UNIVERSIDAD SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1

MANUAL USUARIO

CARLOS AUGUSTO HERNÁNDEZ ORDOÑEZ
201611269
21-02-2020

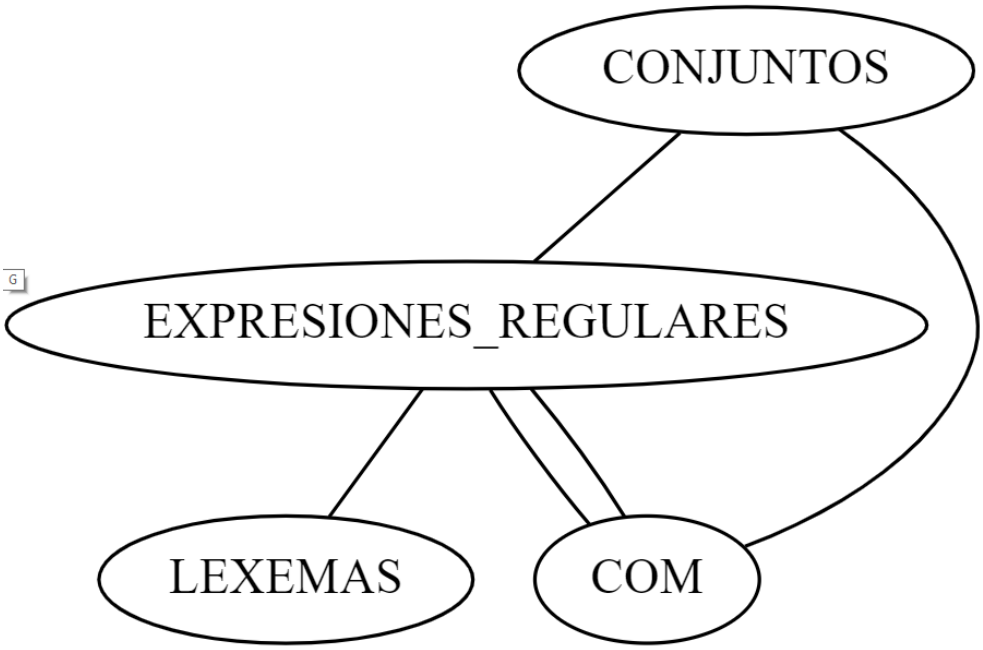
INTRODUCCIÓN

El curso de Organización de Lenguajes y Compiladores, necesita que los alumnos aprendan de forma más formal y específica, sobre los tipos de lenguajes y entrando de lleno a muchas de sus fases, desde análisis léxico, hasta generador de código intermedio, por lo cual se necesita que se aprendan muchos conocimientos, como técnicas para aplicar a la hora de crear su propio compilador con diferentes tipos de tareas en especial como saber los diferentes tipos de lenguajes a desarrollar.

FLUJO DEL PROGRAMA

Para encontrar los diferentes elementos del dato de entrada se hicieron trabajos por “fases” unas continuas, de otras, con sus excepciones, donde también esperábamos información irrelevante para evitar algunos conflictos

Básicamente existían 4 fases del análisis léxico:



Donde Los **conjuntos** Son los elementos que agrupan a todos los caracteres, bajo el siguiente algoritmo:

```
//AQUI VAMOS A DAR LOS ELEMENTOS DE UN CONJUNTO, ESTO SERVIRA PARA LA EXPRE
String contenido="[";
try {
    if(ListaElementos.size()>2){
    //EN EL CASO DE QUE LA LISTA SEA MAYOR A 2 VAMOS ENVIAR TODA LA INFO AS
    for(int i=0;i<ListaElementos.size();i++){
        if(i==ListaElementos.size()-1) contenido+=ListaElementos.get(i);
        else contenido+=ListaElementos.get(i)+",";
    }
}else{

    //SI EL TAMA;O ES IGUAL A 2 ENTONCES DEBEMOS AGREGAR ELEMENTO POR ELEM
    String primero=ListaElementos.get(0);
    String ultimo=ListaElementos.get(1);

    //EN EL CASO DE QUE SEA UN GRUPO DE CARACTERES, LOS AGRUPAREMOS

    //EN ESTE CASO CUANDO SON DE TAMA;O MAYOR O IGUAL A 2 QUIERE DECIR QUE
    if(primero.length()>=2 && ultimo.length()>=2){
        int primI=Integer.parseInt(primero);
        int ultI=Integer.parseInt(ultimo);

        for(int i=primI;i<=ultI;i++){
            if(i==ultI)contenido+=i;
            else contenido+=i+",";
        }
```

