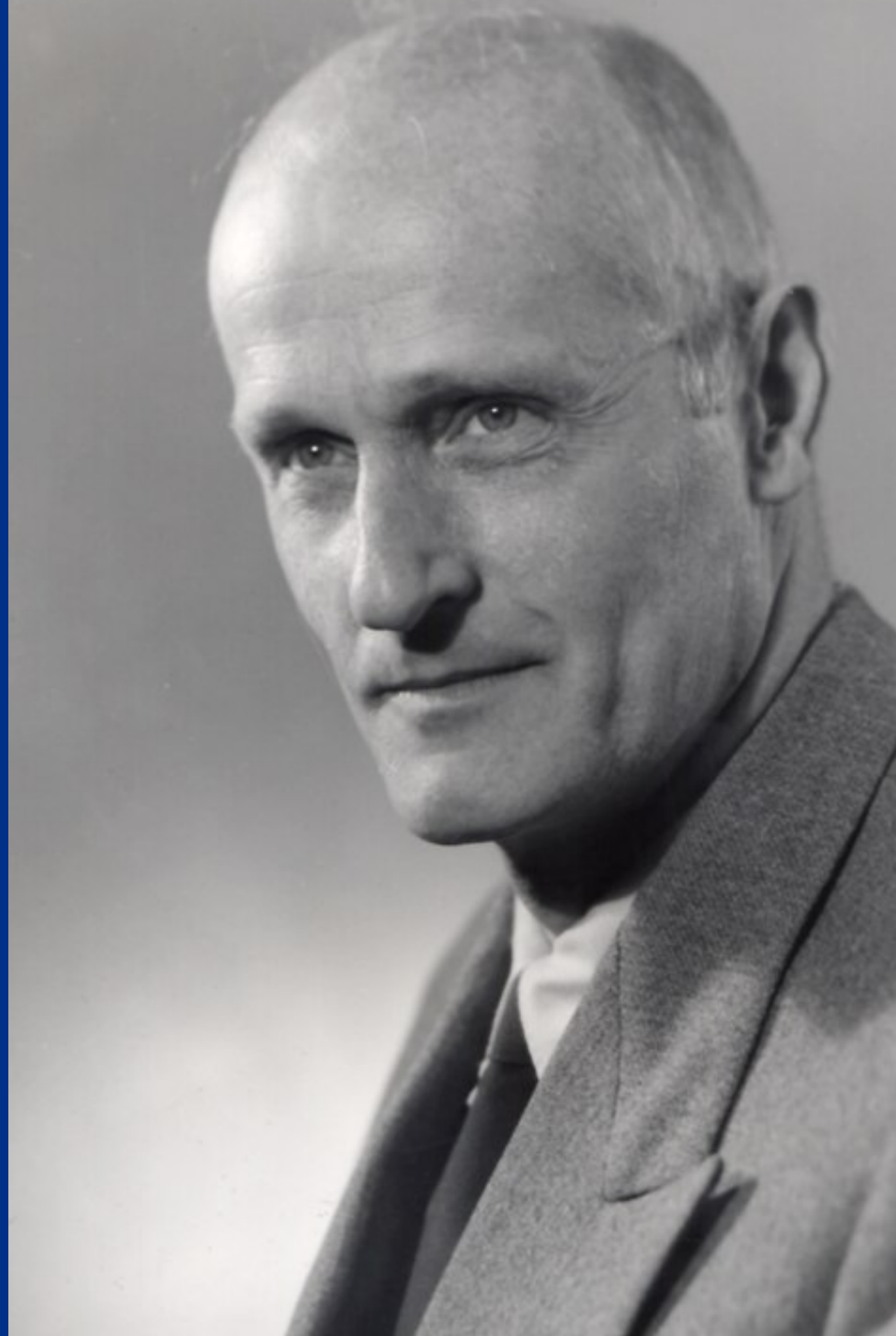
# Expresiones regulares



UNIVERSIDAD DE CHILE

Arturo Merino

# Recuerdo / Plan

# Recuerdo/Plan

- **Recuerdo:**

# Recuerdo/Plan

- **Recuerdo:**
  - Autómatas finitos **no deterministas** $\leftarrow (Q, \Sigma, \delta, q_0, F)$.

# Recuerdo/Plan

- **Recuerdo:**
  - Autómatas finitos **no deterministas** $\leftarrow (Q, \Sigma, \delta, q_0, F)$.
  - Computación: paralelismo/árbol/adivinar

# Recuerdo/Plan

- **Recuerdo:**
  - Autómatas finitos **no deterministas** $\leftarrow (Q, \Sigma, \delta, q_0, F)$.
  - Computación: paralelismo/árbol/adivinar
  - Equivalencia AFND-AFD

# Recuerdo/Plan

- **Recuerdo:**
  - Autómatas finitos **no deterministas** $\leftarrow (Q, \Sigma, \delta, q_0, F)$.
  - Computación: paralelismo/árbol/adivinar
  - Equivalencia AFND-AFD

- **Plan:**

# Recuerdo/Plan

- **Recuerdo:**
  - Autómatas finitos **no deterministas** $\leftarrow (Q, \Sigma, \delta, q_0, F)$.
  - Computación: paralelismo/árbol/adivinar
  - Equivalencia AFND-AFD

- **Plan:**
  - Mostrar equivalencia AFD-AFND

# Recuerdo/Plan

- **Recuerdo:**
  - Autómatas finitos **no deterministas** $\leftarrow (Q, \Sigma, \delta, q_0, F)$.
  - Computación: paralelismo/árbol/adivinar
  - Equivalencia AFND-AFD

- **Plan:**
  - Mostrar equivalencia AFD-AFND
  - Operaciones y expresiones regulares

