UNIVERSIDAD AUTÓNOMA DE CHIAPAS

Facultad de Contaduría y Administración Campus

Nombre: Jesús Iker Fernández Constantino

Docente: Dr. Gutiérrez Alfaro Luis

Materia: COMPILADORES

Grado y Grupo: “6-M”

Fecha: 27/01/2024

INDICE

ACTIVIDAD I

I.- Explicar los tipos de operadores de expresiones regulares.

Una expresión regular es un modelo con el que el motor de expresiones regulares intenta buscar una coincidencia en el texto de entrada. Un modelo consta de uno o más literales de carácter, operadores o estructuras. Para obtener una breve introducción, consulte Expresiones regulares de .NET.

Cada sección de esta referencia rápida enumera una categoría determinada de caracteres, operadores y construcciones que puede usar para definir expresiones regulares.

Existen 3 tipos de expresiones regulares unión, concatenación y cerradura

Unión de dos idiomas L y M se escribe como
LUM = {s | s en Los es en M}
La concatenación de dos lenguajes L y M se escribe como
LM = {st | s es en L yt se encuentra en M}

Concatenación: (r)(s) es una expresión regular que denota L(r)L(s)
Si *E* y *F* son expresiones regulares, entonces *EF* es una expresión regular que representa la concatenación de *L*(*E*) y *L*(*F*).

Cerradura o también conocida como clausura de Kleene

Ejemplo de clausura de Kleene aplicada a un carácter:

$$\{a\}^* = \{\lambda, a, aa, aaa, aaaa, aaaaa, aaaaaa, \ldots\}$$

Ejemplo de clausura de Kleene aplicada a un conjunto de cadenas:

$$\{ab, c\}^* = \{\lambda, ab, c, abab, abc, cab, cc, ababab, ababc, abcab, abcc, cabab, cabc, ccab, ccc, \ldots\}$$

Ejemplo de clausura de Kleene aplicada a un conjunto de caracteres:

$$\{a, b, c\}^* = \{\lambda, a, b, c, aa, ab, ac, ba, bb, bc, \ldots\}$$

II.- Explicar el proceso de conversión de DFA a expresiones regulares.

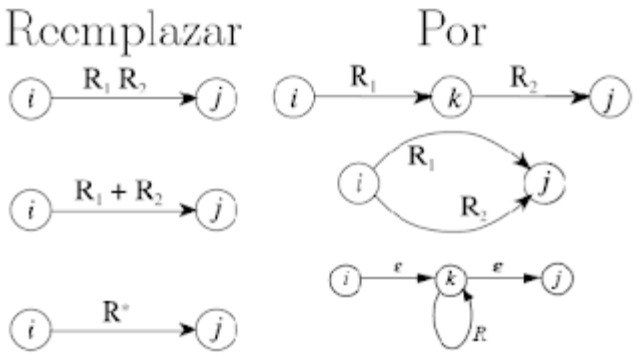
1 - Eliminar estados inalcanzables: Antes de comenzar la conversión, es importante eliminar cualquier estado inalcanzable en el DFA. Esto implica eliminar cualquier estado que no sea accesible desde el estado inicial.

2 - Crear un conjunto de ecuaciones de expresiones regulares: Para cada par de estados i y j en el DFA, se define una expresión regular que describe el conjunto de cadenas que llevan de i a j directamente. Estas expresiones regulares pueden ser representadas como una matriz de expresiones regulares.

3 - Resolver las ecuaciones de expresiones regulares: Después de crear las ecuaciones de expresiones regulares, el siguiente paso es resolverlas. Esto se puede hacer utilizando un algoritmo de eliminación de estados, que es una forma de resolver sistemas de ecuaciones lineales. El objetivo es encontrar una expresión regular para cada par de estados que describe todas las cadenas posibles entre ellos.

4 - Obtener la expresión regular final: Una vez resueltas todas las ecuaciones, se obtiene una expresión regular que describe el lenguaje aceptado por el DFA completo. Esta expresión regular puede ser la que describe el conjunto de cadenas que llevan del estado inicial a cualquiera de los estados de aceptación.

Es importante destacar que la conversión de DFA a expresiones regulares no es trivial y puede requerir un conocimiento sólido de teoría de autómatas y álgebra de expresiones regulares. Además, el proceso puede variar dependiendo de la complejidad del DFA y de la técnica utilizada para resolver las ecuaciones de expresiones regulares.



III.- Explicar leyes algebraicas de expresiones regulares.

Las "leyes algebraicas" de las expresiones regulares son reglas fundamentales que rigen la manipulación y la simplificación de las expresiones regulares. Estas leyes son similares a las leyes algebraicas en álgebra booleana y están diseñadas para ayudar en la simplificación y comprensión de las expresiones regulares. Aquí hay algunas de las leyes más comunes:

1. **Ley de la identidad**:
   - $R + \varnothing = R$
   - $R \cdot \varepsilon = R$
2. **Ley de la anulación**:
   - $R + \emptyset = R$
   - $R \cdot \emptyset = \emptyset$
3. **Ley del cero**:
   - $R \cdot \emptyset = \emptyset$
   - $\emptyset \cdot R = \emptyset$
4. **Ley de la absorción**:
   - $R + (R \cdot S) = R$
   - $R \cdot (R + S) = R$
5. **Ley de la idempotencia**:
   - $R + R = R$
   - $R \cdot R = R$
6. **Ley de la distribución**:
   - $R \cdot (S + T) = R \cdot S + R \cdot T$
   - $(S + T) \cdot R = S \cdot R + T \cdot R$
7. **Ley de De Morgan**:
   - $\overline{R + S} = \overline{R} \cdot \overline{S}$
   - $\overline{R \cdot S} = \overline{R} + \overline{S}$
8. **Ley de complemento**:
   - $R + \overline{R} = \Sigma$ (donde Σ es el conjunto de todos los caracteres)
   - $R \cdot \overline{R} = \emptyset$

Estas leyes son útiles para simplificar expresiones regulares, para entender su comportamiento y para realizar operaciones como la unión, la concatenación y la clausura de Kleene de manera más eficiente. Al igual que las leyes algebraicas, las leyes de las expresiones regulares proporcionan reglas claras y consistentes para manipular y analizar expresiones regulares en el contexto de la teoría de autómatas y la teoría de lenguajes formales.

BIBLIOGRAFIA

https://learn.microsoft.com/es-es/dotnet/standard/base-types/regular-expression-language-quick-reference

https://www.tutorialspoint.com/es/compiler_design/compiler_design_regular_expressions.htm

https://cs.famaf.unc.edu.ar/~hoffmann/md18/10.html

https://www.wikiwand.com/es/Clausura_de_Kleene

https://www.luzem.com/2010/04/27/practica-7-conversiones-automatas-lenguajes-regexp-en-jflap/