Universidad del Valle de Guatemala
Facultad de Ingeniería
Teoría de la Computación

INFORME PROYECTO DE TEORÍA DE LENGUAJES Y AUTÓMATAS

Diederich Solis - 22952
Davis Roldan -

Diseño de la aplicación

En este proyecto, nuestro objetivo es desarrollar una aplicación que convierta gramáticas en su Forma Normal de Chomsky (CNF) y aplique el algoritmo CYK para verificar si una cadena pertenece al lenguaje generado por dicha gramática. Para lograr esto, hemos optado por un diseño modular, que facilita la claridad en el proceso, la mantenibilidad del código y la posible escalabilidad futura.

- Metodología de Diseño

La aplicación está diseñada siguiendo una arquitectura modular centrada en componentes. Cada fase del análisis de gramáticas se divide en módulos especializados, cada uno con funciones y responsabilidades bien definidas. Este enfoque no solo mejora la organización, sino que también simplifica el desarrollo y garantiza que cada componente sea fácilmente testeable de forma independiente.

Composicion del Código

- **CYK:**
  Implementa el algoritmo CYK (Cocke-Younger-Kasami) para determinar si una palabra dada puede ser generada por una gramática en forma normal de Chomsky (CNF). Recibe como entrada una gramática representada como un diccionario, una lista de cadenas que conforman la palabra a analizar y el símbolo de inicio de la gramática. Inicialmente, crea una tabla bidimensional donde cada celda almacena los símbolos no terminales que pueden generar las subcadenas de la palabra. Primero, llena la tabla con los símbolos que corresponden a los terminales de la palabra. Luego, procede a llenar las celdas correspondientes a las combinaciones de dos no terminales, verificando si pueden ser generados a partir de las producciones de la gramática. Finalmente, la función devuelve un valor booleano que indica si la palabra es aceptada por la gramática y, en caso afirmativo, también devuelve la tabla utilizada en el proceso.

- **Grammar:**
  Este código procesa y transforma gramáticas formales para convertirlas a Forma Normal de Chomsky (CNF), simplificando su estructura. Realiza validaciones con expresiones regulares, identifica terminales y no terminales, elimina producciones épsilon, unitarias, símbolos no derivables e inalcanzables, y adapta las reglas para cumplir con CNF. Además, permite generar combinaciones de términos (powerset) y cargar gramáticas desde archivos de texto. Todo esto facilita su uso en análisis sintáctico y procesamiento de lenguajes.

- **Parsetree:**
  Este código permite visualizar un árbol de análisis sintáctico utilizando una tabla CYK y una gramática dada. Define una clase `Node` para representar nodos del árbol, con símbolos y referencias a sus hijos. La función `generate_tree` construye el árbol recursivamente a partir de la tabla CYK, identificando reglas de la gramática que se correspondan con los símbolos. Luego, `draw_tree` dibuja el árbol mediante

`NetworkX` y lo muestra utilizando `matplotlib`. También incluye verificaciones para asegurar que la tabla CYK esté correctamente estructurada y que contenga el símbolo inicial 'S', necesario para generar el árbol.

- **Main:**

  Este código implementa un analizador sintáctico usando el algoritmo CYK. Primero, carga una gramática desde un archivo (`grammar.txt`), convierte las reglas a Forma Normal de Chomsky (CNF) y solicita al usuario una oración en inglés. Luego, utiliza la función `cykParse` para verificar si la oración es aceptada por la gramática, generando la tabla CYK correspondiente. Si la oración es aceptada, visualiza el árbol sintáctico con `visualize_tree`. Finalmente, muestra el tiempo de ejecución total del análisis.

