Conjuntos $\{1,2,3\} = \{1,1,2,3\} = \{3,2,1\}$

$\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R}$

- $x \in A$ $\quad$ $3 \in \mathbb{N}$ $\quad$ $\pi \notin \mathbb{N}$
- $A \cup B$
- $A \cap B$
- $A \setminus B$
- $A \subseteq B \Leftrightarrow \forall x.\ x \in A \Rightarrow x \in B$
- $A = B \Leftrightarrow A \subseteq B \wedge B \subseteq A$
- $\mathcal{P}(A) = \{B \mid B \subseteq A\}$

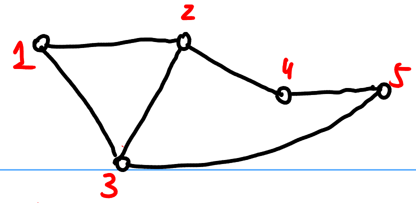
Ej: $\mathcal{P}(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$

Funciones $\quad f: A \to B$

Def. Una función $f: A \to B$ se dice:

1) inyectiva si $\forall a, a' \in A.\ f(a) = f(a') \Rightarrow a = a'$.
2) suryectiva si $\forall b \in B.\ \exists a \in A.\ f(a) = b$.
3) biyectiva si es inyectiva y suryectiva.

Grafos



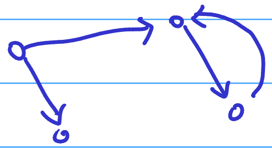
$V = \{1,2,3,4,5\}$

$E = \{ \{1,3\}, \{1,2\}, \{2,3\}, \{2,4\}, \{4,5\}, \{3,5\} \}$

no dirigido

Def. Un grafo G es un par (V,E) donde V es un conjunto finito y E es un conjunto de pares no ordenados de vértices.

[otra noción de grafo: grafo dirigido,


]

Def. Si $v \in V$, notamos $d(v)$ al grado de v, es decir, la cantidad de vecinos de v.

Lenguajes formales

- Sea Σ un conjunto finito, al que llamamos alfabeto.

  Ej.: $\Sigma = \{a, b, c, d\}$

- Una palabra sobre el alfabeto Σ es una secuencia finita de elementos de Σ.

$$\underbrace{w}_{\text{palabra}} = w_1 w_2 \dots w_n$$

donde para todo $i = 1..n$, $w_i \in \Sigma$.

Si $n = 0$, notamos $w = \epsilon$
λ

Además, notamos $|w|$ a la longitud de w.

Ej. $|abbb| = 4$ $\quad |acbda| = 5$

$|\epsilon| = 0$

- Notamos $w_1 \cdot w_2$ o $w_1 w_2$ para la concatenación de palabras.
- Notamos w^n para la palabra que resulta de concatenar $\underbrace{w \cdot w \cdot \ldots \cdot w}_{n \text{ veces}}$

Ej.:

$$(abb)^3 = abbabbabb$$
$$(abb)^1 = abb$$
$$(abb)^0 = \epsilon$$

- Notamos Σ^* al conjunto de todas las palabras que se pueden formar sobre el alfabeto Σ.

Ej. Si $\Sigma = \{0, 1\}$

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$$

- Un <u>lenguaje</u> sobre un alfabeto Σ es un conjunto de palabras sobre Σ.
Es decir, L es un lenguaje si $L \subseteq \Sigma^*$.

- <u>Operaciones con lenguajes:</u>

$$L \cdot L' = \{ w \cdot w' \mid w \in L,\ w' \in L' \}$$

$L \cup L'$ (unión usual de conjuntos)

$$L^* = \{ w_1 w_2 \cdots w_n \mid n \geq 0,\ w_1 \in L,\ w_2 \in L, \dots,\ w_n \in L \}$$

coeficientes en $\mathbb{Q}$
in
$\mathbb{R}$

Polinomios

$$x^3 - 2x^2 + 8/3 = \underbrace{1}\cdot x^3 + \underbrace{(-2)}\cdot x^2 + \underbrace{0}\cdot x^1 + \underbrace{8/3}\cdot x^0$$

Def. Un polinomio con coeficientes en $\overbrace{A}^{\text{anillo}}$

es una suma de la forma:

$$a_0 x^0 + a_1 x^1 + a_2 x^2 + \ldots + a_n x^n$$

donde $a_0, a_1, \ldots, a_n \in A$.

