Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería

Plan de Ingeniero en Software y Tecnologías Emergentes

Meta 2.1
Ejemplo analizador léxico

Materia
Traductores (361)

Docente
Guillermo Licea Sandoval

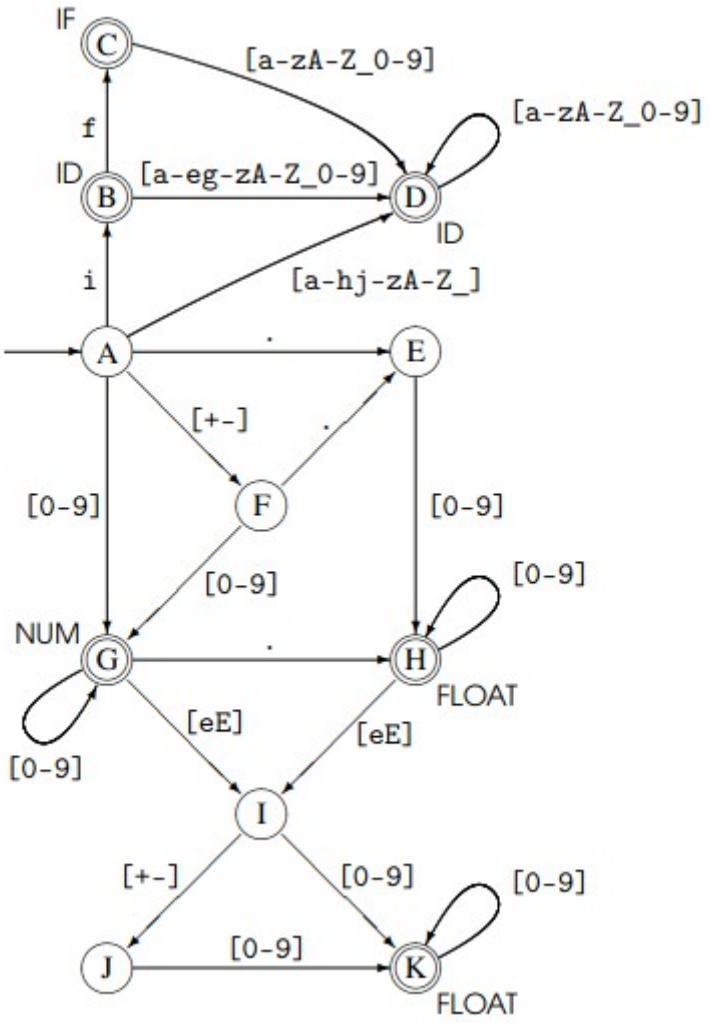
Participante(es)
Luis Eduardo Galindo Amaya
1274895

15 de feb de 2025

Sumario

Instrucciones

Adjunta un archivo con el código de tu analizador léxico basado en ER y AF.



Estado

```
package Automaton2;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class State {
 private final Map<String, State> transitions = new HashMap<>();
 private final boolean isFinal;
 private final String tokenType;

 public boolean isFinal() {
  return isFinal;
 }

 public String getTokenType() {
  return tokenType;
 }

 public State(boolean isFinal, String tokenType) {
  this.isFinal = isFinal;
  this.tokenType = tokenType;
 }

 public State addTransition(String condition, State state) {
  transitions.put(condition, state);
  return this;
 }
```

```
 public State nextState(String test) {

  for (String regex : transitions.keySet()) {

   Pattern pattern = Pattern.compile(regex);
   Matcher matcher = pattern.matcher(test);

   if (matcher.find()) {
    // Retorna el estado del patron
    return transitions.get(regex);
   }
  }

  return null;
 }

}
```

Autómata

```
package Automaton2;



public class Automaton {
 State A = new State(false, "A");
 State B = new State(true, "ID");
 State C = new State(true, "IF");
 State D = new State(true, "ID");

 State E = new State(false, "E");
 State F = new State(false, "F");
 State G = new State(true, "NUM");

 State H = new State(true, "FLOAT");
 State I = new State(false, "I");
 State J = new State(false, "J");
 State K = new State(true, "FLOAT");

 public Automaton() {

  // Cargar los estados del automata
  A.addTransition("i", B);
  A.addTransition("[a-hj-zA-Z_]", D);
  A.addTransition("[+-]", F);
  A.addTransition("\\.", E);
  A.addTransition("[0-9]", G);

  B.addTransition("[a-eg-zA-Z_0-9]", D);
  B.addTransition("f", C);
```

```
 C.addTransition("[a-zA-Z_0-9]", D);

 D.addTransition("[a-zA-Z_0-9]", D);

 E.addTransition("[0-9]", H);

 F.addTransition("[0-9]", G);
 F.addTransition("\\.", E);

 G.addTransition("[0-9]", G);
 G.addTransition("\\.", H);
 G.addTransition("e|E", I);

 H.addTransition("[0-9]", H);
 H.addTransition("e|E", I);

 I.addTransition("[+-]", J);
 I.addTransition("[0-9]", K);

 J.addTransition("[0-9]", K);

 K.addTransition("[0-9]", K);
}

public String tokenType(String input) {
 State currentState = A;
 int length = input.length();

 if (input.isBlank()) {
  throw new IllegalArgumentException("Entrada vacia");
```

```
  }

  for (int i = 0; i < length; i++) {
   char c = input.charAt(i);
   State nextState = currentState.nextState(c + "");

   if (nextState == null) {
    throw new IllegalArgumentException("Error de sintaxis");
   }

   if (!nextState.isFinal() && i == length - 1) {
      throw new IllegalArgumentException("No se pudo llegar a un estado final");
   }

   currentState = nextState;
  }

  return currentState.getTokenType();
 }

}
```

Main

```
package org.example;


import Automaton2.Automaton;
import java.util.stream.Stream;


public class Main {
 public static void main(String[] args) {
  Automaton automaton = new Automaton();


  Stream<String> validInputs = Stream.of(
    "iff", "123.", "if", "123.45", "abc123", "1.2e-3", "xyz", "IF"
  );


  Stream<String> invalidInputs = Stream.of(
    "123#", "1abc", ".abc", "1.2.3", "1.2e", "1.2e+",  "", "123e", "abc!",
"1.2e3.4"
  );


  // Combinar los Streams
          String[]     combinedInputs     =     Stream.concat(validInputs,
invalidInputs).toArray(String[]::new);


  for (String input : combinedInputs) {
   try {
            System.out.println("Token:   "   +   input   +   "   →   "   +
automaton.tokenType(input));
   } catch (RuntimeException e) {
    System.err.println("Error en '" + input + "': " + e.getMessage());
   }
  }
```

```
  }
}
```

Salida

```
Token: iff -> ID
Token: 123. -> FLOAT
Token: if -> IF
Token: 123.45 -> FLOAT
Token: abc123 -> ID
Token: 1.2e-3 -> FLOAT
Token: xyz -> ID
Token: IF -> ID
Error en '123#': Error de sintaxis
Error en '1abc': Error de sintaxis
Error en '.abc': Error de sintaxis
Error en '1.2.3': Error de sintaxis
Error en '1.2e': No se pudo llegar a un estado final
Error en '1.2e+': No se pudo llegar a un estado final
Error en '': Entrada vacia
Error en '123e': No se pudo llegar a un estado final
Error en 'abc!': Error de sintaxis
Error en '1.2e3.4': Error de sintaxis
```