid 0 \mid XSX \mid SX \mid XS \\ S &\Rightarrow U_0Y \mid 0 \mid XSX \mid SX \mid XS \\ X &\Rightarrow b \mid U_0Y \mid 0 \mid XSX \mid SX \mid XS \\ Y &\Rightarrow b \\ U_0 &\Rightarrow 0 \\ U_b &\Rightarrow b \end{aligned}$$

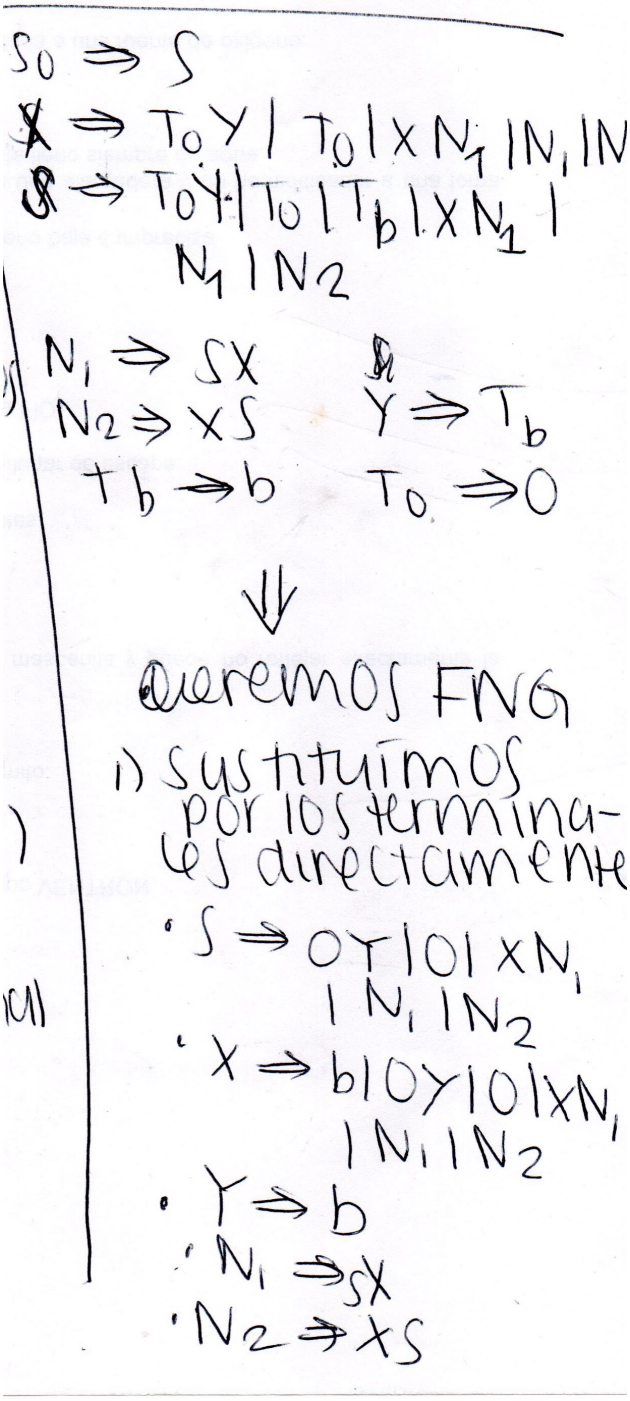
Paso 5. Como no cumplen que sean de longitud < 3, entonces creamos nuevas producciones que también deben cumplir con la longitud. Como XSX tiene 3 símbolos, entonces introducimos $Z \Rightarrow SX(7)$. Luego reemplazamos XSX por XZ en todas partes.