Más concisamente, un polinomio se puede escribir como

$$\sum_{i=0}^{n} a_i \cdot x^i$$

Def. El grado de un polinomio es la potencia más grande de x que esté acompañada por un coeficiente no nulo.

Ej.

$$1 - x^7 = (-1)\cdot x^7 + 1\cdot x^0 = 0\cdot x^8 + 1 - x^7$$

Su grado es 7.

Ej.

$3/4 = (3/4)\cdot x^0$, su grado es 0.

Técnicas de demostración

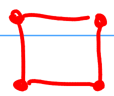
¿Qué es una demostración?

1) Demostración directa (por construcción)

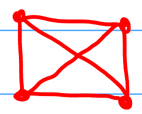
Ejemplo:

Def. Un grafo es k-regular si todos los vértices tienen el mismo grado.

Ej.



2-regular

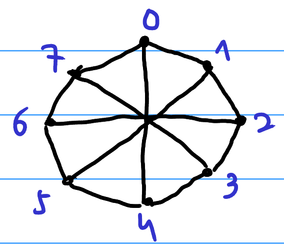


3-regular

Teorema. Para todo $n > 2$ con n par, existe un grafo G que tiene n vértices y además es 3-regular.

Dem.

Ejemplo $n = 8$

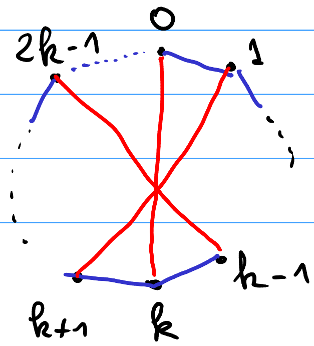


En general, si $n > 2$ y n es par, $n = 2k$

tomemos el grafo $G = (V, E)$.

$$V = \{0, 1, \ldots, 2k-1\}$$

$$E = \{ \{i, i+1\} \mid 0 \leq i < 2k-1 \}$$
$$\cup \{ \{2k-1, 0\} \}$$
$$\cup \{ \{i, i+k\} \mid 0 \leq i \leq k-1 \}$$



2) Demostración por el absurdo (reducción al absurdo).

Idea. Si queremos demostrar P,
una manera de proceder es
suponer que vale $\neg P$,
hacer un razonamiento que llegue
a algo imposible (por ejemplo $1=2$),
y concluir que no es posible
que valga $\neg P$.

Lema. Si $n \in \mathbb{Z}$ es impar, entonces n^2 también es impar.

Dem.

Supongamos que $n \in \mathbb{Z}$ es impar.

Entonces $n = 2k+1$. (ej. $7 = 2\cdot 3+1$)

Entonces

$$
\begin{aligned}
n^2 &= (2k+1)^2 \\
&= (2k+1)\cdot(2k+1) \\
&= (2k)^2 + 2k + 2k + 1 \\
&= \underbrace{4k^2 + 4k}_{= 2(2k^2+2k)} + 1
\end{aligned}
$$

De modo que n^2 es impar.

Teorema. $\sqrt{2}$ es irracional. (Es decir $\sqrt{2} \in \mathbb{R} \setminus \mathbb{Q}$).

Dem.

- Queremos ver que $\sqrt{2} \notin \mathbb{Q}$.
- Para demostrarlo por el absurdo, vamos a suponer que $\sqrt{2} \in \mathbb{Q}$ y vamos a llegar a una contradicción.
- Supongamos, para llegar a un absurdo, que $\sqrt{2} \in \mathbb{Q}$.
  Esto quiere decir que:

$$\sqrt{2} = \frac{p}{q} \quad \text{donde } p, q \in \mathbb{Z}$$

Además, podemos suponer que la fracción está reducida, es decir que p, q no tienen factores primos en común.

