Alan Reyes-Figueroa Teoría de la Computación

(Aula 03b) 10.julio.2023

Expresiones regulares Árboles sintácticos Notación Polaca Reversa

# Operaciones para lenguajes

Dados lenguajes L, L<sub>1</sub> y L<sub>2</sub>, tenemos

- ◆La *concatenación*  $L_1L_2$  es el lenguaje  $L_1L_2 = \{w_1w_2: w_1 \in L_1, w_2 \in L_2\}.$
- La *unión*  $L_1 \cup L_2$  es el lenguaje  $L_1 \cup L_2 = \{w: w \in L_1 \text{ ó } w \in L_2\}.$
- La *cerradura de Kleene* de L es  $L^* = \{w_1w_2 ... w_n : w_i \in L, n \ge 0\}.$

# Operaciones para lenguajes

La cerradura positiva de L es el lenguaje

$$L^+ = \{w_1 w_2 ... w_n : w_i \in L, n \ge 1\}.$$

Ejemplo:  $L_1 = \{a, aa\} L_2 = \{\epsilon, b, c\}.$ 

```
L_1 \cup L_2 = \{\epsilon, a, aa, b, c\}.
```

 $L_1L_2 = \{a, ab, ac, aa, aab, aac\}.$ 

$$L_2^* = \{\epsilon, b, c, bb, bc, cc, cb, bbb, bbc, ...\}.$$

Una expresión regular es una representación de un lenguaje (Obs: no de cualquier lenguaje)

Los lenguajes que son representables mediante expresiones regulares se llaman *lenguajes regulares*.

Dado un alfabeto  $\Sigma$ , para representar un lenguaje regular usamos los símbolos en  $\Sigma$ , y ciertos operadores especiales:

- ab ó a b, para la concatenación
- a υ b, ó a | b ó a+b, para la unión
- a\* para la cerradura de Kleene
- a+ para la cerradura positiva
- (, ), [, ], para definir agrupaciones y jerarquías

También se usan otros símbolos como abreviaturas:  $[a_1,a_2, ..., a_n]$   $[a_1 - a_n]$ 

Las expresiones regulares en un alfabeto Σ se construyen siguiendo las

- ε y cualquier elemento de Σ es una expresión regular.
- 2. Si a y β son expresiones regulares, también lo es aβ.
- 3. Si a y β son expresiones regulares, también lo es a | β.
- 4. Si a es expresión regular, también lo son a\* y a+.
- 5. Sólo las reglas 1-4 generan expresiones regulares.

### Precedencia de operadores

- 1. () operadores de agrupación se aplican primero.
- 2. \* se aplica antes que | y que .
- 3. . Concatenación precede a |.
- 4. | se aplica al final.

#### **Ejemplos:**

```
ab* es equivalente a a(b*)
ab|c es equivalente a (ab)|c
a|b* es equivalente a a|(b*)
```

```
Ejemplo: En \Sigma = \{0,1\}

a = (0|1)*0 es expresión regular.
```

Representa todas las cadenas terminadas en 0.

Si L es el lenguaje representado por la expresión a, en L tenemos las cadenas

```
Ejemplo: En \Sigma = \{a,b\}

\beta = b^*(abb^*)(a|\epsilon) es expresión regular.
```

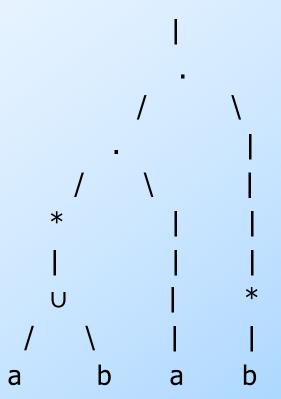
Representa a las cadenas que comienzan con un número cualquiera de b's, luego tiene ab, luego tiene cualquier número de b's luego terminan en a ó en ε.

En el lenguaje L representado por β tenemos ab bab baba babb babb

# Árboles sintácticos

Cada expresión regular tiene asociado un **árbol sintáctico** o (*parse tree*), el cual representa de forma visual la estructura de la regexp

(a|b)\*ab\*



### **Notaciones**

Notación infix: La que usamos regularmente. Los operadores binarios van en medio de los términos.

$$(a | b)^* a b b^*$$

Notación prefix: También se llama notación polaca (PN). Los operadores preceden a los términos.

Notación postfix: También se llama notación polaca revertida (RPN), o notación de Lukasiewicz. Los operadores van después.

# **Ejercicios**

Para cada una de las expresiones regulares siguientes, construir:

- El árbol sintáctico
- La expresión en notación prefix y postfix
- El AFD que representa la expresión
- La tabla de transiciones

# **Ejercicios**

1) 
$$R = a^*$$

Aquí 
$$\Sigma = \{a,b,c\}$$

2) 
$$R = a|b|$$

3) 
$$R = bb^* | a$$

4) 
$$R = abc \mid bac$$

5) 
$$R = (a|b)*$$

6) 
$$R = a^*|b^*|$$