Teoria de la computacion
Avance del proyecto
Analisis sintatico (Parsing) de GIC

Carlos Merma
Jean Paul Melendez

June 2020

1 Introdución

Son los lenguajes formales que engloban a los lenguajes regulares y constituyen los mecanismos de representación y reconocimiento de los lenguajes de programación desde el punto de vista sintáctico. Por tanto, tienen gran aplicación en la teoría y construcción de intérpretes y compiladores de lenguajes de programación (LP) o de especificación o formato de información, específicamente en el analizador sintáctico o parser que comprueba la corrección de sintaxis en los códigos fuentes de los LP.

2 Definición del problema

Determinar si una cadena es producida por una Gramática Independiente del Contexto (GIC) dada.

3 Estado del arte

Identificación	Objetivo General	Instrumentos recolectados de la información	Resultados
CYK algorithm //	Se efectúa el análisis de la cadena mediante la construcción de una tabla que define el algoritmo CYK	https://fliphtml5.com/ejte/lkvc/basic	
Earley parser	Usan la noción de reparto (de cálculos y de estructuras) y que construyen todos los análisis posibles de una frase	https://pdfs.semanticscholar.org/5b27/7a3f9a9f5250939f334e282d1393971722a9.pdf	
GLR parser	Crear un generador de compilador, basado en analizadores ascendentes.	https://dickgrune.com/Books/PTAPG_1st_Edition/BookBody.pdf ()	
LL parser	La gramática LL es un subconjunto de la gramática libre de contexto pero con algunas restricciones para obtener la versión simplificada, a fin de lograr una implementación fácil.		

4 Codigo de c++ (CYK)

Antes de comenzar debemos hacer un breve definición sobre el algoritmo CYK o "Algoritmo de llenado de tabla", el algoritmo construye una tabla que indica si w pertenece a un lenguaje L en un tiempo de O(n3)

Programar en C++ el algoritmo CYK y mostrar el arbol generico de un ejemplo.
S–>AB|BC
A–>BA|a
B–>CC|b

C->AB|a
cadena: aaa

---

tipos.h

```
#ifndef TEORIA_TIPOS_H
#define TEORIA_TIPOS_H

#include<iostream>
#include<string>
#include<iomanip>
#include <queue>
using namespace std;

#endif //TEORIA_TIPOS_H
```

---

funciones.h

```
#ifndef TEORIA_FUNCIONES_H
#define TEORIA_FUNCIONES_H

#include "tipos.h"
#include "arbol/arbol.h"

void prin_level_by_level(Node* root);


#endif //TEORIA_FUNCIONES_H
```

---

funciones.cpp

```
#include "funciones.h"

void prin_level_by_level(Node* root){
    if(root== nullptr)return;
    queue<Node*> q;
    q.push(root);
    q.push(nullptr);
    while (q.size()>1){
        Node* curr = q.front();
        q.pop();
        if(curr== nullptr){
            cout << endl;
            q.push(nullptr);
            continue;
```

```
        }
        cout<< curr->key << " ";
        if(curr->left!=nullptr) q.push(curr->left);
        if(curr->right!=nullptr) q.push(curr->right);
    }
    cout << endl;
}
```

---

ejercicio.h

```
#ifndef TEORIA_EJERCICIO_H
#define TEORIA_EJERCICIO_H

void dibujar_arbol();

#endif //TEORIA_EJERCICIO_H
```

---

ejercicio.cpp

```
#include "ejercicio.h"
#include "funciones.h"

void dibujar_arbol(){
    Node* root = new Node("            S");
    root->left = new Node("        A");
    root->left->left = new Node("    a");
    root->right = new Node("         B");
    root->right->left = new Node("           C");
    root->right->left->left = new Node("                a");
    root->right->right = new Node("     C");
    root->right->right->right = new Node("        a");
    cout << " rbol   generico " << endl;
    prin_level_by_level(root);
    cout << endl;
}
```

Carpeta la carpeta arbol y dentro de la carpeta crear arbol.h

arbol.h

```
#ifndef TEORIA_ARBOL_H
#define TEORIA_ARBOL_H

struct Node{
    string key;
    Node* left;
    Node* right;
```

```
    Node(string k){
        key = k;
        left = right = nullptr;
    }
};

#endif //TEORIA_ARBOL_H
```

