|  |
| --- |
| Eduardo Gil Alba  20-12-2024 |

|  |
| --- |
| GRUPO 63 |
| Procesador de Lenguajes: Javascript JS-- |
| Procesadores de Lenguajes |

Tabla de contenido

[1. Propósito 1](#_Toc187624760)

[2. Diseño de la Estructura 1](#_Toc187624761)

[3. Analizador Léxico 2](#_Toc187624762)

[3.1. Tokens 2](#_Toc187624763)

[3.2. Gramática 3](#_Toc187624764)

[3.3. Autómata 4](#_Toc187624765)

[3.3.1. Acciones Semánticas 5](#_Toc187624766)

[4. Analizador Sintáctico 6](#_Toc187624767)

[4.1. Gramática 6](#_Toc187624768)

[4.2. Tabla Accion de Analizador Sintáctico LR(1) 7](#_Toc187624769)

[Análisis de Conflictos 12](#_Toc187624770)

[4.3. Tabla GOTO de Analizador Sintáctico LR(1) 13](#_Toc187624771)

[5. Analizador Semántico 16](#_Toc187624772)

[4.1 Esquema de Traducción 16](#_Toc187624773)

[4.1.1 Flujo de Programa 16](#_Toc187624774)

[4.1.2 Sentencias Compuestas 16](#_Toc187624775)

[4.1.2.1 Definicion de tipos 17](#_Toc187624776)

[4.1.3 Funciones 18](#_Toc187624777)

[4.1.3.1 Parametros de Función 18](#_Toc187624778)

[4.1.4 Sentencias Simples 19](#_Toc187624779)

[4.1.4.1 Llamadas a funcion como Sentencia 20](#_Toc187624780)

[4.1.5 Expresiones 20](#_Toc187624781)

[6. Tabla de Símbolos 22](#_Toc187624782)

[6.1. Estructura y organización 22](#_Toc187624783)

[7. Gestor de Errores 23](#_Toc187624784)

[6.1 Tipos de Errores 23](#_Toc187624785)

[6.1.1 Errores Léxicos 23](#_Toc187624786)

[6.1.2 Errores Sintácticos 23](#_Toc187624787)

[6.1.3 Errores Semánticos 24](#_Toc187624788)

[6.2 Tolerancia a Fallos 25](#_Toc187624789)

[ANEXO 26](#_Toc187624790)

[Casos de prueba 26](#_Toc187624791)

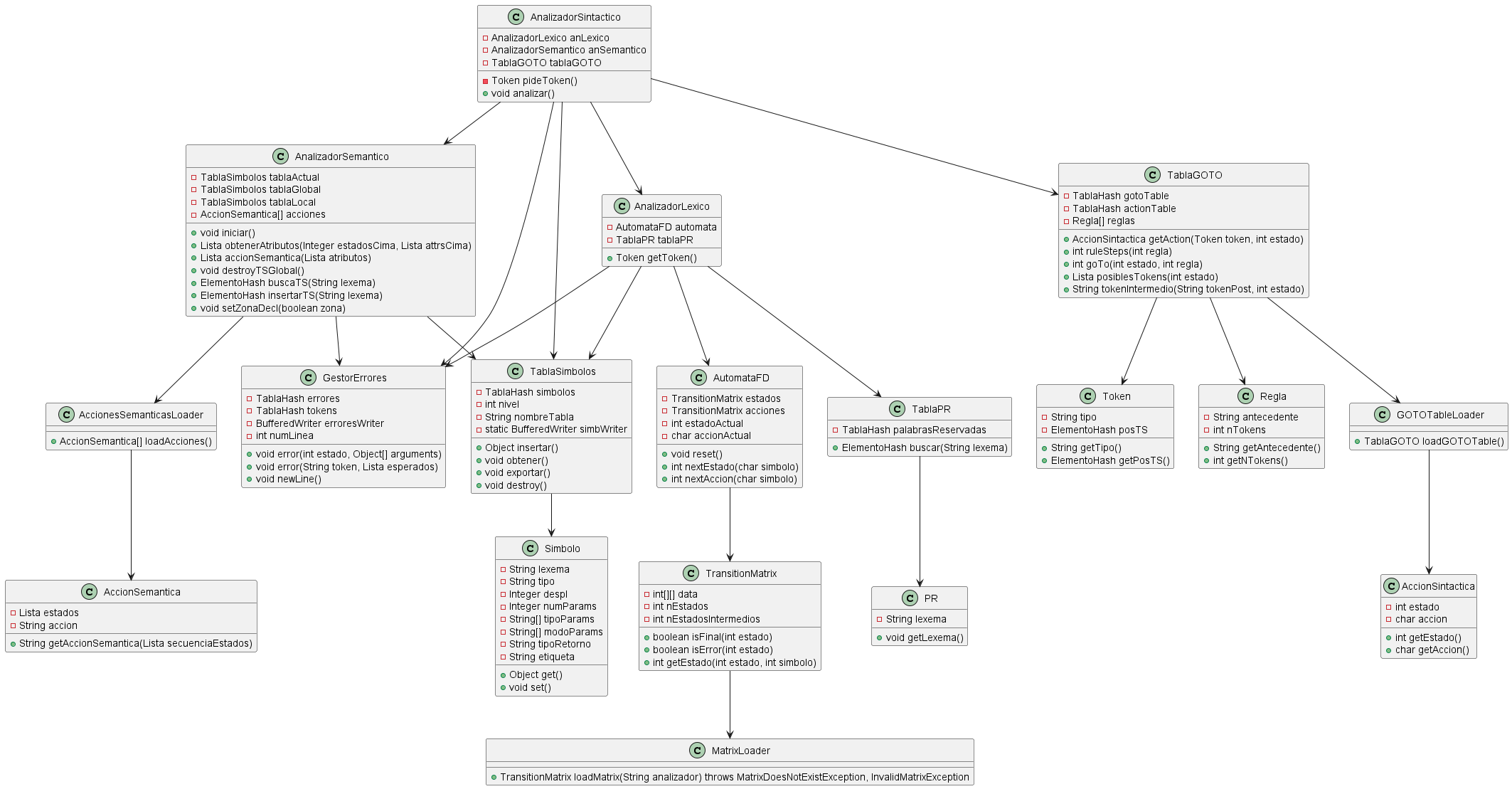
# Propósito

En este documento se detalla el diseño realizado en la implementación de un procesador de lenguajes para el lenguaje Javascript JS--. Para ello, se ha decidido utilizar como lenguaje de soporte Java, con un diseño orientado a objetos.   
Todo el material que se utiliza para ilustrar el diseño se encuentra en el proyecto en la carpeta *doc*. Para probar el procesador son necesarias las dependencias de Java para ejecutar el mandato: *java -jar practicajs-0.0.0.jar [CODIGO\_FUENTE]*. En el mismo directorio dónde se ejecuta el mandato se producen los ficheros de salida en formato .txt:

* *simbolos* : Tablas de símbolos generadas durante la ejecución
* *errores*: Errores contenidos en el fichero CODIGO\_FUENTE
* *tokens* : Secuencia de tokens generada por el analizador léxico
* *parse*: Secuencia de reglas que forman el arbol sintáctico

El diagrama de clases se muestra en el siguiente apartado.

