
# Lenguajes

Matemáticas Computacionales

Dr. Víctor de la Cueva
vcueva@itesm.mx

# Strings y Lenguajes

## Definiciones

- Lenguaje: Es un conjunto de strings de longitud finita sobre un alfabeto
- Alfabeto: Es el conjunto finito de símbolos (objetos indivisibles) de un lenguaje
- String: sobre un alfabeto, es una secuencia finita de símbolos de dicho alfabeto

10.5

- **NOTA**: Los lenguajes no son hechos de string arbitrarios
- Lenguaje: Es el conjunto de strings de longitud finita que satisfacen ciertos criterios y restricciones definidas por la sintaxis del lenguaje

## Notación

- Puesto que los elementos de un alfabeto son indivisibles, generalmente se denotan por caracteres simples
- Se acostumbra usar las primeras letras del alfabeto en minúsculas, siguiendo la notación de conjuntos (a, b, c, d, e) con o sin subíndices
- $\Sigma$ se utiliza para denotar el alfabeto
- Un string sobre un alfabeto se representa con las letras minúsculas del fin del alfabeto (p, q, u, v, w, x, y, z) siguiendo la notación de variables en álgebra

# Definición de un lenguaje

- Para poder definir un lenguaje, es decir, los strings que cumplen con las propiedades, se acostumbra definir el conjunto de strings del lenguaje en forma recursiva:
  - Se parte del string nulo ($\lambda$)
  - Se definen ciertas operaciones que se pueden hacer sobre los strings, donde el operador primitivo es la adhesión de un elemento del alfabeto al lado derecho de un string existente.

# Formalmente

- Sea $\Sigma$ un alfabeto. $\Sigma^*$, el conjunto de strings sobre $\Sigma$, se define recursivamente como sigue:
  i. Base: $\lambda \in \Sigma^*$
  ii. Paso recursivo: si $w \in \Sigma^*$ y $a \in \Sigma$, entonces $wa \in \Sigma^*$
  iii. Cerradura: $w \in \Sigma^*$ sólo si puede ser obtenido de $\lambda$ por medio de un número finito de aplicaciones del paso recursivo

43Su

**NOTA**: Para cualquier alfabeto no vacío $\Sigma$, $\Sigma^*$ contiene un número infinito de elementos

## Longitud de un string w

- Intuitivamente: el número de elementos del alfabeto en el string
- Formalmente: el número de aplicaciones del paso recursivo necesario para producir el string a partir de los elementos del alfabeto
- Se denota por length(w)
- Quiz: Si $\Sigma$ contiene n elementos, ¿cuántos strings de longitud k hay?

11.5

- Lenguaje: Es un subconjunto del conjunto de todos los posibles strings sobre un alfabeto
- Def (formalmente): Un lenguaje sobre un alfabeto $\Sigma$ es un subconjunto de $\Sigma^*$

## Concatenación

- Hay muchas operaciones sobre strings que nos ayudan a dar la definición de un lenguaje, al definir las propiedades que deben cumplir sus strings
- Una de las principales es la concatenación
- Def: Sean u, v$\in\Sigma^*$. La concatenación de u y v, escrita como uv, es una operación binaria en $\Sigma^*$ definida como sigue:
  i. Base: si length(v) = 0, entonces v = $\lambda$ y uv = u
  ii. Paso recursivo: Sea v un string con length(v) = n > 0. Entonces, v = wa, para algún string w con length(w) = n-1 y a$\in\Sigma$, y uv = (uw)a

12

# Teorema

Teorema: Sean u, v, w $\in \Sigma^*$. Entonces,

$$(uv)w = u(vw)$$

Nota: se trata de la propiedad asociativa de la concatenación

Demostración (*Proof*): …

44 Su

Nota: se trata de una demostración por inducción

# Notación

- Los paréntesis se omiten en la concatenación
- Se usan exponentes para evitar repeticiones

