**Analizador sintáctico con CYK**

**Jorge Alberto Padilla Gutiérrez A01635346**

1. **Introducción**

Para la entrega de este proyecto parcial, se desarrolla el analizador sintáctico mediante el algoritmo CYK, el cual requiere de unos cuantos pasos previos a su desarrollo siendo la normalización de la gramática de entrada en la forma normal de Chomsky, y claramente indicar como se recibirá dicha gramática en un archivo de texto para poder introducirlo en el programa y poder trabajar con él.

1. **Formato de la gramática**

Para la entrega la gramática al programa, requerimos definir el formato que esta debe tener para que sea válido, el formato será simple:

S->a|B|&

Así serán todas las líneas del documento, las cuales indican cada símbolo generador que tendrá la gramática, seguido de -> que representa nuestra flecha, a partir de ahí están todos los caracteres resultantes de la producción, considerando el pipe | como un or, que indica que es otra producción del mismo símbolo generador, y tenemos el carácter & que será nuestra épsilon. Por lo tanto, si queremos de múltiples símbolos generadores tendremos diversas líneas.

1. **Almacenamiento de la gramática**

Para poder guardar nuestra gramática requerimos de una clase que tenga la lista de símbolos que estamos guardando, estos símbolos serán por si solos una clase, que llamaremos FNCh, la cual debe tener:

* El carácter que se está guardando
* Un indicador si es terminal o generador
* Un subíndice, si este se generó para la forma normal de Chomsky
* Sus producciones

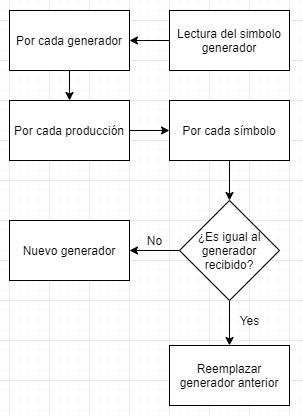
Los primeros requerimientos no tienen complejidad, ya que el carácter es meramente un carácter, el indicador es un booleano y el subíndice es nuevamente un carácter. Para las producciones necesitamos evaluar como son estas

A->abCD|EfGhI|&

Lo primero a notar es que tenemos realmente 1 o más producciones por símbolo generador, por lo que lo primero que requerimos es de un arreglo que nos guarde cada producción. Ahora bien, cada producción no es mas que un arreglo de Símbolos, por lo que ahora lo tenemos definido, que las producciones son arreglos de arreglos de símbolos.

1. **Rectificación de símbolos generadores**

El programa leerá línea por línea del archivo y generará un símbolo generador con la primera letra, para después guardar su siguiente contenido como sus producciones, solo que todo esto requiere de cuidado, ya que podemos tener que el símbolo generador ya haya sido anotado anteriormente.

Lo que se hace es básicamente iterar por cada símbolo que se tiene registro, en todas las producciones anteriores, si se encuentra nuestro nuevo símbolo generador quiere decir que ya antes se había definido como un generador y, por ende, indicaremos que ese generador que teníamos es efectivamente el nuevo que estamos creando.

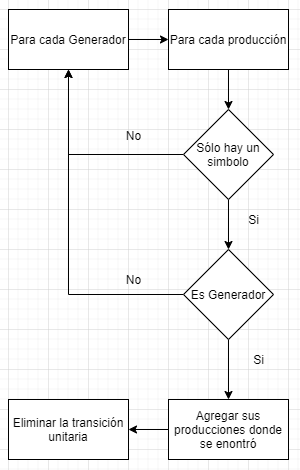
Al llegar a esta decisión podemos también encontrar al generador después de ya haber sido definido, en ese caso simplemente indicamos que es ese generador ya definido.

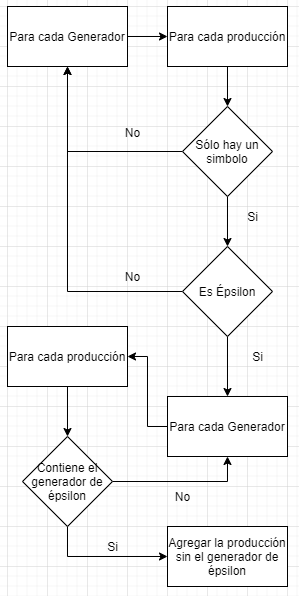
Este procedimiento se hace para que cada símbolo sea exactamente el que decimos, y si nos referimos a un generador en la producción podamos acceder a sus producciones.

1. **Tratamiento de la Gramática**

Nuestra gramática es leída y almacenada, considerando todos los símbolos generadores como ellos mismos dentro de las producciones para que podamos acceder a sus producciones desde cualquier instante que accedamos al símbolo.

