



Universidad Austral de Chile

Tarea Compiladores

Desarrollo de un Traductor Dirigido por Sintaxis

Integrantes: Carlos Denis
Fernando Mora
Tomás Vargas

Profesora: María Eliana de la Maza

Introducción

En el presente informe daremos a conocer el proceso que llevamos a cabo para lograr la implementación de un traductor dirigido por sintaxis en donde su funcionamiento consiste en aceptar una secuencia de instrucciones del lenguaje AZ2012 y efectuar las acciones a medida que se realiza el análisis sintáctico.

Las herramientas que utilizamos para trabajar son:

- Lenguaje de programación Java
- Analizador léxico llamado JFLEX
- Analizador sintáctico llamado CUP

Al finalizar la lectura de este informe, el lector quedará con un conocimiento general del funcionamiento de estos traductores, además tendrá la capacidad de implementar, mediante las herramientas estudiadas, un traductor dirigido por sintaxis.

Objetivos

- Dar solución al problema señalado de manera clara y entendible.
- Entender el funcionamiento de un traductor dirigido por sintaxis.
- Entender el concepto de reglas semánticas.
- Aprender a especificar gramáticas utilizando los complementos de Java: JFLEX y CUP.

Desarrollo

Gramática inicial, entregada en el segundo informe:

| | | |
|---------------------|----|---|
| S | -> | comienzo inst fin |
| inst | -> | inst inst epsilon inst_if inst_asig inst_entrada inst_salida |
| inst_if | -> | if cond then inst |
| inst_asig | -> | id = exp |
| inst_entrada | -> | entrada (id) |
| inst_salida | -> | salida (" string ") salida(id) |
| cond | -> | exp oprel exp |
| exp | -> | exp + exp exp - exp exp * exp exp / exp id const |
| oprel | -> | > < = |
| id | -> | A B C ... X Y Z id |
| const | -> | 0 1 2 ... 8 9 const const |
| string | -> | *todas las combinaciones para hacer palabras* |

- Los identificadores serán solo letras mayúsculas
- Símbolos de puntuación: { (,) , " " }

Ésta gramática debió tener algunos cambios, como por ejemplo la creación de la variable "INSTT" para eliminar la recursividad izquierda que generaba la variable "inst". Quedando la gramática final como sigue:

| | | |
|---------------------|----|--|
| S | -> | COMIENZO ENTER INSTT FIN ENTER |
| INSTT | -> | INST INSTT INST |
| INST | -> | INST_IF INST_ASIG INST_ENTRADA INST_SALIDA |
| INST_IF | -> | IF COND THEN INST |
| INST_ASIG | -> | LETRA IGUAL EXP ENTER |
| INST_ENTRADA | -> | ENTRADA PARA LETRA PARC ENTER DIGITO ENTER |
| INST_SALIDA | -> | SALIDA PARA ESTRING PARC ENTER SALIDA PARA LETRA PARC ENTER |
| COND | -> | EXP MENOR EXP EXP MAYOR EXP EXP IGUAL EXP |
| EXP | -> | EXP MAS T EXP MENOS T T |
| T | -> | T POR F T DIV F F |
| F | -> | LETRA DIGITO |

Código generado con JFLEX

```
import java_cup.runtime.Symbol;|
%%
%cup

variable=[A-Z]
digito=[0-9]+
string="\\"~"\"
enter=[\n]

%%

"=" {return new Symbol(sym.IGUAL);}
"+" {return new Symbol(sym.MAS);}
"-" {return new Symbol(sym.MENOS);}
"*" {return new Symbol(sym.POR);}
"/" {return new Symbol(sym.DIV);}
"<" {return new Symbol(sym.MENOR);}
">" {return new Symbol(sym.MAYOR);}
"(" { return new Symbol(sym.PARA);}
")" { return new Symbol(sym.PARC);}
"comienzo" {return new Symbol(sym.COMIENZO);}
"fin" { return new Symbol(sym.FIN);}
"salida" {return new Symbol(sym.SALIDA);}
"entrada" {return new Symbol(sym.ENTRADA);}
"if" {return new Symbol(sym.IF);}
"then" {return new Symbol(sym.THEN);}

{variable} {return new Symbol(sym.LETRA, new String(yytext()));}
{digito} {return new Symbol(sym.DIGITO, new Integer(yytext()));}
{enter} {return new Symbol(sym.ENTER);}
{string} {return new Symbol(sym.ESTRING, new String(yytext()));}

[\\t\\r\\f ] { /* ignore white space. */ }

. { System.err.println("Illegal character: "+yytext()); }
```

Aquí principalmente es donde se definen los símbolos y palabras reservadas del lenguaje.

Código generado con CUP

A continuación de lograr manejar el complemento JFLEX, comenzamos con el análisis sintáctico utilizando CUP para la implementación del *action code* y *parser code*, terminales, no terminales y las producciones de la gramática de la siguiente forma:

