Diseño de la aplicación:

No se utilizó un método de modelación especifico simplemente se crearon clases las cuales cumplieran con las funciones. Fueron creadas tres clases que realizan tareas específicas para la ejecución del proyecto.

Las clases que existen son:

1. Normalización del CFG: Toma como parámetros para el constructor, una gramática y un símbolo inicial. Esta clase se encarga de Eliminar las producciones épsilon, las producciones unitarias y los caracteres inútiles.
2. Normalización de Chomsky: Toma como parámetro para el constructor, una gramática y un símbolo no terminal con el cual serán creadas las nuevas producciones. Esta clase se encarga de aplicar la normalización de Chomsky a la gramática ya normalizada. Las nuevas producciones nuevas se crean con el símbolo enviado.
3. Algoritmo CYK: Toma como parámetro una gramática normalizada de Chomsky y un símbolo inicial. Esta clase se encarga de determinar si una palabra pertenece a la gramática.
    a. Esta clase contiene una función llamada parser la cual se puede llamar con distintas palabras para verificar varias palabras si pertenecen a la gramática.

Todo esto es consumido por un archivo principal el cual lee un archivo del cual se obtiene la gramática. El primer símbolo encontrado en la gramática será tomado en cuenta como el símbolo inicial. Una vez aplicada la normalización de Chomsky se muestra la gramática y una vez aplicado el algoritmo de CYK se muestra la tabla con la respuesta.

Discusión

La parte más difícil de este proyecto fue la implementación de la normalización del CFG y la normalización de Chomsky. Para la construcción de este proyecto, fue tomada en cuenta solo gramáticas las cuales cuentan con solo un símbolo del lado izquierdo. Así se puede realizar un análisis sobre una gramática sencilla. Todos los algoritmos fueron hechos primero en papel para tener un mejor entendimiento de esto.

Se tomo en cuenta la siguiente gramática:

S→0A0|1B1|BB

A→C

B→S|A

C→S|ε

Primero se tiene que encontrar símbolos anulables, en este caso se encuentra la C, luego mediante un while se vuelve a recorrer todo encontrando que A, C también son, como se agregaron nuevos anulable y así sucesivamente se vuelve a repetir el proceso hasta encontrar

todos S,A,B,C. Para las producciones unitarias desde la primera producción hasta la ultima en ese orden se busca y se aplican. Para los símbolos inútiles primero se revisa que sean símbolos alcanzables y se eliminan los que no. Luego se Busca los símbolos produzcan un terminal y los que no se eliminan al finalizar el proceso quedaría así

S→0A0|1B1|BB|11|00

A→0A0|1B1|BB|11|00

B→0A0|1B1|BB|11|00

Para la forma normal de Chomsky se cambian las producciones = 2 y > 2 se asume que las = 1 son producciones terminales que cumplen con la forma normal de Chomsky. Todas las demás cambian los terminales por nuevas producciones. X0 → 0 y X1 → 1. Cuando longitud = 2 se revisa carácter de la izquierda y de la derecha si uno de los dos es un terminal se cambia por el nuevo generado. Para los de longitud > 2, se tiene que ver el primer símbolo y luego el resto y se verifica si es el penúltimo símbolo para no crear una nueva producción y terminar el proceso. El resultado debería de verse así:

X0 → 0

X1→1

X2→BX1

X3→AX0

S→ X0X3|X1X2|BB|X0X0|X1X1

A→X0X3|X1X2|BB|X0X0|X1X1

B→ X0X3|X1X2|BB|X0X0|X1X1

Para CYK se tiene que tomar en cuenta que no se puede construir como en papel entonces debe tener un enfoque distinto. Se crea una tabla de longitud de la palabra que se quiere y esta se comienza a llenar fila por fila hasta encontrar que en la primera fila se encuentra S, si se encuentra es que es una palabra que pertenece al lenguaje. Ahora solo se tienen que aplicar para gramáticas más complejas con mas caracteres del lado izquierdo.

Obstáculos encontrados, al construir la forma normal del Chomsky es complicada la aplicación cuando las cadenas son de largo > 2. Porque es un proceso cíclico con el cual hay que tener cuidado de no generar nuevas producciones inútiles por lo que hay que estar en constante revisión. Esta transformación me dio mucho problema porque se generaban muchas producciones inútiles y sin sentido. Además, se generaban muchas producciones repetidas. Por lo que, hay que tener en consideración estas cosas para poder aplicar bien este algoritmo.

