INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Práctica 4 - Lenguaje libre de contexto

Unidad de aprendizaje: Teoría computacional

Grupo: 2CM4

Alumno(a):
Nicolás Sayago Abigail

Profesor(a):
Sanchez García Luz María

12 de Abril de 2018

Índice

1. Introducción

En está práctica se mostrará como se implemento un programa que reconoce un **Lenguaje libre de contexto** predeterminado.

Gramáticas Libres de contexto

Estás gramáticas son llamadas también ”Gramática en la forma de Backus-Naur(BNF)üsado para describir lenguajes de programación. Las grámaticas libres de contesto se usan para inferir si ciertas cadenas están en el lenguaje expresado por la grámática. Hay dos tipos de inferencia:

- Inferencia recursivo (cuerpo a cabeza/de cadenas a variables).
- Derivación(cabeza a cuerpo, expansión de producciones).

Árboles de derivación

Son una forma de representar derivaciones, se utilizan en la construcción de compiladores, sólo se pueden utilizar en gramáticas tipo 1,2 y 3. Consta de:

- Axioma — Raíz del árbol
- Terminales — Hojas del árbol
- No terminales — Nodos intermedios
- Derivaciones — Sucesores del símbolo no terminal de la izquierda de producciones.

Ambig$\ddot{u}$edad

Existe más de una forma de generar las palabras desde el axioma. Hay varios niveles:

- Sentencia: Si tiene más de una derivación.
- Si tiene como mínimo una sentencia ambigua.
- Lenguaje: Si existe una gramática ambigua que lo genera.
- Lenguaje inherente ambig$\ddot{u}$o: Si todas la gramáticas son ambiguas.

No debe haber ambig$\ddot{u}$edad para que el análisis sea determinista.

2. Planteamiento del problema

Primero elegí una gramática:

$$S \rightarrow aSb \mid \varepsilon$$

Observo que algunas de las cadenas que esa gramática acepta son:

$$\varepsilon$$

$$ab$$

$$aabb$$

$$aaabbb$$

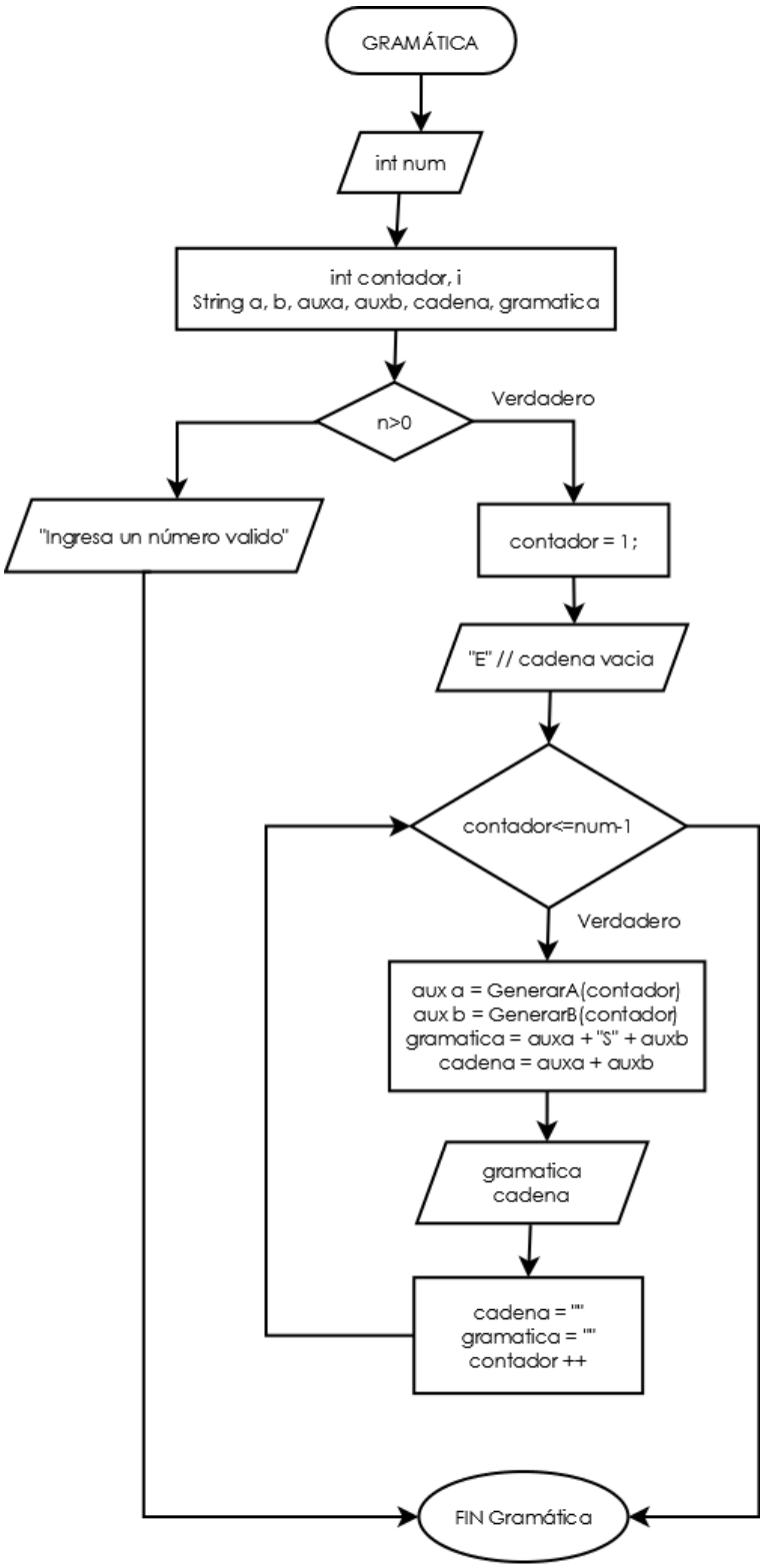
Con eso me percate que el número de **a** y **b** que se generan son iguales, y que van de 1 a n.