Action Code:

```
action code{:
public class AZ2012 extends JFrame{
    public Object [][]ts= new Object[26][2]; /*arreglo que almacena variables creadas
                                                y su valor. pseudotabla de simbolos*/
    private int ultima_posicion = 0; //cumple la funcion de un pseudopuntero
    private JTextArea cuadro = new JTextArea(); /*Lugar en donde desplegaremos las
                                                cosas que haremos.*/

    public boolean flag = true;

    public AZ2012(){
        super("Output");
        setBounds(500,50,300,300);
        setResizable(false);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
        cuadro.setBounds(550, 100, 200, 200);
        cuadro.setEditable(false);
        add(cuadro);
    }

    /*Retorna el indice de la tabla de simbolos en donde se aloja la variable
    buscada 'var' */
    public int busca(String var){
        for(int i=0 ; i<26 ; i++){
            if (ts[i][0].equals(var)){
                return i;
            }
        }
        return -1;
    }

    //funcion que agrega variables y su valor a la 'tabla de simbolos'
    public void agrega(String asdf1, String asdf2){
        ts[ultima_posicion][0] = asdf1;
        ts[ultima_posicion][1] = asdf2;
        ultima_posicion++;
    }

    /*misma funcion aterior. sobregargada que agrega variables y valores a la tabla de
    simbolos.*/
    public void agrega(String asdf1, int asdf2){
        ts[ultima_posicion][0] = asdf1;
        ts[ultima_posicion][1] = asdf2;
        ultima_posicion++;
    }
}
```

```

//Función que usaremos para desplegar en la ventana creada
public void imprime(String asdf){
    String var = cuadro.getText();
    asdf = asdf.replace("'", " ");
    cuadro.setText(var + '\n' + asdf);
}

public void imprime(int asdf){
    String var = cuadro.getText();
    String tmp = asdf+"";
    tmp = tmp.replace("'", " ");
    cuadro.setText(var + '\n' + tmp);
}

public void borrar(){
    cuadro.setText("");
}

}

AZ2012 prog = new AZ2012();
:}

```

Parser Code:

```

parser code{:
    public static void main(String args[])throws Exception{
        System.out.println("----->Traductor Sintactico<-----");
        System.out.println("Para comenzar escriba comienzo");
        System.out.println("Para finalizar escriba fin");
        System.out.println("Porfavor revisar manual usuario");
        System.out.println("*****");
        new parser(new Yylex(System.in)).parse();
    }
:}

```

Terminales y No Terminales:

```

terminal IGUAL,MAS,MENOS,POR,DIV,MENOR,MAYOR,PARA,PARC,COMIENZO,FIN,SALIDA,ENTRADA,
    IF,THEN,ENTER;
terminal Integer DIGITO;
terminal String LETRA,ESTRING;

non terminal S,INST,INSTI,INST_IF,INST_ASIG,INST_ENTRADA,INST_SALIDA;
non terminal Integer COND;
non terminal Integer EXP,T,F;

precedence left MAS, MENOS;
precedence left POR, DIV;

```

Gramática:

```
S ::= COMIENZO ENTER INSTT FIN {:System.exit(0);:} ENTER;

INSTT ::= INST INSTT |
        INST;

INST ::= INST_IF |
        INST_ASIG |
        INST_ENTRADA |
        INST_SALIDA;

INST_IF ::= IF COND:b {:if(b==0) prog.flag = false;:} THEN INST {:prog.flag = true;:};
/*Se revisa la condicion, si es falso se marca una variable publica que restringe la ejecucion
de la proxima instruccion*/

INST_ASIG ::= LETRA:s IGUAL EXP:e {:if (prog.flag) prog.agrega(s,e);:} ENTER;
//Se agrega a la Tabla de Simbolos una variable y un valor dado

INST_ENTRADA ::= ENTRADA PARA LETRA:i PARC ENTER DIGITO:s {:if (prog.flag) prog.agrega(i,s);:} ENTER;
//Asigna a una variable un valor dado, ingresado desde teclado

INST_SALIDA ::= SALIDA PARA ESTRING:s {:if (prog.flag) prog.imprime(">> "+s);:} PARC ENTER |
//Muestra un String en la salida del programa

        SALIDA PARA LETRA:s {:if (prog.flag) prog.imprime(">> "+prog.ts[prog.busca(s)][1]);:} PARC ENTER;
//Muestra en la salida el valor de una variable

COND ::= EXP:l MENOR EXP:r {:if(l.intValue()<r.intValue()) RESULT=1; else RESULT=0;:} |
//Comprueba relaciones entre expresiones, 1->>true, 0->>false

        EXP:l MAYOR EXP:r {:if(l.intValue()>r.intValue()) RESULT=1; else RESULT=0;:} |
        EXP:l IGUAL EXP:r {:if(l.intValue()==r.intValue()) RESULT=1; else RESULT=0;:};

EXP ::= EXP:l MAS T:r {:RESULT=l+r;:} |
        EXP:l MENOS T:r {:RESULT=l.intValue()-r.intValue();:} |
        T:e {:RESULT=e.intValue();:};

T ::= T:l POR F:r {:RESULT=l.intValue()*r.intValue();:} |
        T:l DIV F:r {:RESULT=l.intValue()/r.intValue();:} |
        F:e {:RESULT=e.intValue();:};

F ::= LETRA:s {:RESULT=(Integer)prog.ts[prog.busca(s)][1];:} |
        DIGITO:d {:RESULT=d;:};
```