Discusión

Este proyecto dejó importantes aprendizajes sobre el desarrollo de algoritmos, implementación rigurosa y buenas prácticas en el manejo de gramáticas formales. La validación exhaustiva de las gramáticas antes de su procesamiento por el algoritmo CYK se reveló crucial, ya que errores en la forma o en la conversión a Forma Normal de Chomsky (FNC) pueden desencadenar fallos lógicos difíciles de detectar más adelante. Esto incluye la eliminación de producciones epsilon, unitarias y redundantes para garantizar que el algoritmo funcione de forma óptima. Además, la depuración en tiempo real mediante el seguimiento del estado de la tabla CYK y las combinaciones de subcadenas resultó fundamental para detectar errores complejos que no eran evidentes a simple vista, subrayando la importancia de un flujo de trabajo estructurado que permita inspeccionar las estructuras de datos a lo largo del proceso. A futuro, se recomienda ampliar las pruebas automatizadas para cubrir casos de uso más complejos, garantizando que el algoritmo maneje correctamente tanto gramáticas sofisticadas como cadenas variadas. También es relevante mejorar el manejo de errores mediante mensajes más detallados que orienten al usuario, ofreciendo sugerencias sobre cómo corregir gramáticas mal estructuradas o explicando por qué una cadena no fue aceptada. Esto facilitaría la depuración y aumentaría la usabilidad del sistema. Finalmente, aunque la implementación actual del algoritmo tiene una complejidad temporal de $O(n^3 * |G|)$, sería beneficioso explorar optimizaciones enfocadas en mejorar el rendimiento con gramáticas extensas o cadenas largas, posiblemente mediante paralelización o técnicas más eficientes en términos de uso de memoria.

