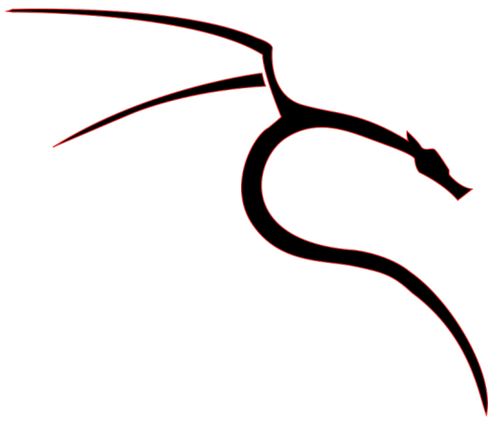

UNSimplexMO: Manual Técnico

Juan David Quintero Perez
Sergio Andres Castro Castro
Juan Sebastián Vivero Jauregui
Marlon Enrique Noguera Ramírez
Cristian Camilo Cristancho Castaño

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial
Bogotá, Colombia
2017

Contents

1 Gramática de UNSimplexMO

A continuación se mostrarán las producciones que genera el lenguaje UNSimplexMO

$$
\begin{aligned}
\langle program \rangle &\models BEGIN\ \langle problem \rangle\ \langle other \rangle\ END \\
\langle other \rangle &\models MFUNCTIONS\ \langle function \rangle \\
\langle problem \rangle &\models VAR\ \langle variables \rangle\ \langle objective \rangle\ \langle function \rangle\ RES\ \langle restric \rangle\ LIM\ \langle bounds \rangle \mid \epsilon \\
\langle variables \rangle &\models \langle ID \rangle \\
\langle objective \rangle &\models MAX \mid MIN \\
\langle function \rangle &\models \langle fItem \rangle\ (\ \langle operator \rangle\ \langle item \rangle\ ) \mid \epsilon \\
\langle item \rangle &\models \langle FLOAT \rangle\ \langle ID \rangle \mid \langle ID \rangle \mid \langle FLOAT \rangle \\
\langle fItem \rangle &\models \langle operator \rangle\ \langle item \rangle \\
\langle operator \rangle &\models MAS \mid MENOS \\
\langle restric \rangle &\models \langle function \rangle\ \langle lComparation \rangle\ \langle operator \rangle\ \langle FLOAT \rangle \\
\langle lComparation \rangle &\models LE \mid GE \mid EQ \\
\langle lComp \rangle &\models LE \mid GE \\
\langle bounds \rangle &\models \langle ID \rangle\ \langle lComp \rangle\ \langle FLOAT \rangle \mid \langle FLOAT \rangle \\
&\quad \mid \langle FLOAT \rangle\ \langle lComp \rangle\ \langle ID \rangle\ \langle lComp \rangle \langle FLOAT \rangle \langle lComp \rangle\ \langle ID \rangle \\
&\quad \mid \langle FLOAT \rangle\ \langle lComp \rangle\ \langle ID \rangle\ \langle lComp \rangle \langle FLOAT \rangle \\
\langle ID \rangle &\models [a-zA-Z\_][a-zA-Z0-9\_] \mid \epsilon \\
\langle FLOAT \rangle &\models (\ [0-9]\ )\ (\ '.'\ (\ [0-9]\ )\ +\ (\ (\ 'e' \mid 'E'\ )\ (\ '+' \mid '-'\ )\ (\ [0-9]\ )\ )\ )
\end{aligned}
$$

2 ANTLR

El lenguaje fue desarrollado en ANTLR[5], definiendo el léxico y la gramática en un archivo propio de ANTLR (.g4). Luego se generó el reconocedor, que creaba todas las clases bases en Java.