# Equivalencia AFD-AFND

# De AFND a AFD

**Teo.**

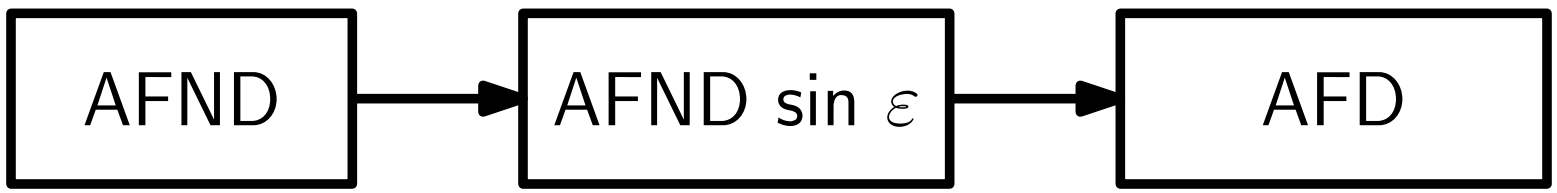
Todo AFND tiene un AFD equivalente.

# De AFND a AFD

**Teo.**

Todo AFND tiene un AFD equivalente.

- **Dem:**



# De AFND a AFD

**Teo.**

Todo AFND tiene un AFD equivalente.

- **Dem:**


**Cor.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

# De AFND a AFD

**Teo.**

Todo AFND tiene un AFD equivalente.

- **Dem:**


**Cor.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe un AFND $M$ tal que $\mathcal{L}(M) = L$.

# De AFND a AFD

**Teo.**

Todo AFND tiene un AFD equivalente.

- **Dem:**


**Cor.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe un AFND $M$ tal que $\mathcal{L}(M) = L$.

- Entender AFD vía AFND.

# De AFND a AFD

**Teo.**

Todo AFND tiene un AFD equivalente.

- **Dem:**


**Cor.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe un AFND $M$ tal que $\mathcal{L}(M) = L$.

- Entender AFD vía AFND.
- Probar que un lenguaje es regular $\leftarrow$ diseñar AFND

# De AFND a AFD

**Teo.**

Todo AFND tiene un AFD equivalente.

- **Dem:**


**Cor.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe un AFND $M$ tal que $\mathcal{L}(M) = L$.

- Entender AFD vía AFND.
- Probar que un lenguaje es regular $\leftarrow$ diseñar AFND
- Nuevas propiedades de cerradura.

# Operaciones regulares

# Operaciones regulares

- Operaciones regulares: $\cup$, $\cdot$, $*$.

# Operaciones regulares

- Operaciones regulares: $\cup$, $\cdot$, $*$.
- ¿Cómo se comportan con los lenguajes regulares?

# Operaciones regulares

- Operaciones regulares: $\cup$, $\cdot$, $*$.
- ¿Cómo se comportan con los lenguajes regulares?

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

# Operaciones regulares

- Operaciones regulares: $\cup$, $\cdot$, $*$.
- ¿Cómo se comportan con los lenguajes regulares?

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

# Operaciones regulares

- Operaciones regulares: $\cup$, $\cdot$, $*$.
- ¿Cómo se comportan con los lenguajes regulares?

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,

# Operaciones regulares

- Operaciones regulares: $\cup$, $\cdot$, $*$.
- ¿Cómo se comportan con los lenguajes regulares?

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$,

# Operaciones regulares

- Operaciones regulares: $\cup$, $\cdot$, $*$.
- ¿Cómo se comportan con los lenguajes regulares?

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$,
- $L_1^*$.

# Operaciones regulares

- Operaciones regulares: $\cup$, $\cdot$, $*$.
- ¿Cómo se comportan con los lenguajes regulares?

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$,
- $L_1^*$.

- Cerrados bajo operaciones regulares.

# Operaciones regulares

- Operaciones regulares: $\cup$, $\cdot$, $*$.
- ¿Cómo se comportan con los lenguajes regulares?

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$,
- $L_1^*$.

- Cerrados bajo operaciones regulares.
- **Dem:**

# Cerradura

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

# Cerradura

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

# Cerradura

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,

# Cerradura

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$,

# Cerradura

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$,
- $L_1^*$.

# Cerradura

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$,
- $L_1^*$.

- **Ej:** $L_1 = \{w \in \{0,1\}^* \mid w$ termina con $(00)^k$ para algún $k \geq 1\}$

# Cerradura

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$,
- $L_1^*$.

- **Ej:** $L_1 = \{w \in \{0,1\}^* \mid w$ termina con $(00)^k$ para algún $k \geq 1\}$

**Obs.**

Todo lenguaje finito es regular.

# Cerradura

**Teo.**

Sean $L_1, L_2 \subseteq \Sigma^*$ lenguajes regulares.

Los siguientes lenguajes son regulares:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$,
- $L_1^*$.

- **Ej:** $L_1 = \{w \in \{0,1\}^* \mid w$ termina con $(00)^k$ para algún $k \geq 1\}$

**Obs.**

Todo lenguaje finito es regular.

- $L_2 =$
  $\{w \in \{a, \ldots, z, \sqcup\}^* \mid w$ es una frase formada por palabras del español$\}$

# Expresiones regulares

# Describiendo vía operaciones

- Describir lenguajes:

# Describiendo vía operaciones

- Describir lenguajes:
  - Vía AFD/AFND.