Especificaciones del programa

Nombre: Traductor Sintáctico

Versión: 1.0

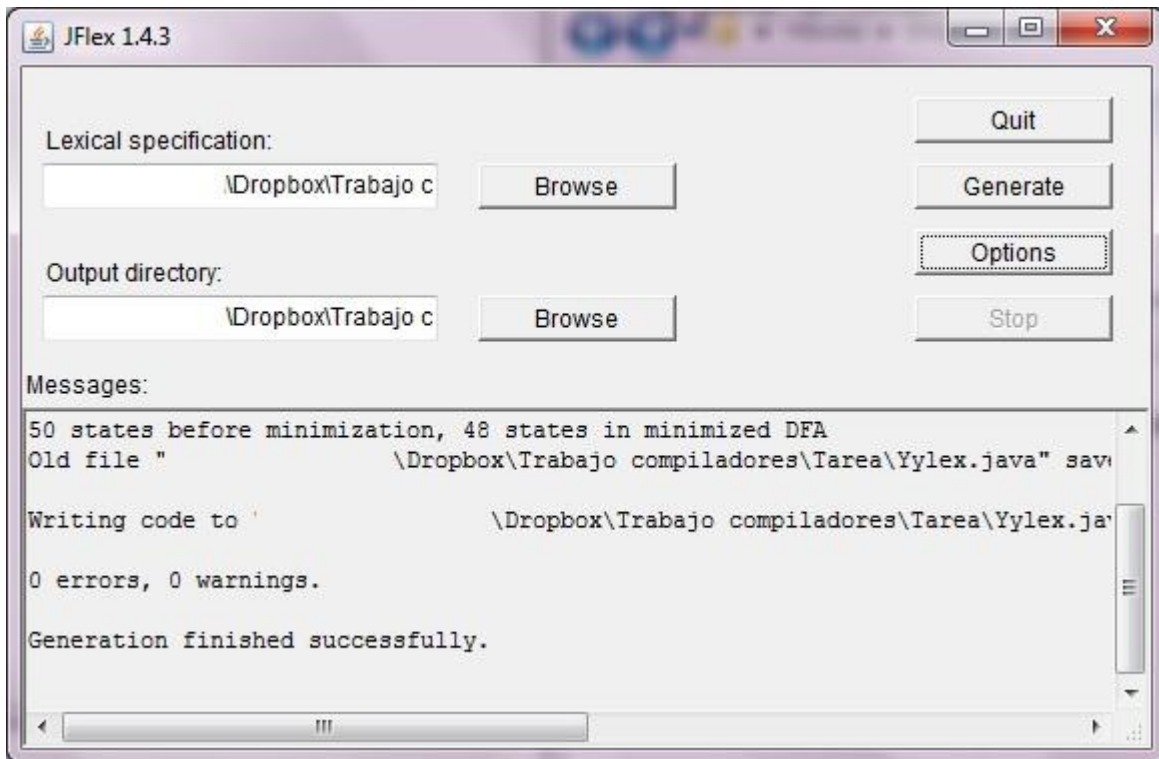
Lenguaje: Java JDK 1.7.0_02 (Build 1320-110325).

Requerimientos básicos de software y hardware:

- Sistema Operativo: Windows 98 o superior.
- Java Runtime Environment (JRE) 1.4 o superior.
- Memoria RAM 64mb. O superior.
- Procesador de 32 o 64 bits.
- Espacio en disco duro: 5 MB.
- Unidad óptica: Lector de CD.

Compilación del programa en Windows

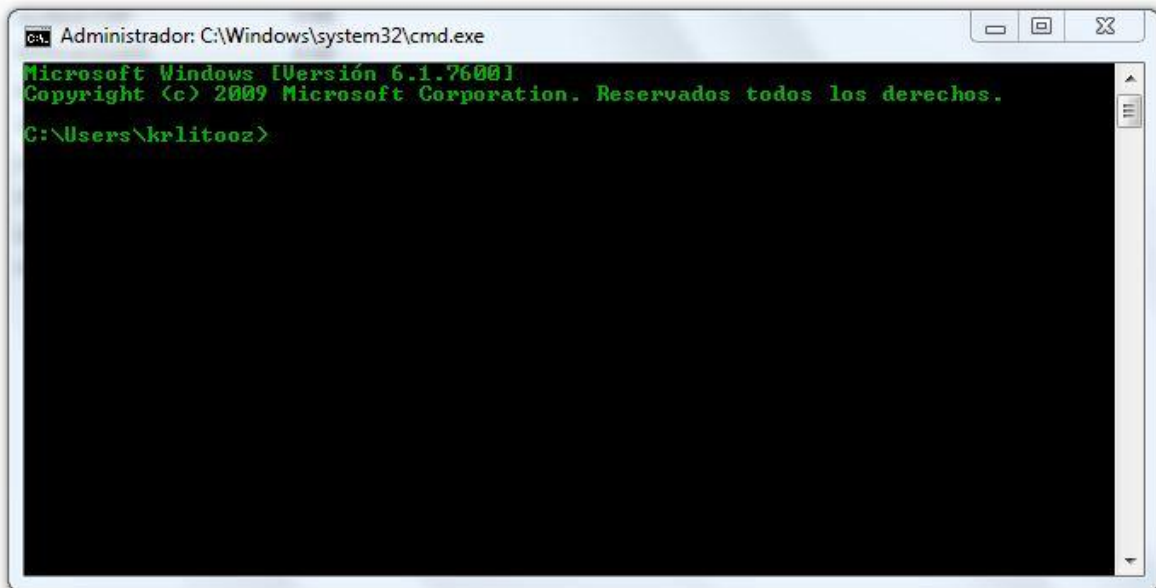
1º- Abrimos el archivo *JFlex.jar* ubicado en la carpeta */jflex/lib*:



2º- En *Lexical specification* ingresamos el archivo *analizador.flex*, luego presionamos el botón *Generate*. De este modo, JFlex nos genera el archivo *Ylex.java*:

