UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN “CAMPUS I”

LICENCIATURA EN INGENIERÍA EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE

6° “M”

ALUMNO:

Lara Clemente Juan Carlos A210573

DOCENTE: Luis Alfaro Gutierrez, Dr.

TAREA: Conceptos Act.1

TUXTLA GUTIÉRREZ, CHIAPAS a de Enero de 2024

Tabla de contenido

Conceptos

Explicar los tipos de operadores de expresiones regulares.

Las expresiones regulares, también conocidas como regex o regexp, son patrones de búsqueda y manipulación de texto que permiten especificar un conjunto de reglas para encontrar cadenas de caracteres dentro de un texto. Estos patrones se utilizan comúnmente en programación, procesamiento de texto y herramientas de búsqueda para realizar operaciones como validación de datos, búsqueda de patrones específicos, reemplazo de texto y más. Las expresiones regulares están compuestas por caracteres literales y metacaracteres que representan clases, cuantificadores, anclajes y otros elementos, lo que les confiere una gran flexibilidad y poder para realizar operaciones complejas en cadenas de texto.

	Descripción	Ejemplo	Resultado
\b	Principio o final de palabra	/\bver\b/	Encuentra ver en *"ver de"*, pero no en *"verde"*
\B	Frontera entre no-palabras	/\Bver\B/	Empareja ver con *"Valverde"* pero no con *"verde"*
\d	Un dígito	/[A-Z]\d/	No falla en "A4"
\D	Alfabético (no dígito)	/[A-Z]\D/	Fallaría en "A4"
\O	Carácter nulo		
\t	Caracter ASCII 9 (tabulador)		
\f	Salto de página		
\n	Salto de línea		
\w	Cualquier alfanumérico, [a-zA-Z0-9_]	/\w+/	Encuentra *frase* en "*frase.*", pero no el . (punto).
\W	Opuesto a \w ([^a-zA-Z0-9_])	/\W/	Hallaría sólo el punto (.)
\s	Carácter tipo espacio (como tab)	/\sSi\s/	Encuentra *Si* en *"Digo Si "*, pero no en *"Digo Sientate"*
\S	Opuesto a \s		
\cX	Carácter de control X	\c9	El tabulador
\oNN	Carácter octal NN		
\xHH	El hexadecimal hh	/\x41/	Encuentra la *A* (ASCII Hex41)

Operadores Literales:

Los caracteres literales en una expresión regular coinciden exactamente con ellos mismos. Por ejemplo, el patrón abc coincidirá con la cadena "abc".

Meta caracteres Básicos:

. (Punto): Coincide con cualquier carácter excepto un salto de línea.

\ (Barra invertida): Se utiliza para escapar metacaracteres, permitiendo que se interpreten literalmente. Por ejemplo, \. coincidirá con un punto literal.

Clases de Caracteres:

[ ]: Define una clase de caracteres. Por ejemplo, [aeiou] coincidirá con cualquier vocal.

[^]: Define una clase de caracteres negada. Por ejemplo, [^0-9] coincidirá con cualquier carácter que no sea un dígito.

Rangos de Caracteres:

- (Guion): Dentro de una clase de caracteres, se puede usar para definir un rango. Por ejemplo, [0-9] coincidirá con cualquier dígito.

Cuantificadores:

: Coincide con cero o más ocurrencias del elemento anterior. Por ejemplo, a coincidirá con "", "a", "aa", etc.

+: Coincide con una o más ocurrencias del elemento anterior.

?: Coincide con cero o una ocurrencia del elemento anterior.

Anclajes:

^: Coincide con el inicio de una cadena.

$: Coincide con el final de una cadena.

Operadores de Agrupación y Alternancia:

( ): Agrupa elementos para aplicar cuantificadores a conjuntos completos.

|: Representa la alternancia, es decir, "o". Por ejemplo, a|b coincidirá con "a" o "b".

Caracteres de Escape:

\: Se utiliza para escapar metacaracteres y permitir que se interpreten literalmente.

Explicar el proceso de conversión de DFA a expresiones regulares.