Aquí vamos a dar los elementos de un conjunto, esto servirá para la expresión regular, lo cual nos pide esos datos

En el caso de que la lista sea mayor a 2 vamos enviar toda la info así a DIRECTAMENTE

Si el tamaño es igual a 2 entonces debemos agregar elemento por elemento

En el caso de que sea un grupo de caracteres, los agruparemos

En este caso cuando son de tamaño mayor o igual a 2 quiere decir que son números. Entonces agregamos una lista de estso números a la cadena

En el caso de que no sean numero, pues agregamos el rango de datos seleccionados

Digamos que de a-z debemos convertirlos a char, luego a int

Por ultimo hacemos un for de tal a tal caracter, entonces luego los volvemos a convertir a char y lo agregamos

Cuando no hay más elementos, volvemos a los que son **EXPRESIONES REGULARES,** las cuales se agregan y desupues se hace un análisis de los posibles grupos de conjuntos, que viene con ello, como por ejemplo el siguiente algoritmo reemplazamos:

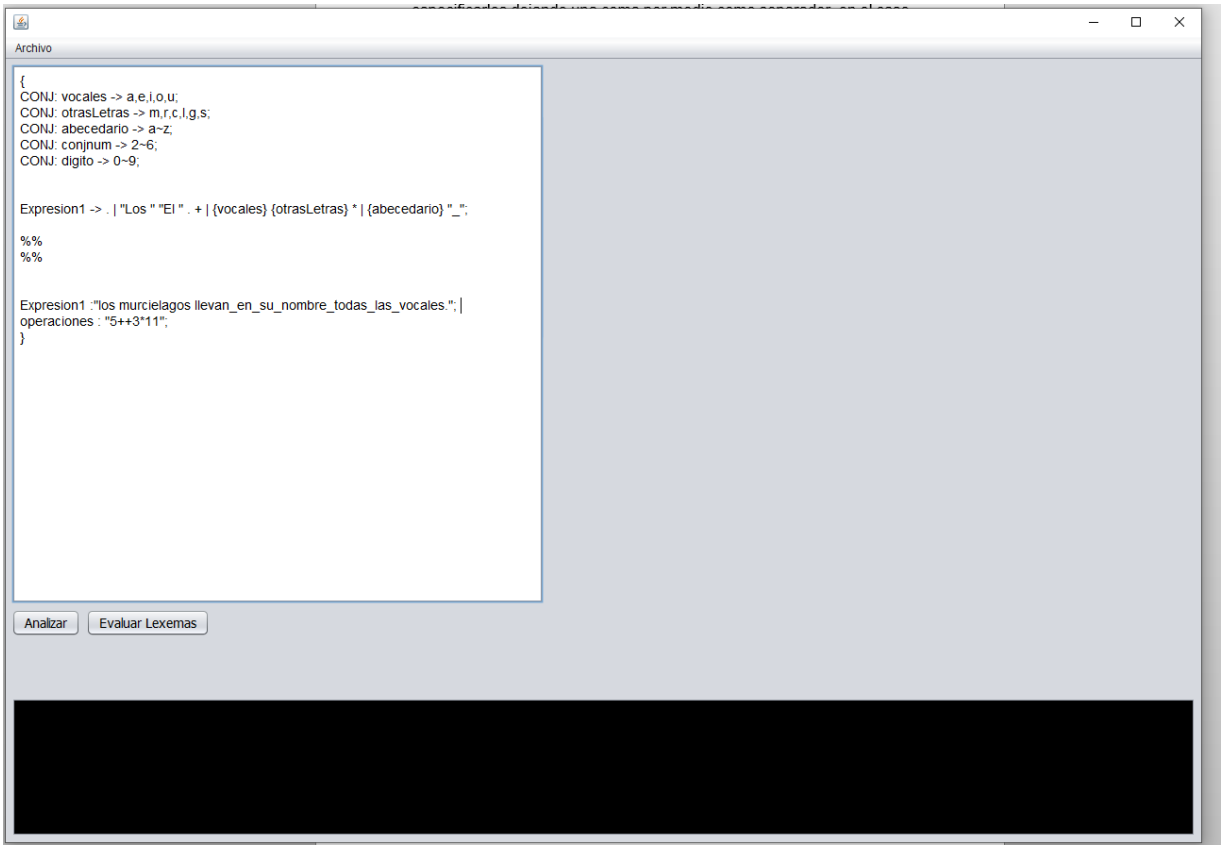
```
//CUANDO LLAMAMOS A UNA EXPRESION REGULAR LO QUE HACEMOS ES:
for(conjunto cn : listaCon){

    //RECORRER CADA CONJUNTO ENVIADO Y VERIFICAR SI HAY ALGUNA COINCIDENCIA
    int index = contenido.indexOf(cn.dameNombre());
    if(index!=-1){


        String contendioCN=cn.dameContenido();
        //System.out.println("ENTRE EN LA CONDICION "+contendioCN);
        contenido=contenido.replace("{"+cn.dameNombre()+"}", contendioCN);
    }
}
```