Propuesta de solución

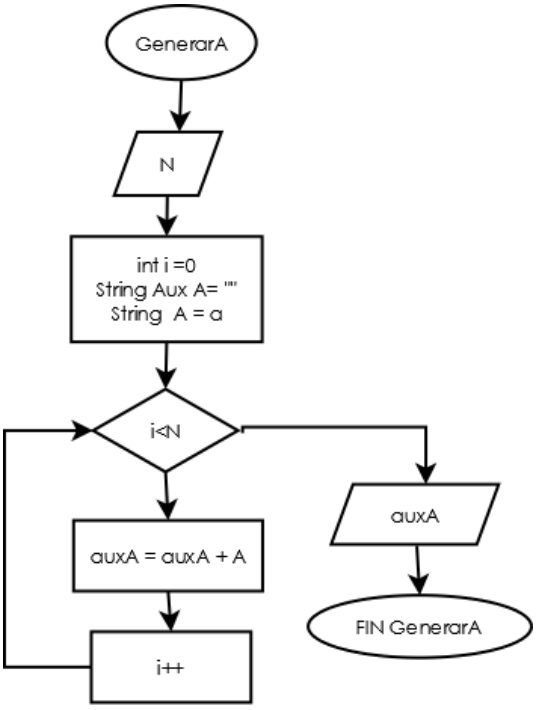
Después de que me percate de lo anterior, entonces pensé en crear un método que generara una cadena de **n** a y otra de **n** b. Y finalmente concatenara ambas cadenas. Dependiendo del número de cadenas que el usuario ingrese, se irán generando e imprimiendo.

