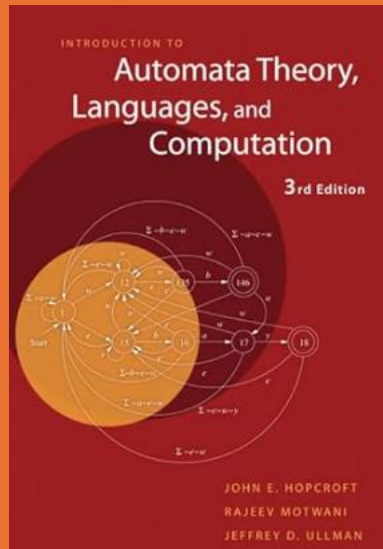




Diseño Lógico y Arquitectura de Sistemas Digitales

2024 – 1er Cuatrimestre

Unidad 1



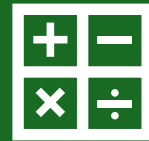
- Autómatas Finitos y Sistemas Secuenciales
 - Introducción a la Teoría de Autómatas.
 - Lenguajes, aceptores y traductores.
 - Modelos de Moore y Mealy.
- Bibliografía:

Introduction to Automata Theory, Languages and Computation, J. Hopcroft, R. Motwani, J. Ullman, Third Edition, Addison Wesley, 2007.

Introducción



Problema: Queremos hacer una máquina de cálculo



Pero... qué es calcular/procesar/computar?



Empecemos por recordar qué es una función para ver luego si la podemos calcular.

- ¿Qué es una función?

Sean X e Y dos conjuntos arbitrarios, una función $f: X \rightarrow Y$ asigna para toda $x \in X$ una $y = f(x) \in Y$ tal que si $y_1, y_2 \in Y$, $y_1 = f(x_1) \neq y_2 = f(x_2)$ se cumple que $x_1 \neq x_2$

- Si X e Y son dos conjuntos de cantidad finita de elementos, $f: X \rightarrow Y$ se puede expresar con una tabla:

f

X	Y
x_1	y_1
x_2	y_2
x_3	y_1
x_4	y_2

- Función inyectiva

Sea f una función cuyo dominio es el conjunto X , se dice que la función f es **inyectiva** si para todo a y b en X , si $f(a) = f(b)$ entonces $a = b$, esto es $f(a) = f(b)$ implica $a = b$. Equivalentemente, si $a \neq b$ entonces $f(a) \neq f(b)$. Simbólicamente,

$$\forall a, b \in X, f(a) = f(b) \implies a = b$$

que es equivalente a su [contrarrecíproco](#)

$$\forall a, b \in X, a \neq b \implies f(a) \neq f(b)$$

Para probar que una función no es inyectiva, basta con hallar dos valores distintos del dominio, cuyas imágenes en el codominio son iguales.

$$f(x) = x + 1$$

$$f(x) = x^2$$

- Función sobreyectiva

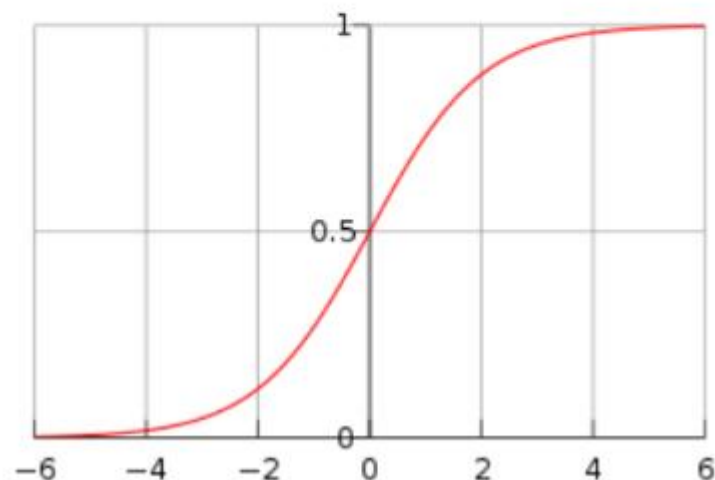
Una **función sobreyectiva** es una función cuya imagen es igual a su **codominio**. Equivalentemente, una función f con dominio X y codominio Y es sobreyectiva si para cada y en Y existe al menos una x en X tal que $f(x) = y$.