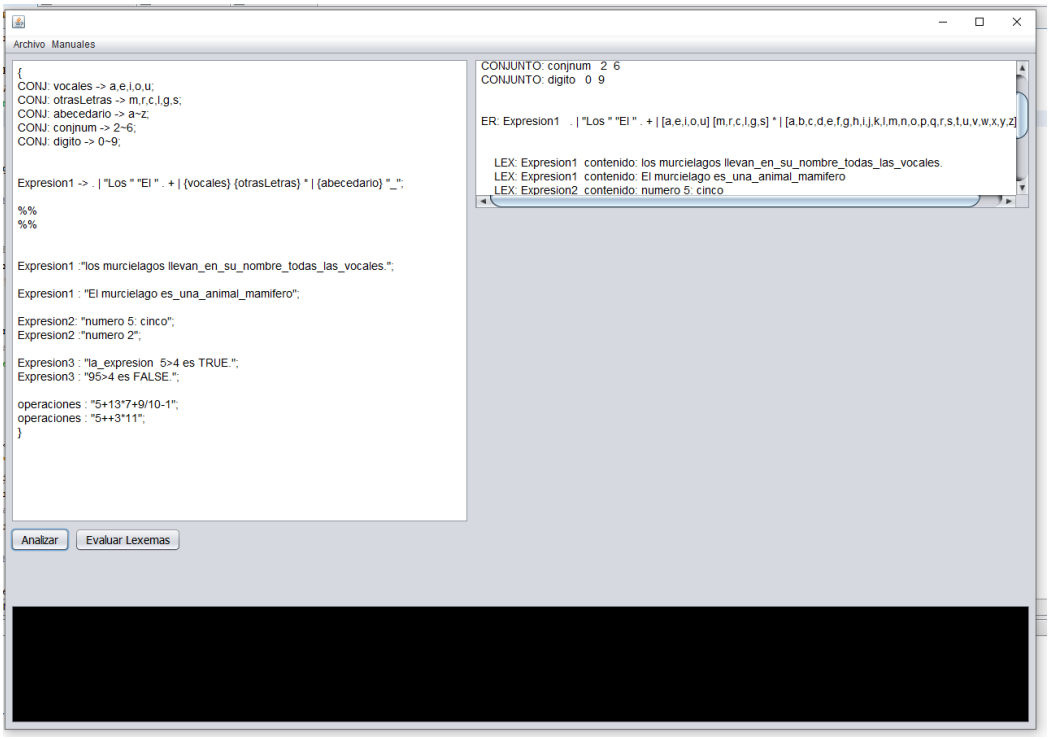
Nos basamos en el ID que se le dio al conjunto previamente para encontrar una coincidencia, si esta existe, reemplazamos los datos.