# Describiendo vía operaciones

- Describir lenguajes:
  - Vía AFD/AFND.
  - Vía *operaciones*

# Describiendo vía operaciones

- Describir lenguajes:
  - Vía AFD/AFND.
  - Vía *operaciones*
  - Vía *operaciones regulares*

# Describiendo vía operaciones

- Describir lenguajes:
  - Vía AFD/AFND.
  - Vía *operaciones*
  - Vía *operaciones regulares*

- **Ej:**

# Describiendo vía operaciones

- Describir lenguajes:
  - Vía AFD/AFND.
  - Vía *operaciones*
  - Vía *operaciones regulares*

- **Ej:**
  - $(\{0\} \cup \{1\}) \cdot \{0\}^*$

# Describiendo vía operaciones

- Describir lenguajes:
  - Vía AFD/AFND.
  - Vía *operaciones*
  - Vía *operaciones regulares*

- **Ej:**
  - $(\{0\} \cup \{1\}) \cdot \{0\}^*$
  - $(0 \mid 1)0^*$

# Describiendo vía operaciones

- Describir lenguajes:
  - Vía AFD/AFND.
  - Vía *operaciones*
  - Vía *operaciones regulares*

- **Ej:**
  - $(\{0\} \cup \{1\}) \cdot \{0\}^*$
  - $(0 \mid 1)0^*$
  - $\Sigma^*$ con $\Sigma = (0|1)$.

# Describiendo vía operaciones

- Describir lenguajes:
  - Vía AFD/AFND.
  - Vía *operaciones*
  - Vía *operaciones regulares*

- **Ej:**
  - $(\{0\} \cup \{1\}) \cdot \{0\}^*$
  - $(0 \mid 1)0^*$
  - $\Sigma^*$ con $\Sigma = (0|1)$.
  - $0\Sigma^* \mid \Sigma^*1$

# Describiendo vía operaciones

- Describir lenguajes:
  - Vía AFD/AFND.
  - Vía *operaciones*
  - Vía *operaciones regulares*

- **Ej:**
  - $(\{0\} \cup \{1\}) \cdot \{0\}^*$
  - $(0 \mid 1)0^*$
  - $\Sigma^*$ con $\Sigma = (0|1)$.
  - $0\Sigma^* \mid \Sigma^*1$

- Describe patrones en texto

# Describiendo vía operaciones

- Describir lenguajes:
  - Vía AFD/AFND.
  - Vía *operaciones*
  - Vía *operaciones regulares*

- **Ej:**
  - $(\{0\} \cup \{1\}) \cdot \{0\}^*$
  - $(0 \mid 1)0^*$
  - $\Sigma^*$ con $\Sigma = (0|1)$.
  - $0\Sigma^* \mid \Sigma^*1$

- Describe patrones en texto

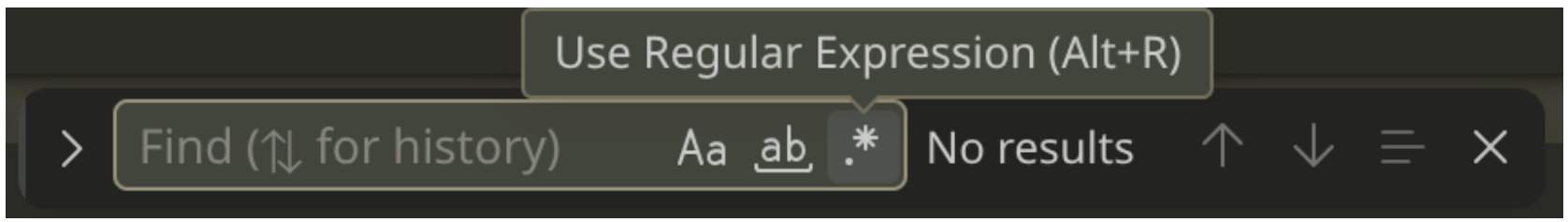
- ¡Varias aplicaciones!

# Expresiones regulares

- Búsqueda en texto

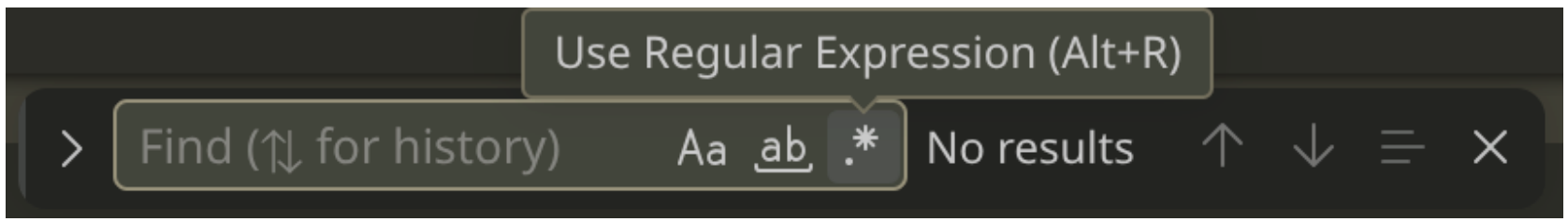
# Expresiones regulares

- Búsqueda en texto
  - `awk`, `grep`, editores de texto, etc.



# Expresiones regulares

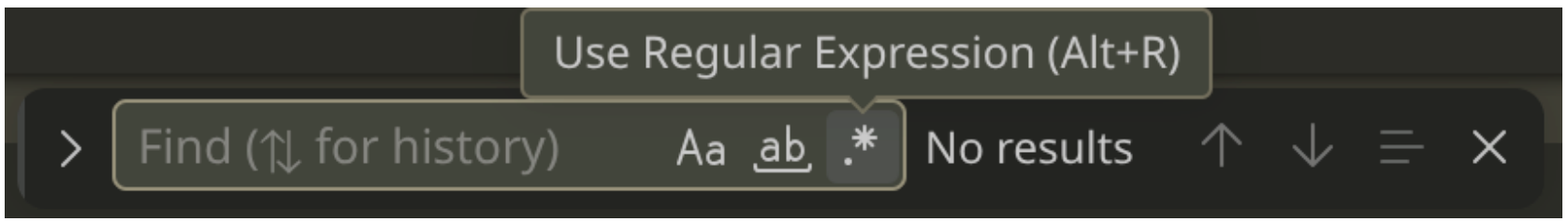
- Búsqueda en texto
  - `awk`, `grep`, editores de texto, etc.




