Laboratorio de Teoría de Lenguajes 2025

Conversión de Expresiones Regulares a AFND-ϵ

Agenda

- Introducción y objetivos del laboratorio.
- Validación de la entrada
- Algoritmo de conversión
- Ejemplo práctico
- Clase AFND_e
- Funciones de comparación (igualdad) y copia de AFND-ϵ
- Conclusiones y discusión

Objetivos del Laboratorio

- **Validar** la expresión regular para asegurarse de que solo contenga:
  - Literales: a, b, c y la tira vacía (e)
  - Operadores: concatenación (.), unión (|) y clausura de Kleene (*).
- **Construir** el AFND-ϵ a partir de la expresión válida.
- **Integrar** la solución con el script de pruebas.

Validación de la Expresión Regular

- Se deberá implementar la función is_entry_valid(expr). Esta función se encarga de verificar que la entrada leída cumple con las siguientes características:
  - Solo contiene los literales: “a”, “b”, “c”, y “e” (este último identifica la tira vacía)
  - Solo contiene los operadores: “|”, “.” y “*”

- Ejemplos de entradas válidas: a.b*|c, a, a|b
- Ejemplos de entradas inválidas: a^b, b*d, (a*), a.(b*|c), *b, |b
- OBSERVACIÓN: Los paréntesis **NO** son caracteres válidos

Biblioteca *re* de Python

- En este laboratorio una de las bibliotecas requeridas es la biblioteca *re* proporcionada por Python
- Contiene funciones para búsqueda de patrones utilizando expresiones regulares
- Una posible expresión regular de entrada es: a*.b|c
- Esta expresión regular reconoce los lenguajes que comienzan con cero o más “a” y luego tienen una “b” o simplemente el terminal “c”

Funciones de la biblioteca *re* de Python

- **re.match**(r"Hello", "Hello World") - Comprueba si el patrón coincide desde el inicio de la cadena.
- **re.search**(r"World", "Hello World") - Busca el patrón en cualquier parte de la cadena.
- **re.fullmatch**(r"\d{3}-\d{2}-\d{4}", "123-45-6789") - Verifica que toda la cadena se ajuste al patrón.
- **re.findall**(r"\d+", "There are 12 apples and 34 oranges") - Devuelve una lista con todas las coincidencias del patrón en la cadena.
- **re.sub**(r"World", "Python", "Hello World") - Sustituye todas las ocurrencias del patrón por un nuevo string.

Secuencias de escape

En python se utilizan las siguientes secuencias:

- **\d**: Coincide con cualquier dígito
- **\D**: Coincide con cualquier carácter que **no** sea un dígito
- **\w**: Coincide con cualquier carácter alfanumérico
- **\W**: Coincide con cualquier carácter que **no** sea alfanumérico
- **\s**: Coincide con cualquier carácter de espacio en blanco
- **\S**: Coincide con cualquier carácter que **no** sea un espacio en blanco.

Uso en el Laboratorio

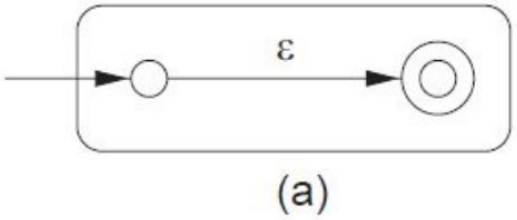
Se utilizará la biblioteca *re* para resolver la función *is_entry_valid(expr)*

ES OBLIGATORIO UTILIZAR ESTA BIBLIOTECA CON SUS FUNCIONES Y NO ESTÁ PERMITIDO EL USO DE NINGUNA OTRA LIBRERÍA PARA RESOLVER ESTA PARTE DEL LABORATORIO

Se podrán utilizar funciones de la librería que no hayan sido nombradas en esta presentación así como cualquier facilidad que pueda proveer la librería *re*.

Algoritmo de Conversión a AFND-ϵ

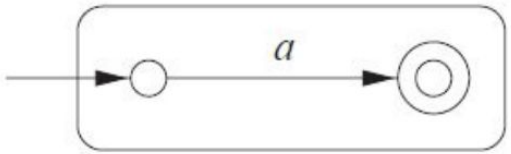
Se utilizará el algoritmo presentado en el teórico para realizar esta parte del laboratorio.



