UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
CAMPUS 1

ING. EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE

6 "M"

COMPILADORES

SUBCOMPETENCIA 1 – ANÁLISIS LÉXICO

DEFINE LOS SIGUIETNES CONCEPTOS Y REALIZAR LOS EJERCICIOS. – ACTIVIDAD I, II.

ALUMNO: MARCO ANTONIO ZÚÑIGA MORALES – A211121

DOCENTE: DR. LUIS GUTIÉRREZ ALFARO

TUXTLA GUTIÉRREZ, CHIAPAS
SÁBADO, 12 DE AGOSTO DE 2023

Marco Antonio Zúñiga Morales
Compiladores

Fecha: 07/Agosto/2023
6to "M"

Definir los siguentes Conceptos

* Definir el concepto de expresión regular
* Explicar los tipos de operadores de expresiones regulares
* Explicar el proceso de conversión de DFA a expresiones regulares
* Explicar leyes algebraicas de expresiones regulares

Expresión regular

Una expresión regular, también conocida como regex o regexp, es una secuencia de caracteres que define un patrón de busqueda en cadenas de texto. Se utiliza principalmente en programación y procesamiento de texto para realizar operaciones de búsqueda, extracción, manipulación o validación de cadenas de caracteres de acuerdo con ciertas reglas predefinidas

Las expresiones regulares son un equivalente algebraico para un autómata. Utilizado en muchos lugares como un lenguaje para describir patrones en texto que son sencillos pero muy útiles

Sirven como lenguaje de entrada a muchos sistemas que procesan cadenas tales como:

* Comandos de búsquda, e.g., grep de UNIX
* Sistemas de formato de texto
* Convierte la expresión regular a un DFA o un NFA y simula el autómata en el archivo de búsqueda
* Generadores de analizadores-léxicos. Como Lex o Flex
* Los analizaderes léxicos son parte de un compilador
* Produce un DFA que reconoce el token.

Ejemplo

[0-9] representa cualquier dígito numérico y puede coincidir con cualquier de los dígitos del 0 al 9 en una cadena de texto.

Marco Antonio Zuñiga Morales
Compiladores

07/Agosto/2023
6 "M"

Los operadores de expresiones regulares

Los operadores en las expresiones regulares son los metacaracteres que se utilizan para definir patrones más complejos y flexibles en las cadenas de texto.

Comunmente existen tres operadores de las expresiones regulares: Unión, concatenación y cerradura.

Unión
L y M son dos lenguajes, su unión se denota como L U M e.g. $L = \{000, 10, 111\}$ $M = \{\varepsilon, 001\}$, entonces la unión será $L \cup M = \{\varepsilon, 10, 001, 111\}$

Concatenación
Se denota como LM o L.M e.g. $L = \{001, 10, 111\}$, $M = \{\varepsilon, 001\}$, entonces la concatenación será $LM = \{001, 10, 111, 001001, 10001, 111001\}$.

Cerradura o cerradura de Kleene
de un lenguaje L se denota como L^*. Representa el conjunto de cadenas que puede que pueden formarse tomando cualquier número de cadenas de L, posiblemente con repeticiones y concatenando todas ellas e.g. Si $L = \{0, 1\}$, L^* son todas las cadenas con 0's y 1's. Si $L = \{0, 11\}$ entonces L^* son todas las cadenas de 0's y 1's tal que los 1's estan en pareja.

Operadores más comunes en las expresiones regulares.

- Operadores de Cuantificación

* '*': Coincide con cero o más repeticiones del elemento anterior. Ejp a* coincide con a, aa, aaa, y así sucesivamente
* +: Coincide con una o más repeticiones del elemento anterior Ejp a+ coincide con a, aa, aaa, Pero no con una cadena vacia

* ? Coincide con cero o una sola repetición del elemento anterior Ejp. a? Coincide con "a" o una cadena vacía.
* {n} Coincide exactamente con "n" repeticiones del elemento anterior Ejp. a{3} Coincide con "aaa"
* {n,} Coincide con almenos "n" repeticiones del elemento anterior
* {n,m} Coincide con un mínimo de "n" y un máximo de "m" repeticiones del elemento anterior.

- Operadores de Caracteres Especiales

* . Coincide con cualquier carácter excepto el salto de línea.
* \ Se utiliza para escapar Caracteres especiales y tratarlos como literales. Ejp \= coincide con el carácter "="
* [ ] Define una clase de caracteres, que especifica un conjunto de caracteres posibles. Ejp [aeiou] coincide con cualquier vocal
* [^] Define una clase de caracteres negados, que coincide con cualquier carácter que no esté en el conjunto especificado Ejp [^0-9] coincide con cualquier carácter que no sea un dígito numérico

- Operadores de Agrupación y Alternancia

* ( ) Crea un grupo de caracteres y permite aplicar operadores a ese grupo. También permite referirse al contenido del grupo más adelante en la expresión
* | Representa una alternancia, es decir, una opción entre dos o más patrones. Ejp (apple|orange) Coincide con "apple" o "orange".