- Compiladores, análisis léxico

# Expresiones regulares

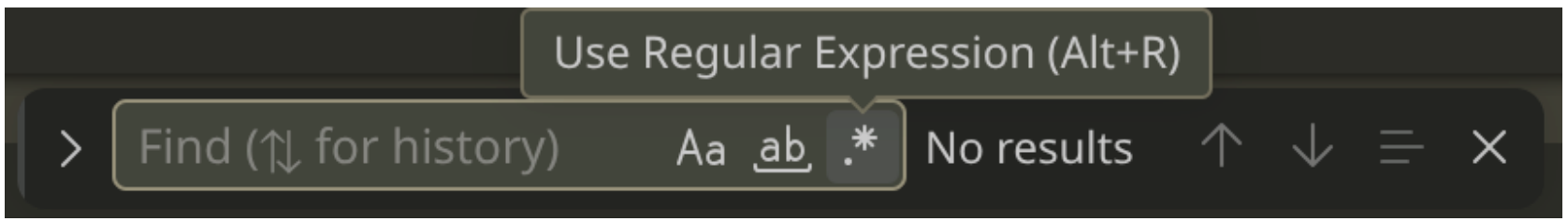
- Búsqueda en texto
  - `awk`, `grep`, editores de texto, etc.




- Compiladores, análisis léxico
  - `texto` $\rightarrow$ `tokens`

# Expresiones regulares

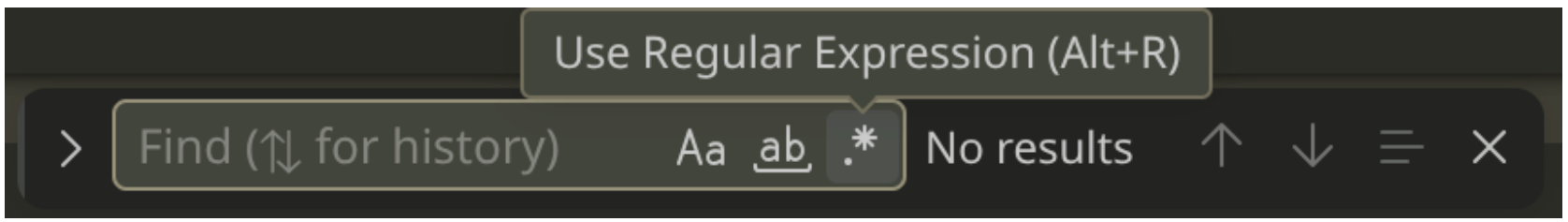
- Búsqueda en texto
  - `awk`, `grep`, editores de texto, etc.




- Compiladores, análisis léxico
  - `texto` $\rightarrow$ `tokens`
  - `lex`, `flex`, etc.

# Expresiones regulares

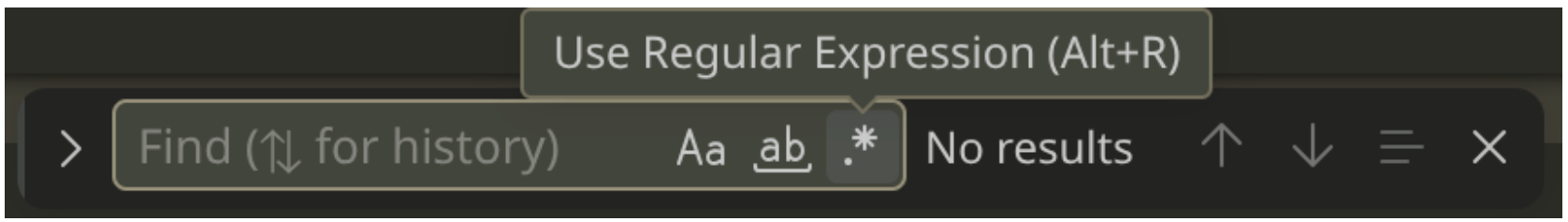
- Búsqueda en texto
  - `awk`, `grep`, editores de texto, etc.




- Compiladores, análisis léxico
  - `texto` $\rightarrow$ `tokens`
  - `lex`, `flex`, etc.
- Validadores de input

# Expresiones regulares

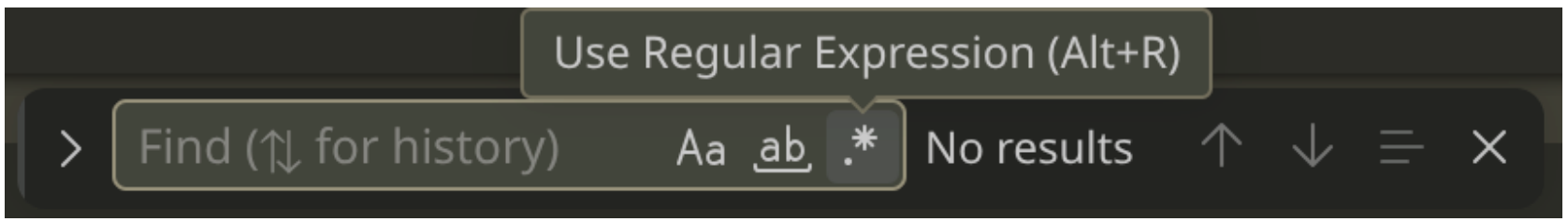
- Búsqueda en texto
  - `awk`, `grep`, editores de texto, etc.