- $$2 = (\sqrt{2})^2 = \left(\frac{p}{q}\right)^2 = \frac{p^2}{q^2}$$

- Entonces $p^2 = 2 \cdot q^2$.
- Con esto sabemos que p^2 es par.
  Por el lema anterior, p es par.
  Por lo tanto $p = 2k$ para algún $k \in \mathbb{Z}$.

- Además, $2q^2 = p^2 = (2k)^2 = 4k^2$
  Por lo tanto:

$$q^2 = 2k^2$$

  Entonces q^2 es par.
  Por el lema anterior, q es par.

- Observemos que

$$p \text{ es par} \quad y \quad q \text{ es par}$$

Entonces 2 es un factor de p y q.
Entonces la fracción $\frac{p}{q}$ no estaba reducida.
Esto es una contradicción. Como esto es imposible, llegamos a que $\sqrt{2} \notin \mathbb{Q}$.

3) <u>Demostración por inducción</u>

Si queremos demostrar que todos los números naturales cumplen con alguna propiedad "P", el principio de inducción afirma que alcanza con demostrar dos cosas:

(1) $P(0)$

(2) $\forall n \in \mathbb{N}.\ P(n) \Rightarrow P(n+1)$.

<u>Ejemplo</u>:

<u>Teorema</u>. $\sum_{i=1}^{n} 2i-1 = n^2$ <u>Ejemplo</u>: <u>$n=3$</u> $1+3+5 = 9$

<u>Dem</u>. Por inducción en n.

$$P(n) \equiv \text{"} \sum_{i=1}^{n} 2i-1 = n^2 \text{"}$$

(1) Veamos que se verifica $P(0)$.

$$P(0) \equiv \text{"} \underbrace{\sum_{i=1}^{0} 2i-1}_{0} = \underbrace{0^2}_{0} \text{"} \quad \checkmark$$

(2) Veamos que $\forall n \in \mathbb{N}.\ P(n) \Rightarrow P(n+1)$.

Sea n un número natural arbitrario.

Podemos suponer que vale la hipótesis inductiva

$$P(n) = \text{"} \sum_{i=1}^{n} 2i-1 = n^2 \text{"}$$

Veamos que vale $P(n+1) \equiv \text{"} \sum_{i=1}^{n+1} 2i-1 = (n+1)^2 \text{"}$.

$$\sum_{i=1}^{n+1} 2i-1 = \left(\sum_{i=1}^{n} 2i-1\right) + 2(n+1)-1 \underset{\uparrow \text{POR HI}}{=} n^2 + 2(n+1) - 1 = n^2 + 2n + 2 - 1$$
$$= n^2 + 2n + 1 = (n+1)^2.$$

Cardinalidad

$\#\{1,2,3\} = 3 = \#\{a,b,c\}$

$\#\{1,2,3,4\} = 4$

Cantor.

$\mathbb{N}$ $\qquad 2\mathbb{N} = \{n \in \mathbb{N} \mid n \text{ par}\} = \{0,2,4,\dots\}$

$\#\mathbb{N} \overset{?}{=} \#(2\mathbb{N})$ $\qquad \#\mathbb{N} \overset{?}{=} \#\mathbb{Q}$ $\qquad \#\mathbb{N} \overset{?}{=} \#\mathbb{R}$

Def. Dados dos conjuntos A, B
decimos que tienen el mismo cardinal,
y lo notamos $A \approx B$,
si y sólo si existe una función biyectiva $f : A \to B$.

Obs. (Simetría). $A \approx B \iff B \approx A$

Obs. (Reflexividad). $A \approx A$ $\qquad id : A \to A$

Obs. (Transitividad). Si $A \approx B$, $B \approx C$, entonces $A \approx C$.
Si $f : A \to B$ biyectiva,
$g : B \to C$ biyectiva
entonces $g \circ f : A \to C$.