Marco Antonio Zuñiga Morales
Compiladores

07/Agosto/2023
6"M"

- Anclajes
  - * ^ Coincide con el inicio de una línea o cadena
  - * $ Coincide con el final de una línea o cadena.

Explicar el proceso de conversión de DFA a expresiones regulares.

Por eliminación de estados

Evita duplicar trabajo en algunos puntos

* Se eliminan todos los arcos que incluyen a "s"
* Se introduce, Para cada predecesor q_i de s y cada sucesor P_j de s, una RE que representa todas las rutas que inician en q_i van a s, quizás hacen un loop en s cero o más veces, y finalmente van a P_j.
* La expresión para estas rutas es $Q_i S^* P_j$
* Esta expresión se suma (con el operador union) al arco que va de q_i a P_j.
* Si este arco no existe, se añade primero uno con la RE $\emptyset$
* El automata resultante

Proceso Paso a Paso

1. Identificar el DFA original
   Comienza con el DFA que deseas convertir a una expresión regular. El DFA consta de estados, simbolos de entrada, transiciones y estados finales

2. Agregar un estado de inicio ficticio
Si el DFA original no tiene un único estado inicial, Crea un nuevo estado inicial ficticio y agrega transiciones ε (transiciones nulas) desde este estado hacia todos los estados iniciales originales

3. Eliminar estados finales
Crea un nuevo estado final ficticio y agrega transiciones ε desde los estados finales originales hacia este nuevo estado final.

4. Eliminar estados intermedios
En este paso comenzamos a eliminar gradualmente los estados intermedios mientras deducimos las expresiones regulares asociadas a las transiciones

5. Actualizar las transiciones y estados
Después de calcular las expresiones regulares para cada par de estados, actualiza las transiciones y estados del DFA.

6. Simplificar la expresión regular final
Una vez que hayas reducido el DFA a una única expresión regular que conecta el estado inicial ficticio, puedes simplificar aún más esta expresión utilizando propiedades algebraicas de las expresiones regulares, como la distribución, la absorción y la identidad

7. Obtener la expresión regular final
La expresión regular final representa el lenguaje original del DFA. Esta expresión regular puede ser más larga y compleja, por lo que es importante verificar cuando sea posible

Otros Procesos

* Crear estados de eliminación (Construcción de Thompson)
* Construcción de fragmentos de expresiones regulares
* Eliminación de estados intermedios
* Obtención de la expresión regular final
* Simplificación adicional

Marco Antonio Zúñiga Morales
Compiladores

07/Agosto/2023
6"M"

Explicar leyes algebraicas de expresiones regulares

Existen un conjunto de leyes algebraicas que se pueden utilizar para las expresiones regulares

* Ley conmutativa para la unión : $L+M = M+L$
* Ley asociativa para la unión : $(L+M)+N = L+(M+N)$
* Ley asociativa para la concatenación : $(LM)N = L(MN)$

- Ley de Idempotencia
  Una expresión regular combinada con sí misma produce el mismo resultado EJP
  $A|A = A$ y $A.A = A$
- Ley de Anulación
  La concatenación de una expresión regular con la expresión vacía (ε) produce la misma expresión regular EJP $A.\varepsilon = A$
- Leyes relacionadas con las propiedad de cerradura
  * $(L^*)^* = L^*$ (idempotencia para la cerradura)
  * $\emptyset^* = \varepsilon$
  * $\varepsilon^* = \varepsilon$
  * $L^+ = LL^* = L^*L$, L^+ se define como $L+LL+LLL+\ldots$
  * $L^* = \varepsilon + L + LL + LLL + \ldots$
  * $LL^* = L\varepsilon + LL + LLL + LLLL + \ldots$
  * $L^* = L^+ + \varepsilon$
  * $L? = \varepsilon + L$

Marco Antonio Zúñiga Morales
Compiladores

07 / Agosto / 2023
6 "M"

* Ley distributiva

Ley distributiva izquierda para la concatenación sobre unión: $L(M+N) = LM + LN$

Ley distributiva derecha para la concatenación sobre unión $(M+N)L = ML + NL$

* Ley de absorción

La expresión regular combinada con la expresión vacía (ε) en una operación de alternancia produce la expresión vacío (ε) EJP $A \mid \varepsilon = A$ y $\varepsilon \mid A = A$