- Compiladores, análisis léxico
  - `texto` $\rightarrow$ `tokens`
  - `lex`, `flex`, etc.
- Validadores de input
  - `texto` $\rightarrow$ `bool`

# Expresiones regulares

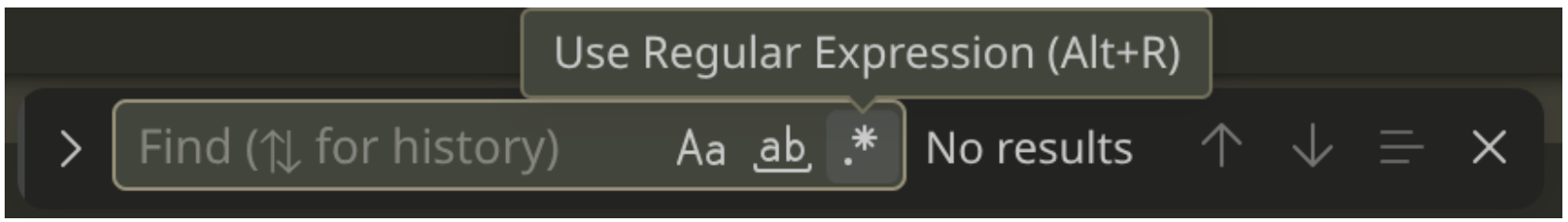
- Búsqueda en texto
  - `awk`, `grep`, editores de texto, etc.




- Compiladores, análisis léxico
  - `texto` → `tokens`
  - `lex`, `flex`, etc.
- Validadores de input
  - `texto` → `bool`
  - Emails, URLs, fechas, etc.

# Expresiones regulares

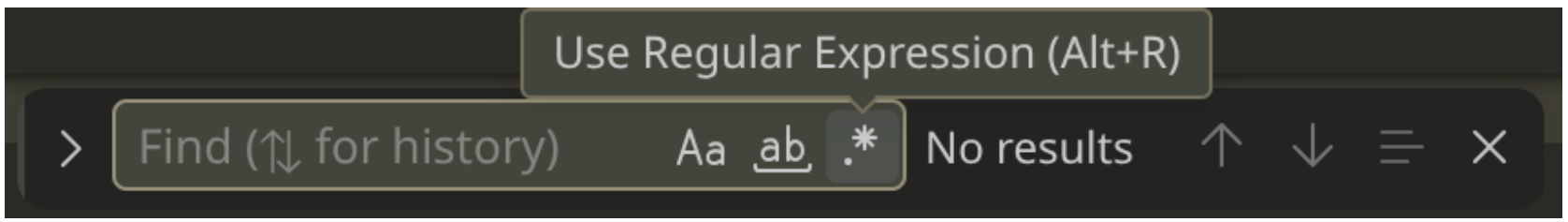
- Búsqueda en texto
  - `awk`, `grep`, editores de texto, etc.




- Compiladores, análisis léxico
  - `texto` → `tokens`
  - `lex`, `flex`, etc.
- Validadores de input
  - `texto` → `bool`
  - Emails, URLs, fechas, etc.
- Análisis de texto/strings

# Expresiones regulares

- Búsqueda en texto
  - `awk`, `grep`, editores de texto, etc.




- Compiladores, análisis léxico
  - `texto` $\rightarrow$ `tokens`
  - `lex`, `flex`, etc.
- Validadores de input
  - `texto` $\rightarrow$ `bool`
  - Emails, URLs, fechas, etc.
- Análisis de texto/strings
  - Pattern matching, bioinformática, etc.

# Sintaxis

- Definición *recursiva*

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

- $R = \sigma$ para algún $\sigma \in \Sigma$

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

- $R = \sigma$ para algún $\sigma \in \Sigma$
- $R = \emptyset$

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

- $R = \sigma$ para algún $\sigma \in \Sigma$
- $R = \emptyset$
- $R = \varepsilon$

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

- $R = \sigma$ para algún $\sigma \in \Sigma$
- $R = \emptyset$
- $R = \varepsilon$
- $R = (R_1|R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

- $R = \sigma$ para algún $\sigma \in \Sigma$
- $R = \emptyset$
- $R = \varepsilon$
- $R = (R_1|R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1 R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

- $R = \sigma$ para algún $\sigma \in \Sigma$
- $R = \emptyset$
- $R = \varepsilon$
- $R = (R_1|R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1^*)$ con $R_1$ una expresión regular sobre $\Sigma$,

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

- $R = \sigma$ para algún $\sigma \in \Sigma$
- $R = \emptyset$
- $R = \varepsilon$
- $R = (R_1|R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1 R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1^*)$ con $R_1$ una expresión regular sobre $\Sigma$,

- **Notación:**

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

- $R = \sigma$ para algún $\sigma \in \Sigma$
- $R = \emptyset$
- $R = \varepsilon$
- $R = (R_1|R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1 R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1^*)$ con $R_1$ una expresión regular sobre $\Sigma$,

- **Notación:**
  - $R_1 R_2 = R_1 \circ R_2 = R_1 \cdot R_2$.

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

- $R = \sigma$ para algún $\sigma \in \Sigma$
- $R = \emptyset$
- $R = \varepsilon$
- $R = (R_1|R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1 R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1^*)$ con $R_1$ una expresión regular sobre $\Sigma$,

- **Notación:**
  - $R_1 R_2 = R_1 \circ R_2 = R_1 \cdot R_2$.
  - $R_1|R_2 = R_1 \cup R_2 = R_1 + R_2$.

# Sintaxis

- Definición *recursiva*

**Def.**

Diremos que $R$ es una *expresión regular* sobre un alfabeto $\Sigma$ si

- $R = \sigma$ para algún $\sigma \in \Sigma$
- $R = \emptyset$
- $R = \varepsilon$
- $R = (R_1|R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1 R_2)$ con $R_1$ y $R_2$ dos expresiones regulares sobre $\Sigma$,
- $R = (R_1^*)$ con $R_1$ una expresión regular sobre $\Sigma$,

- **Notación:**
  - $R_1 R_2 = R_1 \circ R_2 = R_1 \cdot R_2$.
  - $R_1|R_2 = R_1 \cup R_2 = R_1 + R_2$.
  - $R^+ = RR^*$, $R^k = R \dots R$.