# Diseño de la Estructura



# Analizador Léxico

Es el encargado de generar una secuencia de tokens a partir de la entrada y proporcionárselos al Analizador Sintáctico.

## Tokens

Cada uno de los elementos del lenguaje con significado propio. Estos son los tokens que se ha decidido que pueda generar:

|  |  |  |
| --- | --- | --- |
| <CADENA, lexema> **Cadenas de caracteres** | <ID, posTS>  **Identificadores** | <ENTERO, valor>  **Constantes enteras** |
| <DOBLEAND, -> **&&**  **Operador lógico** | <SUMA, -> **+**  **Operador aritmético** | <ASIG, -> **=**  **Operador asignación** |
| < ASIGAND , -> **&=**  **Operador y lógico** | < DOBLEEQ , -> **==**  **Operador relacional** | <COMA, -> **,**  **Coma** |
| < PARENTIZDA , -> **(**  **Apertura paréntesis** | < PARENTDCHA , ->  **)**  **Cierre paréntesis** | < LLAVEIZDA , -> **{**  **Apertura bloque** |
| < LLAVEDCHA , -> **}**  **Cierre bloque** | <PYC, -> **;**  **Punto y coma** |  |

Las palabras reservadas estarán indexadas en una Tabla Hash y al generar el token no requiere de atributo. Los identificadores devuelven su posición en la Tabla de Símbolos.

|  |  |  |
| --- | --- | --- |
| <IF, ->  **if** | <ELSE, ->  **else** | <FUNCTION, ->  **function** |
| <VAR, ->  **var** | <STRING, ->  **string** | <INT, ->  **int** |
| <VOID, ->  **void** | <RETURN, ->  **return** | <BOOLEAN, -> **boolean** |
| <INPUT, ->  **input** | <OUTPUT, ->  **output** |  |

Los tokens que se visualizan en la tabla, aparecerán de esa misma forma en el fichero generado ‘tokens’. Este analizador proporciona los tokens al sintáctico, tal que Token <String, Object>. El primer parámetro corresponde al tipo del token y el segundo parámetro, su atributo. Ejemplos:

<ID, posTS> será Token<”ID”, Entry<String, Simbolo>>; <ENTERO, 5> será Token<”ENTERO”, 5>; <CADENA, “ej3mpl%o”> será Token<”CADENA”, “ej3mpl%o”>; <IF, -> será Token <”IF”, null>

## Gramática

Texto, Carta

Descripción generada automáticamente

La gramática que emplea el analizador léxico es una gramática regular, de tipo 3 según establece la jerarquía de Chomsky. Esta gramática establece los patrones para encontrar los tokens en un codigo fuente según las especificaciones del lenguaje.

A partir de esta gramática, se genera un autómata que determina el comportamiento de este analizador. La columna de la izquierda indica el estado del autómata.

Se puede observar que: la trasición al estado 1, corresponde al procesado de identificadores válidos; la transición al estado 2, al procesado de números enteros; la transición al estado 3, al procesado de las cadenas de caracteres; la transicion al estado 4, comprueba si se trata del operador lógico (&&) o el operador de asignación lógico (&=); la transición al estado 5, comprueba si se trata del operador de asignación (=) o el operador relacional (==) ; las transiciones 6, 7 y 8, al procesado de los comentarios.

En los estados finales, el analizador generará los tokens. También se realizan las comprobación de las restricciones. Los comentarios y delimitadores no conducen a estados finales, por lo que, no generarán token.

Cualquier transición no prevista, se considera un error, se informa al Gestor de Errores que presentará un mensaje descriptivo al usuario.

## Autómata

Diagrama, Esquemático

Descripción generada automáticamente

El automata generado por la gramática es finito determinista (AFD), esto quiere decir que, para una misma entrada, siempre se produce el mismo resultado. La notación en color azul, denota las acciones semánticas que se realizan en las transiciones.

En la implementación, existe una clase AutomataFD, que posee el Analizador Lexico, contiene toda la información del automata: el estado actual, estados finales, dos matrices de transición, TransitionMatrix. Cada una de las matrices se cargan de un fichero, que contiene una cabecera seguido de los elementos de la matriz tabulados. Las columnas de la matriz que corresponden a los caracteres de entrada, van numeradas desde el 0 al 127 ( codificación ASCII). Una de las matrices determina el estado siguiente y la otra la acción siguiente.

Esta clase contiene métodos para conocer si se encuentra en un estado final o en un estado de error.

### Acciones Semánticas

Las acciones semánticas tienen una notación diferente en el diagrama respecto al código, por comodidad. En la implementación, una vez el analizador conoce la acción, entonces se lleva a cabo en una sentencia switch.

|  |  |  |
| --- | --- | --- |
| Acción | Acción en el  código | Descripción |
| A | A | CONCAT  LEE |
| B | B | num := valor(d) |
| C | C | num := num\*10 + valor(d) |
| L | L | LEE |
| D | D | contador := 0 |
| E | E | CONCAT  contador := contador + 1  LEE |
| G1 | G | p := BUSCATPR(lex)  if (p != NULL) { PR := TPR(p); GENTOKEN(PR, -) }  else {  p := BUSCATS(lex)  if (p == NULL) { p := INSERTATS(lex) }  GENTOKEN(ID, p)  } |
| G2 | H | if (num > 32767) { ERROR(36, num) }  else GENTOKEN(ENTERO, num) |
| G3 | I | if (contador > 64) { ERROR(37, lex) }  else { LEE ; GENTOKEN(CADENA, lex) } |
| G4a | J | LEE  GENTOKEN(DOBLEAND, -) |
| G4b | K | LEE  GENTOKEN(ASIGAND, -) |
| G5a | M | LEE  GENTOKEN(DOBLEEQ, -) |
| G5b | N | LEE  GENTOKEN(ASIG, -) |
| G6 | O | LEE  GENTOKEN(SUMA, -) |
| G7 | P | LEE  GENTOKEN(COMA, -) |
| G8 | Q | LEE  GENTOKEN(PYC, -) |
| G9 | R | LEE  GENTOKEN(PARENTIZDA, -) |
| G10 | S | LEE  GENTOKEN(PARENTDCHA, -) |
| G11 | T | LEE  GENTOKEN(LLAVEIZDA, -) |
| G12 | U | LEE  GENTOKEN(LLAVEDCHA, -) |

# Analizador Sintáctico

Encargado de evaluar si la sentencia es válida para el lenguaje generado por la gramática a partir de los tokens recibidos.