* Ley de identidad
La concatenación de una expresión regular con ε no cambia la expresión original EJP $A . \varepsilon = A$

* Ley de Clausura
La expresión regular combinada con ε en una operación de clausura produce una expresión regular que incluye la repetición de la expresión original EJP $A^* = A . A^*$

* Ley de Absorción de clausura
La clausura de una expresión regular A incluye ε, por lo que $A^* = \varepsilon \mid A . A^*$

* Ley de complemento
El complemento de una expresión regular A es equivalente a $(\Sigma^* - A)$, donde Σ^* es el conjunto de todas las cadenas posibles EJP $-A = \Sigma^* - A$

* Ley de Dominancia
La expresión regular combinada con la expresión vacía (ε) en una operación de concatenación se reduce a la propia expresión regular EJP $A . \varepsilon = A$

Marco Antonio Zúñiga Morales
Compiladores

07 | Agosto | 2023
6 M

A.F.D

Un autómata Finito deterministico (AFD) es un concepto fondamental en la teoría de autómatas y lenguajes formales. Es un modelo matemático abstracto utilizado para describir y reconocer patrones y cadenas de símbolos en un lenguaje especifico

un AFD se caracteriza por los siguientes elementos:

* Conjunto de Estados (Q)
  Un conjunto finito de estados en las que se puede encontrarse el autómata

* Alfabeto (Σ)
  Un conjunto finito de símbolos de entrada que el automata puede leer en cada paso

* Función de Transición (δ)
  Una función que especifica cómo el autómata cambia de un estado a otro en respuesta a un símbolo de entrada

* Estado inicial (q_0)
  El estado en el que el autómata se encuentra inicialmente cuando comienza a procesar una cadena.

* Conjunto de Estados Finales (F)
  Un conjunto de estados en los que el autómata considera que la cadena de entrada ha sido aceptado y el proceso de reconocimiento termina

Ejemplo

AFD que reconoce cadenas con un número par de ceros (00) y un número impar de unos (1)

- Estados : q_0, q_1
- Símbolos de entrada : 0, 1
- Estados iniciales : q_0
- Estados finales : q_0

Transiciones

- $q_0 \rightarrow 0q_1$
- $q_1 \rightarrow 0q_0$
- $q_0 \rightarrow 1q_1$
- $q_1 \rightarrow 1q_0$

→ $q_0 \rightarrow 0q_1 \mid 1q_1$
$q_1 \rightarrow 0q_0 \mid 1q_0$

Ejemplo de Expresiones Regulares

* `^[A-Za-z]+$`

Esta expresión regular valida si una cadena contiene solo letras mayúsculas o minúsculas y no contiene espacios ni otros caracteres.

* `(a+b)* b`

La E.R. del lenguaje cuyas palabras están formadas por las letras del alfabeto V = {a, b} y terminan en "b"

* `^\d{3}-\d{2}-\d{4}$`

Esta expresión regular valida un número de seguro social en formato ###-##-####, donde cada "#" representa un dígito numérico

Token

Un token es una unidad léxica básica que representa una categoría de elementos en un lenguaje de programación

Lexema

Es la secuencia completa de caracteres que forma un token en un programa

Patrón

Un patrón es una descripción formal de cómo se ven los lexemas válidos para un tipo particular de token.

Referencias Bibliográficas

Expresiones regulares - JavaScript | MDN. (2023) https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_expressions

Expresiones regulares. Conócelas y pierdeles el miedo. (s.f.) SG Buzz. https://sg.com.mx/content/view/545

INade (s.f.). Teoría de Autómatas y Lenguajes Formales Expresiones regulares y lenguajes Recuperado de https://Posgrados.inaoep.mx/archivos/PosCsComputacionales/curso_Propedeutico/Automatas/03_Automatas_expresiones RegularesLenguajes/Captul1.PDF

Ruiz, L. (2003). Conversión de un AFN a un AFD. Universidad Nacional Mayor de San Marcos Lima, Perú. Recuperado de https://www.redaly.org/PdF/816/81606107.PdF

Expresiones Regulares (2015). Inaoep. Recuperado de https://CCC.inaoep.mx/Ingreso/Automatas/expresiones Regulares.pdf

Sánchez, S (s.f) Procesadores de lenguaje. Departamento de Ciencias de la Computación Universidad de Alcalá de Henares. Recuperado de www.cc.uah.es/ie/docencia/Procesadores de Lenguaje/Procesadores de lenguaje Tema 2 - 1xpagina.PdF