Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 1

Análisis de Expresiones Regulares

ALUMNO: Cortes Santos Maria Fernanda
Dr. Eduardo Cornejo-Velázquez

Introducción

Una Gramática Formal definida sobre un alfabeto $\sum$ es una tupla de la forma:

$$\mathbf{G} = \{\sum_{\mathbf{T}}, \sum_{\mathbf{N}}, \mathbf{S}, \mathbf{P}\}$$

donde:

- $\sum_{\mathbf{T}}$ es un alfabeto de símbolos terminales
- $\sum_{\mathbf{N}}$ es un alfabeto de símbolos no terminales
- $\boldsymbol{S}$ es el símbolo inicial de la gramática
- $\mathbf{P}$ es un conjunto de producciones gramaticales

Además, se cumple:

$$\mathbf{S} \in \sum_{\mathbf{N}}$$
$$\sum_{\mathbf{T}} \cap \sum_{\mathbf{N}} = \emptyset$$
$$\sum = \sum_{\mathbf{T}} \cup \sum_{\mathbf{N}}$$

La gramática formal $\mathbf{G}$ permite generan un lenguaje $\mathbf{L} = \{\mathbf{x} \in \sum_{\mathbf{T}}^{*} | \mathbf{S} \overset{*}{\rightarrow} \mathbf{x}\}$

Por lo tanto, las palabras del lenguaje estarán formadas por cadenas de símbolos terminales generadas a partir del símbolo inicial de la gramática utilizando las producciones que la definen. Una expresión regular es una notación normalizada para representar lenguajes regulares, es decir, lenguajes generados por gramáticas regulares (Tipo 3). Las expresiones regulares permiten describir con exactitud y sencillez cualquier lenguaje regular.

Marco teórico

¿Qué son las Expresiones Regulares?

Las expresiones regulares, comúnmente conocidas como Regex, son secuencias de caracteres que conforman un patrón de búsqueda. Su función principal es la de describir un conjunto de cadenas de caracteres sin tener que enumerarlas una por una. Se basan en la teoría de lenguajes formales y son fundamentales en la computación para tareas de procesamiento de texto, validación de formatos (como correos electrónicos o contraseñas) y operaciones de búsqueda y reemplazo.

Componentes y Uso

Las Regex utilizan una combinación de caracteres literales y metacaracteres (caracteres con significados especiales):

- Literales: Caracteres que se buscan tal cual (ej. a, 9).
- Clases de caracteres: Definen rangos, como [a-z] para letras minúsculas o [0-9] para dígitos.
- Cuantificadores: Indican la frecuencia de aparición de un carácter o grupo:
  - *: Cero o más veces.
  - +: Una o más veces.
  - ?: Cero o una vez.
  - n,m: Entre n y m veces.

Implementación en el Lenguaje Java

En Java, el manejo de expresiones regulares se realiza a través del paquete java.util.regex, el cual introdujo dos clases principales y una interfaz:

- Clase Pattern: Es la representación compilada de una expresión regular. Para usarla, se debe invocar el método estático Pattern.compile().
- Clase Matcher: Es el motor que interpreta el patrón y realiza operaciones de coincidencia contra una cadena de entrada. Se obtiene mediante el método matcher() de un objeto Pattern.
- Métodos de la clase String: Java permite usar Regex directamente en la clase String mediante métodos simplificados como matches(), split(), y replaceAll(), aunque internamente estos utilizan las clases mencionadas anteriormente.

Objetivo General

Construir programas en un lenguaje de programación de alto nivel que realicen el análisis de cadenas de símbolos que forman palabras basado en la implementación de estructuras de datos y algoritmos de análisis, además del uso de expresiones regulares como segunda estrategia de implementación.

Objetivos Específicos

1. Identificar el proceso básico de análisis de cadenas de símbolos de un alfabeto.
2. Implementar estructuras de datos y algoritmos de análisis de cadenas de símbolos en un lenguaje de programación de alto nivel.
3. Implementar expresiones regulares para el análisis de cadenas de símbolos en un lenguaje de programación de alto nivel.