## Gramática

La gramática contiene la sintaxis del lenguaje. En otras palabras, el conjunto de sentencias válidas para el lenguaje. La gramática debe ser de tipo 2, independiente del contexto, lo más general posible, para cubrir todos los posibles casos de uso y no debe ser ambigua, es decir, no puede dar lugar a dos o más arboles sintácticos diferentes.

Tabla

Descripción generada automáticamente

## Tabla Accion de Analizador Sintáctico LR(1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **E** | **ACCION** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| id | + | == | && | &= | = | ( | ) | { | } | ; | , | function | var | return | if | else | int | boolean | string | void | ent | cad | bool | output | input | $ |
| 0 | d9 |  |  |  |  |  |  |  |  |  |  |  | d8 | d5 | d12 | d6 |  |  |  |  |  |  |  |  | d10 | d11 | r3 |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | a |
| 2 | d9 |  |  |  |  |  |  |  |  |  |  |  | d8 | d5 | d12 | d6 |  |  |  |  |  |  |  |  | d10 | d11 | r3 |
| 3 | d9 |  |  |  |  |  |  |  |  |  |  |  | d8 | d5 | d12 | d6 |  |  |  |  |  |  |  |  | d10 | d11 | r3 |
| 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d16 | d17 | d18 |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  | d19 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | r7 |  |  |  |  |  |  |  |  | r7 |  |  | r7 | r7 | r7 | r7 |  |  |  |  |  |  |  |  | r7 | r7 | r7 |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d16 | d17 | d18 | d23 |  |  |  |  |  |  |
| 9 |  |  |  |  | d25 | d24 | d26 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | d32 |  |  |  |  |  | d31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 11 | d36 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 | d32 |  |  |  |  |  | d31 |  |  |  | r36 |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | r1 |
| 14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | r2 |
| 15 | d39 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 16 | r22 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 17 | r23 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 18 | r24 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **E** | **ACCION** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| id | + | == | && | &= | = | ( | ) | { | } | ; | , | function | var | return | if | else | int | boolean | string | void | ent | cad | bool | output | input | $ |
| 19 | d32 |  |  |  |  |  | d31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 20 | d42 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 21 | r9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 22 | r14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 23 | r15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 24 | d32 |  |  |  |  |  | d31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 25 | d32 |  |  |  |  |  | d31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 26 | d32 |  |  |  |  |  | d31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 27 |  |  |  | d48 |  |  |  |  |  |  | d47 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 28 |  |  | d49 | r38 |  |  |  | r38 |  |  | r38 | r38 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 29 |  | d50 | r40 | r40 |  |  |  | r40 |  |  | r40 | r40 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 30 |  | r42 | r42 | r42 |  |  |  | r42 |  |  | r42 | r42 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 31 | d32 |  |  |  |  |  | d31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 32 |  | r43 | r43 | r43 |  |  | d52 | r43 |  |  | r43 | r43 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 33 |  | r45 | r45 | r45 |  |  |  | r45 |  |  | r45 | r45 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 34 |  | r47 | r47 | r47 |  |  |  | r47 |  |  | r47 | r47 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 35 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 36 |  |  |  |  |  |  |  |  |  |  | d53 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 37 |  |  |  |  |  |  |  |  |  |  | d54 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **E** | **ACCION** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| id | + | == | && | &= | = | ( | ) | { | } | ; | , | function | var | return | if | else | int | boolean | string | void | ent | cad | bool | output | input | $ |
| 38 |  |  |  | d48 |  |  |  |  |  |  | r35 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 39 |  |  |  |  |  |  |  |  |  |  | d55 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 40 |  |  |  | d48 |  |  |  | d56 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 41 |  |  |  |  |  |  | d58 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 42 | r10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 43 |  |  |  | d48 |  |  |  |  |  |  | d59 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 44 |  |  |  | d48 |  |  |  |  |  |  | d60 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 45 |  |  |  |  |  |  |  | d61 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 46 |  |  |  | d48 |  |  |  | r34 |  |  |  | d63 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 47 | r28 |  |  |  |  |  |  |  |  | r28 |  |  | r28 | r28 | r28 | r28 |  |  |  |  |  |  |  |  | r28 | r28 | r28 |
| 48 | d32 |  |  |  |  |  | d31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 49 | d32 |  |  |  |  |  | d31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 50 | d32 |  |  |  |  |  | d31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 51 |  |  |  | d48 |  |  |  | d67 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 52 | d32 |  |  |  |  |  | d31 | r32 |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 53 | r29 |  |  |  |  |  |  |  |  | r29 |  |  | r29 | r29 | r29 | r29 |  |  |  |  |  |  |  |  | r29 | r29 | r29 |
| 54 | r30 |  |  |  |  |  |  |  |  | r30 |  |  | r30 | r30 | r30 | r30 |  |  |  |  |  |  |  |  | r30 | r30 | r30 |
| 55 | r4 |  |  |  |  |  |  |  |  | r4 |  |  | r4 | r4 | r4 | r4 |  |  |  |  |  |  |  |  | r4 | r4 | r4 |
| 56 | d9 |  |  |  |  |  |  |  | d69 |  |  |  |  |  | d12 |  |  |  |  |  |  |  |  |  | d10 | d11 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **E** | **ACCION** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| id | + | == | && | &= | = | ( | ) | { | } | ; | , | function | var | return | if | else | int | boolean | string | void | ent | cad | bool | output | input | $ |
| 57 |  |  |  |  |  |  |  |  | d71 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 58 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d16 | d17 | d18 | d74 |  |  |  |  |  |  |
| 59 | r25 |  |  |  |  |  |  |  |  | r25 |  |  | r25 | r25 | r25 | r25 |  |  |  |  |  |  |  |  | r25 | r25 | r25 |
| 60 | r26 |  |  |  |  |  |  |  |  | r26 |  |  | r26 | r26 | r26 | r26 |  |  |  |  |  |  |  |  | r26 | r26 | r26 |
| 61 |  |  |  |  |  |  |  |  |  |  | d75 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 62 |  |  |  |  |  |  |  | r31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 63 | d32 |  |  |  |  |  | d31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d33 | d34 | d35 |  |  |  |
| 64 |  |  | d49 | r37 |  |  |  | r37 |  |  | r37 | r37 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 65 |  |  | r39 | r39 |  |  |  | r39 |  |  | r39 | r39 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 66 |  | r41 | r41 | r41 |  |  |  | r41 |  |  | r41 | r41 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 67 |  | r44 | r44 | r44 |  |  |  | r44 |  |  | r44 | r44 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 68 |  |  |  |  |  |  |  | d77 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 69 | d9 |  |  |  |  |  |  |  |  | r13 |  |  |  | d5 | d12 | d6 |  |  |  |  |  |  |  |  | d10 | d11 |  |
| 70 | r6 |  |  |  |  |  |  |  |  | r6 |  |  | r6 | r6 | r6 | r6 |  |  |  |  |  |  |  |  | r6 | r6 | r6 |
| 71 | d9 |  |  |  |  |  |  |  |  | r13 |  |  |  | d5 | d12 | d6 |  |  |  |  |  |  |  |  | d10 | d11 |  |
| 72 |  |  |  |  |  |  |  | d81 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 73 | d82 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 74 |  |  |  |  |  |  |  | r17 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 75 | r27 |  |  |  |  |  |  |  |  | r27 |  |  | r27 | r27 | r27 | r27 |  |  |  |  |  |  |  |  | r27 | r27 | r27 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **E** | **ACCION** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| id | + | == | && | &= | = | ( | ) | { | } | ; | , | function | var | return | if | else | int | boolean | string | void | ent | cad | bool | output | input | $ |
| 76 |  |  |  | d48 |  |  |  | r34 |  |  |  | d63 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 77 |  | r46 | r46 | r46 |  |  |  | r46 |  |  | r46 | r46 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 78 |  |  |  |  |  |  |  |  |  | d84 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 79 | d9 |  |  |  |  |  |  |  |  | r13 |  |  |  | d5 | d12 | d6 |  |  |  |  |  |  |  |  | d10 | d11 |  |
| 80 |  |  |  |  |  |  |  |  |  | d86 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 81 |  |  |  |  |  |  |  |  | r11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 82 |  |  |  |  |  |  |  | r19 |  |  |  | d88 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 83 |  |  |  |  |  |  |  | r33 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 84 | r21 |  |  |  |  |  |  |  |  | r21 |  |  | r21 | r21 | r21 | r21 | d90 |  |  |  |  |  |  |  | r21 | r21 | r21 |
| 85 |  |  |  |  |  |  |  |  |  | r12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 86 | r8 |  |  |  |  |  |  |  |  |  |  |  | r8 | r8 | r8 | r8 |  |  |  |  |  |  |  |  | r8 | r8 | r8 |
| 87 |  |  |  |  |  |  |  | r16 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 88 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | d16 | d17 | d18 |  |  |  |  |  |  |  |
| 89 | r5 |  |  |  |  |  |  |  |  | r5 |  |  | r5 | r5 | r5 | r5 |  |  |  |  |  |  |  |  | r5 | r5 | r5 |
| 90 |  |  |  |  |  |  |  |  | d92 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 91 | d93 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 92 | d9 |  |  |  |  |  |  |  |  | r13 |  |  |  | d5 | d12 | d6 |  |  |  |  |  |  |  |  | d10 | d11 |  |
| 93 |  |  |  |  |  |  |  | r19 |  |  |  | d88 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 94 |  |  |  |  |  |  |  |  |  | d96 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **E** | **ACCION** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| id | + | == | && | &= | = | ( | ) | { | } | ; | , | function | var | return | if | else | int | boolean | string | void | ent | cad | bool | output | input | $ |
| 95 |  |  |  |  |  |  |  | r18 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 96 | r20 |  |  |  |  |  |  |  |  | r20 |  |  | r20 | r20 | r20 | r20 |  |  |  |  |  |  |  |  | r20 | r20 | r20 |