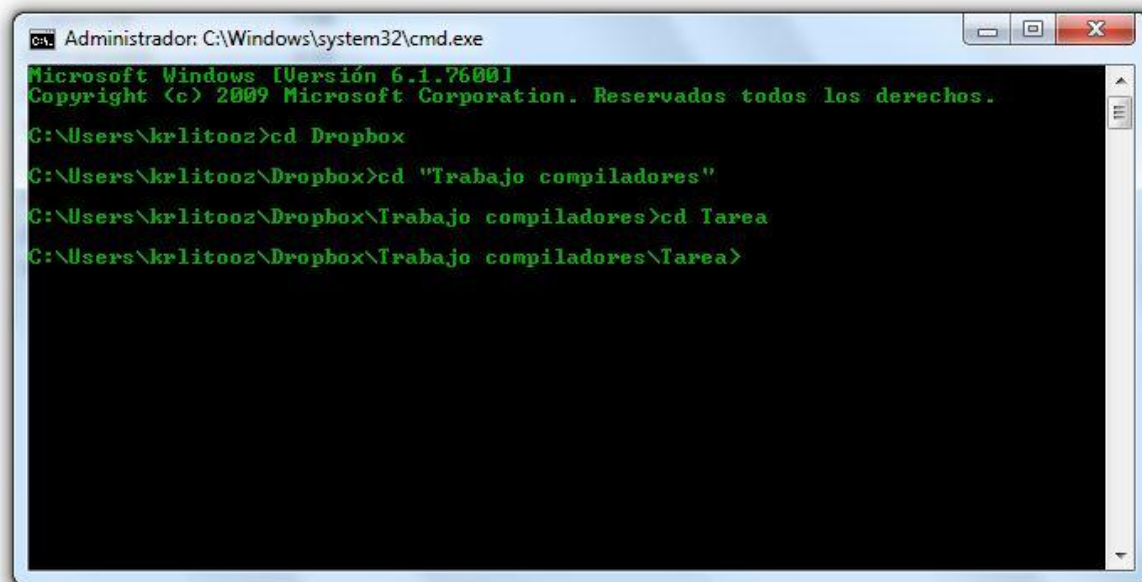
| Nombre | Fecha de modifica... | Tipo | Tamaño |
|------------------------------------|----------------------|---------------------|--------|
| Capturas | 13-11-2012 15:12 | Carpeta de archivos | |
| java_cup | 12-11-2012 13:28 | Carpeta de archivos | |
| jflex | 12-11-2012 12:26 | Carpeta de archivos | |
| analizador.flex | 13-11-2012 14:53 | Archivo FLEX | 2 KB |
| CUP\$parser\$actions\$AZ2012.class | 13-11-2012 15:10 | Archivo CLASS | 3 KB |
| CUP\$parser\$actions.class | 13-11-2012 15:10 | Archivo CLASS | 7 KB |
| miparser.cup | 13-11-2012 14:59 | Archivo CUP | 5 KB |
| parser.class | 13-11-2012 15:10 | Archivo CLASS | 4 KB |
| parser.java | 13-11-2012 15:10 | java_auto_file | 36 KB |
| sym.class | 13-11-2012 15:10 | Archivo CLASS | 1 KB |
| sym.java | 13-11-2012 15:10 | java_auto_file | 2 KB |
| Ylex.class | 13-11-2012 15:10 | Archivo CLASS | 7 KB |
| Ylex.java | 13-11-2012 14:56 | java_auto_file | 18 KB |
| Ylex.java~ | 13-11-2012 13:03 | Archivo JAVA~ | 18 KB |

3º- Ejecutamos la consola CMD:



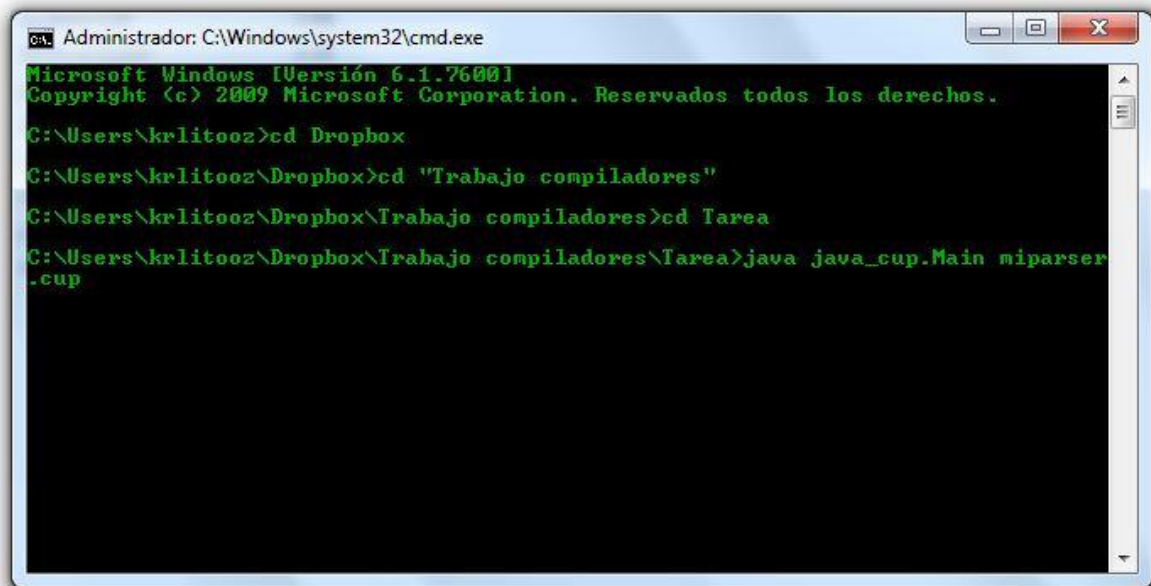
```
Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\krlitooz>
```