(a)

$r = \varepsilon$

$L(r) = \{\varepsilon\}$

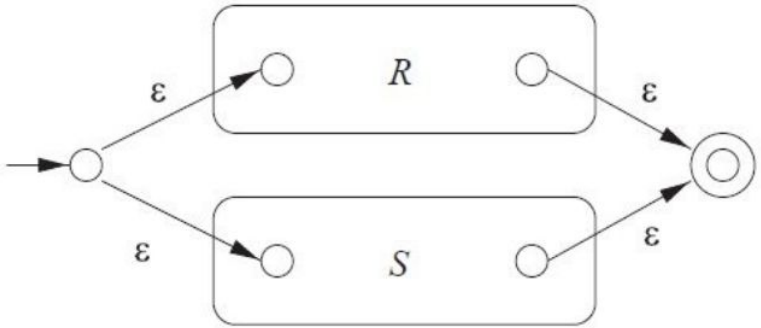


(c)

$r = a \quad \forall\, a \in \Sigma$

$L(r) = \{a\}$

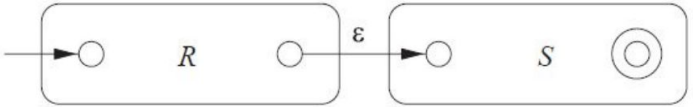
Algoritmo de Conversión a AFND-ϵ



(a)

e = *r* | *s*

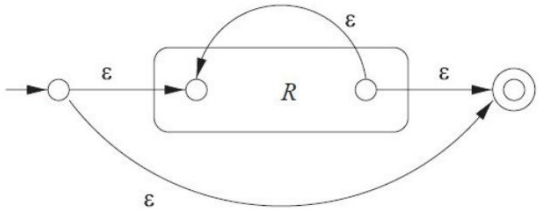
L(e) = R ∪ S



(b)

e = *r* . *s*

L(e) = R . S



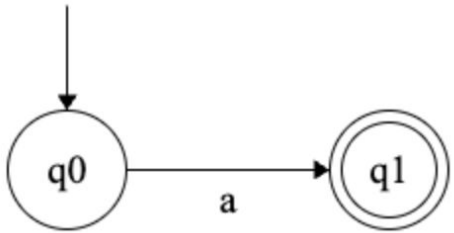
(c)

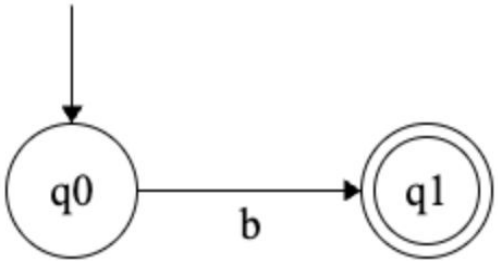
e = *r* *

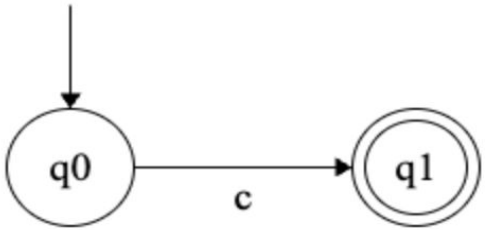
L(e) = R*

Ejemplo práctico: a.b*.c

- Descomponer la expresión:
  - Literal “a”
  - Literal “b” con Kleene (*)
  - Literal “c”

