INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ZEPEDA FLORES ALEJANDRO DE JESÚS

NO. BOLETA 2016601853

TEORÍA COMPUTACIONAL

PROF. LUZ MARÍA SANCHÉZ GARCÍA

11 DE MAYO DE 2018

INTRODUCCIÓN

GRAMÁTICAS INDEPENDIENTES DEL CONTEXTO

Las gramáticas de tipo 2 o gramáticas independientes del contexto, son las que generan los lenguajes libres o independientes del contexto. Los lenguajes libres del contexto son aquellos que pueden ser reconocidos por un autómata de pila determinístico o no determinístico.

- Son útiles para describir bloques anidados en lenguajes de programación ya que describen su sintaxis.
- Son llamadas así porque el elemento no terminal del lado derecho se puede sustituir sin importar el contexto en que este.
- Su característica es que piden que solamente exista un no terminal del lado izquierdo de la producción.

Si se obtiene una gramática que cumpla estas tres condiciones se puede asegurar que en cada derivación que se realiza se introduce información relevante.

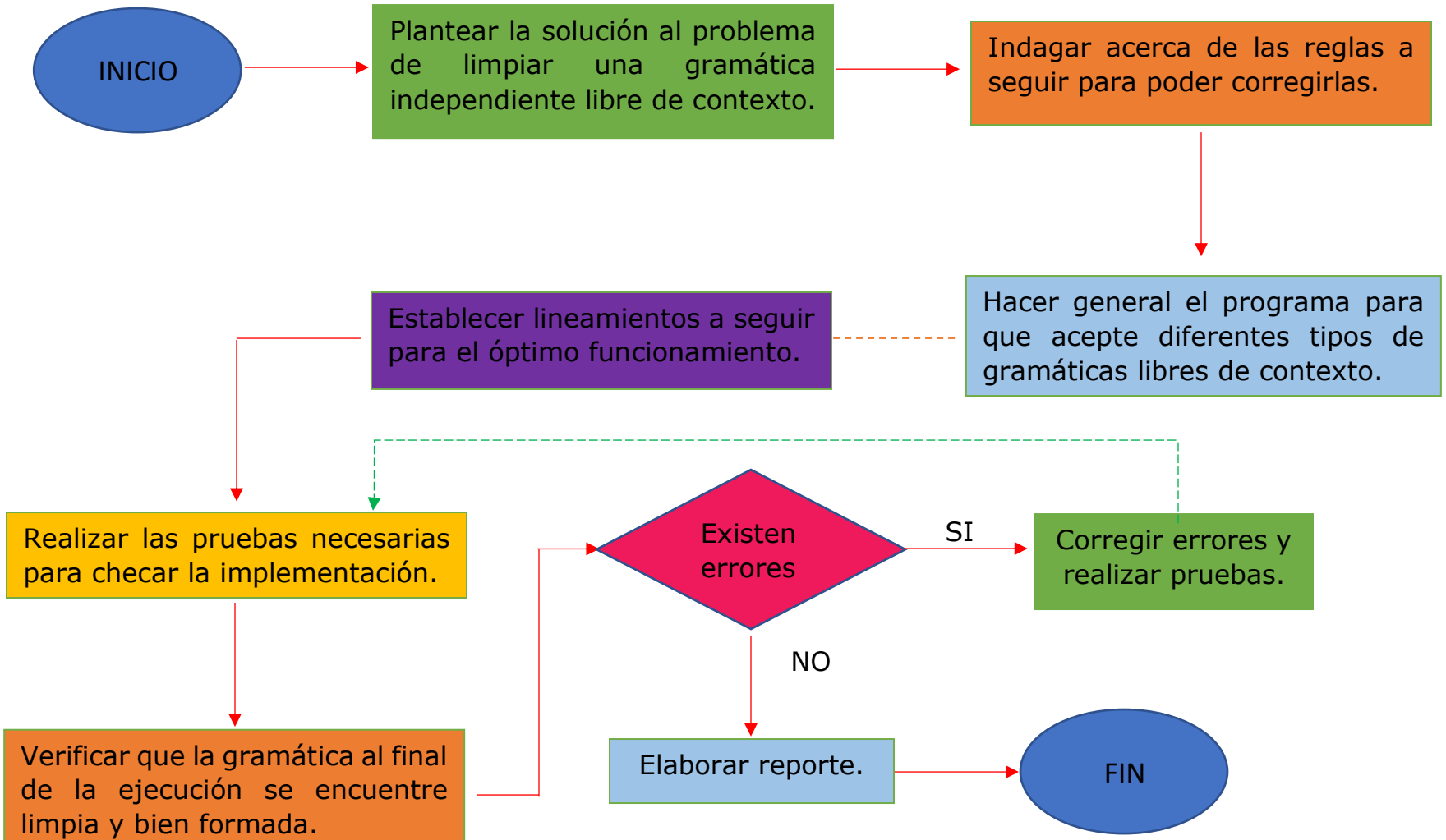
PLANTEAMIENTO DEL PROBLEMA

El programa aceptará una gramática libre de contexto (GLC) con sus reglas de producción leídas desde teclado o desde un archivo y realizará la limpieza de la misma para obtener una GLC limpia y bien formada.

El programa debe limpiar una GLC de producciones:

- muertas (inútiles)
- inaccesibles
- Vacías

DIAGRAMA


INICIO
Plantear la solución al problema de limpiar una gramática independiente libre de contexto.
Indagar acerca de las reglas a seguir para poder corregirlas.
Hacer general el programa para que acepte diferentes tipos de gramáticas libres de contexto.
Establecer lineamientos a seguir para el óptimo funcionamiento.
Realizar las pruebas necesarias para checar la implementación.
Existen errores
SI
Corregir errores y realizar pruebas.
NO
Verificar que la gramática al final de la ejecución se encuentre limpia y bien formada.
Elaborar reporte.
FIN

IMPLEMENTACIÓN DE LA SOLUCIÓN

Este programa si es un poco delicado, en el sentido de que el usuario tiene que seguir ciertas especificaciones para su óptimo rendimiento. A diferencia de las otras prácticas, se programó una función de lineamientos a seguir.

```
void Information(){
     system("cls");
     int opcion;
     cout << "INFORMACION \n "<< endl;
     cout << "SOLO se aceptaran 4 proyecciones (S, A ,B ,C) con el
siguiente formato:\n" << endl;
     cout << "\t" << "S => Aab | B | Csa" << endl;
     cout << "\t" << "A => aA | cb | ABc" << endl;
     cout << "\t" << "B => CG | DC " << endl;