Por ultimo los dato de los lexemas, básicamente estamos en diferentes puntos, muy sencillos para lograr una equidad, como tal el algoritmo:

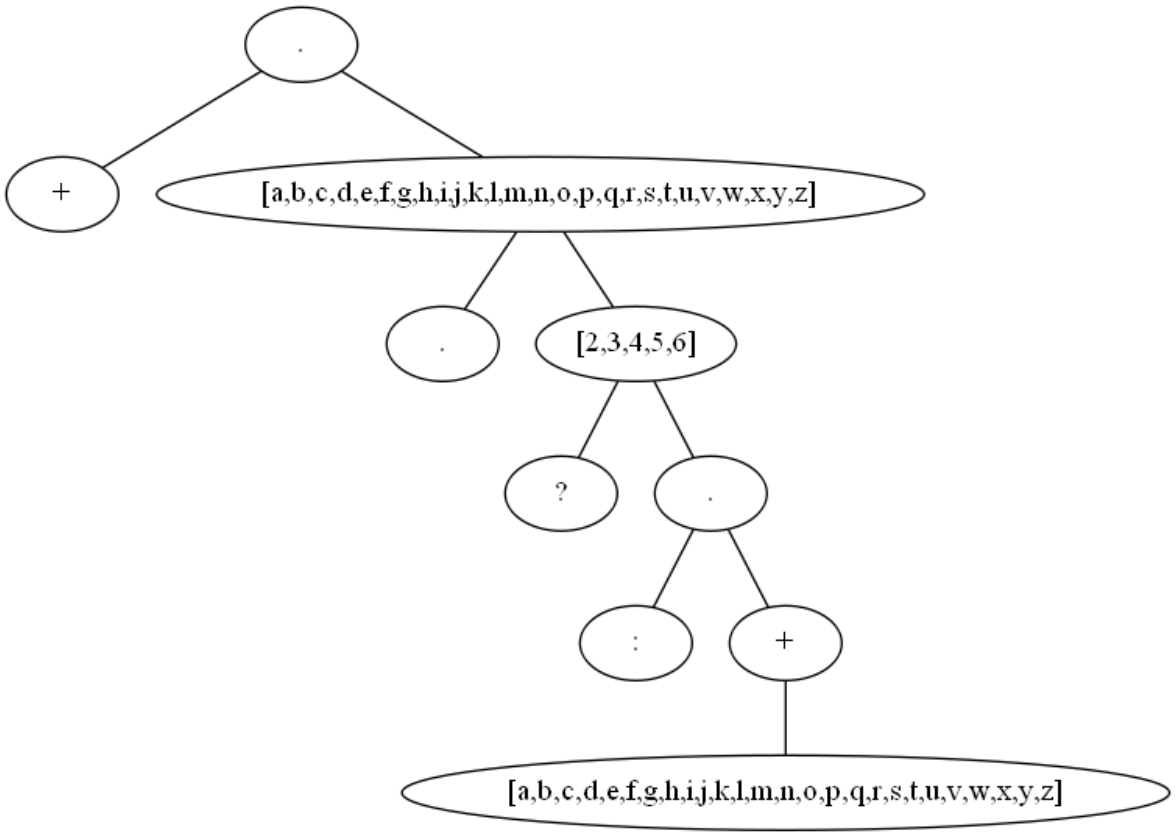
```
switch(EstadoLEX){
                        case 0:
                            if(actual=='%')contadorPorcentaje++;
                            if(contadorPorcentaje>=4            &&
actual=='\n'){
                                EstadoLEX++;
                                contadorPorcentaje=0;
                            }
                            break;

                        case 1:
                            //AQUI  VAMOS  RELLENAR  EL  NOMBRE  Y
ESPERAR OTRO TOQUEN
                            if(actual==':') EstadoLEX++;
                            else if(actual!='\n')nombreLEX+=actual;

                        case 2:
                            //EN ESTE CASO INICIAREMOS AGREGANDO LOS
LEXEMAS A UN STRING Y LUEGO A SU RESPECTIVO CONJUNTO
                            if(actual=='"')EstadoLEX++;
                            break;
                        case 3:
                            if(actual=='"'){
                                //AQUI VAMOS A CREAR EL OBJETO Y LO
VAMOS A MANDAR A SU RESPECTIVA LISTA
                                //nombreLEX=nombreLEX.replaceAll("
", "");

//nombreLEX=nombreLEX.replaceAll("\n", "");

nombreLEX=nombreLEX.replaceAll("\t", "");
                                nombreLEX=nombreLEX.replaceAll(" ",
"");

nombreLEX=nombreLEX.replaceAll("/", "");

nombreLEX=nombreLEX.replaceAll("<", "");
```

```
                                lexema lexi = new lexema(nombreLEX,
contenidoLEX);
                                listaLEX.add(lexi);
                                nombreLEX=contenidoLEX="";
                                EstadoLEX=4;

                            }else contenidoLEX+=actual;
                            break;


                        case 4:
                            if(actual=='}'                          ||
siguiente=='}')EstadoPrimario++;
                            else EstadoLEX=1;
                            break;
                        default:
                            System.out.println("Error en el lexema
ANALIZADOR1");
                            break;
                    }
                    break;
```