# Lenguaje descrito

- Definición *recursiva*

# Lenguaje descrito

- Definición *recursiva*

**Def.**

Sea $R$ una *expresión regular* sobre $\Sigma$.

# Lenguaje descrito

- Definición *recursiva*

**Def.**

Sea $R$ una *expresión regular* sobre $\Sigma$.

El lenguaje descrito por $R$, denotado por $\mathcal{L}(R)$, se define recursivamente:

# Lenguaje descrito

- Definición *recursiva*

**Def.**

Sea $R$ una *expresión regular* sobre $\Sigma$.

El lenguaje descrito por $R$, denotado por $\mathcal{L}(R)$, se define recursivamente:

- Si $R = \sigma$, entonces $\mathcal{L}(R) = \{\sigma\}$.

# Lenguaje descrito

- Definición *recursiva*

**Def.**

Sea $R$ una *expresión regular* sobre $\Sigma$.

El lenguaje descrito por $R$, denotado por $\mathcal{L}(R)$, se define recursivamente:

- Si $R = \sigma$, entonces $\mathcal{L}(R) = \{\sigma\}$.
- Si $R = \emptyset$, entonces $\mathcal{L}(R) = \emptyset$.

# Lenguaje descrito

- Definición *recursiva*

**Def.**

Sea $R$ una *expresión regular* sobre $\Sigma$.

El lenguaje descrito por $R$, denotado por $\mathcal{L}(R)$, se define recursivamente:

- Si $R = \sigma$, entonces $\mathcal{L}(R) = \{\sigma\}$.
- Si $R = \emptyset$, entonces $\mathcal{L}(R) = \emptyset$.
- Si $R = \varepsilon$, entonces $\mathcal{L}(R) = \{\varepsilon\}$.

# Lenguaje descrito

- Definición *recursiva*

**Def.**

Sea $R$ una *expresión regular* sobre $\Sigma$.

El lenguaje descrito por $R$, denotado por $\mathcal{L}(R)$, se define recursivamente:

- Si $R = \sigma$, entonces $\mathcal{L}(R) = \{\sigma\}$.
- Si $R = \emptyset$, entonces $\mathcal{L}(R) = \emptyset$.
- Si $R = \varepsilon$, entonces $\mathcal{L}(R) = \{\varepsilon\}$.
- Si $R = (R_1|R_2)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$.

# Lenguaje descrito

- Definición *recursiva*

**Def.**

Sea $R$ una *expresión regular* sobre $\Sigma$.

El lenguaje descrito por $R$, denotado por $\mathcal{L}(R)$, se define recursivamente:

- Si $R = \sigma$, entonces $\mathcal{L}(R) = \{\sigma\}$.
- Si $R = \emptyset$, entonces $\mathcal{L}(R) = \emptyset$.
- Si $R = \varepsilon$, entonces $\mathcal{L}(R) = \{\varepsilon\}$.
- Si $R = (R_1|R_2)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$.
- Si $R = (R_1 R_2)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1) \cdot \mathcal{L}(R_2)$.

# Lenguaje descrito

- Definición *recursiva*

**Def.**

Sea $R$ una *expresión regular* sobre $\Sigma$.

El lenguaje descrito por $R$, denotado por $\mathcal{L}(R)$, se define recursivamente:

- Si $R = \sigma$, entonces $\mathcal{L}(R) = \{\sigma\}$.
- Si $R = \emptyset$, entonces $\mathcal{L}(R) = \emptyset$.
- Si $R = \varepsilon$, entonces $\mathcal{L}(R) = \{\varepsilon\}$.
- Si $R = (R_1|R_2)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$.
- Si $R = (R_1 R_2)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1) \cdot \mathcal{L}(R_2)$.
- Si $R = (R_1^*)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1)^*$.

# Lenguaje descrito

- Definición *recursiva*

**Def.**

Sea $R$ una *expresión regular* sobre $\Sigma$.

El lenguaje descrito por $R$, denotado por $\mathcal{L}(R)$, se define recursivamente:

- Si $R = \sigma$, entonces $\mathcal{L}(R) = \{\sigma\}$.
- Si $R = \emptyset$, entonces $\mathcal{L}(R) = \emptyset$.
- Si $R = \varepsilon$, entonces $\mathcal{L}(R) = \{\varepsilon\}$.
- Si $R = (R_1|R_2)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$.
- Si $R = (R_1 R_2)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1) \cdot \mathcal{L}(R_2)$.
- Si $R = (R_1^*)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1)^*$.

(donde $R_1$ y $R_2$ son expresiones regulares)

# Lenguaje descrito

- Definición *recursiva*

**Def.**

Sea $R$ una *expresión regular* sobre $\Sigma$.

El lenguaje descrito por $R$, denotado por $\mathcal{L}(R)$, se define recursivamente:

- Si $R = \sigma$, entonces $\mathcal{L}(R) = \{\sigma\}$.
- Si $R = \emptyset$, entonces $\mathcal{L}(R) = \emptyset$.
- Si $R = \varepsilon$, entonces $\mathcal{L}(R) = \{\varepsilon\}$.
- Si $R = (R_1|R_2)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$.
- Si $R = (R_1 R_2)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1) \cdot \mathcal{L}(R_2)$.
- Si $R = (R_1^*)$, entonces $\mathcal{L}(R) = \mathcal{L}(R_1)^*$.

(donde $R_1$ y $R_2$ son expresiones regulares)

- Asociatividad de $|$ y concatenación $\rightarrow$ omitir paréntesis.

