10/3/24

Universidad de Guadalajara
CUCEI

Ejemplo gramática LR utilizando la tabla del compilador

Centro de Ciencias Exactas e Ingenierías
Departamento de Ciencias Universitario
Computacionales

Carrera: Ingeniería en Computación

Materia: Seminario de solución de problemas de traductores de lenguaje II

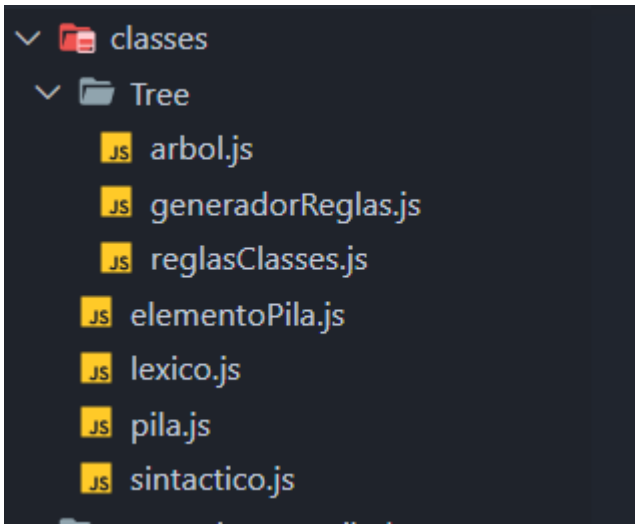
Profesor: Michel Emanuel López Franco

Alumno: Nudelstejer Gómez Iván

código: 218130122

Sección: D02

1. **Ejemplo de Gramática Simple**:
    - Consideremos la siguiente gramática simple:
    - S -> E
    - E -> E + T | T
    - T -> id
    - Construyamos los elementos LR(0) y las cerraduras:
        - Los elementos LR(0) serían: { S -> . E }, { E -> . E + T }, { E -> . T }, { T -> . id }.
        - La cerradura del conjunto { E -> . E + T } incluiría { T -> . id } y { E -> . T }.
        - Continuaríamos aplicando la regla hasta que no podamos agregar más elementos.
2. **Ejemplo de Gramática con Reducciones**:
    - Consideremos la siguiente gramática:
    - S -> E
    - E -> E + T | T
    - T -> T * F | F
    - F -> (E) | id
    - Generemos los elementos LR(0) y las cerraduras:
        - Los elementos LR(0) incluirían producciones como { E -> . E + T }, { T -> . T * F }, { F -> . (E) }, etc.
        - Las cerraduras se construirían siguiendo el proceso explicado anteriormente.
3. **Ejemplo de Tabla de Análisis Sintáctico LR**:
    - Utilizando los elementos LR(0) y las cerraduras, construimos una tabla de análisis sintáctico LR.
    - Las entradas en la tabla indican acciones como desplazamiento (d), reducción ® o aceptación (a).


classes
Tree
arbol.js
generadorReglas.js
reglasClasses.js
elementoPila.js
lexico.js
pila.js
sintactico.js


COMPILADOR
Analizador Sintactico
int hola;
Analizar
Aceptacion
Pila
Entrada
Accion
$0
int
5