12.5

# Definición de substring

- Los substrings pueden ser definidos usando concatenación
- Intuitivamente: u es un substring de v si u está dentro de v
- Formalmente: u es un substring de v si existen unos strings x y y, tales que v = xuy
  - Un prefijo de v es un substring u en el cual x es el string nulo en la descomposición de v. Esto es, v = uy
  - Similarmente, u es un sufijo de v si v = xu

# Reversa

- La reversa de un string es el string escrito al revés
- Se puede definir formalmente en forma recursiva como sigue:
- Def: sea u un string en $\Sigma^*$. La revesa de u, denotado $u^R$, se define como:
  i. Base: Si length(u)=0, entonces, $u=\lambda$ y $\lambda^R=\lambda$
  ii. Paso recursivo: si length(u)=n>0, entonces, u=wa para algún string w de longitud n-1 y algún $a \in \Sigma$, y $u^R=aw^R$

12.5

## Teorema

Teorema: Sean u, v $\in \Sigma^*$. Entonces, $(uv)^R = v^R u^R$

Demostración: … inducción sobre la longitud del string v

13
45 Su

# Especificación finita de lenguajes

## Especificación finita de lenguajes

- Los lenguajes de interés no consisten de un conjunto arbitrario de strings sino de los strings que satisfagan ciertos requerimientos sintácticos.
- La especificación de un lenguaje requiere una descripción no ambigua de los strings que pertenecen al lenguaje.
- Muchos lenguaje infinitos con requerimientos sintácticos simples, son definidos usando recursión.
- Desde luego, para requerimientos sintácticos más complejos esta definición recursiva no es adecuada (e.g. lenguaje natural o lenguajes computacionales)

46Su

## Operaciones de conjuntos

- Otra técnica para la construcción de lenguajes es el uso de operaciones de conjuntos para construir conjuntos de strings más complejos a partir de algunos simples.
- Una operación definida sobre strings puede ser extendida a una operación sobre conjuntos o incluso sobre lenguajes.
- Se puede hacer la descripción de lenguajes infinitos a partir de conjuntos finitos usando operaciones con conjuntos.

# Concatenación de lenguajes

Def: La concatenación de los lenguajes X y Y, denotada por XY, es el lenguaje:

$$XY=\{uv \mid u \in X \text{ y } v \in Y\}$$

La concatenación de X con él mismo, n veces, se denota como $X^n$. $X^0$ es definido como $\{\lambda\}$

47 Su

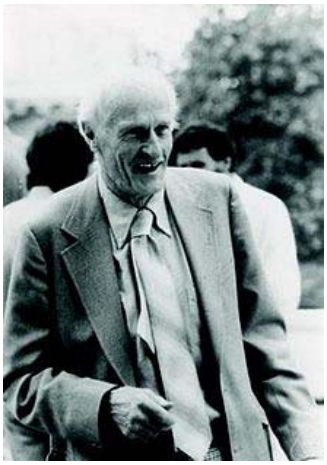
# Kleene Star

- Si para i, $X^i$ contiene los strings de longitud *i* en $\Sigma^*$ entonces la operación de conjuntos llamada Kleene Star sobre un conjunto X, se denota como X* (con el operador *), el cual contiene todos los strings de cualquier longitud, incluyendo $\lambda$.
- Def: Sea X un conjunto, entonces

$$X^* = \bigcup_{i=0}^{\infty} X^i \quad y \quad X^+ = \bigcup_{i=1}^{\infty} X^i$$

- El conjunto X* contiene todos los strings que pueden ser construidos a partir de los elementos de X.
- Si X es un alfabeto, $X^+$ es el conjunto de los strings no nulos sobre X.
- Una definición alternativa para $X^+$ es $X^+=XX^*$

# Stephen Kleene



- Stephen Cole Kleene (n. Hartford, Connecticut, Estados Unidos; 5 de enero de 1909 - f. Madison, Wisconsin; 25 de enero de 1994) fue un lógico y matemático estadounidense.
- Introdujo la operación Clausura de Kleene, denotada por el símbolo V*.
- Desde 1930 a 1935 fue estudiante y asistente investigador en la Universidad de Princeton, donde recibió su doctorado en Matemáticas en 1934, supervisado por Alonzo Church, por una tesis titulada "Una teoría de Enteros Positivos en Lógica Formal".
- En 1935 ingresó en el departamento de matemáticas UW-Madison como instructor. Se convirtió en asistente de profesor en 1937.