Simbólicamente

Si $f : X \rightarrow Y$ entonces se dice que f es sobreyectiva si

$$\forall y \in Y, \exists x \in X : f(x) = y$$

$$f(x) = \frac{1}{1 + e^{-x}}$$



- Función biyectiva

Formalmente, dada una función f :

$$\begin{aligned} f: X &\longrightarrow Y \\ x &\longmapsto y = f(x) \end{aligned}$$

La función es biyectiva si se cumple la siguiente condición:

$$\forall y \in Y : \exists! x \in X / f(x) = y$$

Es decir, para todo y de Y se cumple que existe un único x de X , tal que la función evaluada en x es igual a y .

una **función** es **biyectiva** si es al mismo tiempo **inyectiva** y **sobreyectiva**; es decir, si todos los elementos del **conjunto de salida** tienen una **imagen** distinta en el **conjunto de llegada**, y a cada elemento del conjunto de llegada le **corresponde** un **elemento del conjunto** de salida.

Cardinalidad de un conjunto

- Definición de cardinal:

Si un conjunto X es finito, el cardinal $\#X$ es la cantidad de elementos que posee.

Ejemplo: $X=\{a,b,c,d\}$, $\#X = 4$

- Definición:

Sean dos conjuntos arbitrarios X, Y no necesariamente finitos. Si existe una biyección $f: X \rightarrow Y$, decimos que $\#X = \#Y$ (es decir que “tienen la misma cantidad de elementos”).

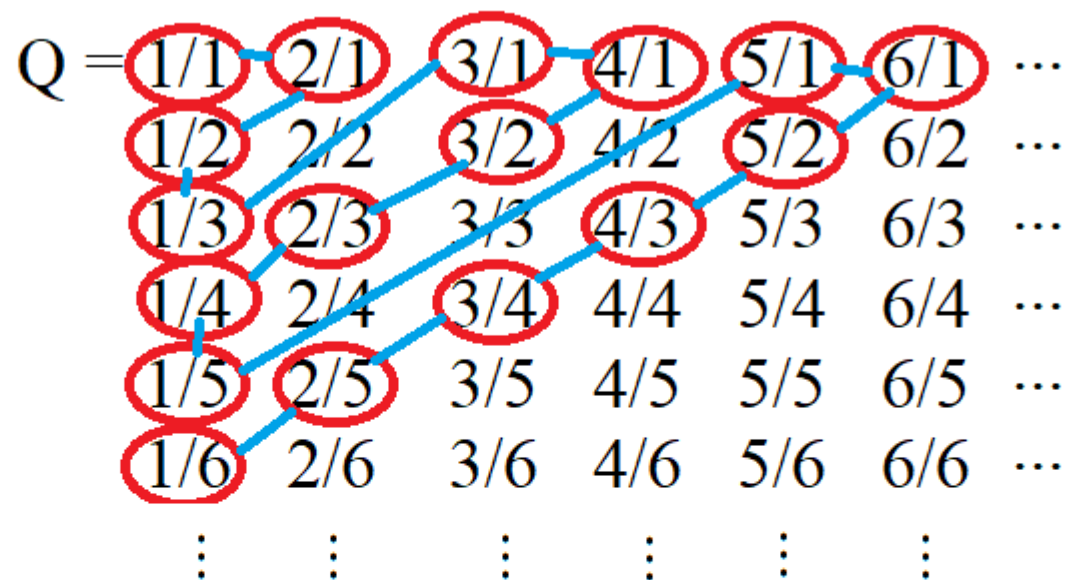
- Los enteros, cuántos son?

$f: \mathbb{N} \rightarrow \mathbb{Z}$
Es biyectiva?

$\mathbb{N} =$	1	2	3	4	5	6	7	8	...
$\mathbb{Z} =$	1	-1	2	-2	3	-3	4	-4	...