Arbol Sintactico

0. programa

__1. Definiciones

____2. Definicion

______3. DefVar

________4. int

________4. hola

________4. ListaVar

________4. ;

____2. Definiciones

Codigo:

```
import {Lexico} from './classes/lexico.js';
import { Sintactico } from './classes/sintactico.js';
import {Nodo, Arbol} from './classes/Tree/arbol.js';

let sint = new Sintactico();

const botonlex1 = document.getElementById("boton");
botonlex1.addEventListener("click", () => {
    ejecutarlexalizador(1);
});

const obtenerFuente = () => {
    const input = document.getElementById("input");
    let fuente = input.value;
    fuente = fuente.trim();
    return fuente;
}

const imprimirRubros = (columna1, columna2, columna3) => {
    for (let i = columna1.length; i < 30; i++) {
        columna1 += " ";
    }

    for (let i = columna2.length; i < 20; i++) {
        columna2 += " ";
    }

    console.log(columna1 + columna2 + columna3);
}

const generarFilasTabla = (columna1, columna2, columna3) => {
    let resultado = document.getElementById("resultado");
    let tabla = resultado.firstElementChild;

    let row = document.createElement("DIV");

    let column1 = document.createElement("DIV");
    let span1 = document.createElement("SPAN");
    column1.classList.add("column");
    span1.textContent = columna1;
    column1.appendChild(span1);
    row.appendChild(column1);

    let column2 = document.createElement("DIV");
    let span2 = document.createElement("SPAN");
    column2.classList.add("column");
```

```
    span2.textContent = columna2;
    column2.appendChild(span2);
    row.appendChild(column2);

    let column3 = document.createElement("DIV");
    let span3 = document.createElement("SPAN");
    column3.classList.add("column");
    span3.textContent = columna3;
    column3.appendChild(span3);
    row.appendChild(column3);

    row.classList.add("row", "row-content");

    tabla.appendChild(row);
}

const reiniciaTabla = () => {
    let resultado = document.getElementById("resultado");
    let tabla = resultado.firstElementChild;
    let vacio = document.getElementById("vacio");
    let mensaje = document.getElementById("mensaje-
resultado");
    vacio.classList.add("oculto");
    mensaje.classList.add("oculto");


    resultado.classList.remove("oculto");

    [...tabla.children].forEach(element => {
        if(element != tabla.firstElementChild) {
            tabla.removeChild(element);
        }
    });
}

const reiniciarArbol = () => {
    let titulo = document.getElementById("titulo-arbol");
    let arbol = document.getElementById("arbol");
    let lista = arbol.firstElementChild;
    titulo.classList.add("oculto");
    arbol.classList.add("oculto");
    [...lista.children].forEach(element => {
        lista.removeChild(element);
    });
}

const entradaVaciaMsj = () => {
```

```
    let resultado = document.getElementById("resultado");
    resultado.classList.add("oculto");

    let aceptacion = document.getElementById("mensaje-
resultado");
    aceptacion.classList.add("oculto");

    let titulo = document.getElementById("titulo-arbol");
    let arbol = document.getElementById("arbol");
    titulo.classList.add("oculto");
    arbol.classList.add("oculto");

    let vacio = document.getElementById("vacio");
    vacio.classList.remove("oculto");
}

const resultadoMsj = (texto) => {
    let aceptacion = document.getElementById("mensaje-
resultado");
    aceptacion.classList.remove("oculto");
    aceptacion.firstElementChild.textContent = texto;
}

const imprimirArbol = (arbol) => {
    let tituloDom = document.getElementById("titulo-arbol");
    let arbolDom = document.getElementById("arbol");
    let listaDom = arbolDom.firstElementChild;
    arbolDom.classList.remove("oculto");
    tituloDom.classList.remove("oculto");

    let espacio = 0;
    let fragmento = new DocumentFragment();
    arbol.raiz.forEach(rama => {
        imprimirNodo(rama, espacio, fragmento);
    });
    listaDom.appendChild(fragmento);
}

const imprimirNodo = (rama, espacio, fragmento) => {
    let espaciado = "";
    for (let i = 0; i < espacio; i++) {
        espaciado += "__";
    }
    console.log(espaciado + espacio + ". " + rama.simbolo);
    let ramaDom = document.createElement("li");
    ramaDom.textContent = espaciado + espacio + ". " +
rama.simbolo;
```

```
    ramaDom.classList.add("parte-arbol");
    if(rama.ramas.length == 0) {
        ramaDom.classList.add("hoja");
    }
    fragmento.appendChild(ramaDom);
    rama.ramas.forEach(elemento => {
        imprimirNodo(elemento, espacio + 1, fragmento);
    });
}

const ejecutarlexalizador = () => {
    let fuente = "";
    fuente = obtenerFuente();
    let lex = new Lexico(fuente);

    //console.log("Resultado del analisis Sintactico");
    //console.log("Resultado del lexalisis Lexico");
    console.log("Arbol Sintactico");

    if(fuente.length != 0) {
        //console.log("Pila                          Entrada
Accion   ");
        reiniciaTabla();
        reiniciarArbol();

        sint.inicializarPila();
        sint.inicializarArbol();
        let arbol = new Arbol();

        while(!lex.termina()) {
            lex.obtenerSimbolo();

            let opcion = 0;

            do {
                sint.sigEntrada(lex.tipo);

                //imprimirRubros(sint.pila.toString(),
lex.simbolo, sint.accion);
                generarFilasTabla(sint.pila.toString(),
lex.simbolo, sint.accion);

                opcion = sint.sigAccion(lex.simbolo);

                if(opcion == 2) {
                    //imprimirRubros(sint.pila.toString(),
lex.simbolo, sint.accion);
```

```
                    generarFilasTabla(sint.pila.toString(),
lex.simbolo, sint.accion);

                } else if(opcion == 3) {
                    resultadoMsj("Error sintactico");
                    console.log("Error sintactico");
                    lex.caracter = "$";
                    break;

                } else if(opcion == 4) {
                    resultadoMsj("Aceptacion");
                    console.log("Aceptacion");
                    arbol = sint.arbol;
                    imprimirArbol(arbol);
                    break;
                }
            } while(opcion == 2);
        }
    } else {
        console.log("Entrada vacia");
        entradaVaciaMsj();
    }
}
```