El analizador sintáctico, es de tipo Ascendente, es decir, el árbol generado se construye desde las hojas. En concreto, necesitará un token, en cada iteración. Para su implementación, se ha construido la tabla LR que determina la acción a realizar en un determinado estado, compuesta por los tokens en cada columna y los estados en las filas. Las acciones definidas pueden ser reducir por la Regla i (ri), desplazar el token a la pila y apila estado m (d m), aceptar o error. De tal forma que, las acciones de reducción generarán un “parse”, una secuencia de reglas, que determina la construcción del árbol sintáctico, sin necesidad de una estructura para ello. Existirán dos pilas, una de ellas en la que se iran introduciendo, pares de símbolos y estados y otra, los atributos calculados por el Analizador Semántico. En las acciones de reducción por regla i, desapilaremos 2\*m elementos de ambas pilas, siendo m el número de símbolos de la regla i. Con los atributos desapilados, el Analizador Semántico calculará los siguientes. En cada iteración, el estado de la cima de la pila y el token proporcionado por el léxico determina la siguiente acción. Se ha decidido eliminar token <eof> y la regla V→bool, por lo que, la regla P→eof, se sustituye por P→λ y los estado 4 y 35 desaparecen.

#### Análisis de Conflictos

Existe un conflicto, si en un estado para un determinado token, existen dos o más acciones. No se podría determinar cuál de las acciones se debe ejecutar. Los conflictos que se pudieran producir serían de tipo reducción-reducción o reducción-desplazamiento. Dicho de otro modo, en el primer caso, que hubiese dos o más acciones de reducción para un mismo token y en el segundo, una o más acciones de reducción y una o más acciones de desplazar.

Podrían haberse presentado conflictos de reducción-desplazamiento, en 16 estados. Los estados 12, 26, 28, 29, 38, 46, 52, 65, 69, 71, 76, 79, 82, 84, 92 y 93. Se observa que, cada columna, token, en ese estado, fila, tiene una sola acción. Por lo tanto, está gramática es válida y la tabla está lista para su implementación.