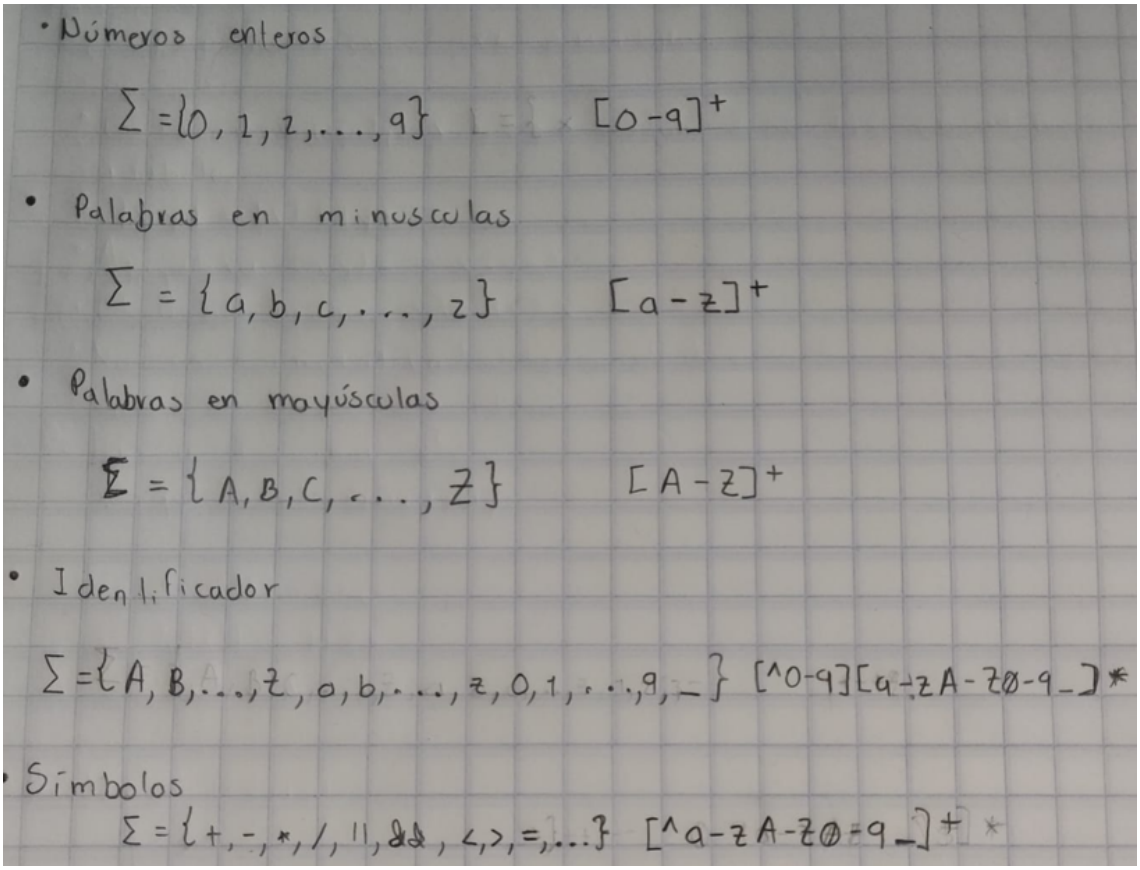
Equipo y software a utilizar

Equipo de cómputo con software para programación a elección de los alumnos.

Procedimiento para el desarrollo de la práctica

1. Utilizar un lenguaje de programación para construir programas basados en estructuras de datos para analizar cadenas de símbolos con el propósito de identificar:
   - Número entero... https://github.com/MaFecs/Actividades_Automatas_y_Compiladores/blob/a28d086a9cf1d89c917198c11a7571a97a5994b5/Punto1NumeroEntero.java
   - Palabras en minúsculas... https://github.com/MaFecs/Actividades_Automatas_y_Compiladores/blob/a28d086a9cf1d89c917198c11a7571a97a5994b5/Punto2Minusculas.java
   - Palabras en mayusculas... https://github.com/MaFecs/Actividades_Automatas_y_Compiladores/blob/a28d086a9cf1d89c917198c11a7571a97a5994b5/Punto3Mayusculas.java
   - dentificadores(Nombres de variables)... https://github.com/MaFecs/Actividades_Automatas_y_Compiladores/blob/a28d086a9cf1d89c917198c11a7571a97a5994b5/Punto4Identificadores.java
2. Definir el alfabeto de símbolos y las expresiones regulares que permita identificar las siguientes tipos de palabras de un lenguaje regular:
   - Número entero
   - Palabras en minúsculas