4º- Nos dirigimos a la carpeta donde tenemos guardado el programa:



```
Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\krlitooz>cd Dropbox
C:\Users\krlitooz\Dropbox>cd "Trabajo compiladores"
C:\Users\krlitooz\Dropbox\Trabajo compiladores>cd Tarea
C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>
```

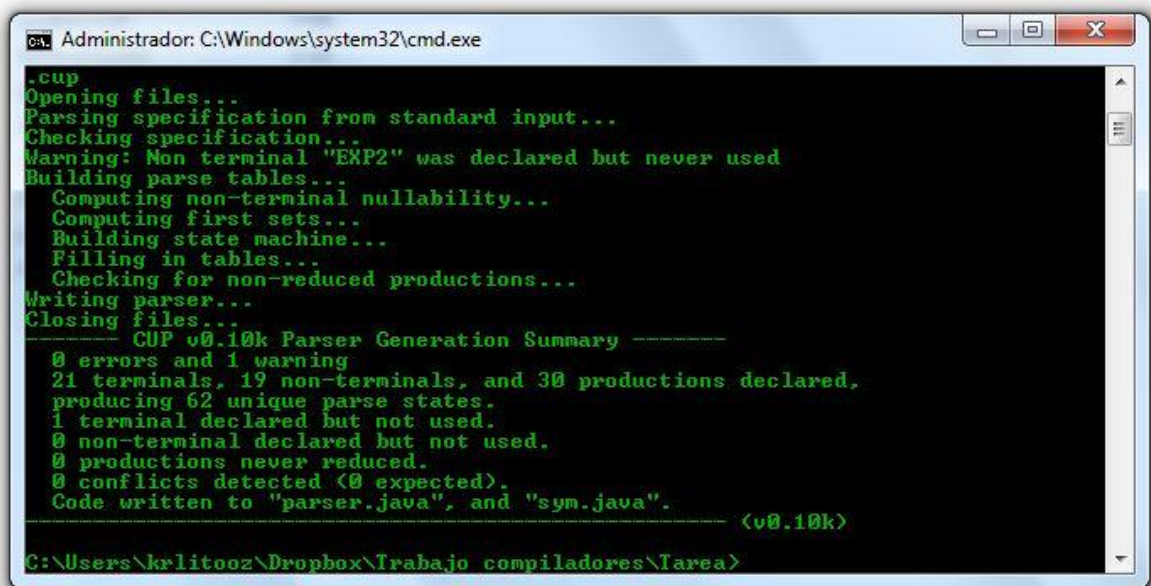
5º- Ingresamos el primer comando *java java_cup.Main miparser.cup*:



```
Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\krlitooz>cd Dropbox
C:\Users\krlitooz\Dropbox>cd "Trabajo compiladores"
C:\Users\krlitooz\Dropbox\Trabajo compiladores>cd Tarea
C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>java java_cup.Main miparser
.cup
```

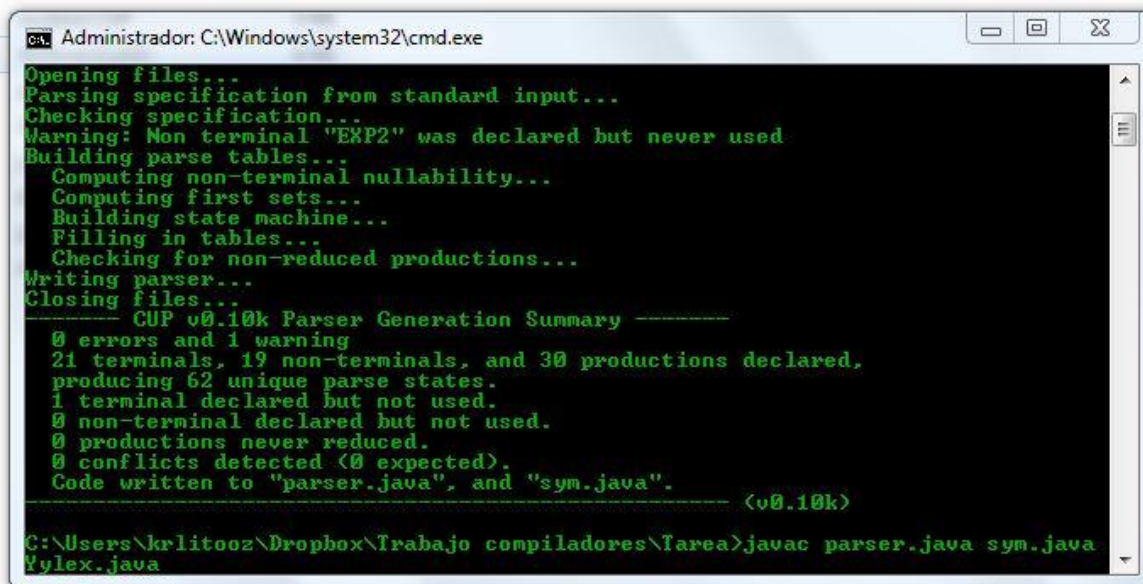
Obteniendo el primer resultado:



```
Administrador: C:\Windows\system32\cmd.exe
.cup
Opening files...
Parsing specification from standard input...
Checking specification...
Warning: Non terminal "EXP2" was declared but never used
Building parse tables...
Computing non-terminal nullability...
Computing first sets...
Building state machine...
Filling in tables...
Checking for non-reduced productions...
Writing parser...
Closing files...
CUP v0.10k Parser Generation Summary -----
0 errors and 1 warning
21 terminals, 19 non-terminals, and 30 productions declared,
producing 62 unique parse states.
1 terminal declared but not used.
0 non-terminal declared but not used.
0 productions never reduced.
0 conflicts detected (0 expected).
Code written to "parser.java", and "sym.java".
                                         (v0.10k)

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>
```