## Tabla GOTO de Analizador Sintáctico LR(1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **E** | **GOTO** | | | | | | | | | | | | | | | | | | | |
| P | B | F | T | E | C | I | S | F1 | F2 | F3 | H | A | K | L | X | Q | R | U | V |
| 0 | 1 | 2 | 3 |  |  |  |  | 7 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | 13 | 2 | 3 |  |  |  |  | 7 |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | 14 | 2 | 3 |  |  |  |  | 7 |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  | 15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  | 22 |  |  |  |  | 20 |  |  | 21 |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  | 27 |  |  |  |  |  |  |  |  |  |  |  |  | 28 | 29 | 30 |
| 11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 |  |  |  |  | 38 |  |  |  |  |  |  |  |  |  |  | 37 |  | 28 | 29 | 30 |
| 13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 16 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 17 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 18 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 19 |  |  |  |  | 40 |  |  |  |  |  |  |  |  |  |  |  |  | 28 | 29 | 30 |
| 20 |  |  |  |  |  |  |  |  |  | 41 |  |  |  |  |  |  |  |  |  |  |
| 21 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 22 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 23 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 24 |  |  |  |  | 43 |  |  |  |  |  |  |  |  |  |  |  |  | 28 | 29 | 30 |
| 25 |  |  |  |  | 44 |  |  |  |  |  |  |  |  |  |  |  |  | 28 | 29 | 30 |
| 26 |  |  |  |  | 46 |  |  |  |  |  |  |  |  |  | 45 |  |  | 28 | 29 | 30 |
| 27 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 28 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 29 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **E** | **GOTO** | | | | | | | | | | | | | | | | | | | |
| P | B | F | T | E | C | I | S | F1 | F2 | F3 | H | A | K | L | X | Q | R | U | V |
| 30 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 31 |  |  |  |  | 51 |  |  |  |  |  |  |  |  |  |  |  |  | 28 | 29 | 30 |
| 32 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 33 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 34 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 35 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 36 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 37 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 38 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 39 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 40 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 41 |  |  |  |  |  |  |  |  |  |  | 57 |  |  |  |  |  |  |  |  |  |
| 42 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 43 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 44 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 45 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 46 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 62 |  |  |  |
| 47 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 48 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 64 | 29 | 30 |
| 49 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 65 | 30 |
| 50 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 66 |
| 51 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 52 |  |  |  |  | 46 |  |  |  |  |  |  |  |  |  | 68 |  |  | 28 | 29 | 30 |
| 53 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 54 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 55 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 56 |  |  |  |  |  |  |  | 70 |  |  |  |  |  |  |  |  |  |  |  |  |
| 57 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 58 |  |  |  | 73 |  |  |  |  |  |  |  |  | 72 |  |  |  |  |  |  |  |
| 59 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 60 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 61 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 62 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 63 |  |  |  |  | 76 |  |  |  |  |  |  |  |  |  |  |  |  | 28 | 29 | 30 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **E** | **GOTO** | | | | | | | | | | | | | | | | | | | |
| P | B | F | T | E | C | I | S | F1 | F2 | F3 | H | A | K | L | X | Q | R | U | V |
| 63 |  |  |  |  | 76 |  |  |  |  |  |  |  |  |  |  |  |  | 28 | 29 | 30 |
| 64 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 65 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 66 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 67 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 68 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 69 |  | 79 |  |  |  | 78 |  | 7 |  |  |  |  |  |  |  |  |  |  |  |  |
| 70 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 71 |  | 79 |  |  |  | 80 |  | 7 |  |  |  |  |  |  |  |  |  |  |  |  |
| 72 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 73 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 74 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 75 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 76 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 83 |  |  |  |
| 77 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 78 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 79 |  | 79 |  |  |  | 85 |  | 7 |  |  |  |  |  |  |  |  |  |  |  |  |
| 80 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 81 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 82 |  |  |  |  |  |  |  |  |  |  |  |  |  | 87 |  |  |  |  |  |  |
| 83 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 84 |  |  |  |  |  |  | 89 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 85 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 86 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 87 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 88 |  |  |  | 91 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 89 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 90 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 91 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 92 |  | 79 |  |  |  | 94 |  | 7 |  |  |  |  |  |  |  |  |  |  |  |  |
| 93 |  |  |  |  |  |  |  |  |  |  |  |  |  | 95 |  |  |  |  |  |  |
| 94 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 95 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 96 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# Analizador Semántico

En el diseño de este analizador es necesario definir una Traducción Dirigida por la Sintaxis para determinar las acciones semánticas que se ejecutan por cada regla. En este caso, se ha decidido su definición mediante un Esquema de Traducción (EdT)

## 4.1 Esquema de Traducción

### 4.1.1 Flujo de Programa

P' -> {TSG := CrearTS(),

TSA := TSG,

desplG := 0,

zonaDecl := false}

P {destruirTS(TSG)}

P -> B {if (B.tipoRet != vacio) {

Error('Las sentencias return solo pueden ir dentro de las funciones')

then tipo\_err

}

} P {}

P -> F P {}

P -> lambda {}

### 4.1.2 Sentencias Compuestas

B -> var {zonaDecl := true}

T id ; { insertaTipoTS(id.pos, T.tipo),

B.tipo :=

if (TSA = TSG) {

insertaDesplTS(id.pos, desplG)

desplG += T.ancho

then tipo\_ok

} else {

insertaDesplTS(id.pos, desplL)

desplL += T.ancho

then tipo\_ok

}

},

zonaDecl := false}

B -> if ( E ) S {B.tipo :=

if (E.tipo = ent) {

then S.tipo

} else {

Error('En las sentencias condicionales, se necesita una condicion logica, en vez de E.tipo')

then tipo\_err

},

B.tipoRet := S.tipoRet

}

B -> S {B.tipo := S.tipo, B.tipoRet := S.tipoRet}

B -> if ( E ) { C } I {B.tipo :=

if (E.tipo = ent) {

if (C.tipo = tipo\_err) {

then tipo\_err

} else if (I.tipo = tipo\_err) {

then tipo\_err

} else {

then tipo\_ok

} else {

Error('En las sentencias condicionales, se necesita una condicion logica, en vez de E.tipo')

then tipo\_err

},

B.tipoRet := if (C.tipoRet = vacio)

then I.tipoRet

else

then C.tipoRet

}

C -> B C1 {C.tipo :=

if (B.tipo = tipo\_err)

then tipo\_err

else

then C1.tipo

C.tipoRet :=

if (B.tipoRet = vacio)