- Palabras en mayusculas
- Identificadores (Nombres de variables)
- Símbolos



3. Utilizar un lenguaje de programación para construir programas basados en expresiones regulares para analizar cadenas de símbolos con el propósito de clasificar y contabilizar las palabras en las categorias:
   - Número entero... https://github.com/MaFecs/Actividades_Automatas_y_Compiladores/blob/a28d086a9cf1d89c917198c11a7571a97a5994b5/Punto5NumeroEntero.java
   - Palabras en minúsculas... https://github.com/MaFecs/Actividades_Automatas_y_Compiladores/blob/a28d086a9cf1d89c917198c11a7571a97a5994b5/Punto6Minusculas.java
   - Palabras en mayusculas... https://github.com/MaFecs/Actividades_Automatas_y_Compiladores/blob/a28d086a9cf1d89c917198c11a7571a97a5994b5/Punto7Mayusculas.java
   - Identificador (Nombre de variables)... https://github.com/MaFecs/Actividades_Automatas_y_Compiladores/blob/a28d086a9cf1d89c917198c11a7571a97a5994b5/Punto8Identificador.java
   - Símbolo... https://github.com/MaFecs/Actividades_Automatas_y_Compiladores/blob/a28d086a9cf1d89c9
     Punto9Simbolos.java

Cuestionario

- ¿Cúal es utilidad de las expresiones regulares en la identificación de palabras de un lenguajes?

  Permiten definir patrones abstractos para validar si una secuencia de caracteres pertenece a una categoría específica (números, identificadores, etc.) de forma eficiente, sin necesidad de escribir múltiples condiciones if-else manuales para cada carácter.

- ¿Cuáles elementos del lenguaje de programación utilizaste para la implementación de las estructuras de datos?, ¿Cómo las utilizaste?

  Se utilizó la clase String para capturar la entrada y el tipo String[] (Arreglo de cadenas) para almacenar las palabras individuales tras dividirlas con el método .split(). Se usaron bucles for indexados para recorrer cada posición del arreglo y procesar los datos uno a uno.

- ¿Cuál es el proceso que seguiste para analizar las cadenas de símbolos?

  - Captura: Lectura del texto completo mediante Scanner.
  - Limpieza y Segmentación: Uso de .trim() y .split("doble diagonal invertida"s+") para separar el texto en palabras eliminando espacios innecesarios.
  - Evaluación: Recorrido del arreglo mediante un ciclo y aplicación del método .matches() en cada palabra para clasificarla según el patrón.

- ¿Qué elementos del lenguaje de programación te permitieron implementar expresiones reguales?, ¿qué condiciones o recomendaciones se deben seguir al escribir las expresiones regulares?

  Se utilizaron los métodos de la clase String (como .matches()) y las clases especializadas Pattern y Matcher.

Conclusiones

El uso de expresiones regulares (Regex) es fundamental en la computación porque permite implementar de forma práctica la teoría de autómatas finitos. A través de esta práctica, se comprobó que las Regex optimizan drásticamente el análisis léxico, permitiendo validar patrones complejos (como identificadores o números) con una sintaxis compacta que sustituye múltiples estructuras condicionales.
En Java, su integración mediante la clase Pattern y el método matches() facilita la creación de software robusto para la validación y limpieza de datos. En definitiva, dominar esta herramienta es clave para cualquier desarrollador, ya que constituye la base para el diseño de compiladores y el procesamiento eficiente de grandes volúmenes de texto.

Bibliografía

- Deitel, P., y Deitel, H. (2016). Java: How to Program (11a ed.). Pearson.
- Oracle. (s.f.). Lesson: Regular Expressions. The Java™ Tutorials. https://docs.oracle.com/javase/tutorial/essential/reg
- Sierra, K., y Bates, B. (2017). OCA Java SE 8 Programmer I Exam Guide. McGraw-Hill Education.
- Friedl, J. E. (2006). Mastering Regular Expressions (3a ed.). O'Reilly Media.