2.1 Tokens del lenguaje

```
MAX: 'maximieren';
MIN: 'minimieren';
RES: 'cortapisa';
VAR: 'variabilis';
LIM: 'fines';

MAS: '+';
MENOS: '-';
MULT: '*';
DIV: '/';

GT: '>';
LT: '<';
LE: '<=';
GE: '>=';
EQ: '=';
COMA: ',';

BEGIN: 'Dantzig';
END: 'MayoDe2005' ;
```

```
MFUNCTIONS: 'Goicochea';

ID: [a-zA-Z_][a-zA-Z0-9_]*;
FLOAT : ([0-9])+ ('.' ([0-9])+(('e' | 'E')('+'|'-')?([0-9])+)?)?;

WS:[ \t\r\n]+ -> skip;
```

2.2 Gramática (en notación E-BNF, Extended Backus–Naur Form)

```
program
 : BEGIN problem other* END
 ;

other
 : MFUNCTIONS function
 ;

problem
 : VAR variables+ objective function RES restric* LIM bounds*
 ;

variables
 : ID #Variable
 ;

objective
 : MAX | MIN #Objective
 ;

function
 : fItem (operator item)* #Func
 ;

item
 : FLOAT ID
 | ID
 | FLOAT
 ;

fItem
 : operator? item
 ;

operator
 : MAS
 | MENOS
 ;

restric
 : function lComparation operator? FLOAT #Res
 ;

lComparation
 : LE
 | GE
 | EQ
 ;

lComp
```

```
 : LE
 | GE
 ;

bounds
 : ID lComp FLOAT
 | FLOAT lComp ID
 | FLOAT lComp ID lComp FLOAT
 ;
```

2.3 Clases de Java

- **Simplex.java** en donde se ejecuta el método simplex cuando ya se tiene la matriz normalizada, el proceso se cubre en esta matriz, hasta conseguir la matriz con las soluciones deseadas.
- **SimplexCustomVisitor.java** en donde se toman los valores de las restricciones, variables y función(es) objetivo que el usuario pone y se guardan en las tablas de símbolos respectivas. También aquí se encuentran las funciones que se encargan de normalizar las líneas de las restricciones y las funciones objetivo.
- **Main.java** en donde se decide que procedimiento ejecutar, esto basado en si hay una o varias funciones objetivo. Se usa para otros propósitos como terminar de normalizar la matriz antes de que Simplex.

2.4 Diagramas de sintaxis

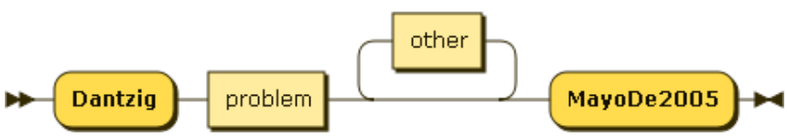

Imagen 1: Program.

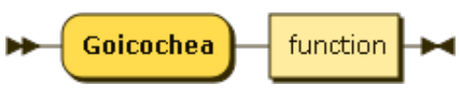

Imagen 2: Other.

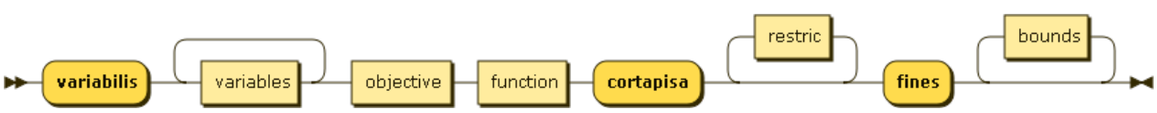

Imagen 3: Problem.

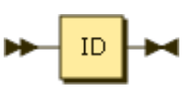

Imagen 4: Variables.

Imagen 5: Objective.

Imagen 6: Function.

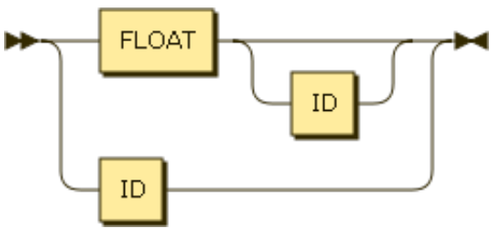

Imagen 7: Item.