El siguiente paso es transformar nuestra gramática a la forma normal de Chomsky para que podamos realizar el algoritmo CYK. Pero previo a esto debemos hacer que la gramática sea apta para poder transformarla a la forma normal de Chomsky, esto requiere eliminar las épsilon transiciones, las unitarias y las inútiles.

Para las transiciones unitarias, requerimos iterar en cada producción de nuestra gramática y comprobar 2 cosas: que la longitud de la producción sea 1, y que el símbolo que este tiene sea un símbolo generador.

Si ambas cosas se cumplen, quiere decir que tenemos una transición unitaria, la cual para sustituir requerimos insertar todas sus producciones en el generador donde la encontramos y, después, eliminar esa producción unitaria.

Una vez que hicimos este procedimiento nos será más fácil hacer el siguiente, que consiste en eliminar las épsilon transiciones de la gramática.

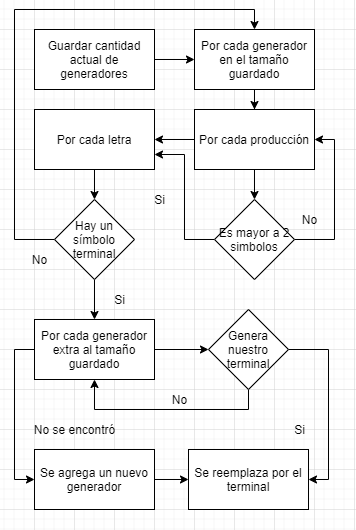
Para las épsilon producciones iteraremos nuevamente en cada producción y compararemos su tamaño a 1, ahora si esta es épsilon, quiere decir que este generador tiene una épsilon transición, por lo que hay que iterar nuevamente en cada producción hasta encontrarlo y agregar ahora las producciones sin el generador de la épsilon transición. También si no hay otra producción en ese generador se elimina la producción que lo contiene.

Finalmente, se eliminan las producciones que después de estos procedimientos no tengan ahora uso, es decir, sean inútiles, esto se hace simplemente iterando en cada símbolo generador por cada símbolo de cada producción, y si no se encuentra nunca este símbolo quiere decir que es inútil, y por ende lo podemos eliminar.

1. **Forma Normal de Chomsky**

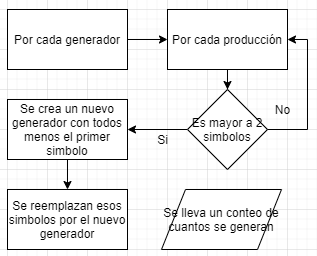
Con estos procedimientos podemos ahora proceder a convertir la gramática en la forma normal de Chomsky, lo cual requiere de 2 pasos:

* Por cada terminal que no este solo, le asignamos un generador único y reemplazamos sus ocurrencias por el generador
* Por cada cadena de generadores mas larga de 2 símbolos, se crea un nuevo generador que genere todos menos el primer símbolo, y los reemplaza en la producción dejándola con longitud de 2

Para el primer punto se debe iterar en cada generador, recordando la cantidad actual, ya que se generarán mas que no debemos considerar, en estos por cada producción si se encuentra un símbolo terminal en cadenas de más de dos símbolos, debemos buscar en la lista de nuevos símbolos generadores si ya hay uno que genere este terminal.

Si se encuentra uno, se reemplaza, si no quiere decir que debemos crear uno, entonces se añade este generador nuevo a la lista y se reemplaza el terminal por este nuevo generador.

Con esto al final tendremos ya terminales solos o cadenas de 2 o mas generadores, por lo que ahora debemos proceder a reducir estas cadenas a solamente dos símbolos generadores.

Ahora debemos iterar nuevamente en cada producción, si una es mayor a 2 símbolos debemos reemplazar todos menos el primer símbolo por un nuevo generador que produzca todos estos generadores. Este proceso incluye a los nuevos que se generen para poder también estos reducirlos si se requiere.

Se llamarán Tx todos los símbolos generadores que generen un terminal, y G# todos los extras que hayan reducido un tamaño mayor a 2.

Con esto ahora tenemos una gramática en la forma normal de Chomsky lista para utilizarla en el algoritmo CYK, para el cual requerimos de una palabra.

1. **La Clase CYK**

Para la elaboración del algoritmo CYK requerimos nuevamente una clase, esta deberá tener los siguientes atributos:

* La gramática en la forma normal de Chomsky
* La palabra por evaluar
* El resultado de si la palabra es válida en la gramática

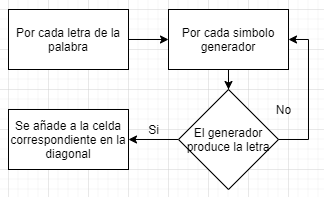
Estos atributos son fáciles de definir, ya tenemos la clase FNCh que será la gramática, la palabra será un arreglo de caracteres para poder acceder a ellos con mayor facilidad, y el resultado es un booleano que se definirá al terminar de realizar el algoritmo CYK en la palabra.