<u>Pregunta</u>. $\mathbb{N} \overset{?}{\approx} 2\mathbb{N}$

$$\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & \dots \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \\ 0 & 2 & 4 & 6 & 8 & \dots \end{array}$$

Sí, $\mathbb{N} \approx 2\mathbb{N}$, porque $f : \mathbb{N} \longrightarrow 2\mathbb{N}$
$f(n) = 2 \cdot n$
es biyectiva.

<u>Pregunta</u>. $\mathbb{N} \overset{?}{\approx} \mathbb{Q}$

Sí, $\mathbb{N} \approx \mathbb{Q}$.

Vamos a ver algo un poco más fácil, $\mathbb{N} \approx \mathbb{Q}_{>0}$.

	1	2	3	4	5	6	7	...
1	1/1	2/1	3/1	4/1	5/1	6/1	7/1	...
2	1/2	~~2/2~~	3/2	~~4/2~~	5/2	~~6/2~~	7/2	...
3	1/3	2/3	~~3/3~~	4/3	5/3	~~6/3~~	7/3	...
4	1/4	2/4	3/4	~~4/4~~	5/4	6/4	7/4	...
5								
6								
⋮								

Suman 2: $1/1$
Suman 3: $1/2,\ 2/1$
Suman 4: $1/3,\ 3/1$
Suman 5: $1/4,\ 2/3,\ 3/2,\ 5/1$
⋮

70/9

$$\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \dots \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \\ 1/1 & 1/2 & 2/1 & 1/3 & 3/1 & 1/4 & 2/3 & 3/2 & 5/1 & \dots \end{array}$$

Esta construcción nos da una función $f : \mathbb{N} \to \mathbb{Q}_{>0}$ y además f es biyectiva.

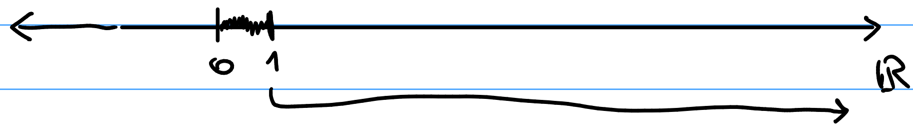
$$\mathbb{N} \approx \mathbb{Q}_{>0}$$

$$\mathbb{N} \approx \mathbb{Q}$$

$\underbrace{\mathbb{N} \approx \mathbb{Z} \approx \mathbb{Q}} \not\approx \underline{\mathbb{R}}$

Pregunta. $\mathbb{N} \approx \mathbb{R}$ ¡No!

1) $\mathbb{R} \approx [0,1]$



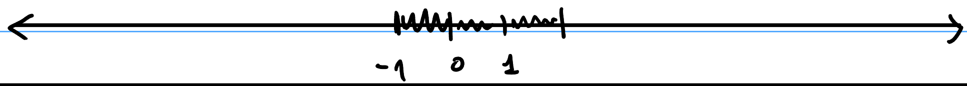
$[1,+\infty) \approx (0,1]$ $f: [1,+\infty) \longrightarrow (0,1]$

$f(x) = 1/x$

$0 < x \leq 1$ $1/x = 1/y$

$1 \leq 1/x$

$1/(1/x) = x$



---

2) Para ver $\mathbb{N} \not\approx \mathbb{R}$, veamos $\mathbb{N} \not\approx [0,1]$

Por el absurdo, supongamos que $\mathbb{N} \approx [0,1]$,
y a partir de esto lleguemos a una contradicción.
En ese caso, existiría una función biyectiva
$f: \mathbb{N} \longrightarrow [0,1]$.

$$
\begin{aligned}
0 &\longmapsto f(0) = 0{,}d_1^{(0)}\, d_2^{(0)}\, d_3^{(0)}\, d_4^{(0)} \ldots \\
1 &\longmapsto f(1) = 0{,}d_1^{(1)}\, d_2^{(1)}\, d_3^{(1)}\, d_4^{(1)} \ldots \\
2 &\longmapsto f(2) = 0{,}d_1^{(2)}\, d_2^{(2)}\, d_3^{(2)}\, d_4^{(2)} \ldots \\
3 &\longmapsto f(3) = 0{,}d_1^{(3)}\, d_2^{(3)}\, d_3^{(3)}\, d_4^{(3)} \ldots \\
4 &\longmapsto f(4) = \vdots \\
5 &\longmapsto f(5) = \vdots \\
6 &\longmapsto f(6) = \vdots \\
\vdots &
\end{aligned}
$$

