Memoria 2ª Entrega Práctica PDL:

Analizador sintáctico

|  |  |
| --- | --- |
| Grupo 54 | Marcos Carnerero Blanco 22M039 |
| Grado GMI | Rodrigo Elola Torrijos 22M088 |
|  | Miguel López García 22M108 |

ÍNDICE

[ÍNDICE 1](#_Toc182827403)

[CONTENIDOS 1](#_Toc182827404)

[1. INTRODUCCIÓN 1](#_Toc182827405)

[2. GRAMÁTICA Y AFD 1](#_Toc182827406)

[3. TABLA LL 3](#_Toc182827407)

[4. ANEXO 3](#_Toc182827408)

[4.1. Casos Correctos 3](#_Toc182827409)

[4.2. Casos Erróneos 3](#_Toc182827410)

CONTENIDOS

[Ecuación 1 Reacción estequiométrica de la magnetita 1](#_Toc67495211)

**Ilustraciones**

[1 Reacción con el amoniaco 2](file:///C:\yo%20y%20mis%20cosas\%23%23INSTI%23%23\Informe%20UCAV%20Ferrofluidos%20Marcos%20Carnerero%20Blanco.docx#_Toc67519056)

[2 Emulsión con el aceite 3](file:///C:\yo%20y%20mis%20cosas\%23%23INSTI%23%23\Informe%20UCAV%20Ferrofluidos%20Marcos%20Carnerero%20Blanco.docx#_Toc67519057)

[3 Producto final 3](file:///C:\yo%20y%20mis%20cosas\%23%23INSTI%23%23\Informe%20UCAV%20Ferrofluidos%20Marcos%20Carnerero%20Blanco.docx#_Toc67519058)

[4 Reacción a campos magnéticos de la muestra 3](#_Toc67519059)

[5 Reacción a campos magnéticos del sobrenadante 4](#_Toc67519060)

1. INTRODUCCIÓN

Continuamos con el diseño de nuestro compilador de JS-- con las especificaciones propuestas para nuestro grupo 54:

* Sentencias: Sentencia de selección múltiple (switch-case)
* Operadores especiales: Pre-auto-incremento (++ como prefijo)
* Técnicas de Análisis Sintáctico: Ascendente LR
* Comentarios: Comentario de bloque (/\* \*/)
* Cadenas: Con comillas dobles (" ")

En esta segunda entrega hemos implementado el Analizador Sintáctico y, al igual que en la primera entrega, continuamos utilizando Java como lenguaje de programación para evitar posibles confusiones y/o errores además de las razones mencionadas en dicha entrega.

1. GRAMÁTICA Y AFD

El primer paso para crear el analizador sintáctico es la creación de una gramática junto con su autómata correspondiente.

Como terminales tenemos: { boolean break case function if input int output return string switch var void ent cad ++ id = , ; : ( ) { } \* ! > default true false eof }

Por parte de los no terminales: { P F F1 F2 F3 A K C B B1 T W S S1 L Q X E R U V V1 }

Nuestro axioma será P, y nuestras producciones quedan reflejadas en la siguiente tabla:

|  |
| --- |
| P’🡪P |
| P🡪BP | FP | eof |
| F🡪function F1 F2 F3 {C} |
| F1🡪T | void |
| F2🡪id |
| F3🡪(A) |
| A🡪T id K | void |
| K🡪T id K | λ |
| C🡪BC | λ |
| B🡪if B1 S | var T id; | S |
| B🡪switch B1 {W} |
| B1🡪(E) //Facilita el autómata |
| W🡪case ent: CW | default: C | λ |
| S🡪output E; | break; | id S1 |
| S🡪input id; | return X; |
| S1🡪(L); | =E; |
| L🡪EQ | λ |
| Q🡪,EQ | λ |
| X🡪E | λ |
| E🡪E>R | R |
| R🡪R\*U | U |
| U🡪!U | ++U | V |
| V🡪(E) | id V1 | ent | cad | true | false |
| V1🡪(L) | λ //La mantenemos por compartimentación |