Ejemplos y pruebas realizadas

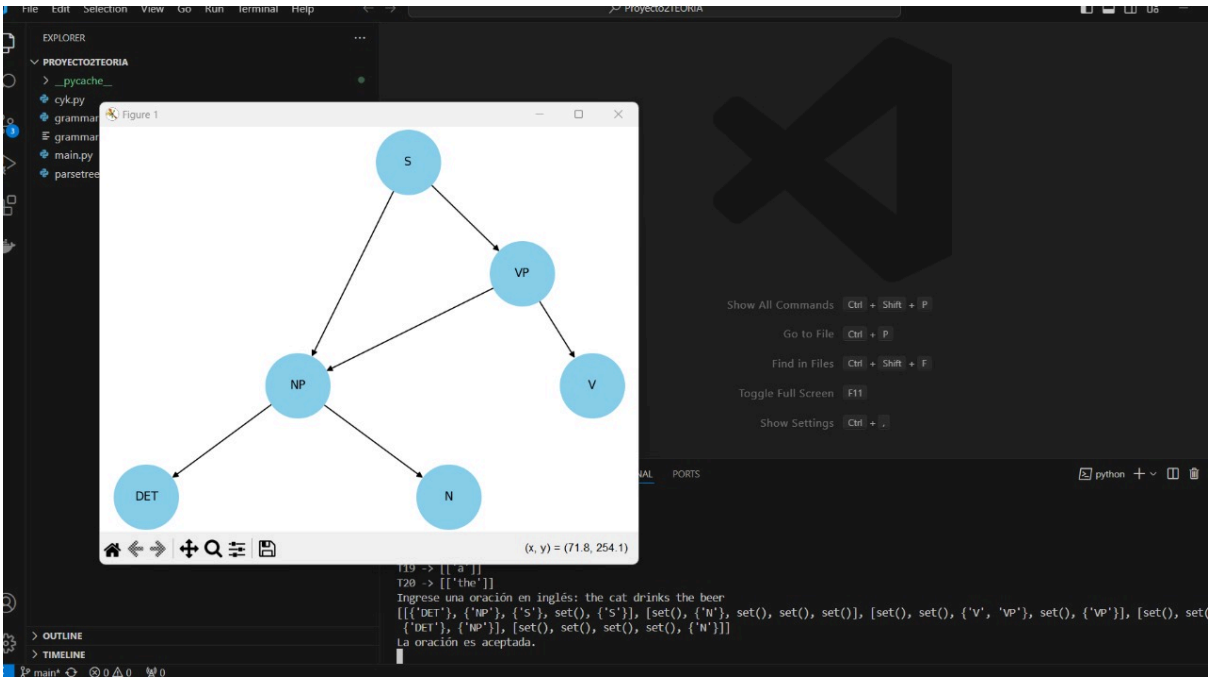


Figure 1
S
VP
NP
V
DET
N
(x, y) = (71.8, 254.1)
Show All Commands Ctrl + Shift + P
Go to File Ctrl + P
Find in Files Ctrl + Shift + F
Toggle Full Screen F11
Show Settings Ctrl + ,
T19 -> [['a']]
T20 -> [['the']]
Ingrese una oración en inglés: the cat drinks the beer
La oración es aceptada.