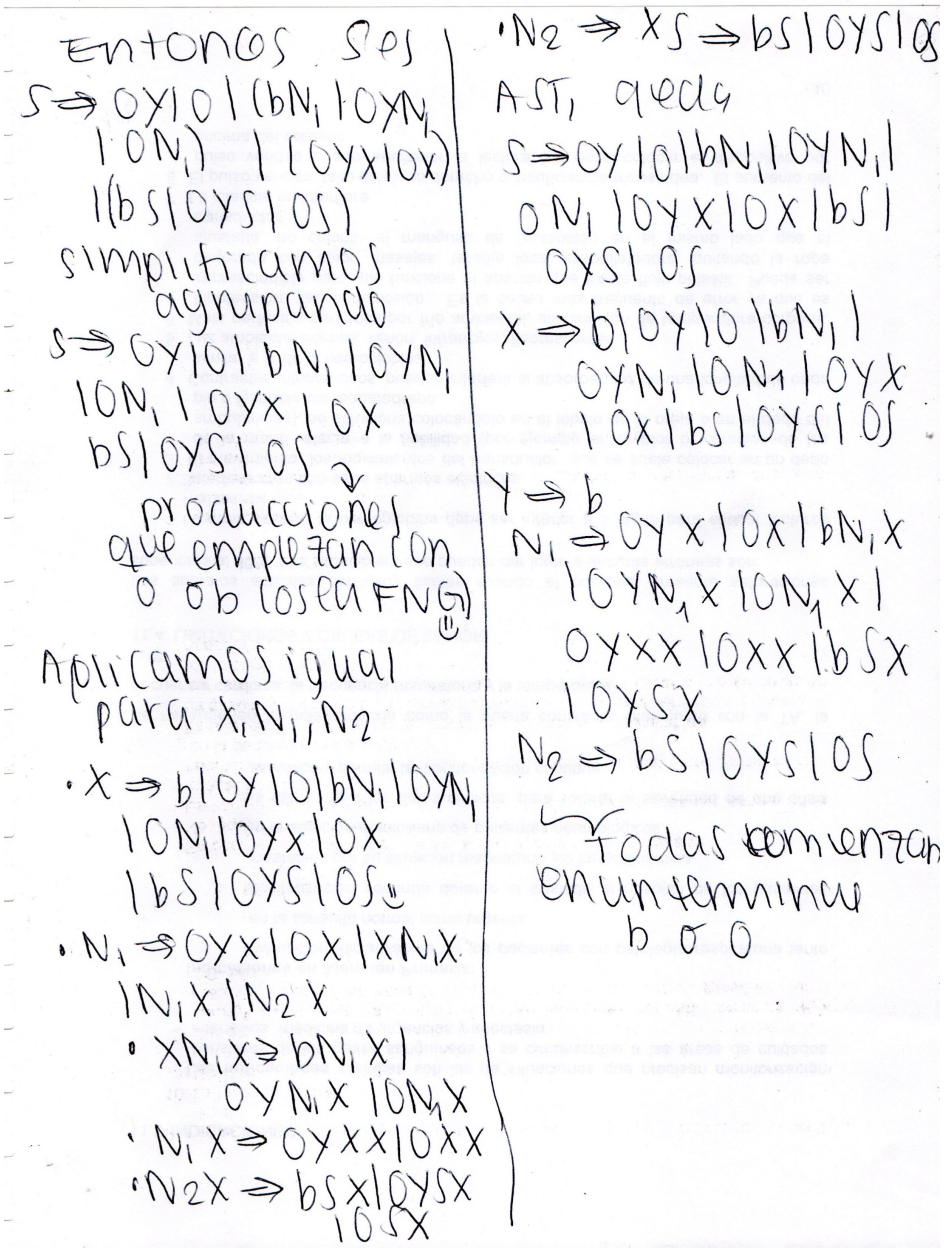
Ahora las reglas de producción son:

$$\begin{aligned} S_0 &\Rightarrow U_0Y \mid 0 \mid XZ \mid SX \mid XS \\ S &\Rightarrow U_0Y \mid 0 \mid XZ \mid SX \mid XS \\ X &\Rightarrow b \mid U_0Y \mid 0 \mid XZ \mid SX \mid XS \\ Y &\Rightarrow b \\ U_0 &\Rightarrow 0 \\ Z &\Rightarrow SX \qquad (7) \end{aligned}$$

5. **(1.5 pt)** Convierte la Grámatica Libre de Contexto del inciso anterior a su correspondiente Forma Noral de Greibach. Puedes apoyarte del resultado del ejercicio anterior para realizarlo.

Respuesta: Tenemos





6. **(2 pt)** Construye un autómata con pila general (no - determinista) para la gramática del inciso anterior. Además, propón para cada uno una cadena legal de longitud 8 y muestra el cómputo de cada autómata usando la **descripción inmediata** ($\vdash$).

   Respuesta: Una disculpita - unu - no pude hacerlo.

7. **(1 pt)** Investiga y describe para qué sirve la construcción de analizadores sintácticos hacia arriba o hacia abajo (*top-down/botton-up parsers*) y describe los algoritmos para cada uno.