Tabla 1: Gramática

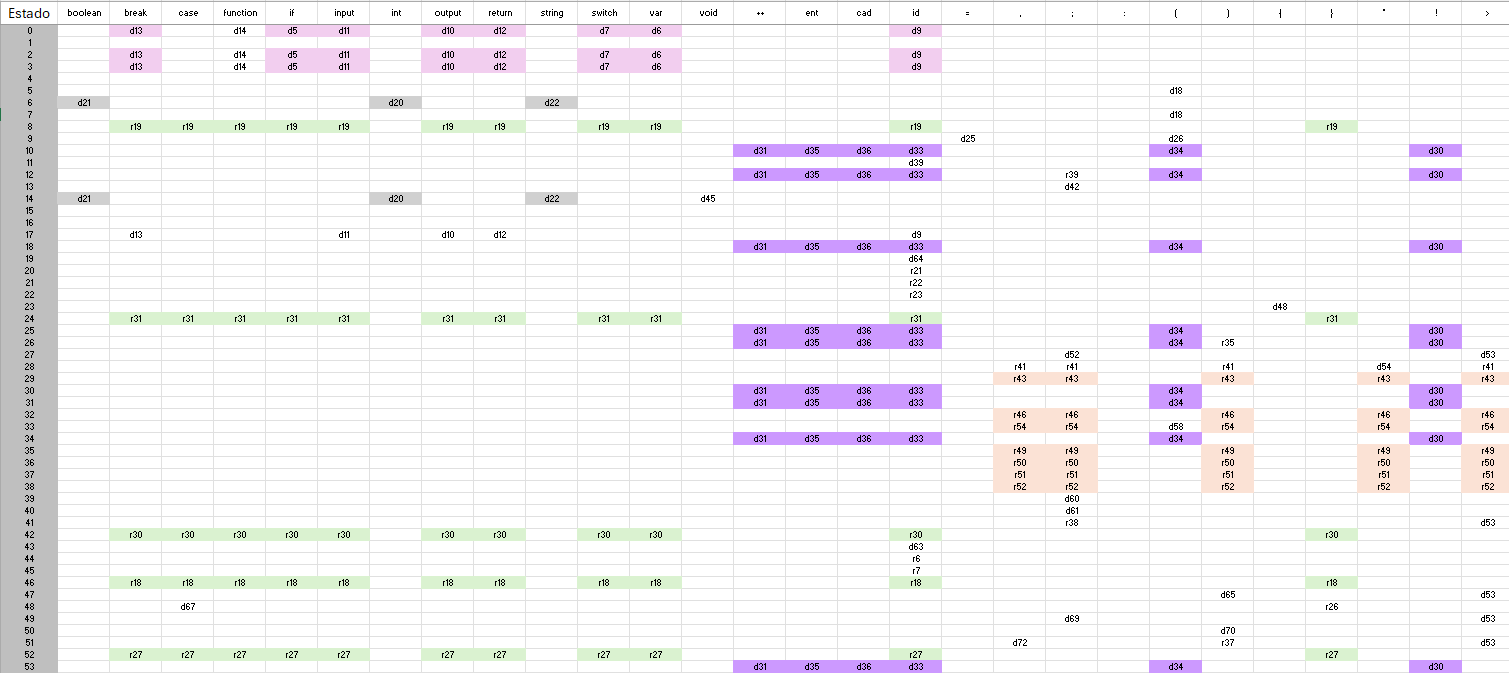
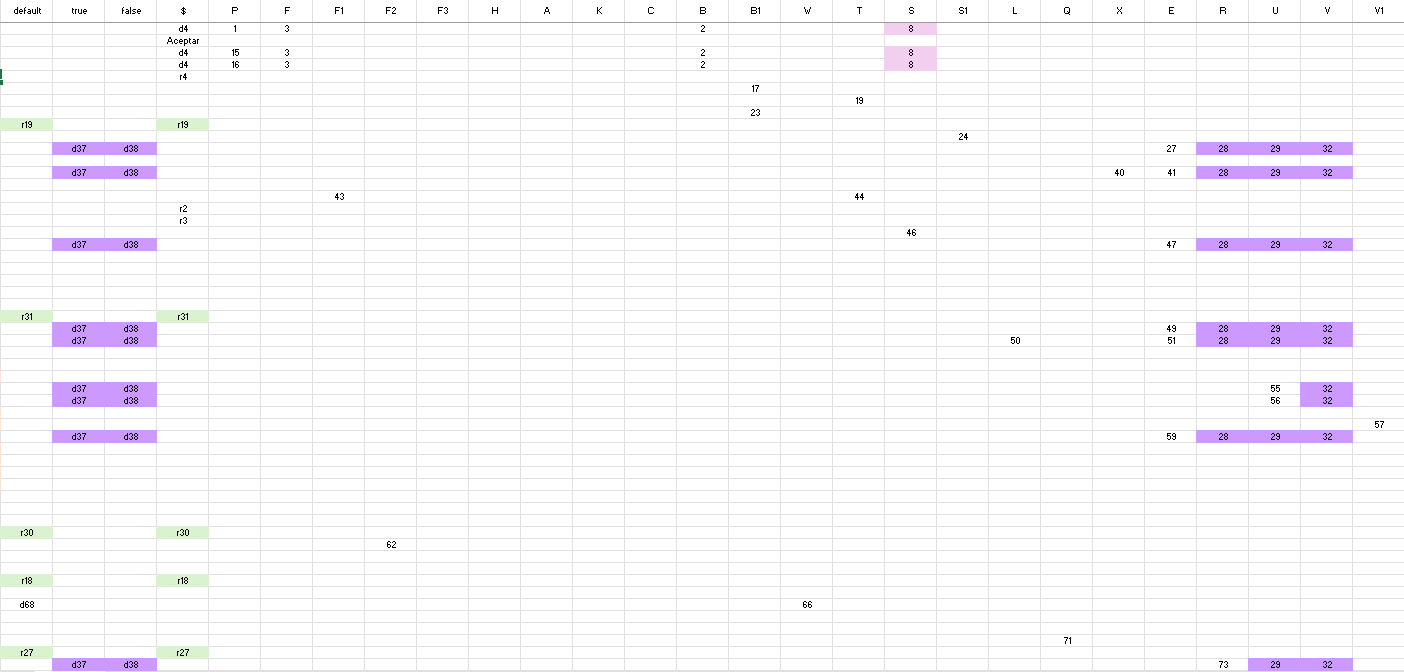
Realizamos además la comprobación de conflictos:

|  |  |
| --- | --- |
| FOLLOW | //Omitimos las comas de separación para evitar confusiones |
| E | > ; ) , |
| V1 | > ; ) \* , |
| K | ) |
| L | ) |
| X | ; |
| C | { case default |

También comprobamos las reducciones:

|  |  |
| --- | --- |
| FOLLOW | //Al igual que en la tabla anterior omitimos las comas |
| P | $ |
| F | $ function if var switch id output break input return |
| B | $ function if var switch id output break input return case default } |
| F1 | Id |
| F2 | ( |
| F3 | { |
| A | ) |
| B1 | { output break id input return |
| W | } |
| T | Id |
| S | $ function if var switch id output break input return case default } |
| S1 | $ function if var switch id output break input return case default } |
| Q | ) |
| R | \* > ; ) , |
| U | \* > ; ) , |
| V | \* > ; ) , |

Para demostrar que la gramática utilizada es válida simplemente comprobamos la tabla parser correspondiente (debido a la longitud de la tabla se mostrará por partes):



Calendario

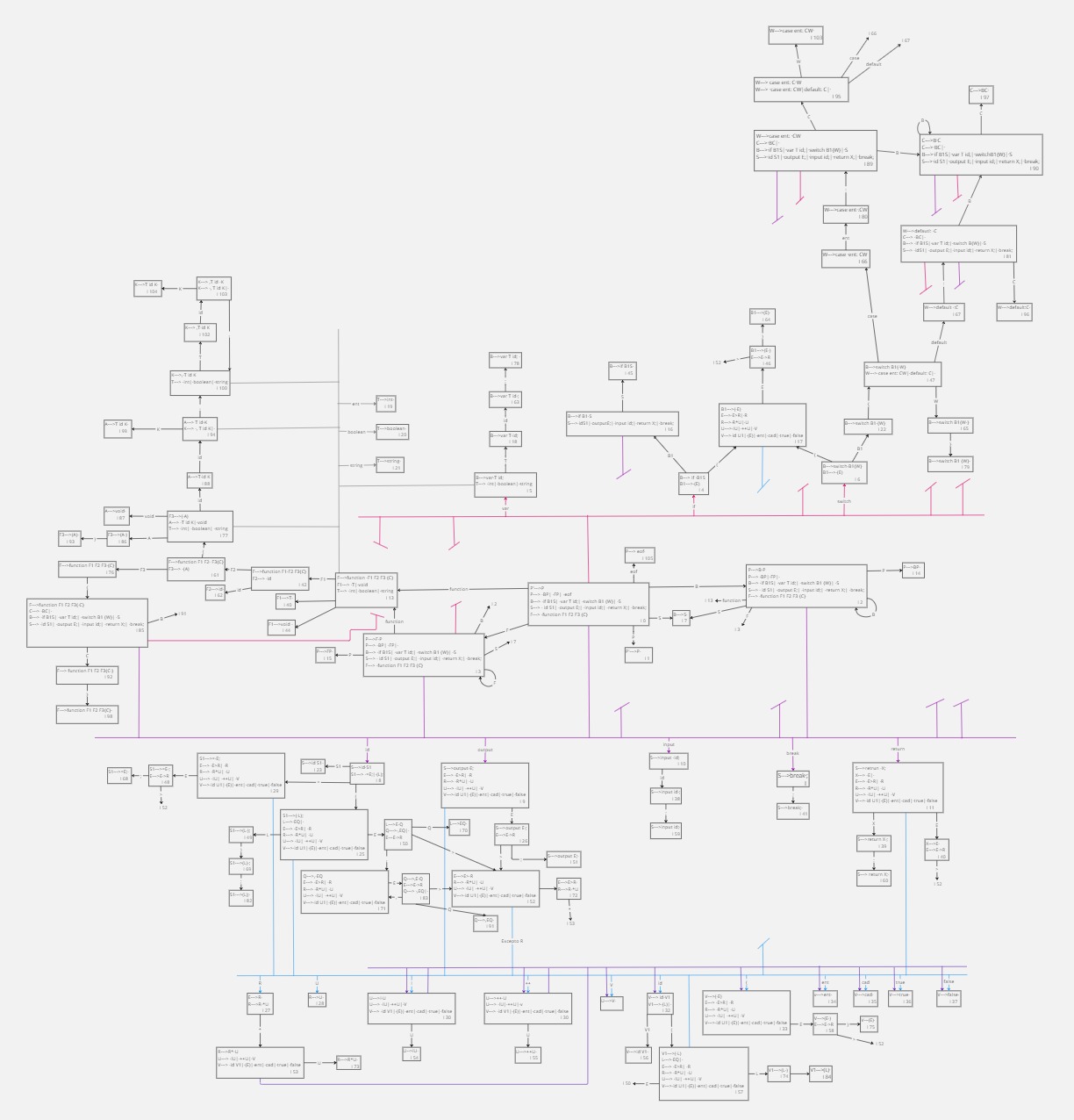
Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Como podemos comprobar, no hay ninguna celda que contenga más de una acción, por lo que queda demostrado que la gramática usada es la correcta.

Una vez comprobada la validez de la gramática, creamos el AFD asociado a ella:

1. TABLA LL

1. ANEXO
   1. Casos Correctos

Código 1:

function void signo(int a, boolean b, string c){

if(a>0) output("Positivo");

if(!(a>0)) output("Negativo");

}

var int c;

signo(c, e, 7 \* 54);

switch (a) {}

Árbol generado:

Gráfico