# Descripción

- Describir patrones vía expresiones regulares

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$
  - $\Sigma^*1\Sigma^*$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$
  - $\Sigma^*1\Sigma^*$
  - $\Sigma^*001\Sigma^*$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$
  - $\Sigma^*1\Sigma^*$
  - $\Sigma^*001\Sigma^*$
  - $1^*(01^+)^*$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$
  - $\Sigma^*1\Sigma^*$
  - $\Sigma^*001\Sigma^*$
  - $1^*(01^+)^*$
  - $(\Sigma\Sigma)^*$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$
  - $\Sigma^*1\Sigma^*$
  - $\Sigma^*001\Sigma^*$
  - $1^*(01^+)^*$
  - $(\Sigma\Sigma)^*$
  - $(\Sigma\Sigma\Sigma\Sigma\Sigma)^*$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$
  - $\Sigma^*1\Sigma^*$
  - $\Sigma^*001\Sigma^*$
  - $1^*(01^+)^*$
  - $(\Sigma\Sigma)^*$
  - $(\Sigma\Sigma\Sigma\Sigma\Sigma)^*$
  - $01|10|1001|101$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$
  - $\Sigma^*1\Sigma^*$
  - $\Sigma^*001\Sigma^*$
  - $1^*(01^+)^*$
  - $(\Sigma\Sigma)^*$
  - $(\Sigma\Sigma\Sigma\Sigma\Sigma)^*$
  - $01|10|1001|101$
  - $0\Sigma^*0|1\Sigma^*1|0|1$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$
  - $\Sigma^*1\Sigma^*$
  - $\Sigma^*001\Sigma^*$
  - $1^*(01^+)^*$
  - $(\Sigma\Sigma)^*$
  - $(\Sigma\Sigma\Sigma\Sigma\Sigma)^*$
  - $01|10|1001|101$
  - $0\Sigma^*0|1\Sigma^*1|0|1$
  - $(0|\epsilon)(1|\epsilon)$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$
  - $\Sigma^*1\Sigma^*$
  - $\Sigma^*001\Sigma^*$
  - $1^*(01^+)^*$
  - $(\Sigma\Sigma)^*$
  - $(\Sigma\Sigma\Sigma\Sigma\Sigma)^*$
  - $01|10|1001|101$
  - $0\Sigma^*0|1\Sigma^*1|0|1$
  - $(0|\epsilon)(1|\epsilon)$
  - $1^*\emptyset$

# Descripción

- Describir patrones vía expresiones regulares
- **Ej:** $\Sigma = (0|1)$
  - $0^*10^*$
  - $\Sigma^*1\Sigma^*$
  - $\Sigma^*001\Sigma^*$
  - $1^*(01^+)^*$
  - $(\Sigma\Sigma)^*$
  - $(\Sigma\Sigma\Sigma\Sigma\Sigma)^*$
  - $01|10|1001|101$
  - $0\Sigma^*0|1\Sigma^*1|0|1$
  - $(0|\epsilon)(1|\epsilon)$
  - $1^*\emptyset$
  - $\emptyset^*$

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?
  - Separar en bloques claves

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?
  - Separar en bloques claves
  - Patrones forzados

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?
  - Separar en bloques claves
  - Patrones forzados
- **Ej:**

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?
  - Separar en bloques claves
  - Patrones forzados

- **Ej:**
  - $\{w \in \{0,1\}^* : w$ contiene $0001$ como substring$\}$

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?
  - Separar en bloques claves
  - Patrones forzados

- **Ej:**
  - $\{w \in \{0,1\}^* : w$ contiene $0001$ como substring$\}$
  - $\{w \in \{0,1\}^* : |w| \geq 3$ y el antepenúltimo carácter es un $1\}$

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?
  - Separar en bloques claves
  - Patrones forzados

- **Ej:**
  - $\{w \in \{0,1\}^* : w$ contiene $0001$ como substring$\}$
  - $\{w \in \{0,1\}^* : |w| \geq 3$ y el antepenúltimo carácter es un $1\}$
  - $\{w \in \{0,1\}^* : w$ contiene al menos tres símbolos $1\}$

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?
  - Separar en bloques claves
  - Patrones forzados

- **Ej:**
  - $\{w \in \{0,1\}^* : w$ contiene $0001$ como substring$\}$
  - $\{w \in \{0,1\}^* : |w| \geq 3$ y el antepenúltimo carácter es un $1\}$
  - $\{w \in \{0,1\}^* : w$ contiene al menos tres símbolos $1\}$
  - $\{w \in \{0,1\}^* : w$ contiene una cantidad par de símbolos $1\}$

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?
  - Separar en bloques claves
  - Patrones forzados

- **Ej:**
  - $\{w \in \{0,1\}^* : w$ contiene $0001$ como substring$\}$
  - $\{w \in \{0,1\}^* : |w| \geq 3$ y el antepenúltimo carácter es un $1\}$
  - $\{w \in \{0,1\}^* : w$ contiene al menos tres símbolos $1\}$
  - $\{w \in \{0,1\}^* : w$ contiene una cantidad par de símbolos $1\}$
  - $\{w \in \{0,1\}^* : w$ no es $11$ ni $111\}$

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?
  - Separar en bloques claves
  - Patrones forzados

- **Ej:**
  - $\{w \in \{0,1\}^* : w$ contiene $0001$ como substring$\}$
  - $\{w \in \{0,1\}^* : |w| \geq 3$ y el antepenúltimo carácter es un $1\}$
  - $\{w \in \{0,1\}^* : w$ contiene al menos tres símbolos $1\}$
  - $\{w \in \{0,1\}^* : w$ contiene una cantidad par de símbolos $1\}$
  - $\{w \in \{0,1\}^* : w$ no es $11$ ni $111\}$
  - $\{w \in \{0,1\}^* :$ las posiciones impares de $w$ contienen un $1\}$

# Diseño de ER

- ¿Cómo diseño ER para un lenguaje?
  - Separar en bloques claves
  - Patrones forzados

- **Ej:**
  - $\{w \in \{0,1\}^* : w$ contiene $0001$ como substring$\}$
  - $\{w \in \{0,1\}^* : |w| \geq 3$ y el antepenúltimo carácter es un $1\}$
  - $\{w \in \{0,1\}^* : w$ contiene al menos tres símbolos $1\}$
  - $\{w \in \{0,1\}^* : w$ contiene una cantidad par de símbolos $1\}$
  - $\{w \in \{0,1\}^* : w$ no es $11$ ni $111\}$
  - $\{w \in \{0,1\}^* :$ las posiciones impares de $w$ contienen un $1\}$

- **Obs.:** Las ER no suelen llevarse bien con $\cap$ o complemento.