También, hubo obstáculos en las lecturas más avanzadas como VP → VP P P, se tuvo que cambiar la lectura y se asumió que las gramáticas venían en esa forma con un espacio de por medio.

Se Recomienda que los algoritmos se hagan primero a papel para tener un mejor entendimiento sobre el tema. Además, si se quiere utilizar inteligencia artificial para solucionar cosas, es recomendable tener un modelo propio ya hecho con las especificaciones porque usualmente esta falla y no da un buen resultado. También formular bien los props para que esta no se equivoque y haga algo que no tiene.

Ejemplos Realizados:

Ejemplo 1:

Entrada:

```
S      → X1 X2
X1     → if C
X2     → X3 else S | ACTION
X3     → then S
C      → true | false
ACTION → A
A      → run | stop
```

Palabra:

```
string = "if true stop"
```

Resultado:

Ejemplo 2:

Entrada:

```
S    → NP VP
VP   → VP PP | V NP | cooks | drinks | eats | cuts
PP   → P NP
NP   → DET N | he | she
V    → cooks | drinks | eats | cuts
P    → in | with
N    → cat | dog | beer | cake | juice | meat | soup | fork | knife | oven | soup
DET → a | the
```

Palabra:

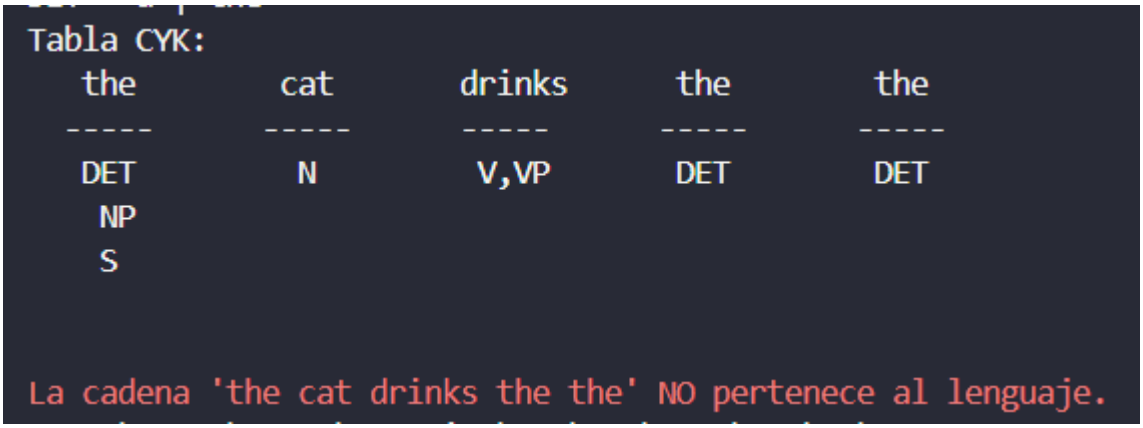
```
string = "the cat drinks the beer"
```

Resultado:

Palabra mal ingresada:

```
string = "the cat drinks the the"
```

Resultado:



Ejemplo 3:

Entrada:

```
S   → 0 A 0 | 1 B 1| B B
A   → C
B   → S | A
C   → S | ε
```

Palabra:

```
string = "0 0 1 1"
```

Resultado:

```
Gramática 1:
{'S': [['0', 'A', '0'], ['1', 'B', '1'], ['B', 'B']], 'A': [['C']], 'B': [['S'], ['A']], 'C': [['S'], ['ε']]}
Gramatica Transformada:
X0 → 1
X1 → 0
X2 → B X0
X3 → A X1
S → X1 X1 | X1 X3 | X0 X2 | X0 X0 | B B
A → X1 X1 | X1 X3 | X0 X2 | X0 X0 | B B
B → X1 X1 | X1 X3 | X0 X2 | X0 X0 | B B
Tabla CYK:
    0         0         1         1
 -----     -----     -----     -----
   X1        X1        X0        X0
 S,B,A               S,B,A
   X2
 S,B,A
La cadena '0 0 1 1' pertenece al lenguaje.
```