3. Diseño de la solución

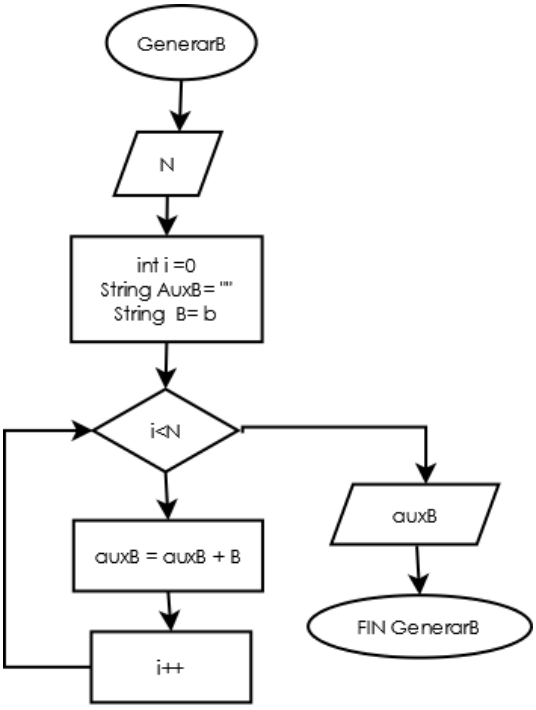
GRAMÁTICA



GenerarA



GenerarB



4. Implementación de la solución

De forma general el programa esta estructurado de tal forma que se introduce una cadena, al dar click al boton **OK** el programa empieza a validar cada uno de los estados.

A continuación se muestra cada parte fundamental para que el programa funcione.

IMPLEMENTACIÓN DE GRAMÁTICA

```
public void Gramatica()
{
 int num = Integer.parseInt(Cadena.getText());
 int contador, i;
 String a, b;
 String auxa, auxb, cadena, gramatica;
 a = "a"; auxa = "";
 b = "b"; auxb = "";
 if(num > 0)
 {
  contador = 1;
  AreaSalida.setText("E");
  while (contador<=num-1) // Imprime el n mero de cadenas que se eligen
  {
   // Se crea una cadena que contenga s a
   auxa = GenerarA(contador);
   // Se crea una cadena que contenta s b
   auxb = GenerarB(contador);
   // Cadena que muestra COMO funciona
   gramatica = auxa + "S" + auxb;
   // Cadena que muestra el resultado
   cadena = auxa + auxb;
   AreaSalida.setText(" "+AreaSalida.getText()+ "\n" + gramatica + "    " + cadena);
   cadena = ""; gramatica = "";
   contador++;
  }
 }
 else
  AreaSalida.setText("El numero que ingresaste no es valido");
}
```

IMPLEMENTACIÓN DE GENERAR A

```
public String GenerarA(int N)
{
 int i;
 String auxA = "", A = "a";
 // Concatena el numero de a que se pidan
 for (i=0; i<N; i++)
  auxA = auxA + A;
 return auxA;
}
```

