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. Esta gramática es la siguiente:

|  |
| --- |
| 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 |

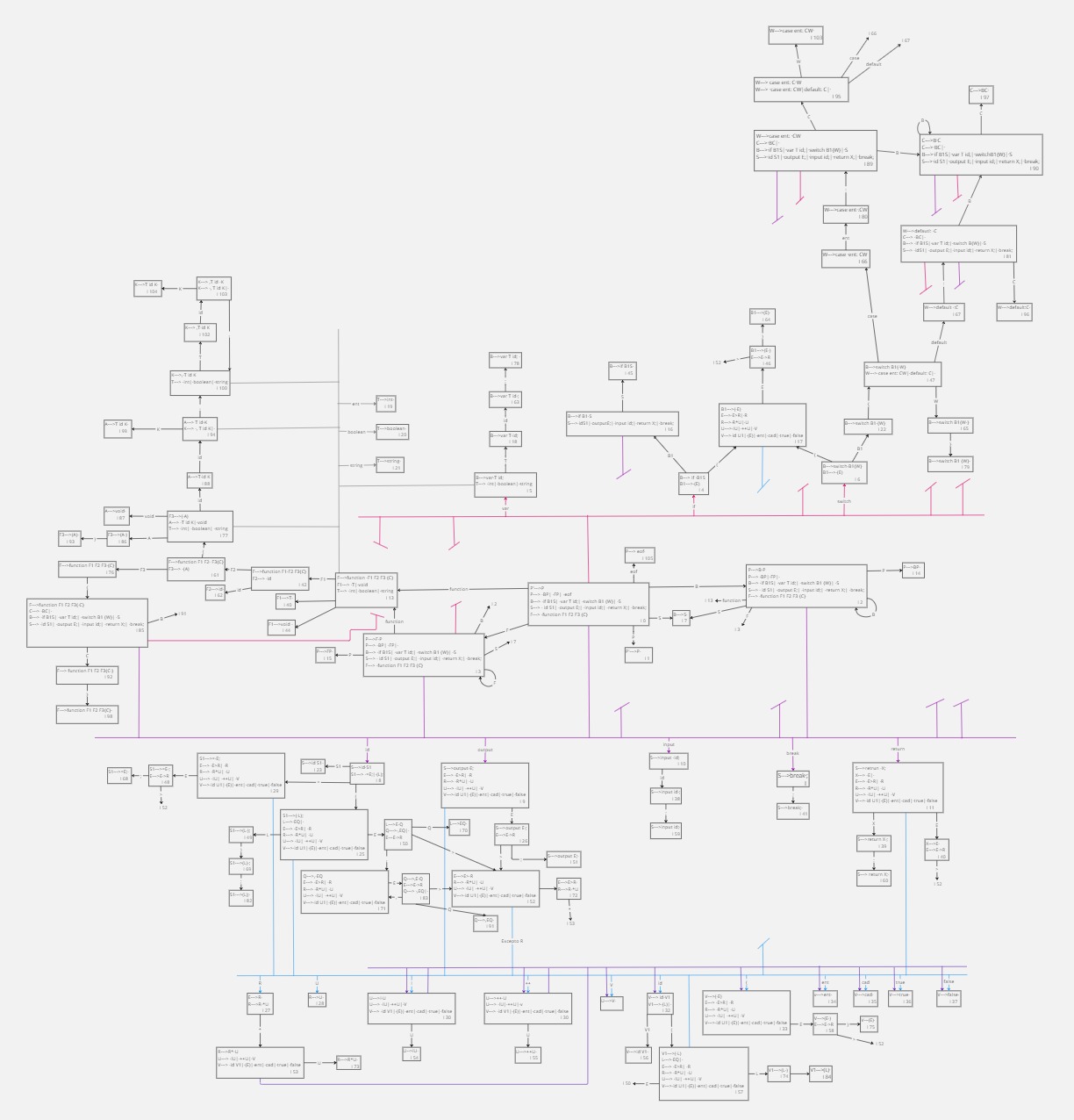
Tabla 1: Gramática

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 | \* > ; ) , |

Cuyo AFD es:

1. TABLA LL

1. ANEXO
   1. Casos Correctos
   2. Casos Erróneos