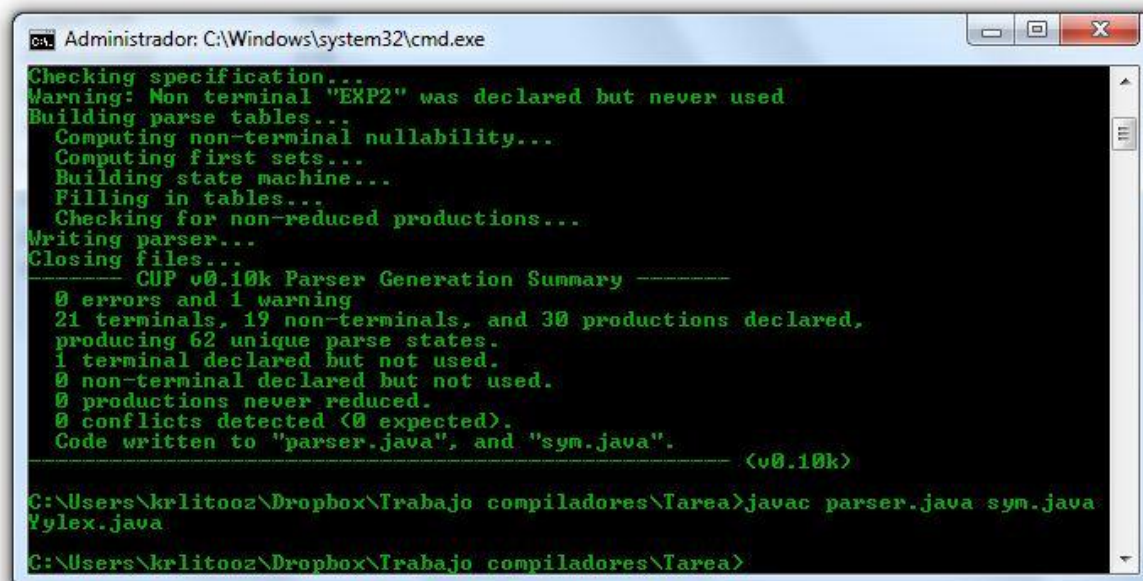
6º- Ingresamos el segundo comando `javac parser.java sym.java Yylex.java`:



```
CA: Administrador: C:\Windows\system32\cmd.exe
Opening files...
Parsing specification from standard input...
Checking specification...
Warning: Non terminal "EXP2" was declared but never used
Building parse tables...
Computing non-terminal nullability...
Computing first sets...
Building state machine...
Filling in tables...
Checking for non-reduced productions...
Writing parser...
Closing files...
CUP v0.10k Parser Generation Summary -----
0 errors and 1 warning
21 terminals, 19 non-terminals, and 30 productions declared,
producing 62 unique parse states.
1 terminal declared but not used.
0 non-terminal declared but not used.
0 productions never reduced.
0 conflicts detected (0 expected).
Code written to "parser.java", and "sym.java".
----- (v0.10k)

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>javac parser.java sym.java
Yylex.java
```

Obteniendo el segundo resultado:



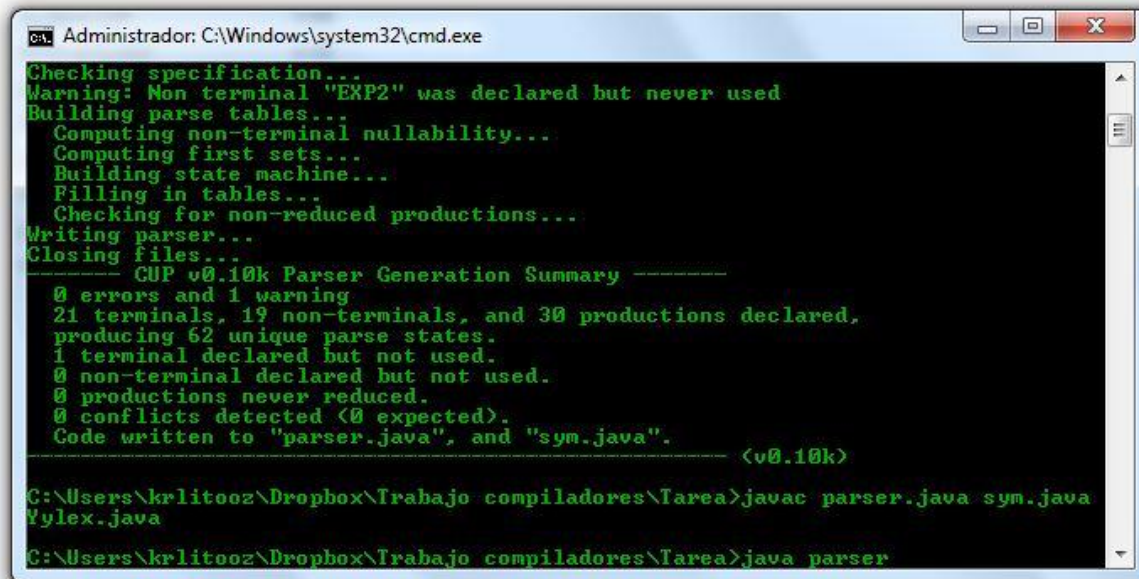
```
CA: Administrador: C:\Windows\system32\cmd.exe
Checking specification...
Warning: Non terminal "EXP2" was declared but never used
Building parse tables...
Computing non-terminal nullability...
Computing first sets...
Building state machine...
Filling in tables...
Checking for non-reduced productions...
Writing parser...
Closing files...
CUP v0.10k Parser Generation Summary -----
0 errors and 1 warning
21 terminals, 19 non-terminals, and 30 productions declared,
producing 62 unique parse states.
1 terminal declared but not used.
0 non-terminal declared but not used.
0 productions never reduced.
0 conflicts detected (0 expected).
Code written to "parser.java", and "sym.java".
----- (v0.10k)

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>javac parser.java sym.java
Yylex.java

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>
```