1. **El algoritmo CYK**

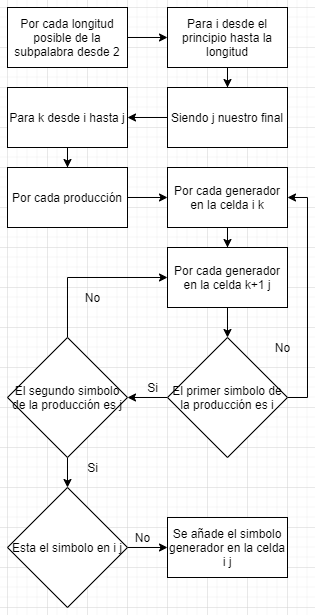
Para comenzar con el algoritmo requerimos de una matriz en la que vamos a trabajar, en esta se almacenarán los datos resultantes a las iteraciones realizadas. Esta matriz contendrá en cada celda una lista de Símbolos, debido a que puede haber diversos símbolos generadores que nos den dichas producciones.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |  |
|  |  |  |  |  | 1 |
|  |  |  |  |  | 2 |
|  |  |  |  |  | 3 |
|  |  |  |  |  | 4 |
|  |  |  |  |  | 5 |

Con esta tabla trabajaremos el algoritmo CYK, el primer paso consiste en obtener en la diagonal los generadores de los símbolos terminales de la palabra.

Este paso es sencillo, por cada letra de la palabra se evalúa cada símbolo generador, si este generador produce la letra, se añade a la lista de su respectiva celda, donde la celda respectiva es la celda i i siendo i el índice de la letra en la palabra, es por esto que se maneja como cadena de caracteres en lugar de un String.

Ahora procedemos a la parte de programación dinámica, para la cual el algoritmo CYK se basará. Por lo que procederemos a explicar su procedimiento.

Primero requerimos definir las longitudes que evaluaremos, van desde 2 hasta n, en cada longitud iteramos para i desde el principio hasta la longitud teniendo j como el tope, en este itera k que va desde i hasta j.

Con esto tenemos también iteración en cada producción de la gramática, y a la par iteramos en la lista de las celdas generadas por nuestras i j y k, donde i k es la celda a evaluar con el primer símbolo de la producción y k+1 j con el segundo. Si el actual generador de la lista de i k es igual al primer generador de la producción actual, y el actual generador de la lista de k+1 j es igual al segundo generador de la producción actual quiere decir que esa producción genera esa parte de la palabra, el siguiente paso es comprobar si este ya fue agregado antes a la lista de la celda i j, si no lo fue la agregamos.

Al final llegamos a comparar la celda 1 n, si está el símbolo generador inicial en la lista de esta celda, la palabra es aceptada por esta gramática, si no está, la palabra no es aceptada.

1. **El árbol de derivación**

El ultimo requerimiento para este proyecto es el mostrar el árbol de derivación de la palabra si esta es válida, por lo que se requiere una implementación del mismo que, originalmente se soluciona con back tracking, pero si desde el principio del algoritmo se realiza una pequeña modificación, podemos obtener este árbol desde el principio, esta modificación, o más bien, adición, es tener otra matriz donde en lugar de guardar la lista de símbolos generadores, guardaremos una clase Tree, la cual tiene

* El símbolo generador actual
* El árbol hijo izquierdo
* El árbol hijo derecho

El símbolo generador actual es básicamente el símbolo que guardaremos en la otra matriz, y los hijos izquierdo y derecho refieren a los arboles de la celda izquierda y derecha que se utilizaron para obtener el símbolo de la celda actual.

Así, la matriz no solo guarda l símbolo que genera esa sub-palabra, sino también que producción se utilizó, con esto podemos obtener el árbol de derivación.

1. **El final del algoritmo**

Al final, el algoritmo comprueba, como se mencionó al final del capítulo VIII, si la palabra es aceptada por la gramática o no, esto lo hace buscando en 1 n si está el símbolo generador, si no se encuentra, imprime que no es aceptada la palabra, pero si sí esta, imprime el árbol de derivación, accediendo a ese mismo índice del arreglo en i n, pero de la matriz de los árboles.

La impresión del árbol es recursiva, donde conserva la cantidad de recursividad con un parámetro de tabulación el cual hace que se identen los hijos para mejor visualización.

Con esto termina el algoritmo, el cual, recibiendo un archivo con una gramática y una cadena, te dice si la cadena es aceptada por la gramática y, si lo es, te muestra el árbol de derivación de esa cadena.

<https://youtu.be/VRKo2SFkE1M>