---

main.cpp

```
#include "tipos.h"
#include "ejercicio.h"
#define MAX 100
#define for(i,a,b) for(i=a;i<b; i++)

string gram[MAX][MAX]; //Para almacenar la gram tica ingresada
string dpr[MAX];
int p,np;      //np-> n mero de producciones

inline string concat( string a, string b) //concatena no terminales
  nicos
    int i;
    string r = a;
    for(i, 0, b.length()) {
    if (r.find(b[i]) > r.length()) {
        r += b[i];
    }
    }
    return r;
}
//separa el lado derecho de la gram tica ingresada
inline void break_gram(string a){
    int i;
    p=0;
    while(a.length()){
        i=a.find("|");
        if(i>a.length()){
            dpr[p++] = a;
            a="";
        }
        else{
            dpr[p++] = a.substr(0,i);
            a=a.substr(i+1,a.length());
        }
    }
}
```

```
//comprobar si el LHS de la gram tica ingresada est  en FNC
inline int lchomsky(string a){
   if(a.length()==1 && a[0]>='A' && a[0]<='Z')
       return 1;
   return 0;
}
  //comprobar si RHS de gram tica est  en FNC
inline int rchomsky(string a){
   if (a.length() == 1 && a[0]>='a' && a[0] <='z')
       return 1;
   if (a.length()==2 && a[0]>='A' && a[0]<='Z' && a[1]>='A' && a[1]<='Z' )
       return 1;
   return 0;
}
//devuelve una cadena concatenada de variables que pueden producir la cadena p
inline string search_prod(string p){
   int j,k;
   string r="";
   for(j,0,np){
       k=1;
       while(gram[j][k] != ""){
           if(gram[j][k] == p){
               r=concat(r,gram[j][0]);
           }
           k++;
       }
   }
   return r;
}
//crea todas las combinaciones de variables de a y b.
//Por ejemplo: BA * AB = {BA, BB, AA, BB}
inline string gen_comb(string a, string b){
   int i,j;
   string pri=a,re="";
   for(i,0,a.length()) {
       for(j, 0, b.length()) {
           pri = "";
           pri = pri + a[i] + b[j];
           re = re + search_prod(pri);
            //busca si las producciones generadas pueden ser creadas o no
       }
   }
   return re;
}
```

```
int main(){
   int i, pt, j, l, k;
   string a, str, r, pr, start;
   cout << endl;
   cout << "Ingrese la variable de inicio: ";
   cin >> start;
   cout << endl;
   cout << "Numero de producciones: ";
   cin >> np;
   for(i, 0, np){
       cin >> a;
       pt = a.find("->");
       gram[i][0] = a.substr(0,pt);
       if (lchomsky(gram[i][0]) == 0){
           cout << endl;
           cout << "La gram tica no est  en forma de Chomsky";
           abort();
       }
       a = a.substr(pt+2, a.length());
       break_gram(a);
       for(j, 0 ,p){
           gram[i][j+1]=dpr[j];
           if (rchomsky(dpr[j]) == 0){
               cout << endl;
               cout << "La gram tica no est  en forma de Chomsky";
               abort();
           }
       }
   }
   string matrix[MAX][MAX],st;
   cout << endl;
   cout << "Ingrese la cadena a generar: ";
   cin >> str;
   for(i, 0, str.length()){
   //Asigna valores a la diagonal principal de la matriz
       r = "";
       st = "";
       st += str[i];
       for(j, 0, np){
           k = 1;
           while(gram[j][k] != ""){
               if(gram[j][k] == st){
                   r = concat(r,gram[j][0]);
               }
               k++;
           }
```

```
        }
        matrix[i][i] = r;
    }

    for(k, 1, str.length()){
    //Asigna valores a la mitad superior de la matriz.
        for(j, k, str.length()){
            r = "";
            for(l, j-k ,j){
                pr = gen_comb(matrix[j-k][l],matrix[l+1][j]);
                r = concat(r,pr);
            }
            matrix[j-k][j] = r;
        }
    }

    for(i, 0, str.length()){
    //Imprime la matriz
        k = 0;
        l = str.length()-i-1;
        for(j, l, str.length()){
            cout << "|" << setw(5)<< matrix[k++][j] << "|";
        }
        cout << endl;
    }

    for(i, 0, start.length()) {
        if (matrix[0][str.length() - 1].find(start[i]) <= matrix[0][str.length()
        // si el ultimo elemento de la primera //fila contiene una variable
         //de inicio
            cout << "Cadena generado. " << endl;
            cout << endl;
            dibujar_arbol();
            return 0;
        }
    }
    cout <<"Gramatica no generada";

    return 0;
}
```

```
Ingrese la variable de inicio:S

Numero de producciones:4
S->AB|BC
A->BA|a
B->CC|b
C->AB|a

Ingrese la cadena a generar:aaa
 |  SCA|
|    B||    B|
|   AC||   AC||   AC|
Cadena generado.

Arbol generico
           S
       A         B
   a          C     C
              a        a


Process finished with exit code 0
```

5 Bibliografia

$https://ddd.uab.cat/pub/trerecpro/2009/hdl_2072_43845/PFC_LaiaFelipMolina.pdf...pag(16)$

$https://mascvuex.unex.es/ebooks/sites/mascvuex.unex.es.mascvuex.ebooks/files/files/file/TeoriaAutomatas_9788469163450.pdf..pag(15)$

$https://core.ac.uk/download/pdf/44311153.pdf..pag(24)$