Fuente: https://es.wikipedia.org/wiki/Stephen_Kleene

# La ambigüedad

- La definición de un lenguaje formal requiere una especificación no ambigua de los strings que pertenecen al lenguaje.
- Una definición informal (o intuitiva) carece del rigor requerido para una definición precisa. 14 48 Su
- La precisión proporcionada por las operaciones de conjuntos se puede usar para dar una descripción no ambigua de los strings de un lenguaje. 48,49 Su

# Conjuntos y Expresiones Regulares

## Conjuntos y Expresiones Regulares

- Es posible construir un lenguaje a partir de operaciones con conjuntos pero limitando los conjuntos y las operaciones que son permitidas en el proceso de construcción.
- Def (informal): un conjunto de strings es regular se puede ser generado a partir del conjunto vacío, el conjunto que contiene el string nulo y los conjuntos que contienen un solo elemento del alfabeto, usando las operaciones de unión, concatenación y Kleen star.
- Los conjuntos regulares comprenden una familia de lenguajes que juegan un papel muy importante en lenguajes formales, reconocimiento de patrones y teoría de máquinas de estado finito.

## Conjunto regular: Definición (formal)

- Sea $\Sigma$ un alfabeto. Los conjuntos regulares sobre $\Sigma$ son definidos recursivamente como sigue:
  i. Base: $\varnothing$, $\{\lambda\}$ y $\{a\}$, para cada $a \in \Sigma$, son conjuntos regulares sobre $\Sigma$.
  ii. Paso recursivo: Sean X y Y conjuntos regulares sobre $\Sigma$. Los conjuntos:

  $X \cup Y$ (unión)

  XY (concatenación)

  $X^*$ (Kleene star)

  son conjuntos regulares sobre $\Sigma$.
  iii. Cerradura: X es un conjunto regular sobre $\Sigma$ sólo si puede obtenerse de los elementos básicos por medio de un número finito de aplicaciones del paso recursivo.

# Expresiones Regulares

# Lenguaje y expresión regular

- Def: Un lenguaje es llamado regular si está definido por un conjunto regular. 49 y 50 Su
- Las expresiones regulares son usadas para abreviar la descripción de conjuntos regulares:
  - Los conjuntos regulares $\varnothing$, $\{\lambda\}$ y $\{a\}$ son representados por $\varnothing$, $\lambda$ y a, respectivamente.
  - Las operaciones unión, Kleene Star y concatenación, son designadas por $\cup$, * y yuxtaposición (poner una letra junto a otra), respectivamente.

# Expresiones regulares

- Def: Sea $\Sigma$ un alfabeto. Las expresiones regulares sobre $\Sigma$ son definidas recursivamente como sigue:
  i. Base: $\varnothing$, $\lambda$ y a, para cada $a \in \Sigma$, son expresiones regulares sobre $\Sigma$.
  ii. Paso recursivo: Sean u y v expresiones regulares sobre $\Sigma$. Las expresiones

  $$(u \cup v)$$
  $$(uv)$$
  $$(u^*)$$

  Son expresiones regulares sobre $\Sigma$.
  iii. Cerradura: u es una expresión regular sobre $\Sigma$ si puede ser obtenida de los elementos base por medio de un número finito de aplicaciones del paso recursivo.

## Notación

- Como la unión y la concatenación son asociativas, se pueden omitir los paréntesis de una expresión regular que consista en una secuencia de esas operaciones.
- Para reducir más el número de paréntesis se asigna una precedencia a los operadores: Kleene star, concatenación y unión. 15.5
- La expresión $u^+$ se usa para abreviar $uu^*$.
- $u^2$ denota $uu$, …, $u^n$ denota $uu…u$, $n$ veces. 51 y 52 Su
- Una expresión regular define un patrón y un string está en el lenguaje de la expresión regular sólo si hace match con el patrón.