La conversión de un Autómata Finito Determinista (DFA, por sus siglas en inglés) a una expresión regular es un proceso que implica representar el lenguaje reconocido por el DFA mediante una expresión regular equivalente. Aquí hay una descripción general del proceso:

Paso 1: Eliminar Estados Inalcanzables

Antes de comenzar la conversión, es recomendable eliminar cualquier estado que no sea alcanzable desde el estado inicial. Esto simplifica el DFA y facilita el proceso de conversión.

Paso 2: Eliminar Estados Inalcanzables

Crear una expresión para cada transición directa:

Para cada transición directa del estado p al estado q con el símbolo a, se crea la expresión regular a.

Manejar los Estados de Transición Indirecta:

Para cada par de estados p y q sin una transición directa, se consideran los estados intermedios posibles r. Se crea una expresión regular para la transición indirecta de p a q mediante la combinación de las expresiones de las transiciones directas desde p hasta r, seguido de la expresión de la transición directa desde r hasta q.

Matemáticamente, si hay una transición directa de p a r con la expresión regular E1 y otra transición de r a q con la expresión regular E2, entonces la expresión regular para la transición indirecta de p a q será E1E2.

Paso 3: Eliminar Estados Intermedios

Identificar Estados Intermedios:

Por cada par de estados distintos p y q, donde r es un estado intermedio, se considera la posibilidad de eliminar r.

Actualizar las Expresiones Regulares:

Para cada par de estados p y q, se actualizan las expresiones regulares entre p y q considerando la eliminación de r. La nueva expresión regular se calcula como la unión de las expresiones sin r, utilizando la expresión regular de la transición directa de p a q y las expresiones de las transiciones indirectas de p a r y de r a q.

Repetir hasta que no haya más estados intermedios a eliminar.

Paso 4: Resultado Final

Al final del proceso, la expresión regular resultante estará asociada con la transición directa entre el estado inicial y el estado final del DFA. Esta expresión regular representará el lenguaje reconocido por el DFA original.

Es importante señalar que este proceso puede ser complejo y que no todos los DFA tienen una representación de expresión regular simple. Sin embargo, para DFAs finitos y completos, este método suele ser aplicable.

Explicar leyes algebraicas de expresiones regulares.

Las leyes algebraicas de las expresiones regulares son reglas que permiten manipular y simplificar expresiones regulares de manera algebraica, similar a cómo se realizan operaciones algebraicas en álgebra convencional. Estas leyes son útiles para simplificar y transformar expresiones regulares manteniendo su equivalencia. Aquí se presentan algunas de las leyes algebraicas comunes:

1. **Ley de Idempotencia:**
    - $=E+E=E$

Esta ley indica que la unión de una expresión regular consigo misma es equivalente a la expresión regular original.

2. **Ley de Anulación:**
    - $E \cdot \emptyset = \emptyset$
    - $\emptyset \cdot E = \emptyset$

La concatenación de una expresión regular con el conjunto vacío ($\emptyset\emptyset$) resulta en el conjunto vacío.

3. **Ley de Identidad:**
    - $E+\emptyset=E$
    - $\emptyset+E=E$
    - $E \cdot \epsilon = E$
    - $\epsilon \cdot E = E$

Estas leyes indican que la unión con el conjunto vacío y la concatenación con la cadena vacía (�ϵ) no alteran la expresión regular original.

4. **Ley de Absorción:**

    - $E+E\cdot F=E$
    - $E\cdot(E+F)=E$

Estas leyes reflejan que, en ciertos contextos, la unión y la concatenación pueden ser "absorbidas" por una de las expresiones regulares, resultando en la misma expresión regular.

5. **Ley de Complemento:**

    - $\Sigma*E+E=\Sigma*$
    - $\emptyset E\cdot E=\emptyset$

Aquí, $\Sigma*\Sigma*$ representa el conjunto de todas las cadenas sobre el alfabeto ΣΣ. Estas leyes indican que la unión con el complemento de una expresión regular es equivalente al conjunto de todas las cadenas, y la concatenación con el complemento de una expresión regular es equivalente al conjunto vacío.

6. **Ley de Distribución:**

    - $E\cdot(F+G)=E\cdot F+E\cdot G$

Esta ley refleja la propiedad de distribución de la concatenación sobre la unión.

Estas leyes algebraicas proporcionan herramientas para simplificar expresiones regulares y pueden ser útiles en el diseño y análisis de autómatas finitos y en el procesamiento de lenguajes formales.