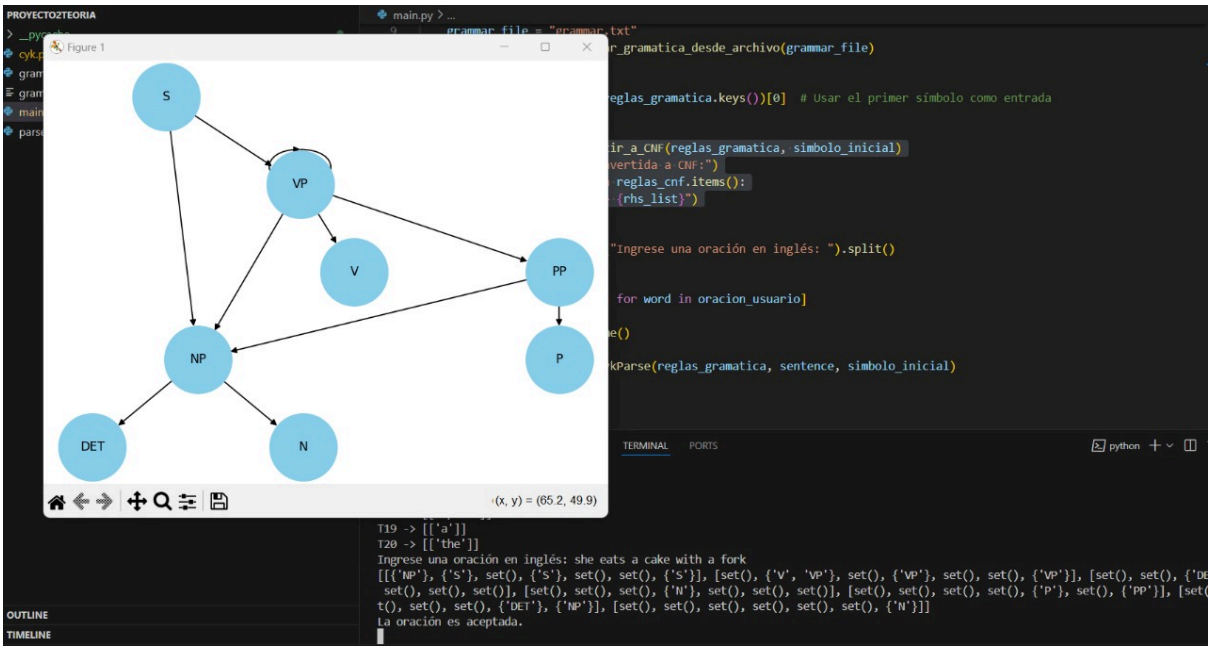
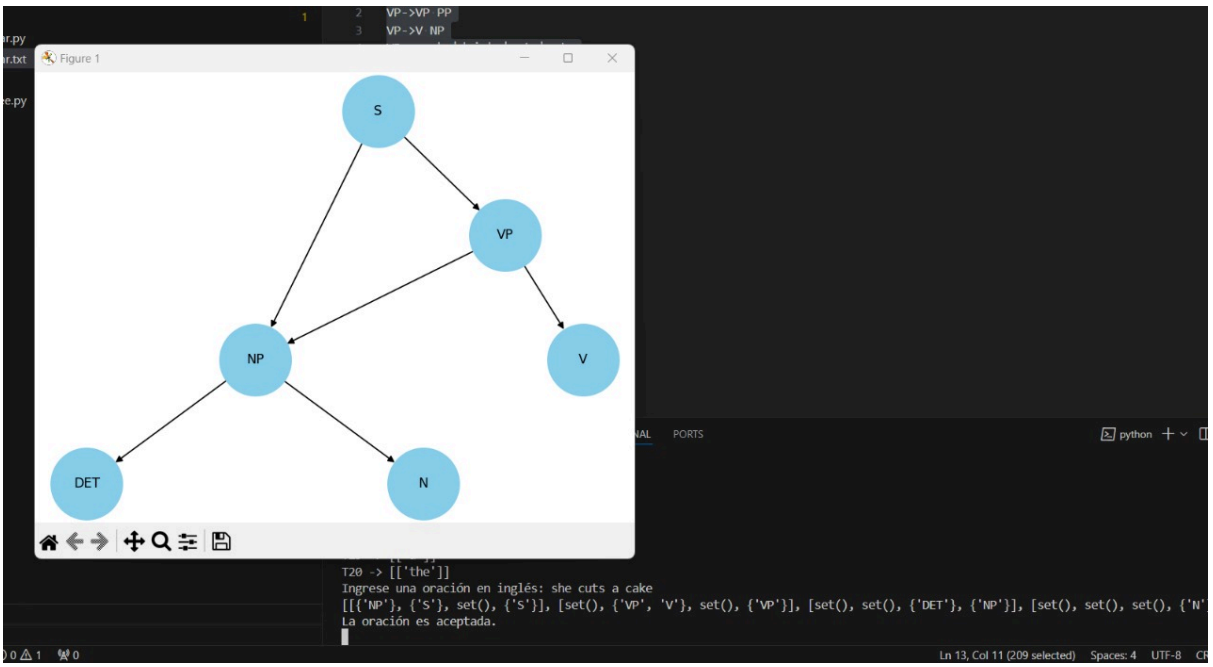