     cout << "\t" << "C => C\n" << endl;
     cout << "En caso de utilizar el operador ""|"" debe haber espacios"
<< endl;
     cout << "Se utilizara "" E "" para denotar EPSILON \n" << endl;
     cout << "Presiona 1 para continuar  -> ";
     cin  >> opcion; system("cls");
     switch(opcion){
          case 1: main(); break;
          default: Information();
     }
}
```

Ahora implementamos una función para leer la gramática del usuario, es importante recordar que se deben seguir las especificaciones previamente mencionadas, de lo contrario, el programa terminra.

```
void InsertGram(){
     system("cls");
     int cant, ind;
     cout << "Gramatica libre de contexto para limpiar"<< endl;
     string nueva;
     vector <vector<string> > prod;
     vector <int> eps;
     prod.push_back( vector <string> () );
     prod.push_back( vector <string> () );
     prod.push_back( vector <string> () );
     prod.push_back( vector <string> () );

// SE PREPARA EL PORGRAMA PARA INGRESAR TODAS LAS PRODUCCIONES DE S
     cout << "\nPROYECCIONES PARA ""S""  " << endl;
```

```
     cout << "Numero de proyeciones :> ";  cin >> cant;
     for(int i =0; i< (cant+(cant-1)); i++){
          cin >> nueva;
          if (nueva != "|" &&  nueva != "E")
               prod[0].push_back(nueva);

          if (nueva == "E")
               prod[0].push_back(nueva);

     }
     cout << "\n\nPROYECCIONES PARA ""A""  " << endl;
     cout << "Numero de proyecciones :> ";  cin >> cant;
     for(int i =0; i< (cant+(cant-1)); i++){
          cin >> nueva;
          if (nueva != "|" &&  nueva != "E")
               prod[1].push_back(nueva);

          if (nueva == "E"){
               eps.push_back(1);
          }
     }
     cout << "\n\nPROYECCIONES PARA ""B""  " << endl;
     cout << "Numero de proyecciones :> ";  cin >> cant;
     for(int i =0; i<(cant+(cant-1)); i++){
          cin >> nueva;
          if (nueva != "|" &&  nueva != "E")
               prod[2].push_back(nueva);
          if (nueva == "E"){
               eps.push_back(2);
          }
     }
     cout << "\n\nPROYECCIONES PARA ""C""  " << endl;
     cout << "Numero de proyecciones :> ";  cin >> cant;
     for(int i =0; i<(cant+(cant-1)); i++){
          cin >> nueva;
          if (nueva != "|" &&  nueva != "E")
               prod[3].push_back(nueva);
          if (nueva == "E"){
               eps.push_back(3);
          }
     }
if(eps.size() != 0)
vacia(prod,eps);
else
     Inaccesible(prod);
}
```

El resto de las funciones de validación y limpieza de gramáticas se ven en el código.