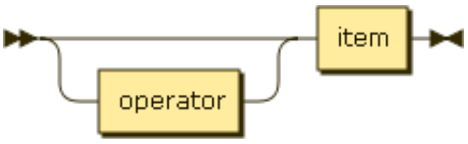

Imagen 8: FItem.

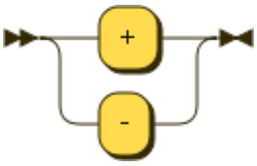

Imagen 9: Operator.

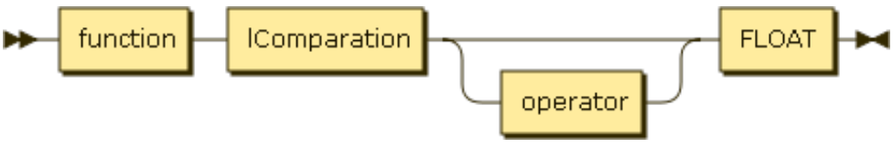

Imagen 10: Restric.

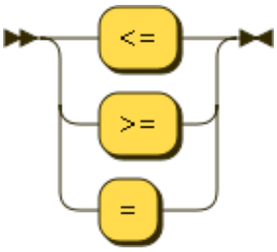

Imagen 11: LComparation.

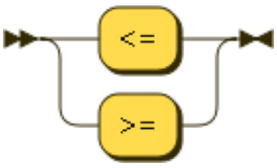

Imagen 12: LComp.

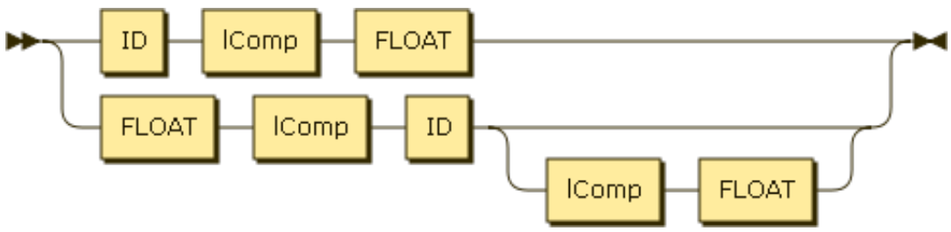

Imagen 13: Bounds.

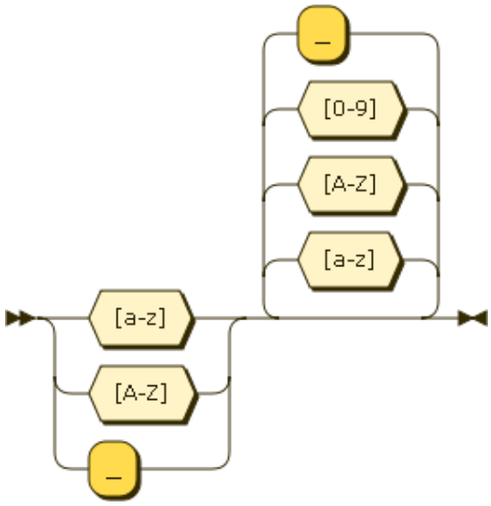

Imagen 14: ID.

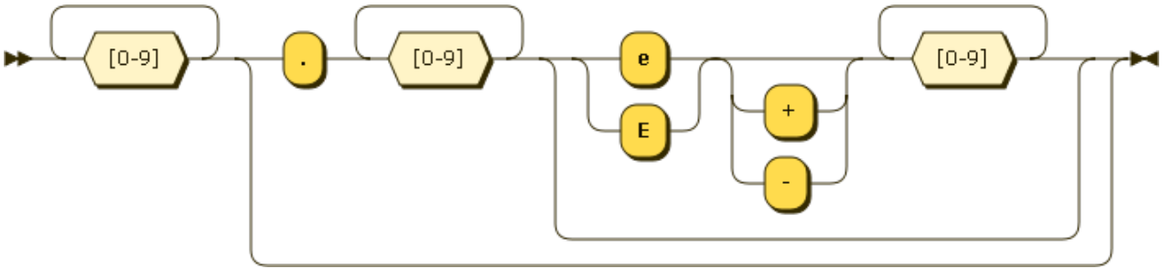

Imagen 15: Float.