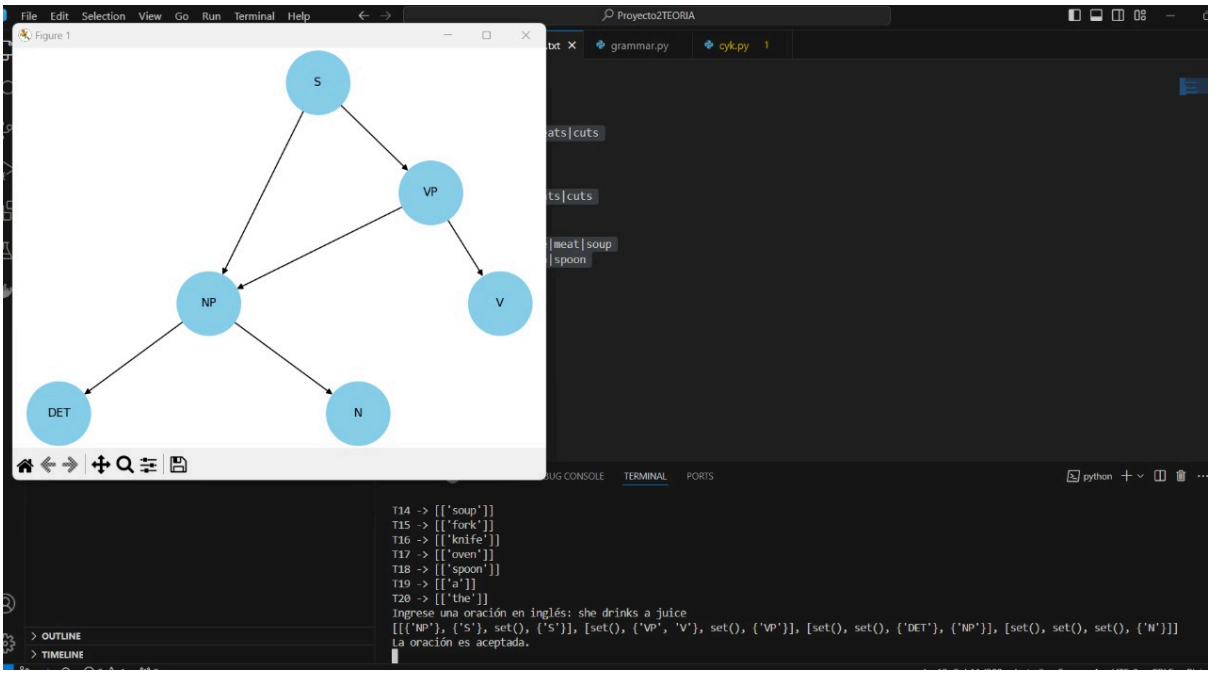
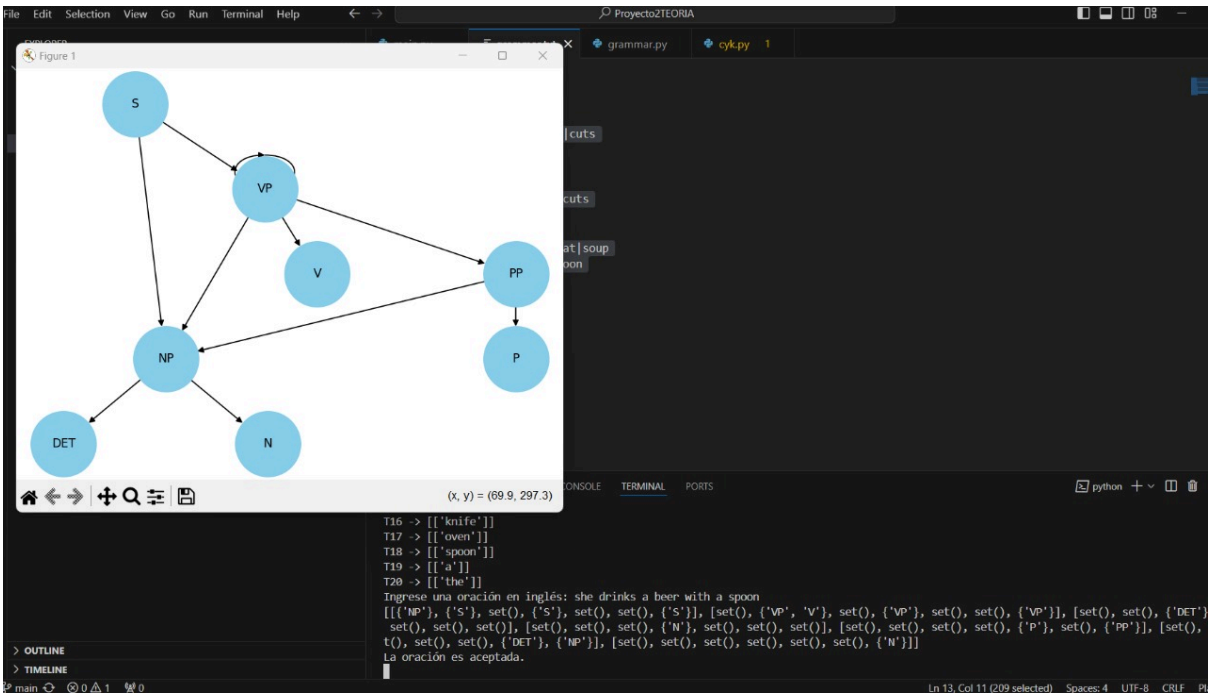