Sí, entonces $\#\mathbb{N} = \#\mathbb{Z}$

- Y los racionales, cuántos son?



- Los reales, cuántos son?
- En el intervalo $(0,1)$ hay tantos elementos como en todo \mathbb{R} .

Solo tenemos que hallar una $f: (0,1) \rightarrow \mathbb{R}$ biyectiva

$$y = f(x) = \tan\left[\pi\left(x - \frac{1}{2}\right)\right]$$

Entonces $\#(0,1) = \#\mathbb{R}$

- En el intervalo $(0,1)$ hay más racionales o reales?

Argumento diagonal de Cantor:

La demostración es por **reducción al absurdo**:

1. Se supone que el intervalo $[0,1]$ es infinito numerable.
2. En ese caso se podría elaborar una secuencia de números, (r_1, r_2, r_3, \dots) .
3. Se sabe que los reales entre 0 y 1 pueden ser representados solamente escribiendo sus decimales.
4. Se colocan los números en la lista (no necesariamente en orden). Considerando los decimales periódicos, como $0.499\dots = 0.500\dots$, como los que tienen infinitos nueves.

La secuencia podría tener un aspecto similar a:

$$r_1 = 0.5105110\dots$$

$$r_2 = 0.4132043\dots$$

$$r_3 = 0.8245026\dots$$

$$r_4 = 0.2330126\dots$$

$$r_5 = 0.4107246\dots$$

$$r_6 = 0.9937838\dots$$

$$r_7 = 0.0105135\dots$$

...

Dada la primera premisa dicha lista contiene todos los números reales entre 0 y 1. Con esto, se puede construir un número x que debería estar en la lista. Para eso usamos los números de la diagonal.

$$r_1 = 0. \mathbf{5} 1 0 5 1 1 0 \dots$$

$$r_2 = 0. 4 \mathbf{1} 3 2 0 4 3 \dots$$

$$r_3 = 0. 8 2 \mathbf{4} 5 0 2 6 \dots$$

$$r_4 = 0. 2 3 3 \mathbf{0} 1 2 6 \dots$$

$$r_5 = 0. 4 1 0 7 \mathbf{2} 4 6 \dots$$

$$r_6 = 0. 9 9 3 7 8 \mathbf{3} 8 \dots$$

$$r_7 = 0. 0 1 0 5 1 3 \mathbf{5} \dots$$

...

- El número x está definido así: al k -ésimo dígito decimal de x le corresponde el k -ésimo dígito decimal de r_k más 1 (en caso de que fuera un nueve, se le asigna el dígito cero)

Entonces $x = 0.6251346\dots$

El número x es claramente un real. Pero... ¿Dónde está x ?

- Si yo quisiera decir que x está en el n -ésimo lugar de mi lista, no sería cierto, ya que el n -ésimo dígito de r_n es distinto al de x .
- Entonces esta no es una lista completa de los reales en el intervalo $[0, 1]$.
- Existe una contradicción, que nace de la premisa de suponer que estos números son infinitos numerables.

- Entonces $\#N = \#Q = \#Z < \#(0,1) = \#R$
- Ejercicio: Falta probar que $\#Q = \#Z$ (obvio)

Qué función se puede computar?

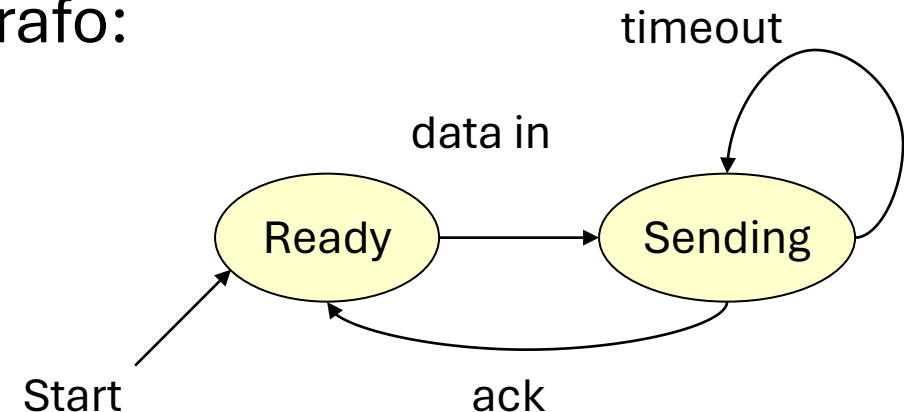
- Una función se puede computar si tengo un método de cálculo “efectivo”
- Sea $f: \mathbb{R} \rightarrow \mathbb{R}$, ¿puedo computar $f(x) = x^2$?
- Sea $f: \mathbb{N} \rightarrow \mathbb{N}$, ¿puedo computar $f(x) = x^2$?