Ejemplo de funcionamiento

1º- Luego de compilar, ejecutamos el programa ingresando el comando *java parser*:

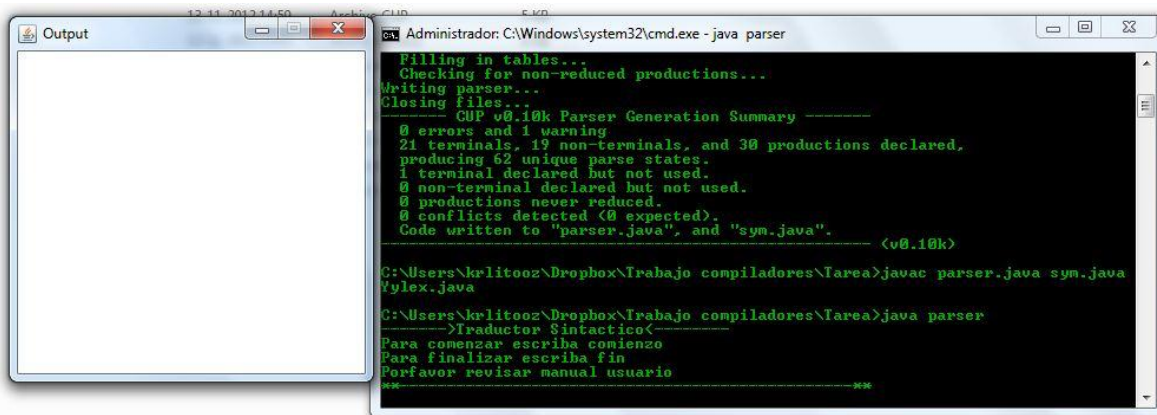


```
Administrador: C:\Windows\system32\cmd.exe
Checking specification...
Warning: Non terminal "EXP2" was declared but never used
Building parse tables...
  Computing non-terminal nullability...
  Computing first sets...
  Building state machine...
  Filling in tables...
  Checking for non-reduced productions...
Writing parser...
Closing files...
----- CUP v0.10k Parser Generation Summary -----
0 errors and 1 warning
21 terminals, 19 non-terminals, and 30 productions declared,
producing 62 unique parse states.
1 terminal declared but not used.
0 non-terminal declared but not used.
0 productions never reduced.
0 conflicts detected (0 expected).
Code written to "parser.java", and "sym.java".
----- (v0.10k) -----

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>javac parser.java sym.java
Yylex.java

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>java parser
```

Obtenemos la ventana de entrada y la ventana de salida llamada "Output":

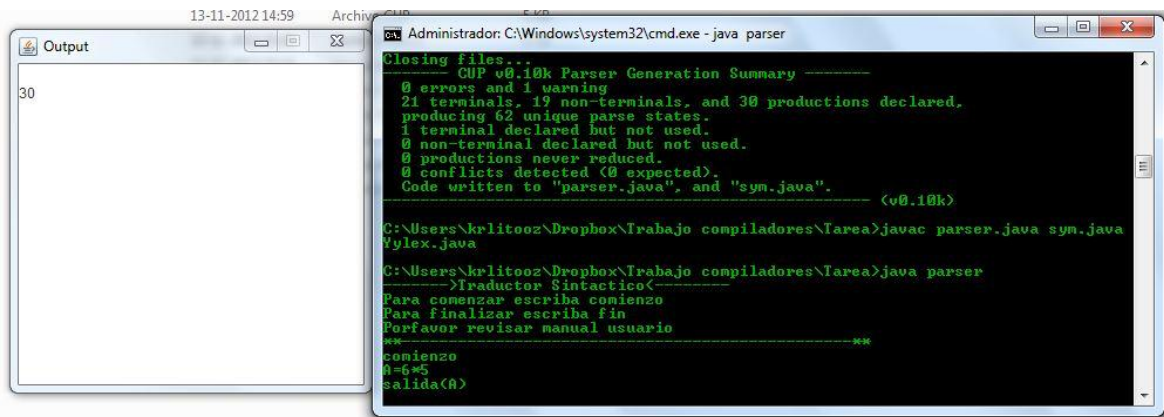


```
Output
Administrador: C:\Windows\system32\cmd.exe - java parser
Filling in tables...
Checking for non-reduced productions...
Writing parser...
Closing files...
----- CUP v0.10k Parser Generation Summary -----
0 errors and 1 warning
21 terminals, 19 non-terminals, and 30 productions declared,
producing 62 unique parse states.
1 terminal declared but not used.
0 non-terminal declared but not used.
0 productions never reduced.
0 conflicts detected (0 expected).
Code written to "parser.java", and "sym.java".
----- (v0.10k) -----

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>javac parser.java sym.java
Yylex.java

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>java parser
----->Traductor Sintactico<-----
Para comenzar escriba comienzo
Para finalizar escriba fin
Porfavor revisar manual usuario
***
```