Figure 1
S
VP
V
PP
NP
P
DET
N
(x, y) = (65.2, 49.9)
T19 -> [['a']]
T20 -> [['the']]
Ingrese una oración en inglés: she eats a cake with a fork
La oración es aceptada.


VP->VP PP
VP->V NP
Figure 1
S
VP
NP
V
DET
N
T20 -> [['the']]
Ingrese una oración en inglés: she cuts a cake
[[{'NP'}, {'S'}, set(), {'S'}], [set(), {'VP', 'V'}, set(), {'VP'}], [set(), set(), {'DET'}, {'NP'}], [set(), set(), set(), {'N'
La oración es aceptada.


File Edit Selection View Go Run Terminal Help
Proyecto2TEORIA
Figure 1
grammar.py
cyk.py
S
VP
NP
V
DET
N
TERMINAL
PORTS
T14 -> [['soup']]
T15 -> [['fork']]
T16 -> [['knife']]
T17 -> [['oven']]
T18 -> [['spoon']]
T19 -> [['a']]
T20 -> [['the']]
Ingrese una oración en inglés: she drinks a juice
[[{'NP'}, {'S'}, set(), {'S'}], [set(), {'VP', 'V'}, set(), {'VP'}], [set(), set(), {'DET'}, {'NP'}], [set(), set(), set(), {'N'}]]
La oración es aceptada.
OUTLINE
TIMELINE



Indique en su reporte ejemplos de frases en el lenguaje, y ejemplos que no estén en el lenguaje:

-LAS QUE NO:

- **He runs fast**
- **She eats the table**
- **They drink juice**
- **He cuts with the knife**
- **She cooks the cake and the soup**
- **He eats**
- **The dog drinks beer**
- **She eats quickly**
- **He cuts meat with the knife and fork**
- **She bakes bread**

1. He runs fast
Problema: El verbo "runs" y el adverbio "fast" no están definidos en tu gramática.
Explicación: Tu gramática solo incluye los verbos "cooks", "drinks", "eats", "cuts" y no maneja adverbios como "fast".

2. She eats the table
Problema: El sustantivo "table" no está incluido en la lista de sustantivos (N) de tu gramática.
Explicación: Tus sustantivos incluyen "cat", "dog", "beer", "cake", "juice", "meat", "soup", "fork", "knife", "oven", "spoon", pero no "table".

3. They drink juice
Problema 1: El pronombre "they" no está definido en tu gramática.

Problema 2: Aunque "juice" está en tu gramática, el sujeto "they" no lo está.
Explicación: En tu gramática, NP puede derivar "he" o "she", pero no "they".

4. He cuts with the knife
Problema: Falta el objeto directo después del verbo "cuts".
Explicación: Según tu gramática, VP → V NP, por lo que después de "cuts" debería haber un sintagma nominal (NP). La estructura "cuts with the knife" no cumple con las reglas, ya que "with the knife" es un complemento preposicional (PP), pero falta el objeto directo (NP) que debe seguir al verbo.

5. She cooks the cake and the soup
Problema: La conjunción "and" no está contemplada en tu gramática.
Explicación: Tu gramática no maneja estructuras coordinadas con "and". No hay reglas para unir dos NP con una conjunción.

6. He eats
Problema: Falta el objeto directo después del verbo "eats".
Explicación: Según tu gramática, VP → V NP, por lo que "eats" debería ir seguido de un NP. La oración "He eats" está incompleta en este contexto.

7. The dog drinks beer
Problema: El sintagma nominal "The dog" no está correctamente definido en tu gramática.
Explicación: Aunque "dog" es un sustantivo en tu gramática, y "the" es un determinante, tu gramática define NP → DET N, pero "The dog" requiere que DET y N estén en las reglas apropiadas. Sin embargo, en tu gramática, NP → DET N y N → dog, por lo que en teoría debería aceptarse. Sin embargo, si tu parser no está manejando correctamente estas reglas, podría no aceptar la oración.

LINK REPO:

https://github.com/DiederichSolis/Proyecto2TEORIA.git