- Pareciera que el problema son los irracionales...
- Ejemplo de Numberphile:
- Sea $D=\{1,2,3,4\}$, sea $I=\{0,1\}$
- Sea $f : D \rightarrow I$ tal que
 - $f(n) = 1$ si existen z_1, z_2, \dots, z_n tales que $z_i + z_j = 2^k$ para $i \neq j$; $i, j = 1, 2, \dots, n$; n perteneciente a D .
 - $f(n) = 0$ en otro caso
- Calculemos f . Como D, I son finitos temenos que hallar una tabla:
 - $f(1) = ?$ Cómo calculamos $f \dots ?$
 - $f(2) = ?$
 - $f(3) = ?$
 - $f(4) = ?$

Hay funciones entonces que aunque cuyo dominio e imagen sean ambos finitos no las podemos calcular!!!

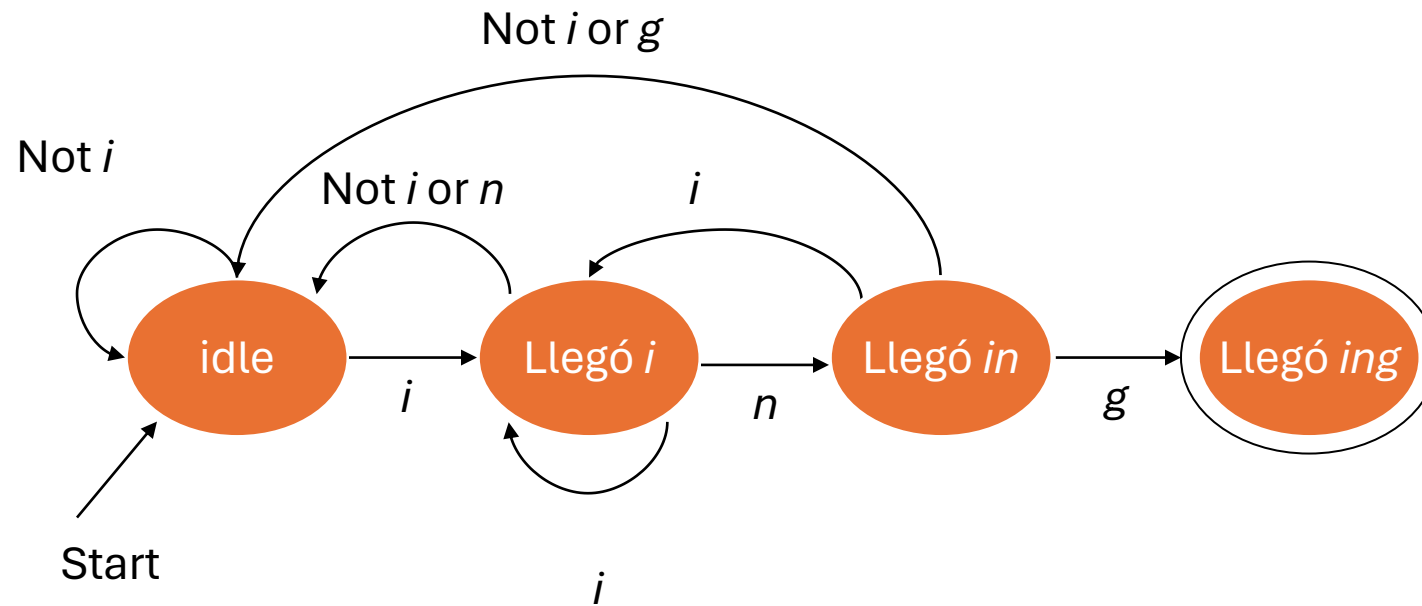
Autómatas Finitos

- Explicación Informal:
 - Colección finita de estados con reglas de transición que te permiten ir de un estado a otro
 - La aplicación original fueron los circuitos switching secuenciales, donde el “estado” era la configuración interna de bits
 - Hoy en día, múltiples implementaciones en hardware y software pueden ser modelados utilizando autómatas Finitos (FA).
- Representación más simple es mediante un grafo:
 - Nodos = estados
 - Arcs indican las transiciones entre estado
 - Arcs labels indican las causas de la transición



FA: Ejemplo

- Reconocer strings que terminan en “ing”



FA: Definiciones

Σ

Alfabeto: Cualquier conjunto finito de símbolos. Ejemplos:

ASCII

Unicode

$\{0, 1\}$ (alfabeto binario)

$\{a, b, c\}$



Cadena de caracteres (strings): Secuencia finita de símbolos seleccionados de algún alfabeto. Ejemplo, 01101 es una cadena del alfabeto binario $\Sigma = \{0, 1\}$.

