TRABAJO PRÁCTICO Nº 2

CONSTRUCCION DE THOMPSON Y CONVERSIÓN DE AFN EN AFD

IMPLEMENTACIÓN CON PYTHON PARA VALIDACIÓN DE DATOS MEDIANTE EXPRESIONES REGULARES

Contenido Conceptual

Autómatas finitos deterministas y no deterministas, aceptación de cadenas.

Obtención de autómatas a partir de expresiones regulares. Conversión de AFN en AFD.

Expresiones regulares y definiciones regulares.

Objetivos

- Identificar y diferenciar autómatas finitos deterministas y no deterministas.
- Lograr la capacidad para representar e implementar autómatas que reconozcan patrones en cadenas de entrada para obtener los componentes léxicos a partir de expresiones regulares.
- Obtener la habilidad para construir autómatas finitos no deterministas mediante "Construcción de Thompson" y convertirlos en autómatas finitos deterministas.
- Adquirir la habilidad para identificar y crear expresiones regulares que definan cadenas.

PARTE A

Conceptos fundamentales: autómatas finitos deterministas y no deterministas, aceptación de cadenas, obtención de autómatas a partir de expresiones regulares, y conversión de AFN en AFD.

DEFINICIONES

Diagrama de transiciones representa las acciones que tienen lugar cuando el analizador léxico es llamado por el analizador sintáctico para obtener el siguiente componente léxico.

Construcción del diagrama de transiciones*:*

- Las *posiciones* en un diagrama de transición se representan con un círculo y se llaman *estados*.
- Los *estados* se conectan mediante flechas, llamadas *aristas.*
- Las *aristas* tienen etiquetas que indican los caracteres de entrada.
- El *estado de inicio* es el estado inicial del diagrama de transición.
- El *estado de aceptación* es el estado en el cual se ha encontrado un token y se indica con un círculo doble.

Autómatas finitos: o máquinas de estados finitos, son una forma matemática para describir clases particulares de algoritmos (o "máquinas"). Se pueden utilizar para describir el proceso de reconocimiento de patrones en cadenas de entrada, y de este modo construir analizadores léxicos. Se representan a través de **diagramas de transiciones**.

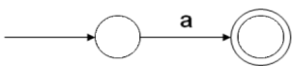
Autómatas finitos determinista AFD: Se limitan a aceptar o no una determinada cadena recibida en la entrada. La salida solo tendrá dos valores posibles aceptar o no aceptar la entrada. Deben tener una transición para cada estado y carácter. Aquellas transiciones que dan errores como resultado, se dejan fuera del diagrama para el AFD. La definición de un AFD no especifica lo que ocurre cuando se presenta un error. No especifica la acción que tomará un programa al alcanzar un estado de aceptación.

Autómatas finitos no determinista AFN: Pueden tener ninguna, una o más transiciones de un estado sobre el mismo símbolo de entrada. Pueden tener transiciones vacías o transiciones ε.

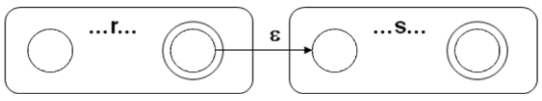
Construcción de Thompson permite la obtención de cada operación de la expresión regular al conectar entre sí los AFN de las subexpresiones.

Representaciones de la Construcción de Thompson:

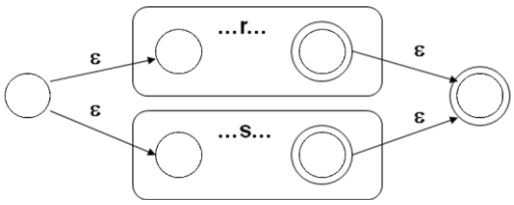
- *Expresiones regulares básicas:* a representa una correspondencia con un carácter simple del alfabeto



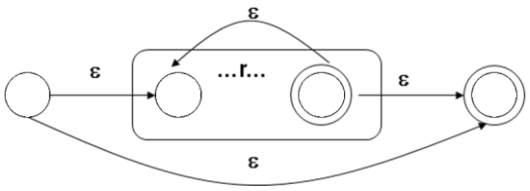
- *Concatenación:* AFN para la expresión regular rs , donde r y s son expresiones regulares.