# Expresividad

- ¿Qué tiene más poder de cómputo: una ER o un AFD?

# Expresividad

- ¿Qué tiene más poder de cómputo: una ER o un AFD?
  - ¡Mismo poder que un AFD!

# Expresividad

- ¿Qué tiene más poder de cómputo: una ER o un AFD?
  - ¡Mismo poder que un AFD!

**Teo.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

# Expresividad

- ¿Qué tiene más poder de cómputo: una ER o un AFD?
  - ¡Mismo poder que un AFD!

**Teo.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe una expresión regular $R$ tal que $\mathcal{L}(R) = L$.

# Expresividad

- ¿Qué tiene más poder de cómputo: una ER o un AFD?
  - ¡Mismo poder que un AFD!

**Teo.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe una expresión regular $R$ tal que $\mathcal{L}(R) = L$.

- Entender AFD vía ER.

# Expresividad

- ¿Qué tiene más poder de cómputo: una ER o un AFD?
  - ¡Mismo poder que un AFD!

**Teo.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe una expresión regular $R$ tal que $\mathcal{L}(R) = L$.

- Entender AFD vía ER.
- Probar que un lenguaje es regular

# Expresividad

- ¿Qué tiene más poder de cómputo: una ER o un AFD?
  - ¡Mismo poder que un AFD!

**Teo.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe una expresión regular $R$ tal que $\mathcal{L}(R) = L$.

- Entender AFD vía ER.
- Probar que un lenguaje es regular
  - Diseñar un AFD

# Expresividad

- ¿Qué tiene más poder de cómputo: una ER o un AFD?
  - ¡Mismo poder que un AFD!

**Teo.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe una expresión regular $R$ tal que $\mathcal{L}(R) = L$.

- Entender AFD vía ER.
- Probar que un lenguaje es regular
  - Diseñar un AFD
  - Diseñar un AFND

# Expresividad

- ¿Qué tiene más poder de cómputo: una ER o un AFD?
  - ¡Mismo poder que un AFD!

**Teo.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe una expresión regular $R$ tal que $\mathcal{L}(R) = L$.

- Entender AFD vía ER.
- Probar que un lenguaje es regular
  - Diseñar un AFD
  - Diseñar un AFND
  - Diseñar una ER

# Expresividad

- ¿Qué tiene más poder de cómputo: una ER o un AFD?
  - ¡Mismo poder que un AFD!

**Teo.**

Sea $L \subseteq \Sigma^*$ un lenguaje.

$L$ es regular si y sólo si existe una expresión regular $R$ tal que $\mathcal{L}(R) = L$.

- Entender AFD vía ER.
- Probar que un lenguaje es regular
  - Diseñar un AFD
  - Diseñar un AFND
  - Diseñar una ER
- **Dem:** próxima clase.

# Dudas/Preguntas

# Problema del día

2. La siguiente pregunta modela el daño que puede sufrir un mensaje durante su transmisión: algunos caracteres pueden perderse, pero el orden de los caracteres que llegan es el mismo que el del mensaje original. Diremos que $x$ es una *versión dañada* de $w$ si $x$ puede obtenerse borrando cero o más símbolos de $w$. Para un lenguaje $L \subseteq \Sigma^*$, definimos

$$\text{Daño}(L) = \{x \in \Sigma^* : x \text{ es una versión dañada de algún } w \in L\}.$$

*a*) Un protocolo acepta los mensajes binarios descritos por

$$R_3 = 0^*10^*10^*10^*.$$

Entregue una expresión regular que describa todas las versiones dañadas de los mensajes aceptados por el protocolo.

*b*) Sea $R$ una expresión regular cualquiera. Defina recursivamente una expresión regular $D(R)$ que describa todas las versiones dañadas de los strings descritos por $R$.

*c*) Demuestre por inducción estructural sobre $R$ que su construcción satisface

$$\mathcal{L}(D(R)) = \text{Daño}(\mathcal{L}(R)).$$

Deduzca que, si $L$ es regular, entonces $\text{Daño}(L)$ también es regular.

*d*) Demuestre además que, para todos $L, K \subseteq \Sigma^*$ tenemos que $L \subseteq \text{Daño}(L)$, si $L \subseteq K$ entonces $\text{Daño}(L) \subseteq \text{Daño}(K)$. Concluya que $\text{Daño}(\text{Daño}(L)) = \text{Daño}(L)$.

*e*) Considere el siguiente problema computacional: dados una expresión regular $R$ y un string recibido $x$, decidir si $x$ podría ser una versión dañada de algún mensaje de $\mathcal{L}(R)$. Proponga un algoritmo eficiente para resolver este problema y discuta sobre su análisis. ¿Que pasa si el problema busca recuperar el mensaje original $w$ a partir de $x$ y $R$?

# Wrap-Up

- **Hoy:**

# Wrap-Up

- **Hoy:**
  - Mostrar equivalencia AFD-AFND

# Wrap-Up

- **Hoy:**
  - Mostrar equivalencia AFD-AFND
  - Operaciones y expresiones regulares

# Wrap-Up

- **Hoy:**
  - Mostrar equivalencia AFD-AFND
  - Operaciones y expresiones regulares

- **Próx. clase:**

# Wrap-Up

- **Hoy:**
  - Mostrar equivalencia AFD-AFND
  - Operaciones y expresiones regulares

- **Próx. clase:**
  - Equivalencia de lenguajes regulares