λ

Cadena vacía: Presenta cero apariciones de símbolos y puede construirse en cualquier alfabeto. La representamos con λ .

FA: Definiciones

- **Potencia de un alfabeto:** Conjunto de todas las cadenas de una determinada longitud de dicho alfabeto.
 - Definimos Σ^k al conjunto de cadenas de longitud k , donde cada uno de los simbolos pertenece a Σ
 - Observar que $\Sigma^0 = \{\lambda\}$, independientemente de Σ
 - Si $\Sigma = \{0,1\}$, entonces $\Sigma^1 = \{0,1\}$, $\Sigma^2 = \{00,01,10,11\}$, $\Sigma^3 = \{000,001,010,011,100,101,110,111\}$
 - Al conjunto de todas las cadenas de un alfabeto Σ se la designa Σ^*

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots$$

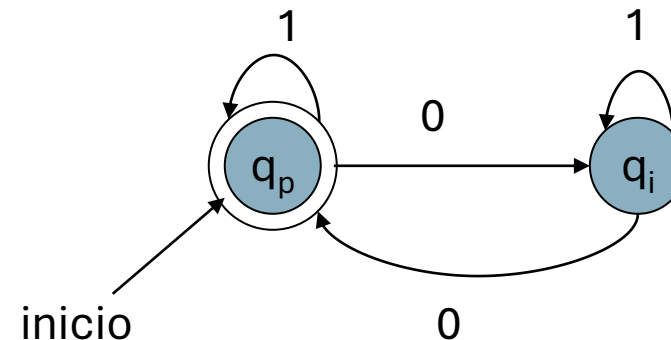
FA: Definiciones

- Un lenguaje es un conjunto de cadenas de Σ^* para algún alfabeto Σ
- Ejemplos:
 - Lenguaje de todas las cadenas que constan de n ceros seguidos de n unos para cualquier $n \geq 0$: $\{\lambda, 01, 0011, 000111, \dots\}$
 - Conjunto de cadenas formadas por el mismo número de ceros que de unos: $\{\lambda, 01, 10, 0011, 0101, 1001, \dots\}$
 - El conjunto de números binarios cuyo valor es un número primo: $\{10, 11, 101, 111, 1011, \dots\}$
 - Σ^* es un lenguaje para cualquier alfabeto Σ .
 - \emptyset , el lenguaje vacío, es un lenguaje para cualquier alfabeto
 - $\{\lambda\}$ también es un lenguaje de cualquier alfabeto. Nota: $\emptyset \neq \{\lambda\}$; el primero no contiene ninguna cadena y el Segundo solo tiene una cadena