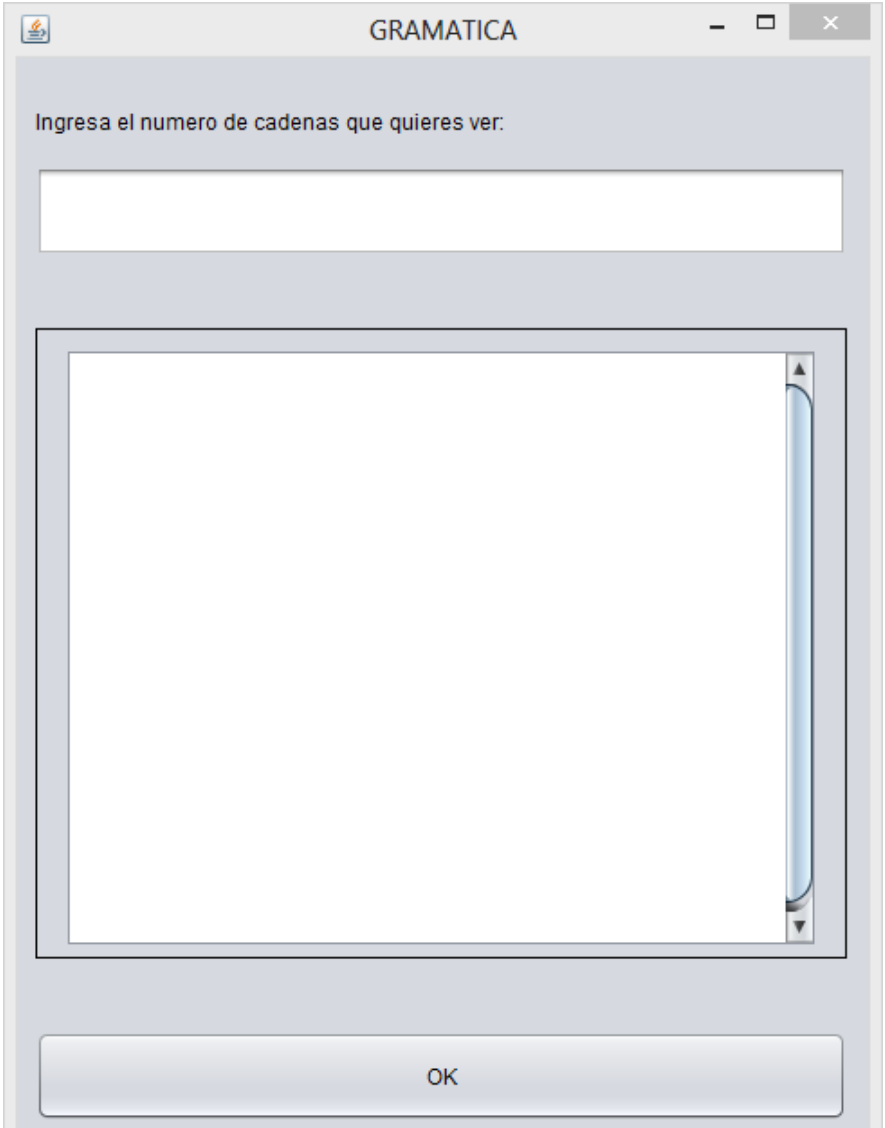
IMPLEMENTACIÓN DE GENERAR B

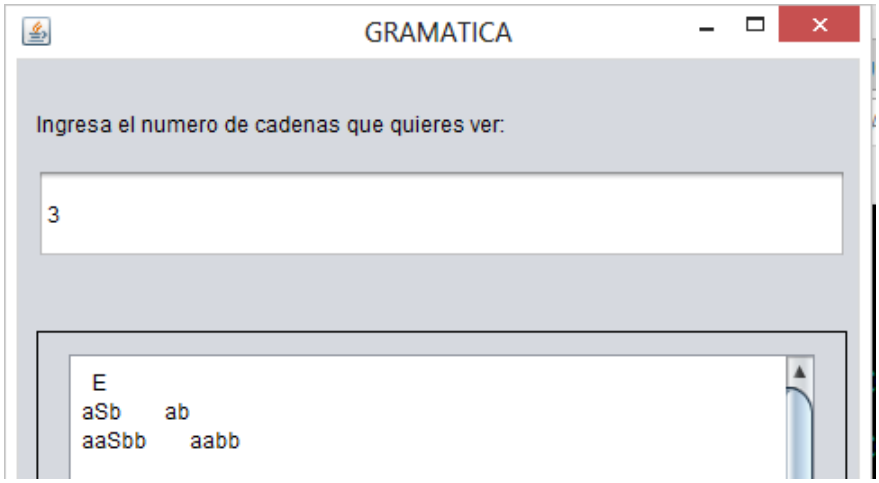
```
1  public String GenerarB(int N)
2  {
3   int i;
4   String auxB = "", B = "b";
5   // Concatena el numero de  b que se pidan
6   for (i=0; i<N; i++)
7    auxB = auxB + B;
8   return auxB;
9  }
```