Archivo
{
CONJ: vocales -> a,e,i,o,u;
CONJ: otrasLetras -> m,r,c,l,g,s;
CONJ: abecedario -> a~z;
CONJ: conjnum -> 2~6;
CONJ: digito -> 0~9;
Expresion1 -> . | "Los " "El " . + | {vocales} {otrasLetras} * | {abecedario} "_";
%%
%%
Expresion1 :"los murcielagos llevan_en_su_nombre_todas_las_vocales."; |
operaciones : "5++3*11";
}
Analizar
Evaluar Lexemas


Archivo Manuales
{
CONJ: vocales -> a,e,i,o,u;
CONJ: otrasLetras -> m,r,c,l,g,s;
CONJ: abecedario -> a~z;
CONJ: conjnum -> 2~6;
CONJ: digito -> 0~9;
Expresion1 -> . | "Los " "El " . + | {vocales} {otrasLetras} * | {abecedario} "_";
%%
%%
Expresion1 :"los murcielagos llevan_en_su_nombre_todas_las_vocales.";
Expresion1 : "El murcielago es_una_animal_mamifero";
Expresion2: "numero 5: cinco";
Expresion2 :"numero 2";
Expresion3 : "la_expresion 5>4 es TRUE.";
Expresion3 : "95>4 es FALSE.";
operaciones : "5+13*7+9/10-1";
operaciones : "5++3*11";
}
CONJUNTO: conjnum 2 6
CONJUNTO: digito 0 9
ER: Expresion1 . | "Los " "El " . + | [a,e,i,o,u] [m,r,c,l,g,s] * | [a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z]
LEX: Expresion1 contenido: los murcielagos llevan_en_su_nombre_todas_las_vocales.
LEX: Expresion1 contenido: El murcielago es_una_animal_mamifero
LEX: Expresion2 contenido: numero 5: cinco
Analizar
Evaluar Lexemas

Su árbol sería el siguiente:



El dato de entrada:

```
{
CONJ: vocales -> a,e,i,o,u;
CONJ: otrasLetras -> m,r,c,l,g,s;
CONJ: abecedario -> a~z;
CONJ: conjnum -> 2~6;
CONJ: digito -> 0~9;


Expresion1 -> . | "Los " "El " . + | {vocales} {otrasLetras} * |
{abecedario} "_";
Expresion2 -> . +{abecedario}  . {conjnum} ? . ":" +{abecedario};
Expresion3 -> . . . . . * | {abecedario} "_" + {conjnum} ">" + {conjnum}
"es " | "TRUE" "FALSE" ".";
operaciones -> . +{digito} + . |"+" |"-" |"*" "/" +{digito};

%%
%%

Expresion1 :"los murcielagos llevan_en_su_nombre_todas_las_vocales.";

}
```