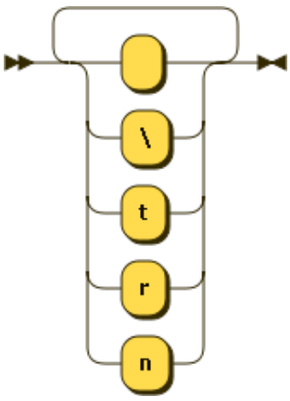

Imagen 16: WS.

2.5 Generación del Árbol de Parseo

Se puede generar con 2 herramientas:

- **Eclipse:** Con esta herramienta, al configurar bien este entorno, se puede ejecutar una ventana donde se muestra el árbol de sintaxis del programa a probar.

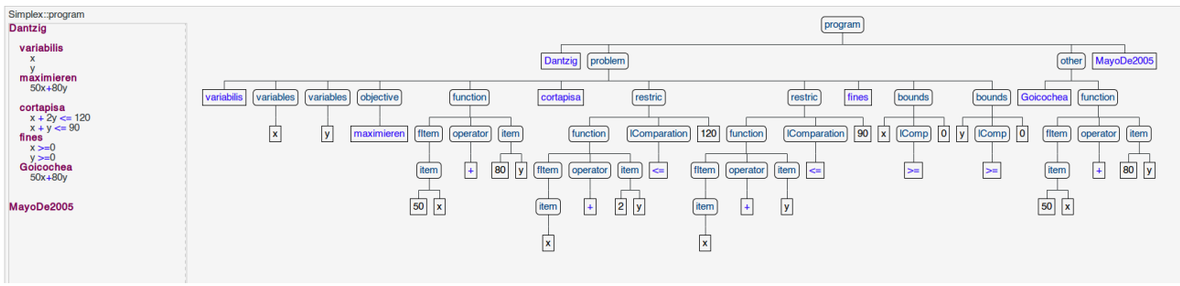

Imagen 17: Arbol de parseo en Eclipse.

- **Grun:** Ejecutando el comando en consola *grun Simplex program -gui test.smp* donde *test.smp* es donde se encuentra el programa que se quiere ver; éste genera una nueva ventana, donde podemos ver el árbol sintáctico.

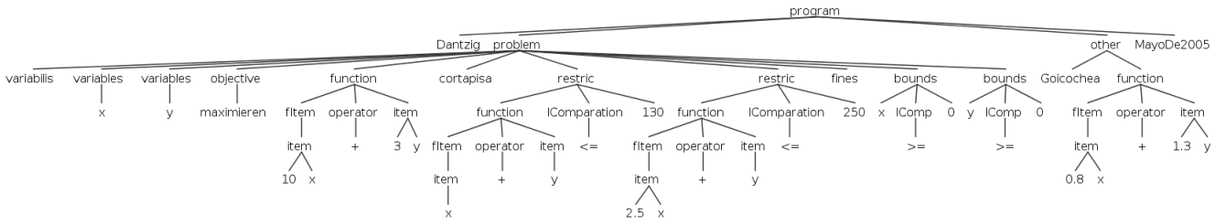

Imagen 18: Arbol de parseo en consola.

References

[1] http://ingenieria-industrial.net/leer/articulo/89

[2] www.ibm.com/support/knowledgecenter/es/SSSA5P_-12.5.1/ilog.odms.ide.help/OPL_Studio/opllangref/topics/opl_langref_-decisiontypes_dvars.html

[3] https://prezi.com/-p-crjmejb6k/variables-de-holgura-y-variables-de-excedente/

[4] http://ingenieria-industrial.net/leer/articulo/91

[5] Página Oficial de ANTLR - www.antlr.org