   Respuesta: Los analizadores sintácticos determinan la estructura sintáctica de una secuencia de tokens (una entrada). Para ello, compara los tokens con las reglas de una gramática formal. El resultado es un árbol de análisis sintáctico, una estructura de datos jerárquica que representa visualmente cómo se combinan las reglas gramaticales para formar el programa. Este árbol es una entrada necesaria para las pasos posteriores de un compilador, como el análisis semántico y la generación de código.

   Los analizadores sintácticos se clasifican según la dirección en la que construyen este árbol de análisis:

   a) Descendentes (hacia abajo): comienzan con el símbolo inicial de la gramática (la raíz del árbol) y aplican producciones para expandir los no terminales hasta que coincidan con los tokens de entrada.
      - Tiene el objetivo de construir el árbol de análisis sintáctico desde la raíz hacia las hojas. El analizador sintáctico encuentra la derivación más a la izquierda para la cadena de entrada expandiendo el no terminal más a la izquierda en cada paso.
      - Su algoritmo principal es el de descenso recursivo/análisis sintáctico LL.
      - Su funcionamiento es que para cada no terminal de la gramática se implementa como una función. Estas funciones se llaman entre sí para reflejar las producciones de la gramática. Luego, utiliza un token de anticipación para predecir qué producción aplicar, lo que le permite funcionar sin retroceso. Este comportamiento predictivo define el análisis LL (escaneo de izquierda a derecha, derivación más a la izquierda). Es decir, el analizador comienza con la función para el símbolo de inicio, y basándose en la anticipación, elige una producción y ejecuta el código correspondiente, lo que normalmente implica hacer coincidir terminales y llamar a funciones para no terminales.

   b) Ascendentes (hacia arriba): comienzan con los tokens de entrada (las hojas del árbol) y aplican reducciones para agrupar los tokens en no terminales hasta que formen el símbolo inicial.
      - Tiene el objetivo de construir el árbol de análisis sintáctico desde las hojas hacia la raíz. El analizador sintáctico realiza una serie de reducciones, que es el proceso de sustituir una cadena que coincide con el lado derecho de una producción por el no terminal de su lado izquierdo. La secuencia de reducciones es una derivación más a la derecha en sentido inverso.
      - Su algoritmo principal es el análisis sintáctico Shift-Reduce / LR
      - Para su funcionamiento, el algoritmo utiliza una pila para almacenar una secuencia de símbolos y estados gramaticales. Lee la entrada de izquierda a derecha. Además, hace las acciones de:
        - Desplazamiento: Mueve el token de entrada actual a la pila.
        - Reducción: Cuando los símbolos en la parte superior de la pila forman el lado derecho de una producción, los extrae y empuja el no terminal del lado izquierdo correspondiente a la pila. Esta acción crea un nodo padre en el árbol de análisis.
        - Control: Una tabla de análisis dicta las acciones. Esta tabla especifica si se debe desplazar, reducir, aceptar o señalar un error en función del estado actual (en la pila) y el token de anticipación. Este método se le llama análisis LR (escaneo de izquierda a derecha, derivación más a la derecha por Left Right).

8. **(1 pt)** Indica que lenguaje formal describe el siguiente autómata de pila:

   Respuesta: Primero vemos que el estado q_0 lo que hace es que el autómata solo mete símbolos$a$la pila. Si lee a, mete una a, y si lee una b, mete una b . Esto significa que está guardando la primera parte de la cadena, símbolo por símbolo.

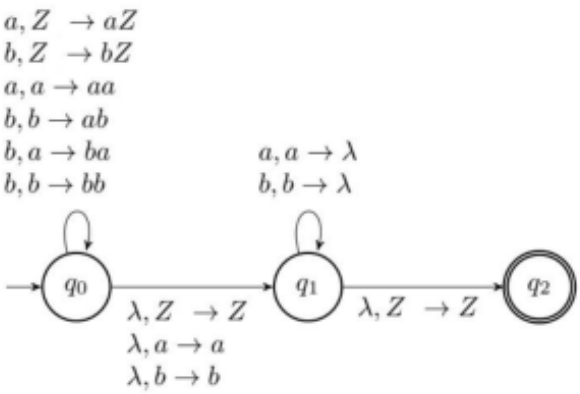


Figura 1: Autómata de Pila para el ejercicio 8:)

Luego está la transición de ϵ a q_1, lo que nos dice que se puede "brincar sin leer nada", por lo que el autómata puede decidir si hasta ahí deja de guardar y ahora empiezo a comparar, donde ese "hasta aquí" es el final de la primera mitad, es decir, es un punto que divide la cadena como si fuera algo que se refleja en un espejo.