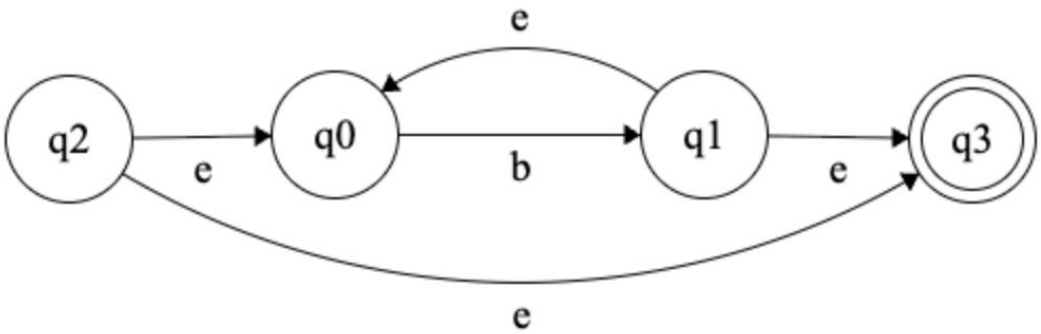






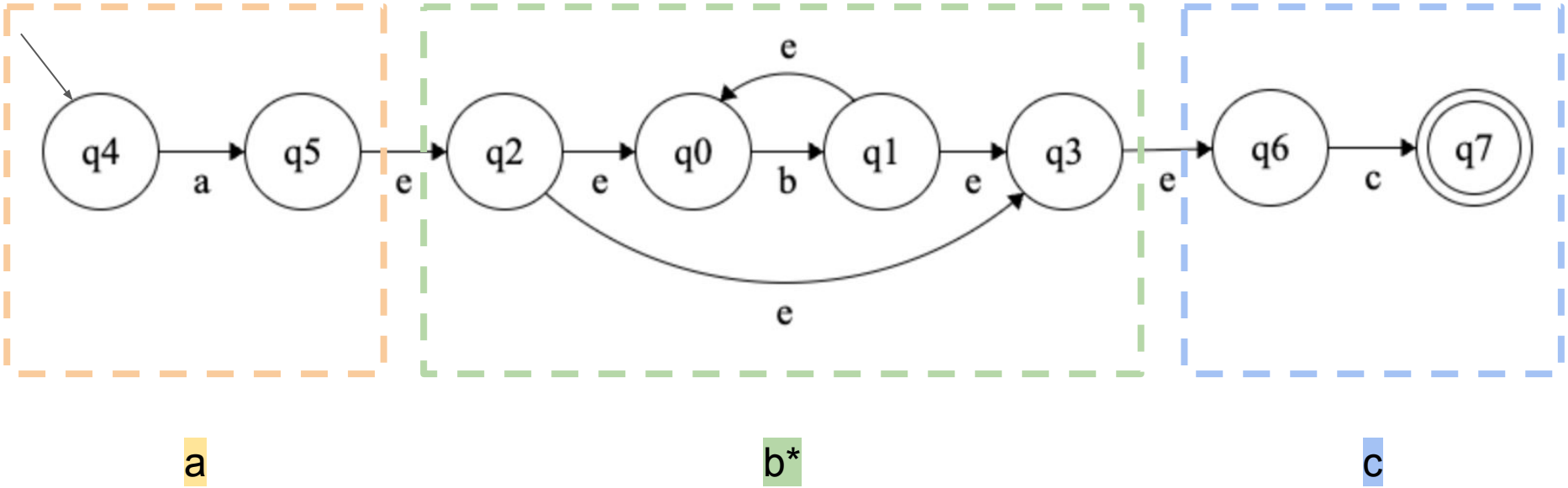
Ejemplo práctico: a.b*.c

- Clausura de Kleene



Ejemplo práctico: a.b*.c

- Concatenación



Clase AFND_e

- Dentro de los materiales del laboratorio tendrán **implementada** la clase AFND_e, esta clase tiene las siguientes funciones:
  - agregar_estado: que recibe el nombre de un estado y lo agrega al afnd_e
  - establecer_inicial: que recibe el nombre de un estado y lo marca como inicial
  - agregar_final: que recibe el nombre de un estado y lo marca como final
  - agregar_transicion: que recibe el nombre del estado del cual parte la transición, el símbolo y el nombre del estado al cual llega la transición
  - mostrar: esta función imprime en pantalla una representación del AFND_e
  - son_isomorfos: es un método estático que recibe dos AFND_e y chequea si son iguales, a excepción de renombramiento de estados

Función regex_to_afnde(expr)

- **Precondición**: La expresión ya pasó la validación con is_entry_valid.
- **Pasos del algoritmo**:
  - *Parsing* de la expresión en componentes (literales y operadores).
  - Construcción de autómatas parciales y su combinación.
- Uso de la pila para administrar la operación de concatenación, unión y Kleene.
- Uso de la función nuevo_estado() para obtener nuevos estados de los autómatas.
- **Resultado**: Un objeto de la clase AFND_e que representa el autómata final.

Función de copia del AFND-ϵ

- copiar(dest, fuente): toma un AFND_e destino y un AFND_e fuente y copia las transiciones de fuente en destino
- Pre-condición: si un estado en el AFND_e destino tiene el **MISMO NOMBRE** que un estado en el AFND_e fuente entonces **SON EL MISMO ESTADO.**
- Debe utilizarse durante la ejecución del algoritmo de pasaje a afnd-e para simplificar el procedimiento.

Ejecución de pruebas

- El script de pruebas se ejecuta con el comando python3 test.py
- Este comando ejecutará los tests e imprimirá en pantalla para cada caso si pasó la validación la expresión regular de la entrada y en caso afirmativo mostrará el AFND_e generado con su estado inicial y final.
- Mostrará para cada caso si la ejecución fue exitosa o no y un recuento de los casos correctos sobre los casos totales. Deben pasarse todos los tests públicos.
- A la hora de evaluar el laboratorio entregado se correrán tests que no son conocidos por el alumno llamados casos privados.

Recapitulación y Conclusiones

- El **primer paso** es implementar la función is_entry_valid que dada una expresión regular devuelve un booleano (True o False) que indica si la expresión es válida (sólo contiene los caracteres a, b, c, e, ., * y |).
- El **segundo paso**, luego de la validación de la expresión regular es realizar el pasaje a AFND_e con el algoritmo visto en el curso, utilizando las funciones de la clase AFND_e que es parte de los materiales.
- Como **último paso** debe ejecutarse el script de pruebas y todos los casos deben ser exitosos.

Entrega

- La entrega deberá realizarse en grupos de entre **2 y 4 integrantes. No se aceptarán entregas individuales.**
- La fecha límite de entrega es a las **23:59 del 24 de abril. No se aceptarán entregas pasado el límite.**
- Habrá un foro para armar grupos.
- Habrá un foro para preguntar específicas del laboratorio, recuerden que **NO SE PUEDE COMPARTIR CÓDIGO** a través de este foro.

PREGUNTAS?