Sabemos que f es suryectiva.
El número en la diagonal es $0{,}d_1^{(0)}\, d_2^{(1)}\, d_3^{(2)}\, d_4^{(3)} \ldots d_{i+1}^{(i)} \ldots$

Consideremos el número $z = 0, e_1 \ e_2 \ e_3 \ e_4 \ \dots \in [0,1]$.

donde $e_i \neq d_i^{(i-1)}$.

Notemos que $z \in [0,1]$ pero $\nexists \, i. \ f(i) = z$.

Por lo tanto f no es surjectiva.

Esto es absurdo.

Teoría de la Computación

Computabilidad — Complejidad

1) Computabilidad

Problema: ordenar una lista. es computable

$$[4,2,1,3] \longrightarrow [1,2,3,4]$$

En la década de 1930 surgieron tres definiciones de computabilidad:

1) Cálculo-λ, Church
2) Funciones recursivas, Kleene
3) Máquinas de turing, Turing

"Tesis de Church"

Ejemplo de problema no computable

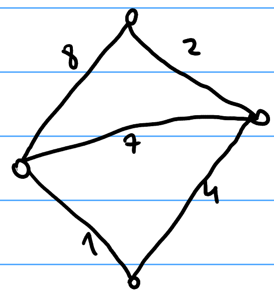
Sistema 1	Sistema 2
$ab \to ba$	$ab \to bbaa$
aaabab	aab
$\to$ aabaab	$\to$ abbaa
$\to$ abaaab	$\to$ bbaabaa
$\to$ baaaab	$\to$ bbabbaaaa
$\to \ldots \to$ bbaaaa	$\to$ bbbbaabaaaa
	$\to \ldots$

Pregunta: ¿Se puede hacer un programa que reciba como input reglas de reescritura y produzca output un booleano que diga si el sistema siempre termina?

2) <u>Complejidad</u>: Eficiencia de la solución a un problema.

<u>Ejemplo</u>. Ordenar una lista de n elementos se puede resolver haciendo $O(n \cdot \log n)$ Comparaciones entre los elementos en peor caso.

<u>Ejemplo</u>.



<u>Problema</u>: Encontrar un camino que recorra todas las ciudades sin repetir y que tenga costo mínimo.

¿Se puede resolver este problema de manera eficiente en tiempo?

Lenguajes regulares

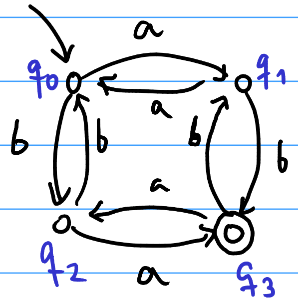
1) Autómata Finito Determinístico. (AFD)

Definición. Un AFD es una 5-upla:

$$(\Sigma, Q, q_0, Q_F, \delta).$$

$Q = \{q_1, q_2, q_3, q_4\}$

$Q_F = \{q_3\}$



aaba

aaaa

$\delta : Q \times \Sigma \longrightarrow Q$

Un AFD acepta una cadena $a_1 \cdots a_n \in \Sigma$ si empezando desde el estado inicial q_0 y consumiendo la cadena se llega a alguno de los estados finales.

$\exists\, q_1, q_2, \ldots, q_n \,.$

$$\delta(q_{i-1}, a_i) = q_i \qquad \forall\, 1 \leq i \leq n$$

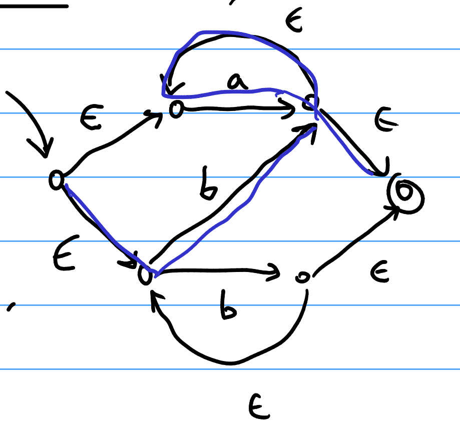
y además $q_n \in Q_F$.