Es decir, acepta las cadenas de la forma ww^r. Entonces, el lenguaje que lo describe es:

$$\mathcal{L} = \{ww^r \mid w \in \{a, b\}^*\}$$

Además, en el estado q_1 el autómata ya no guarda, si no que compara, pues si lee a y la pila tiene a arriba, entonces saca la a ; pero si lee b y la pila tiene b arriba, entonces saca la b . O sea, para cada símbolo que lee en esta etapa, debe coincidir con el que está arriba de la pila. Esto solo funciona si la entrada es exactamente el reverso de lo que metió antes.

Lo interesante es que si la segunda mitad no es el reverso exacto, entonces alguna comparación fallaría. Ya sea que leería a pero habría un b arriba, o leería b pero habría a , o la pila se vaciaría antes. En todos esos casos el autómata se atoraría y no aceptaría.

Así, cuando la pila queda vacía (osea cuando tiene Z) entonces pasa a q_2 y lo acepta. Lo que significa que todas las letras de la primera parte fueron metidas en orden, todas las letras de la segunda parte coincidieron al reverso y que la pila volvió exactamente a como estaba al inicio.

9. **Extra:(1 pt)** Existen problemas indecidibles, es decir, problemas para los cuales no existe un algoritmo que pueda determinar si la respuesta es afirmativa o negativa. Menciona dos problemas indecidibles relacionados con las gramáticas libres de contexto y explica algunas implicaciones que estos tienen para la construcción de compiladores.

   Respuesta: El primer problema consiste en determinar si una gramática libre de contexto es ambigua. Este problema es indecidible, lo que implica que no existe un algoritmo capaz de verificar de manera infalible si una gramática dada puede generar alguna cadena mediante más de un árbol de derivación.

   Además, esta limitación tiene implicaciones en la construcción de compiladores. Dado que los analizadores sintácticos requieren gramáticas no ambiguas para su correcto funcionamiento, la responsabilidad recae en los diseñadores de lenguajes de programación, quienes deben verificar manualmente la ausencia de ambigüedades mediante pruebas exhaustivas y patrones de diseño gramatical conocidos, sin poder depender de una verificación automática completa. Otra implicación es que cuando una ambigüedad es

conocida e inevitable (o muy difícil de eliminar de la gramática), los compiladores implementan reglas de desambiguación en el propio parser.

El segundo problema indecidible es la equivalencia entre gramáticas libres de contexto, que consiste en determinar si dos gramáticas generan exactamente el mismo lenguaje.

Así que, esta limitación también impacta directamente en el diseño de los compiladores. Por ejemplo, cuando se realizan transformaciones de código para optimización o se modifica la gramática de un lenguaje en una nueva versión, resulta imposible verificar de manera general si las representaciones anteriores y posteriores son equivalentes. Entonces, los implementadores de compiladores deben apoyarse en técnicas parciales, como la verificación de equivalencia para casos específicos o la validación empírica mediante baterías de pruebas, en lugar de contar con un método automático y general.

La razón fundamental de que ambos sean indecidibles es que si tal algoritmo existiera, se podría utilizar para resolver otro problema que ya se ha demostrado indecidible: el Problema de la Parada (Halting Problem) de Alan Turing. Como el Problema de la Parada es indecidible, el problema de la ambigüedad también debe serlo

Fuentes:

- ¿Es decidible determinar si una gramática libre de contexto es ambigua? (2023, agosto). Instituto Europeo de Certificación TI. https://es.eitca.org/la-seguridad-cibern%C3%A9tica/eitc-es-fundamentos-de-la-teor%C3%ADa-de-la-complejidad-computacional-cctf/decidibilidad/problemas-relacionados-con-los-lenguajes-libres-de-contexto/problemas-de-revisi%C3%B3n-de-examen-relacionados-con-lenguajes-libres-de-contexto/%C2%BFEs-decidible-determinar-s
C3%A1tica-libre-de-contexto-es-ambigua%3F/
- ¿Es posible determinar si dos gramáticas libres de contexto aceptan el mismo lenguaje? (2023, agosto). Instituto Europeo de Certificación TI. https://es.eitca.org/la-seguridad-cibern%C3%A9tica/eitc-es-fundamentos-de-la-teor%C3%ADa-de-la-complejidad-computacional-cctf/decidibilidad/problemas-relacionados-con-los-lenguajes-libres-de-contexto/problemas-de-revisi%C3%B3n-de-examen-relacionados-con-lenguajes-libres-de-contexto/%C2%BFEs-posible-determinar-si-
C3%A1ticas-independientes-del-contexto-aceptan-el-mismo-idioma%3F-%C2%BFEs-este-problema-deci
3F/

1. Forma Normal de Chomsky(FNC)

Toda grámatica libre de contexto se puede transformar a forma normal de Chomsky, y toda grámatica en forma normal de Chomsky es libre de contexto. En la forma normal de Chomsky todas las reglas de producción tienen que ser la forma

$$A \Rightarrow BC$$
$$A \Rightarrow a$$

Es decir, las reglas solo pueden tener **dos** símbolos no terminales (BC) o un símbolo terminal a.