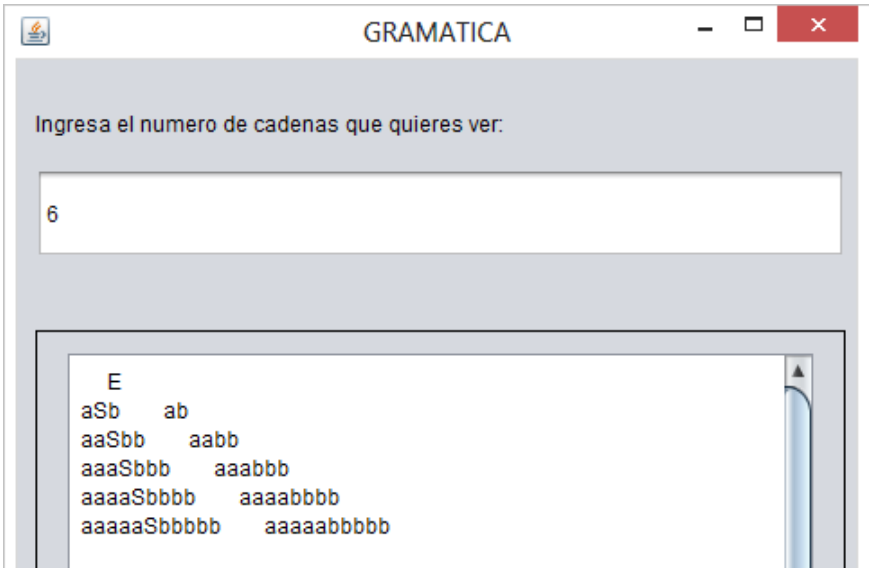
5. Funcionamiento

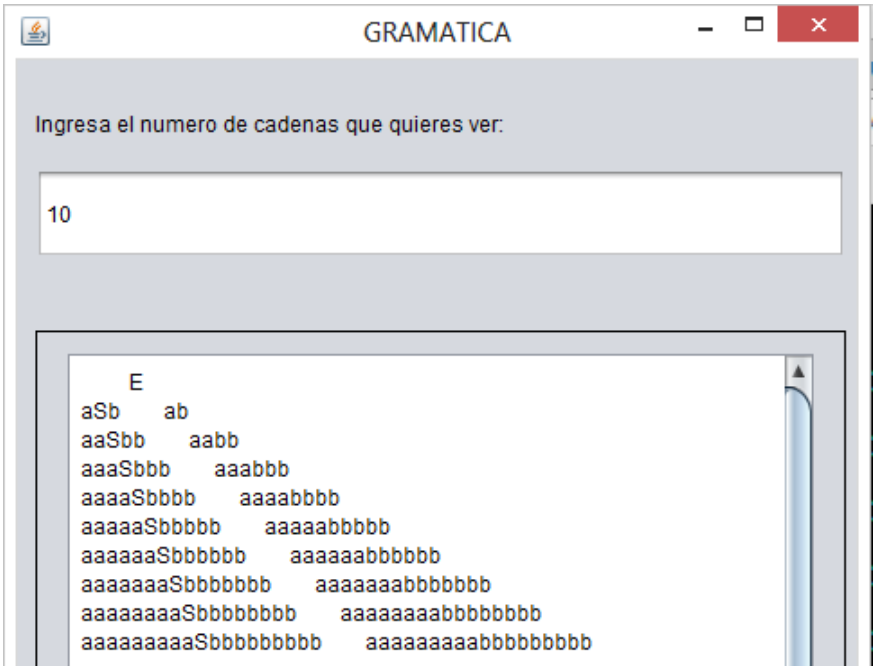
Primero que nada, mostramos la interfaz inicial. Observamos que el usuario tiene la oportunidad de ingresar el número de cadenas que desea que se muestren. Al dar click en el botón OK, se mostrarán las cadenas generadas.

La pantalla inicial es la siguiente:




GRAMATICA
Ingresa el numero de cadenas que quieres ver:
3
E
aSb ab
aaSbb aabb


GRAMATICA
Ingresa el numero de cadenas que quieres ver:
6
E
aSb ab
aaSbb aabb
aaaSbbb aaabbb
aaaaSbbbb aaaabbbb
aaaaaSbbbbb aaaaabbbbb



6. Conclusiones

Al terminar está práctica pude ver que las gramáticas libres de contexto te permiten generar cadenas de forma recursiva y de cierta forma más sencilla que las anteriores, puesto que es muy recursivo.

Además que me parece muy interesante que sean utilizadas para validar las sentencias en los lenguajes de programación.

Referencias

[1] E. A. Martinez, *Alfabetos, símbolos y cadenas*. [Online]. Available: http://eafranco.com/docencia/teoriacomputacional/files/03/Clase$_1$5.*pdf*