- *Selección de alternativas*: AFN para la expresión regular *r*|*s*, donde r y s son expresiones regulares.



- *Repetición o cerradura de Kleene:* AFN para la expresión regular *r**, donde r es una expresión regular.



Construcción del subconjunto:

Para la construcción de un AFD a partir de un AFN M dado, M

- Calcularnos la cerradura ε del estado de inicio de M, esto se convierte en el estado de inicio de M.
- Para este conjunto, y para cada conjunto subsiguiente, calculamos las transiciones en los caracteres.

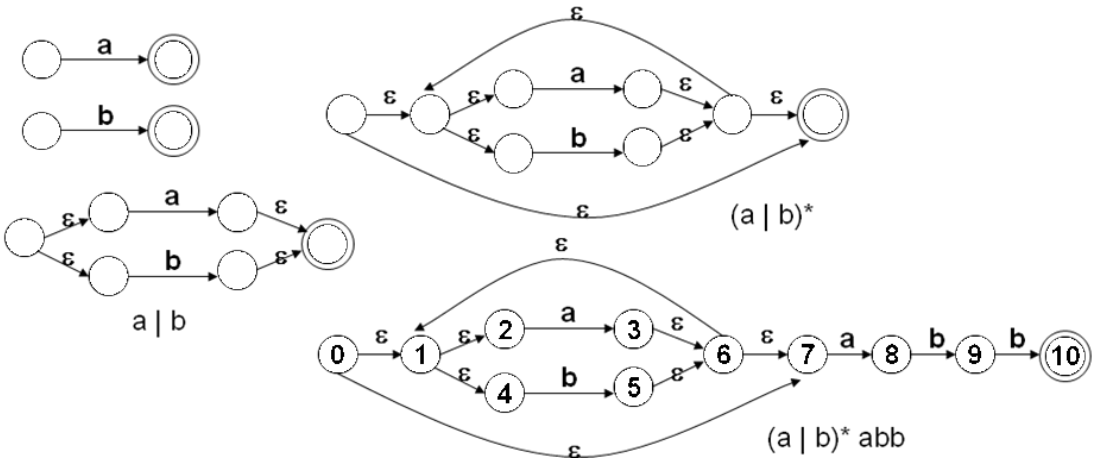
EJERCICIOS

Ejercicio 1:

Cree, mediante "Construcción de Thompson", los siguientes autómatas finitos no deterministas:

Ejemplo:

Dada la expresión regular (a | b)* abb.



a) Que reconozca la expresión regular x (x | y) *.

b) Que reconozca la expresión regular (c | a c) *.

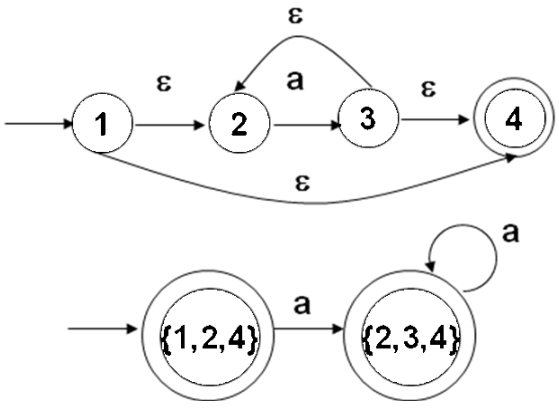
c) Que reconozca la expresión regular (b | (b* a)*)a.

Ejercicio 2:

Convierta, utilizando construcción de subconjuntos, los autómatas finitos no deterministas realizados en el ejercicio anterior, en autómatas finitos deterministas.

Ejemplo:

NFA que corresponde a la expresión regular a* bajo la construcción de Thompson.