2º- Escribimos "comienzo", apretamos la tecla "enter", ingresamos una instrucción y pedimos la salida de la siguiente forma:



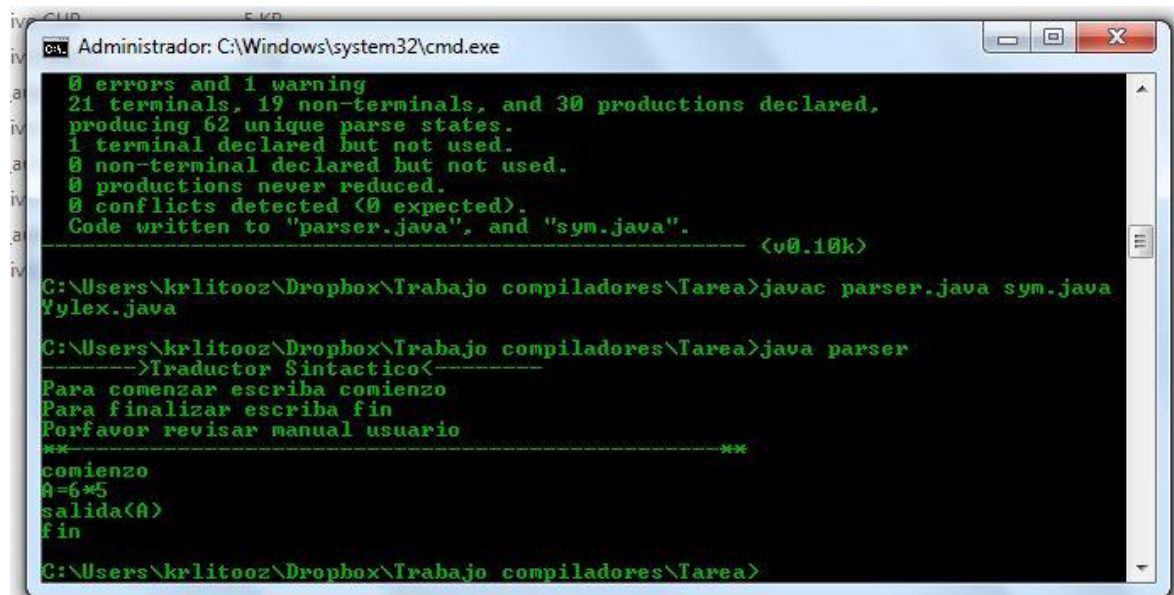
The screenshot shows two windows. On the left, an 'Output' window displays the number '30'. On the right, a command prompt window titled 'Administrador: C:\Windows\system32\cmd.exe - java parser' shows the following text:

```
Closing files...
CUP v0.10k Parser Generation Summary -----
0 errors and 1 warning
21 terminals, 19 non-terminals, and 30 productions declared.
producing 62 unique parse states.
1 terminal declared but not used.
0 non-terminal declared but not used.
0 productions never reduced.
0 conflicts detected (0 expected).
Code written to "parser.java", and "sym.java".
----- (v0.10k)

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>javac parser.java sym.java
Vylex.java

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>java parser
----->Traductor Sintactico<-----
Para comenzar escriba comienzo
Para finalizar escriba fin
Porfavor revisar manual usuario
**
comienzo
A=6*5
salida(A)
```

3º- Finalmente ingresamos "fin" y se termina el programa:



The screenshot shows a command prompt window titled 'Administrador: C:\Windows\system32\cmd.exe' with the following text:

```
0 errors and 1 warning
21 terminals, 19 non-terminals, and 30 productions declared.
producing 62 unique parse states.
1 terminal declared but not used.
0 non-terminal declared but not used.
0 productions never reduced.
0 conflicts detected (0 expected).
Code written to "parser.java", and "sym.java".
----- (v0.10k)

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>javac parser.java sym.java
Vylex.java

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>java parser
----->Traductor Sintactico<-----
Para comenzar escriba comienzo
Para finalizar escriba fin
Porfavor revisar manual usuario
**
comienzo
A=6*5
salida(A)
fin

C:\Users\krlitooz\Dropbox\Trabajo compiladores\Tarea>
```

Conclusión

La construcción del traductor dirigido por sintaxis fue una experiencia muy provechosa para los integrantes de nuestro grupo, pues pudimos visualizar de forma mas clara las etapas y procesos que realiza un computador para interpretar un lenguaje de programación, principalmente en la construcción de un programa.

El mayor problema o dificultad que tuvimos fue el reconocer los saltos de línea además de problemas de recursividad en la gramática. Todo esto solo pudo ser resuelto cambiando detalles en la gramática

Afortunadamente, después de estudiar y buscar soluciones para estas interrogantes, logramos construir nuestro traductor dirigido por sintaxis. Sin embargo, creemos que podríamos implementar algunas mejoras, como por ejemplo una interfaz más elaborada.

Bibliografía

Blog Openfecks:

- <http://openfecks.wordpress.com/jlex-y-cup/>

Documentación Oficial:

- <http://www.cs.princeton.edu/~appel/modern/java/JLex/current/manual.html>
- <http://www.cs.princeton.edu/~appel/modern/java/CUP/manual.html>

Apuntes:

- Clases de Compiladores Universidad Austral de Chile 2012.
Sra. María Eliana de la Maza W.