## Positivo

- La construcción de expresiones regulares es un proceso positivo.
- Las características de los strings son explícitamente insertadas en la expresión usando concatenación, unión o Kleene star.
- NO hay operaciones negativas para omitir strings que tienen una propiedad particular.
- Para construir una ER para el conjunto de strings que no tienen una propiedad, es necesario formular las condiciones en una forma positiva y construir la ER usando la reformulación del lenguaje. 53Su

# Expresiones equivalentes

- Algunos de los ejemplos previos muestran que la definición de un lenguaje por medio de expresiones regulares no es única.
- Dos expresiones que representan el mismo conjunto son equivalentes.
- Existe una tabla de identidades de expresiones regulares

52 Su

# Tabla de identidades

**TABLE 2.1** Regular Expression Identities

| | |
|---|---|
| 1. | $\emptyset u = u\emptyset = \emptyset$ |
| 2. | $\lambda u = u\lambda = u$ |
| 3. | $\emptyset^* = \lambda$ |
| 4. | $\lambda^* = \lambda$ |
| 5. | $u \cup v = v \cup u$ |
| 6. | $u \cup \emptyset = u$ |
| 7. | $u \cup u = u$ |
| 8. | $u^* = (u^*)^*$ |
| 9. | $u(v \cup w) = uv \cup uw$ |
| 10. | $(u \cup v)w = uw \cup vw$ |
| 11. | $(uv)^*u = u(vu)^*$ |
| 12. | $(u \cup v)^* = (u^* \cup v)^*$ <br> $= u^*(u \cup v)^* = (u \cup vu^*)^*$ <br> $= (u^*v^*)^* = u^*(vu^*)^*$ <br> $= (u^*v)^*u^*$ |

Fuente: Sudkamp, página 54.

## Identidades

- Las identidades se pueden usar para manejar algebraicamente expresiones regulares para construir expresiones equivalentes.
- Dichas identidades son la formulación en expresiones regulares de las operaciones de unión, concatenación y Kleene star.
  - La identidad 5 se sigue de la conmutatividad de la unión de conjuntos.
  - Las identidades 1 y 10 son las leyes distributivas de la unión y la concatenación trasladadas a notación de expresiones regulares.
- Las identidades se pueden usar para simplificar o establecer la equivalencia de expresiones regulares.

53Su

## Lenguajes no regulares

- Debemos considerar que hay lenguajes que no pueden ser definidos usando expresiones regulares (más adelante):

Ej. $\{a^i b^i \mid i \geq 0\}$

# Expresiones Regulares y búsqueda de texto

- Una aplicación común de expresiones regulares es la especificación de patrones para búsqueda de documentos y archivos (*text searching*).
- La mayor diferencia entre el uso de expresiones regulares para definir un lenguaje y para buscar texto es el alcance deseado del match. 16
- En *text searching*, buscamos substrings en el texto que hagan match con el patrón deseado.
- Hay dos tipos:
  - On-line, regresa la primera ocurrencia (e.g. procesadores de texto: find)
  - Off-line, regresa las líneas de las ocurrencias.
- Hay más notación que en las expresiones regulares. 16.5

# Tarea 2

- Ejercicios de las páginas 58 a la 61 de [1]:

2, 3, 4, 5, 6, 7, 8

14, 15, 16, 17, 18, 19, 20, 39a, 39b

# Proyecto 2

- Por definir

# Referencias

1. T.A. Sudkamp. Languages and Machines: An Introduction to the Theory of Computer Science. Pearson, 3rd Edition (2005).
2. J.E. Hopcroft, R. Motwani, J.D. Ullman. Introduction to Automata Theory, Languages, and Computation. Pearson, 3rd Edition (2006).