then C1.tipoRet

else

then B.tipoRet

}

C -> lambda {C.tipo := tipo\_ok, C.tipoRet := vacio}

#### 4.1.2.1 Definicion de tipos

T -> int {T.tipo := ent, T.ancho := 2}

T -> boolean {T.tipo := log, T.ancho := 2}

T -> string {T.tipo := cad, T.ancho := 128}

H -> T {H.tipo := T.tipo}

H -> void {H.tipo := vacio}

### 4.1.3 Funciones

F -> function F1 {TSL := CrearTS(), TSA := TSL, desplL := 0}

F2 {insertaTipoRetTS(F2.pos, F1.tipo),

insertaEtiqTS(F2.pos, nuevaEtiqueta())}

F3 {insertaTipoParamsTS (F2.pos, F3.tipoParams),

insertaNParamsTS (F2.pos, F3.nParams),

insertaModoParamsTS(F2.pos, F3.modoParams)}

{ C } {F.tipo :=

if (C.tipo = tipo\_err) { /\* No repetir mensaje \*/

then tipo\_err

} else if (C.tipoRet = tipo\_err) {

then tipo\_err

} else if (C.tipoRet != F1.tipo && C.tipoRet != vacio) {

Error('La funcion debe devolver F1.tipo pero esta devolviendo C.tipoRet')

then tipo\_err

} else {

then funcion

},

destruirTS(TSL),

TSA := TSG

}

F1 -> H {F1.tipo := H.tipo}

F2 -> id {F2.pos = id.pos}

F3 ->

( A ) {F3.tipoParams := A.tipoParams,

F3.nParams = A.nParams,

F3.modoParams := A.modoParams

}

#### 4.1.3.1 Parametros de Función

A -> T {zonaDecl := true} id

K {id.tipo := insertaTipoTS(id.pos, T.tipo),

insertaDesplTS(id.pos, desplL),

desplL += T.ancho,

A.tipoParams := push(T.tipo, K.tipoParams),

A.nParams = K.nParams + 1,

A.modoParams := push(modo\_valor, K.modoParams),

zonaDecl := false

}

A -> void {A.tipoParams = push(vacio, nuevaLista()),

A.nParams = 0,

A.modoParams = push(modo\_valor, nuevaLista()),

zonaDecl := false

}

K -> lambda {K.tipoParams = push(vacio, nuevaLista()),

K.nParams = 0,

K.modoParams = nuevaLista ()}

K -> , T {zonaDecl := true} id

K1 {id.tipo := insertaTipoTS(id.pos, T.tipo),

insertaDesplTS(id.pos, desplL),

desplL += T.ancho,

K.tipoParams := push(T.tipo, K1.tipoParams),

K.nParams = K1.nParams + 1,

K.modoParams := push(modo\_valor, K1.modoParams)

}

### 4.1.4 Sentencias Simples

S -> id = E ; {id.tipo := buscaTipoTS(id.pos)

S.tipo :=

if (id.tipo = E.tipo) {

then tipo\_ok

} else {

Error ('No se pueden asignar valores E.tipo a la variable id que es un id.tipo')

then tipo\_err

},

S.tipoRet := vacio

}

S -> id &= E ; {id.tipo := buscaTipoTS(id.pos)

S.tipo :=

if (id.tipo = E.tipo) {

then tipo\_ok

} else {

Error ('No se pueden asignar valores E.tipo a la variable id que es un id.tipo')

then tipo\_err

},

S.tipoRet := vacio

}

S -> output E ; {S.tipo := if (E.tipo ∈ {ent, cad})

then tipo\_ok

else {

Error('Las operaciones de salida solo admiten enteros

o cadenas, no se permite E.tipo')

then tipo\_err

},

S.tipoRet := vacio

}

S -> input id ; {id.tipo := buscaTipoTS(id.pos),

S.tipo := if (id.tipo ∈ {ent, cad}

then tipo\_ok

else {

Error('Las operaciones de entrada solo admiten enteros o cadenas, no se permite id.tipo')

then tipo\_err

},

S.tipoRet := vacio

}

S -> return X ; {S.tipo := if (X.tipoRet = tipo\_err) {

then tipo\_err

} else {

then tipo\_ok

},

S.tipoRet := X.tipoRet

}

S -> id

( L ) ; {S.tipoParams = buscaTipoParams(id.pos)

S.nParams = buscaNParams(id.pos),

S.tipo := if (S.nParams != L.nParams) {

Error('El numero de argumentos no corresponde con los definidos en esta funcion')

then tipo\_err

}

for (i = 0, error = 0; i < S.nParams && !error; i++){

if (L.tipoParams(i) = tipo\_err) {

error = 1

} else if (S.tipoParams(i) != L.tipoParams(i)){

Error('El tipo de los argumentos no corresponde con los definidos en esta funcion')

error = 1

}

}

if (!error) then tipo\_ok else tipo\_err,

S.tipoRet := vacio

}

#### 4.1.4.1 Llamadas a funcion como Sentencia

L -> E Q {L.tipoParams := push(E.tipo, Q.tipoParams),

L.nParams = Q.nParams + 1}

L -> lambda {L.tipoParams := push(vacio, nuevaLista()),L.nParams := 0}

Q -> , E Q1 {Q.tipoParams := push(E.tipo, Q1.tipoParams),

Q.nParams := Q1.nParams + 1}

Q -> lambda {Q.tipoParams := push(vacio, nuevaLista()),Q.nParams := 0}

### 4.1.5 Expresiones

E -> E1 && R {E.tipo :=

if (E1.tipo = R.tipo) {

if (R.tipo = log)

then ent

else {

Error('Los operandos deben ser de tipo logico')

then tipo\_err

}

} else {

Error('Los operandos deben ser mismo tipo')

then tipo\_err

}}

E -> R {E.tipo := R.tipo}

X -> E {X.tipoRet := E.tipo}

X -> lambda {X.tipoRet := vacio}

R -> R1 == U {R.tipo :=

if (R1.tipo = U.tipo) {

if (U.tipo = ent) {

then log

} else {

Error('Los operandos deben ser de tipo entero')

then tipo\_err

}

} else {

Error('Los operandos deben ser mismo tipo')

then tipo\_err

}}

R -> U {R.tipo := U.tipo}

U -> U1 + V {U.tipo :=

if (U1.tipo = V.tipo) {

if (R.tipo = ent){

then ent

} else {

Error('Los operandos deben ser de tipo entero')

then tipo\_err

}

} else {

Error('Los operandos deben ser mismo tipo')

then tipo\_err

}}

U -> V {U.tipo := V.tipo}

V -> id {V.tipo := buscaTipoTS(id.pos)}

V -> ( E ) {V.tipo := E.tipo}

V -> ent {V.tipo := ent}

V -> id

( L ){V.tipoParams := buscaTipoParams(id.pos),

V.nParams := buscaNParams(id.pos),

V.tipo := if (S.nParams != L.nParams) {

Error('El numero de argumentos no corresponde con los definidos en esta funcion')

then tipo\_err

}

for (i = 0, error = 0; i < S.nParams&& !error; i++) {

if (L.tipoParams(i) = tipo\_err) {

error = 1

} else if (S.tipoParams(i) != L.tipoParams(i)){

Error('El tipo de los argumentos no corresponde con los definidos en esta funcion')

error = 1

}

}

if (!error) then buscaTipoRet(id.pos) else tipo\_err

}

V -> cad {V.tipo := cad}

# Tabla de Símbolos

## Estructura y organización

Para implementar el módulo de la Tabla de Símbolos, se ha decidido que su estructura sea una tabla hash. En este tipo de tabla, cada elemento es indexado por una clave que, internamente corresponde a otro índice resultado de una función hash sobre esa clave. Por esta misma razón, no pueden existir dos elementos con la misma clave, en este caso, dos identificadores con un mismo lexema. La búsqueda por lexema es inmediata.

