

**Analizador Sintáctico descendente con retroceso**

Universidad Tecnológica Nacional - Facultad regional Delta

Materia: Sintaxis y semántica de los lenguajes

Alumno: KUCHARUK, Manuel

Docentes: SANTOS, Juan; MIRANDA, Hernán

Fecha: 26/10/2015

Índice

[Gramática 3](#_Toc433120389)

[Código del programa 4](#_Toc433120390)

[Explicación, observaciones y comentarios 5](#_Toc433120391)

# Gramática y estructuras utilizadas

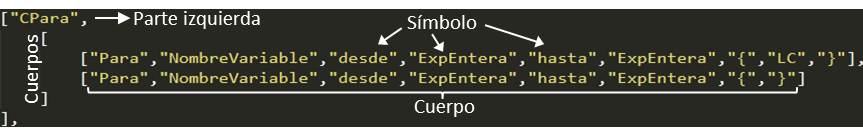
Usé la gramática completa de la consigna del TP, pero modificada para evitar reglas borradoras y recursividad izquierda.

Como el analizador sintáctico usa antes un analizador léxico, algunos símbolos no terminales pasan a ser terminales en la nueva gramática; por ejemplo NombreVariable, NombreFuncion, Operador, etc.

Símbolos No Terminales = {P, DV, DF, LP, LPCont, LC, CAsignacion, CPara, CSiEntonces, CLLamadoFuncion, LLPar, LLParCont, ExpEntera, ExpEntera2, Termino, Termino2, Factor, ExpLogica, ExpLogica2, OperandoLogico, OperandoLogico2, FactorLogico, Comparacion}

Las gramáticas están definidas en el archivo producciones.js en una variable tipo array llamada producciones.

Cada elemento del array de producciones también es un array que se compone de la siguiente manera:



El primer elemento es la parte izquierda de la producción, el segundo elemento es otro array, que contiene todas las producciones (cuerpos) asociadas a la parte izquierda.

Cada cuerpo es un nuevo array compuesto por símbolos.

En la imagen, se muestra la representación de la producción:

CPara -> Para NombreVariable desde ExpEntera hasta ExpEntera { LC }

Sabiendo que LC puede ser λ, se crea también la producción:

CPara -> Para NombreVariable desde ExpEntera hasta ExpEntera { }

# Código del programa

De index.js:

var pathLexer = path.join(\_\_dirname,"../bin","lexico.exe");

lexer.analizar(pathLexer,req.body.entrada,function (resLexer){

parser.analizar(resLexer+"$",function (resParser){

res.render("index",{

entrada: req.body.entrada,

salida: resLexer,

resultadoParser: formatearResultado(resParser)

}

}

}

lexer.analizar(pathLexer,req.body.entrada,function (resLexer){:

El programa llama al ejecutable léxico.exe, que es el analizador del TP1, analizando el contenido del textbox entrada y poniendo el resultado en la variable resLexer.

parser.analizar(resLexer+"$",function (resParser){:

Se ejecuta el analizador sintáctico con la cadena resLexer concatenada con el símbolo de fin de cadena $. El resultado se almacena en resParser (simplemente 0 si la cadena no fue aceptada y 1 si la cadena sí fue aceptada).

res.render("index",{

entrada: req.body.entrada,

salida: resLexer,

resultadoParser: formatearResultado(resParser)

Muestra la entrada del usuario, el resultado del análisis léxico y si la cadena fue aceptada o no por el analizador sintáctico.

De parser.js:

La implementación del algoritmo es básicamente la misma que el pseudocódigo provisto en los apuntes. Excepto por las siguientes modificaciones:

Cada función devuelve la posición a la que se está apuntando en la cadena de entrada, en vez de ser una variable global. De esa forma se puede volver hacia atrás en la función procesar en caso de error.

La función PNi, en vez ser una función por cada no terminal, es una única función que recibe un símbolo no terminal y busca las producciones (cuerpos) asociadas a ese símbolo.

# Explicación, observaciones y comentarios

Lo hice con Javascript del lado del servidor para poder usar ejecutar el programa del TP1 sin necesidad de recodificarlo.

Las producción de CSiEntonces -> --- -> NombreVariable no es correcta, puede haber un match incorrecto así que la saqué.

Tiene que