FUNCIONAMIENTO

```
Practica 5 - Alejandro Zepeda Flores

1.-  Lista de especificaciones
2.-  Nueva gramatica libre de contexto
3.-  Salir del programa
Opcion:
```

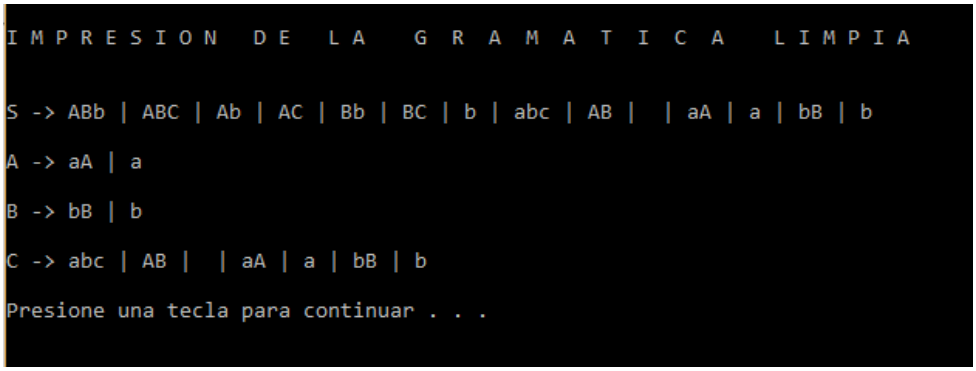
Interfaz de usuario

```
INFORMACION

SOLO se aceptaran 4 proyecciones (S, A ,B ,C) con el siguiente formato:

        S => Aab | B | Csa
        A => aA | cb | ABc
        B => CG | DC
        C => C

En caso de utilizar el operador | debe haber espacios
Se utilizara  E  para denotar EPSILON

Presiona 1 para continuar  ->
```

Lineamientos por seguir

```
Gramatica libre de contexto para limpiar

PROYECCIONES PARA S
Numero de proyeciones :> 2
ABb | ABC


PROYECCIONES PARA A
Numero de proyecciones :> 2
aA | E


PROYECCIONES PARA B
Numero de proyecciones :> 2
bB | E


PROYECCIONES PARA C
Numero de proyecciones :> 2
abc | AB
```

Inserción de gramática libre de contexto



Gramática limpia y bien formada

CONCLUSIÓN

Esta práctica, ha sido hasta el momento, la más complicada y laboriosa de este semestre. Como primer punto, tuve que emplear un arduo trabajo de investigación acerca de como es que se limpia una gramática; nunca encontré un "algoritmo" o alguna serie de pasos a seguir para la limpieza de estas, entonces me base en los apuntes de clase.

Una vez comprendido el tema, tuve que emplear todos los conocimientos previos adquiridos en programación de C y C++ porque el grado de complejidad de la práctica si es alto; tuve que jugar con arreglos y reordenamiento, eliminación y superposición de símbolos para llegar al resultado esperado.

Fue una práctica agradable, ahora sólo me queda tratar de hacerla general, recibir n-cadenas y no especificar al usuario tantos lineamientos, porque el target es facilitar la vida al usuario.

BIBLIOGRAFÍA

- Edgardo Adrián Franco Martínez. (2013). Gramáticas libres de contexto. mayo 11, 2018, de Instituto Politécnico Nacional Sitio web: http://www.eafranco.com/docencia/teoriacomputacional/files/14/Clase_14.pdf
- S/A. (2006). Presentacion sobre gramaticas y lenguajes libres de contexto. mayo 11, 2018, de AlfaOmega Grupo Editor Sitio web: http://libroweb.alfaomega.com.mx/book/685/free/ovas_statics/cap9/Presentacion%20sobre%20gramaticas%20y%20lenguajes%20libres%20de%20contexto..pdf