- El estado de inicio del DFA correspondiente es
  $\overline{1} = \{1,2,4\}$
- Existe una transición desde el estado 2 hasta el estado 3 en a, y no hay transiciones desde los estados 1 o 4 en a
  $\overline{\{1,2,4\}}_a = \{3\} = \{2,3,4\}$
- Existe una transición desde 2 a 3 en a y ninguna transición a desde 3 o 4, de modo que hay una transición desde {2, 3, 4} hasta
  $\overline{\{2,3,4\}}_a = \{3\} = \{2,3,4\}$
- Existe una transición a desde {2, 3, 4} hacia sí misma.

PARTE B

Conceptos fundamentales: Expresiones regulares y definiciones regulares.

DEFINICIONES

Expresiones regulares: representan patrones de cadenas de caracteres. Una expresión regular r se encuentra completamente definida mediante el conjunto de cadenas con las que concuerda.

Expresiones regulares básicas:

- Son los caracteres simples del alfabeto.
- Dado cualquier carácter a del alfabeto Σ, indicamos que la expresión regular a corresponde al carácter a escribiendo L(a) = {a}.
- El símbolo ε denota la cadena vacía, L(ε) = { ε }.
- El símbolo { } corresponde a la ausencia de cadenas, cuyo lenguaje sea el conjunto vacío ϕ, L{ϕ}= { }.
- El conjunto { } no contiene ninguna cadena, y el conjunto { ε } contiene la cadena simple que no se compone de ningún carácter.

Operaciones de expresiones regulares:

- **Selección entre alternativas**: se indica mediante el metacarácter | (barra vertical). Si *r* y *s* son expresiones regulares, entonces *r* | *s* es una expresión regular que define cualquier cadena que concuerda con *r* o con *s*.
- **Concatenación**: se indica mediante yuxtaposición, sin un metacarácter. La concatenación de r y s, rs, corresponde a cualquier cadena que sea la concatenación de dos cadenas, con la primera r y la segunda s.
- **Repetición**: se indica mediante el metacarácter *. La expresión regular r* corresponde a cualquier concatenación finita de cadenas, cada una de las cuales corresponde a r.

EJERCICIOS

Ejercicio 1

Escriba las descripciones para los lenguajes generados por las siguientes expresiones regulares:

1. (ca|b)*(a|cc)*
2. (0|1|.. .|9|A|B|C|D|E|F)(w|W)
3. (A|B|…|Z)(a|b|…z)*

Ejemplo:

> Dada la siguiente expresión regular (a|b)*a(a|b|ε), el lenguaje generado por la misma es: Todas las cadenas de letras minúsculas que comienzan con una o más a o una o con una o más b.

Ejercicio 2

Escriba las expresiones regulares para validar:

1. Fecha con formato dd/mm/yyyy o dd-mm-yyyy
2. Número real con dos decimales y separador de miles.
3. ID de un vídeo de Youtube
4. Cuenta de Email de alumno de la Universidad de Mendoza
5. Número de teléfono móvil de Argentina, que incluya código de país, de provincia, y el 15.
6. CUIL.
7. Seguridad de una contraseña, que incluya:
   a. Que contenga al menos un número.
   b. Que contenga al menos una letra mayúscula.
   c. Que contenga al menos carácter especial.
   d. La longitud sea como mínimo 8 caracteres.

e. La longitud máxima no debe mayor a 16 caracteres.

Ejercicio 3:

Utilizando el lenguaje de programación Python, implementar la validación de las expresiones regulares del ejercicio anterior (Ejercicio 2)

Ejemplo: validación de extensiones para imágenes.

```
import re

regex = re.compile(r"jpg|png|gif|bmp|svg")

img_ext = input("Ingrese una extensión de una imagen: ")

if regex.match(img_ext):

    print('La extensión ', img_ext, 'se corresponde con la extensón de una imagen')

else:

    print('La extensión ', img_ext, 'no se corresponde con la extensón de una imagen')
```