En la implementación existe una clase TablaSimbolos que contiene la tabla hash de objetos Simbolo indexados por su lexema y tiene las operaciones necesarias: insertar, buscar, añadir atributo, consulta de atributos y vaciar la tabla. Cada TablaSimbolos contiene un descriptor de fichero a un archivo temporal de simbolos. La tabla global escribe en un fichero ‘simbolos’. En el momento de su destrucción, se añaden todas las tablas locales de los ficheros temporales y se eliminan.

Cada vez que se realice una búsqueda a la tabla de símbolos, se retornará un puntero a esa entrada de la tabla. La clase Símbolo contiene todos los atributos que pueden ser accedidos y modificados, más adelante, por los demás analizadores.

# Gestor de Errores

## Tipos de Errores

### Errores Léxicos

|  |  |  |
| --- | --- | --- |
| Código de error | Mensaje de error | Posible causa |
| 30 | No se reconoce el carácter LEXEMA en ASCII | Se ha introducido un carácter que su codificación no se encuentra en el rango [0, 127] |
| 31 | Existe un carácter ASCII de control**¹** | Se ha introducido un carácter de control que su codificación se encuentra en el rango [0, 31], exceptuando los códigos 10 y 13 que corresponden al salto de línea. |
| 32 | No se reconoce el operador LEXEMA | El carácter introducido no se reconoce en este lenguaje como operador, palabra reservada o identificador. |
| 33 | LEXEMA no es un carácter ASCII imprimible | Se ha introducido un carácter no imprimible en el lexema de una cadena o un comentario. |
| 34 | El numero LEXEMA ha excedido los 16 bits | El número no es representable en este lenguaje para un entero. Se exceden los 2B para su representación. |
| 35 | La cadena LEXEMA ha excedido los 64 caracteres | El lexema excede el máximo representable para una cadena en este lenguaje. |

**¹.** Se ha tomado la decisión de ignorar estos caracteres y considerarlos como delimitadores en ciertas transiciones.

### Errores Sintácticos

|  |  |  |
| --- | --- | --- |
| Código de error | Mensaje de error | Posible causa |
| 60 | Se esperaba encontrar LISTA\_TOKENS pero se ha recibido TOKEN | Falta algún token para completar una sentencia |
| 61 | Este token TOKEN es incorrecto. Por favor, elimínelo. | El token TOKEN está repetido y es incorrecto. |

### Errores Semánticos

|  |  |  |
| --- | --- | --- |
| Código de error | Mensaje de error | Posible causa |
| 40 | Las sentencias return deben ir dentro de las funciones | Las sentencias return sólo pueden ir dentro de la definición de una función. |
| 41 | En las sentencias condicionales, se necesita una condición lógica, sin embargo, es de tipo EXPRESION\_TIPO | Las sentencias condicionales sólo evalúan condiciones lógicas. |
| 42 | La funcion ID\_LEXEMA debe devolver ID\_TIPORET pero esta devolviendo TIPO\_RET | El tipo de retorno que devuelve la función no coincide con el de la definición. |
| 43 | Ambos lados de la asignacion deben ser del mismo tipo. En cambio el valor de LEXEMA es: ID\_TIPO y su valor asignado EXPRESION\_TIPO | No hay conversión de tipos. Por tanto, el tipo de la expresión debe ser el mismo tipo de la variable a asignar |
| 44 | En ambos lados de la asignacion con operador &= deben ser valores enteros. En cambio el valor de LEXEMA es: ID\_TIPO y su valor asignado EXPRESION\_TIPO | En las asignaciones con el operador &=. Tanto la variable como la expresión deben ser enteros. |
| 45 | Llamada a un funcion ID\_LEXEMA no definida previamente | Las funciones deben estar definidas previamente antes de ser llamadas. |
| 46 | El numero de argumentos no corresponde con los definidos en la funcion ID\_LEXEMA | Tanto el número de argumentos como el tipo de cada uno en una llamada a función debe coincidir con los de la definición. |
| 47 | El tipo de los argumentos no corresponde con los definidos en la funcion ID\_LEXEMA |
| 48 | Las operaciones de salida solo admiten enteros o cadenas, no se admite EXPRESION\_TIPO | Las operaciones de entrada y salida sólo admiten, los tipos entero o cadena, tanto si es una variable como una expresión |
| 49 | Las operaciones de entrada solo admiten enteros o cadenas, no se admite ID\_TIPO |
| 50 | Los operandos en una expresión && deben ser de tipo logico | La operación && sólo se puede realizar entre expresiones lógicas. |
| 51 | Los operandos en una expresión EXRPRESION deben ser del mismo tipo | No existe conversión de tipos, los operandos no pueden ser de diferente tipo |
| 52 | Los operandos en una expresión EXPRESIÓN deben ser de tipo entero | La operaciones de suma e igualdad sólo se pueden realizar entre expresiones enteras. |

## Tolerancia a Fallos

Frente a un error, el procesador debe continuar el análisis para seguir detectando más posibles errores. Tienen en común que todos dan parte del error pero el manejo será distinto. En el léxico, continúa leyendo hasta encontrar un delimitador, en ese momento, reanuda el análisis. En el sintáctico, si no se encuentra una acción, se tratan dos casos. Primer caso, si el token está repetido, se elimina y se pide el siguiente token. Segundo caso, falta un token, se evalúa cuales son los tokens posibles para ese estado y si existe el token de algun estado siguiente a partir de este, añade un token ficticio. El planteamiento que se ha seguido en la inserción de un token faltante, es intentar buscar de entre los posibles tokens cuál de ellos ejecutando su acción es posible encontrar el token actual en la siguiente iteración. Sin embargo, en la mayoría de los casos no se encuentra en la siguiente iteración o hay que aplicar varias reducciones, por lo que, en esta implementación mientras sólo exista una posibilidad prosigue explorando. Si no se encuentra, en cualquiera de los dos casos, se intenta pedir el siguiente token. En el semántico, no hay necesidad de manejar errores porque no alteran el comportamiento del procesador y se pueden observar en el tipo de los identificadores (TIPO\_ERROR) en la Tabla de Símbolos.