Descripción generada automáticamente con confianza baja

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

Código 2:

function void nada(void) {}

b = a > 10;

c = !c \* ++b > (mult() \* true);

var int prueba /\*Aquí cabe un comentario\*/;

Árbol generado:

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Código 3:

var int n;

function int mult(void){

output("Introduce 0 o 1 para hacer 7\*4 o 3\*9");

input n;

switch(n){

case 0: input n;

case 1: return 3\*9; break;

default: return 0;

}

}

Árbol generado:

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Gráfico de dispersión

Descripción generada automáticamente

* 1. Casos Erróneos

Código 1:

var boolean b;

var int c = 3;

signo(c, e, 7 \* 54);

switch (a) {}

Errores detectados:

ParserException: Error en la linea 2 no se esperaba ASIG

at Parser.parse(Parser.java:28)

at Main.main(Main.java:10)

Código 2:

b = a > 10;

c = !c \* ++b > (mult() true);

if(b) output("Preincremento");

var int prueba /\*Aquí cabe un comentario\*/;

var int prueba3;

Errores detectados:

ParserException: Error en la linea 3 no se esperaba TRUE

at Parser.parse(Parser.java:28)

at Main.main(Main.java:10)

Código 3:

output("Introduce tu pais de nacimiento");

input pais;

function int mult(void){

output("Introduce 0 o 1 para hacer 7\*4 o 3\*9");

input n;

switch(n){

case 0: input n;

case 1: return 3\*9; break;

default: return 0;

}

Errores detectados:

ParserException: Error en la linea 12 no se esperaba EOF

at Parser.parse(Parser.java:28)

at Main.main(Main.java:10)