2) Autómata Finito No Determinístico (AFN)

Def. Un AFN es una 5-upla

$$(\Sigma, Q, q_0, Q_F, \delta)$$

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \longrightarrow \mathcal{P}(Q).$$



$ba = \epsilon b \epsilon a \epsilon$

Un AFN acepta una cadena $w \in \Sigma^*$ si la cadena w se puede escribir como $w = x_1 x_2 \cdots x_n$ donde $x_i \in \Sigma \cup \{\epsilon\}$ e (igual que antes) existen estados $q_1, \ldots, q_n$ $\quad q_i \in \delta(q_{i-1}, x_i)$ $\forall\, 1 \leq i \leq n$

y además $q_n \in Q_F$.

3) Expresiones regulares

Una expresión regular es una expresión que se construye así:

1) $\emptyset$ es una ER
2) ϵ es una ER
3) a es una ER si $a \in \Sigma$
4) Si R, S son ER, entonces $R \cdot S$ es una ER
5) si R, S son ER, entonces $R | S$ es una ER
6) si R es una ER, entonces R^* es una ER.

Cada ER denota un lenguaje.
Si R es una ER, $\mathcal{L}(R) \subseteq \Sigma^*$.

$\mathcal{L}(\emptyset) = \emptyset$ $\qquad \mathcal{L}(R \cdot S) = \mathcal{L}(R) \cdot \mathcal{L}(S)$
$\mathcal{L}(\epsilon) = \{\epsilon\}$ $\qquad \mathcal{L}(R|S) = \mathcal{L}(R) \cup \mathcal{L}(S)$
$\mathcal{L}(a) = \{a\}$ $\qquad \mathcal{L}(R^*) = \mathcal{L}(R)^*$

Teorema. Sea $L \subseteq \Sigma^*$ un lenguaje.
Son equivalentes:
1) Hay un AFD que acepta las palabras de L y sólo esas.
2) Hay un AFN que acepta las palabras de L y sólo esas.
3) Hay una ER. R tal que $L = \mathcal{L}(R)$.

Estos lenguajes se llaman <u>lenguajes regulares</u>.

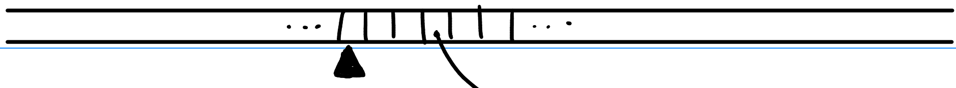
Propiedades de clausura de lenguajes regulares:

• Si L_1, L_2 son lenguajes regulares, entonces:

1) $L_1 \cup L_2$ es regular
2) $L_1 \cdot L_2$ es regular
3) L_1^* es regular
4) $\Sigma^* \setminus L_1$ es regular
5) $L_1 \cap L_2$ es regular.

$L_1 \cap L_2 = \Sigma^* \setminus ((\Sigma^* \setminus L_1) \cup (\Sigma^* \setminus L_2))$

$\neg(A \wedge B) \equiv \neg A \vee \neg B$



En cada celda de la cinta hay un símbolo de un conjunto Γ.