2. Pasos para pasar a FNC

1. Si S está en alguna regla de producción, creamos un nuevo símbolo inicial S_0
2. Quitamos todas las producciones anulables
3. Quitamos las producciones con un único símbolo no terminal ($A \to B$).
4. Quitamos las producciones mezcladas, es decir, que tienen una combinación de variables y símbolos terminales.
5. Quitamos las producciones con más de dos variables.

2.1. Ejemplo

$$S \to ASB$$
$$A \to aAS \mid a \mid \epsilon$$
$$B \to SbS \mid A \mid bb$$

Introducimos un nuevo símbolo inicial

$$\begin{aligned} S_0 &\to S \\ S &\to ASB \\ A &\to aAS \mid a \mid \epsilon \\ B &\to SbS \mid A \mid bb \end{aligned}$$

2.2. Paso 1

Quitamos todas las producciones anulables, sustituyéndolas en las otras producciones. En este caso estamos sustituyendo $A \to \epsilon$

$$\begin{aligned} S_0 &\to S \\ S &\to ASB \mid SB \\ A &\to aAS \mid a \mid aS \\ B &\to SbS \mid A \mid \epsilon \mid bb \end{aligned}$$

Esto nos dejó con que ahora B se puede anular. Entonces también vamos a sustituir $B \to \epsilon$

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASB \mid SB \mid AS \mid S \\
A &\to aAS \mid a \mid aS \\
B &\to SbS \mid A \mid bb
\end{aligned}
$$

2.3. Paso 2

Ahora quitamos las producciones unitarias. Acá tenemos $B \to A$. Entonces sustituimos el valor de A y agregamos sus reglas de producción a B.

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASB \mid SB \mid AS \mid S \\
A &\to aAS \mid a \mid aS \\
B &\to SbS \mid aAS \mid a \mid aS \mid bb
\end{aligned}
$$

2.4. Paso 3

Ahora quitamos también las reglsa unitarias $S_0 \to S$ y $S \to S$.

$$
\begin{aligned}
S_0 &\to ASB \mid SB \mid AS \\
S &\to ASB \mid SB \mid AS \\
A &\to aAS \mid a \mid aS \\
B &\to SbS \mid aAS \mid a \mid aS \mid bb
\end{aligned}
$$

2.5. Paso 4

Quitamos las producciones mezcladas. Para esto añadimos nuevos símbolos para quitar los terminales

$$
\begin{aligned}
S_0 &\to ASB \mid SB \mid AS \\
S &\to ASB \mid SB \mid AS \\
A &\to X_1AS \mid a \mid X_1S \\
B &\to SX_2S \mid X_1AS \mid a \mid X_1S \mid X_2X_2 \\
X_1 &\to a \\
X_2 &\to b
\end{aligned}
$$

2.6. Paso 5

Quitamos las producciones con más de dos variables.

$$
\begin{aligned}
S_0 &\to AX_3 \mid SB \mid AS \\
S &\to AX_3 \mid SB \mid AS \\
X_3 &\to SB \\
A &\to X_1X_4 \mid a \mid X_1S \\
B &\to SX_5 \mid X_1X_4 \mid a \mid X_1S \mid X_2X_2 \\
X_4 &\to AS \\
X_5 &\to X_2S \\
X_1 &\to a \\
X_2 &\to b
\end{aligned}
$$

3. Forma Normal de Greibach

Para convertir a Forma Normal de Greibach queremos que todas nuestras reglas de producción sean de la forma $A \to a\alpha$, donde a es un símbolo **terminal** y α es 0 o más **no-terminales**.

Pasos para pasar a Greibach:

1. Pasar la gramática a Forma Normal de Chomsky
2. Quitar las recursiones izquierdas
3. Convierte la regla de producción a forma GNF en la gramática. (quitamos las producciones mezcladas y dejamos a solo un terminal de lado derecho)