El módulo de Gestor de Errores, contiene el número de línea en el fichero fuente, un descriptor de fichero a un archivo ‘errores’ para el volcado de errores y una tabla indexada por código de error de los mensajes de error al usuario. Los estados de error, se encuentran codificados en la matriz de transición de estados para cada transición no prevista.

# ANEXO

#### Casos de prueba

##### Interfaz de usuario gráfica, Texto, Aplicación Descripción generada automáticamenteCaso de prueba 1

Diagrama

Descripción generada automáticamente

Ilustración 3. Fichero simbolos generado

Imagen que contiene Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente con confianza baja

Ilustración 2.Fichero tokens generado

Ilustración 1. Codigo de prueba

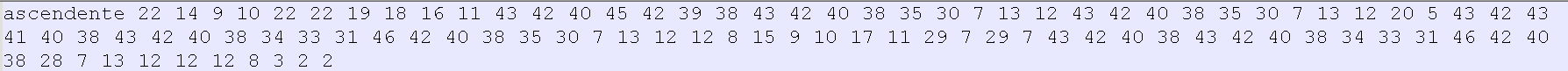


Ilustración 4.Fichero parse generado

Imagen que contiene Gráfico

Descripción generada automáticamenteGráfico, Gráfico de dispersión

Descripción generada automáticamenteDiagrama, Esquemático

Descripción generada automáticamenteGráfico de dispersión

Descripción generada automáticamente con confianza baja

Ilustración . Árbol generado por herramienta Vast

##### Tabla Descripción generada automáticamente con confianza bajaImagen que contiene Aplicación Descripción generada automáticamenteImagen que contiene Diagrama Descripción generada automáticamenteUna captura de pantalla de un celular Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico Descripción generada automáticamenteCaso de prueba 2

Ilustración . Fichero errores generado

Ilustración . Fichero parse generado

Ilustración . Fichero tokens generado

Ilustración . Fichero simbolos generado

Ilustración . Código de prueba

##### Gráfico Descripción generada automáticamente con confianza bajaImagen que contiene Escala de tiempo Descripción generada automáticamenteImagen que contiene Gráfico Descripción generada automáticamenteImagen que contiene Diagrama Descripción generada automáticamenteCaso de prueba 3

Ilustración . Árbol generado por herramienta Vast

Tabla

Descripción generada automáticamente con confianza baja Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Código de prueba

Tabla

Descripción generada automáticamente con confianza media Imagen que contiene Tabla

Descripción generada automáticamenteTabla

Descripción generada automáticamente con confianza baja

Ilustración . Fichero errores generado

Ilustración . Fichero tokens generado

Ilustración . Fichero simbolos generado

Ilustración . Fichero parse generado

Imagen de la pantalla de un celular de un mensaje en letras negras

Descripción generada automáticamente con confianza media

Imagen que contiene Gráfico

Descripción generada automáticamenteEscala de tiempo

Descripción generada automáticamente con confianza mediaGráfico

Descripción generada automáticamente con confianza mediaGráfico de dispersión

Descripción generada automáticamenteImagen que contiene Diagrama

Descripción generada automáticamente

Ilustración . Árbol sintáctico generado por herramienta Vast

##### Caso de prueba 4

Tabla

Descripción generada automáticamente con confianza mediaTexto, Carta

Descripción generada automáticamenteImagen que contiene Tabla

Descripción generada automáticamenteTabla

Descripción generada automáticamenteTexto

Descripción generada automáticamente con confianza mediaInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Fichero errores generado

Ilustración . Fichero parse generado

Ilustración . Fichero tokens generado

Ilustración . Fichero simbolos generado

Ilustración . Código de prueba

##### Imagen que contiene Gráfico de dispersión Descripción generada automáticamenteImagen que contiene Gráfico de dispersión Descripción generada automáticamenteImagen que contiene Diagrama Descripción generada automáticamenteImagen que contiene Gráfico Descripción generada automáticamenteImagen que contiene Diagrama Descripción generada automáticamenteGráfico Descripción generada automáticamente con confianza mediaImagen que contiene Diagrama Descripción generada automáticamenteCaso de prueba 5

Ilustración . Árbol generado por herramienta Vast

Tabla

Descripción generada automáticamenteImagen que contiene Texto

Descripción generada automáticamenteUna captura de pantalla de un celular con letras

Descripción generada automáticamente con confianza mediaInterfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Ilustración . Fichero parse generado

Ilustración . Fichero tokens generado

Ilustración . Fichero simbolos generado

Ilustración . Código de prueba

Gráfico

Descripción generada automáticamenteImagen que contiene Gráfico de dispersión

Descripción generada automáticamenteGráfico, Gráfico de dispersión

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente con confianza mediaCaso de prueba 6

Ilustración . Árbol sintáctico generado por herramienta Vast

Texto, Carta

Descripción generada automáticamenteImagen de la pantalla de un celular de un mensaje en letras negras

Descripción generada automáticamente con confianza mediaInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Fichero tokens generado

Ilustración . Fichero simbolos generado

Ilustración . Fichero errores generado

Ilustración . Código de prueba

##### Caso de prueba 7

Texto

Descripción generada automáticamenteTabla

Descripción generada automáticamente con confianza mediaInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Fichero simbolos generado

Ilustración . Fichero tokens generado

Ilustración . Código de prueba

Ilustración . Fichero errores generado

##### Caso de prueba 8

Diagrama

Descripción generada automáticamente con confianza media

Ilustración .Fichero simbolos generado

Imagen que contiene Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Ilustración . Fichero errores generado

Ilustración . Fichero tokens generado

Ilustración . Código de prueba

##### Caso de prueba 9

Texto

Descripción generada automáticamente con confianza mediaInterfaz de usuario gráfica, Texto

Descripción generada automáticamenteImagen que contiene Tabla

Descripción generada automáticamente

Ilustración . Fichero errores generado

Ilustración . Fichero simbolos generado

Ilustración . Código de prueba

Ilustración . Fichero tokens generado

##### Caso de prueba 10