- Σ es el conjunto de símbolos del alfabeto.
- Γ es el conjunto de símbolos de cinta.

$$\Sigma \subseteq \Gamma \qquad \sqcup \in \Gamma$$

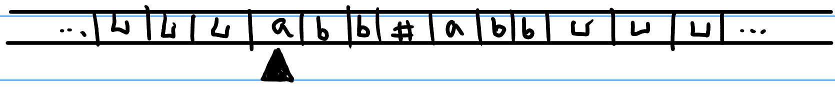
$\sqcup$ blanco

<u>Problema.</u> En el alfabeto $\Sigma = \{a, b, \#\}$ decidir si una cadena $w_0 \in \Sigma^*$ pertenece al lenguaje

$$L = \{w\#w \mid w \in \Sigma^*\}$$

Por ejemplo $abb\#abb \in L$

pero $abb\#bba \notin L$.



si la entrada fuera $abb\#abb$

$$\Sigma = \{a, b, \#\}$$
$$\Gamma = \{a, b, \#, \sqcup, x\}$$

Si la máquina está en el estado q_0 vamos a escribir esa configuración así:

$$q_0\, abb\#abb$$
$$\curvearrowright\ x\, q_1\, bb\#abb$$
$$\curvearrowright\ x\, b\, q_1\, b\#abb$$
$$\curvearrowright\ x\, bb\, q_1\, \#abb$$
$$\curvearrowright\ x\, bb\#\, q_2\, abb$$
$$\curvearrowright\ x\, bb\, q_2\, \#xbb$$
$$\curvearrowright\ \ldots$$

$\cancel{a}\cancel{b}\cancel{b}\#\cancel{a}\cancel{b}\cancel{b}$

Def. Una máquina de Turing es una 7-upla $(\Sigma, \Gamma, Q, \delta, q_0, q_{accept}, q_{reject})$ donde:

- Σ es un conjunto finito, el alfabeto,
- Γ es un conjunto finito tal que $\sqcup \in \Gamma$, $\Sigma \subseteq \Gamma$,
- Q es un conjunto finito, de "estados",
- $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$
- $q_0, q_{accept}, q_{reject} \in Q$
- $q_0 \neq q_{accept}$, $q_0 \neq q_{reject}$, $q_{accept} \neq q_{reject}$

Def. Una configuración de la máquina es una tripla

$$(w_1, q, w_2) \in \Gamma^* \times Q \times \Gamma^*$$

Ej. $aab\,q_0\,\#aab$

$(aab, q_0, \#aab)$

$$\ldots \sqcup\sqcup\sqcup\; a\; a\; b\; \#\; a\; a\; b\; \sqcup\sqcup\sqcup \ldots$$

$\uparrow q_0$

- Hay una transición desde una configuración

$$w_1\, q\, x\, w_2$$

hacia una configuración

$$w_1\, y\, q'\, w_2 \qquad \text{si} \qquad \delta(q, x) = (q', y, R).$$

- Hay una transición desde una configuración

$$w_1\, z\, q\, x\, w_2$$

hacia una configuración

$$w_1\, q'\, z\, y\, w_2 \qquad \text{si} \qquad \delta(q, x) = (q', y, L).$$

Def. Una máquina de Turing $(\Sigma, \Gamma, Q, \delta, q_0, q_{accept}, q_{reject})$

- **acepta** una cadena w
  si desde la configuración $q_0 w$
  se alcanza una configuración $w_1 q_{accept} w_2$
  haciendo transiciones
  y sin pasar por el estado q_{reject}.

- **rechaza** una cadena w
  si desde $q_0 w$
  se alcanza $w_1 q_{reject} w_2$ sin pasar por el estado q_{accept}.

Def. Dado un lenguaje $L \subseteq \Sigma^*$, decimos que una máquina de Turing M:

1) **decide** el lenguaje L si y sólo si:
   - **acepta** todas las cadenas de L,
   - **rechaza** todas las cadenas de $\Sigma^* \setminus L$.

2) **reconoce** (o "**semi-decide**") el lenguaje L si y sólo si:
   - **acepta** todas las cadenas de L,
   - **no acepta** a todas las cadenas de $\Sigma^* \setminus L$.

Obs. Si M decide L, entonces M reconoce L.
(pero no vale al revés).

Problema. Dar una máquina de Turing que decida el lenguaje:

$$L = \{ w w^r \mid w \in \Sigma^* \} \qquad \Sigma = \{a, b\}.$$

$\Gamma = \{a, b, \sqcup\}$