Autómata Finito Determinístico

- Sólo puede estar en un único estado después de leer cualquier secuencia de entradas.
- Un **autómata finito determinístico (AFD)** se define como una 5-upla $\langle Q, \Sigma, \delta, q_0, F \rangle$ donde:
 - Q es el conjunto finito de estados.
 - Σ es el alfabeto de entrada.
 - $\delta : Q \times \Sigma \rightarrow Q$ es la función de transición.
 - $q_0 \in Q$ es el estado inicial.
 - $F \subseteq Q$ es el conjunto de estados finales.
- **Ejemplo:**

$$A = \langle \{q_p, q_i\}, \{0, 1\}, \delta, q_p, \{q_p\} \rangle$$



Autómata Finito Determinístico

- La función de transición de un AFD puede extenderse para aceptar como segundo argumento cadenas en el alfabeto de entrada. La versión extendida de la función de transición δ sería $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$, definida como:
 - $\hat{\delta}(q, \lambda) = q$
 - $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$, con $x \in \Sigma^*$ y $a \in \Sigma$
- Intuitivamente, $\hat{\delta}(q, xa)$ en su definición, implica realizar todas las transiciones para llegar desde el estado q hasta el estado resultante de consumir la cadena x , y luego hacer un paso más para consumir a .

Autómata Finito Determinístico

- Función de transición Delta Extendida

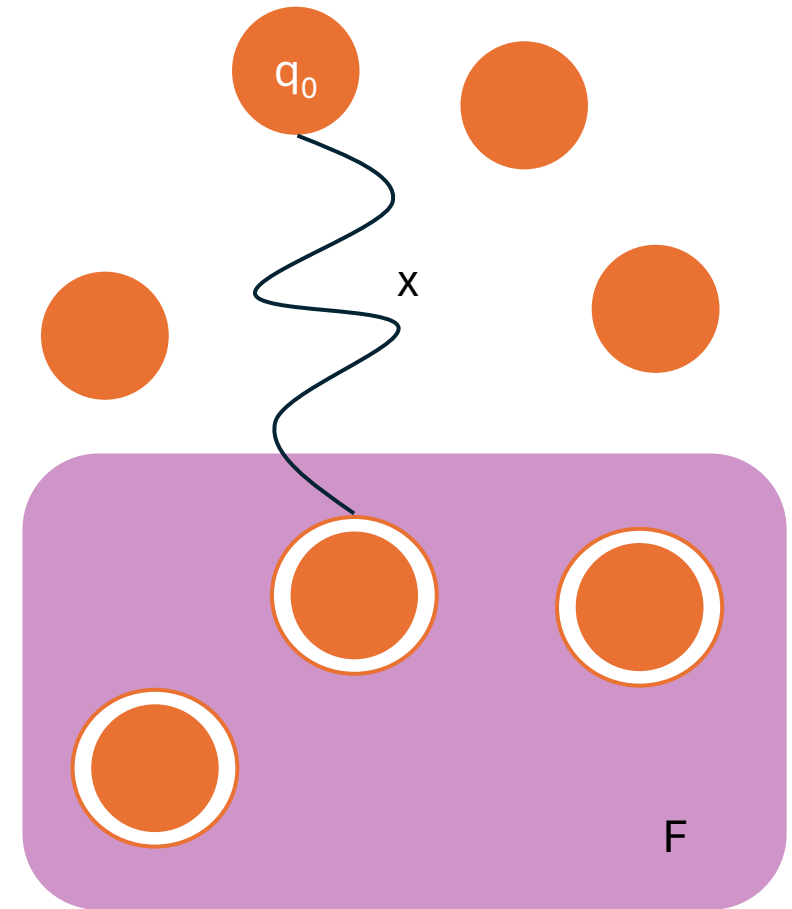
	0	1
A	A	B
B	A	C
C	C	C

$$\begin{aligned} \delta(B, 011) &= \delta(\delta(B, 01), 1) = \delta(\delta(\delta(B, 0), 1), 1) = \\ \delta(\delta(A, 1), 1) &= \delta(B, 1) = C \end{aligned}$$

Autómata Finito Determinístico

- **Cadena Aceptada:** Una cadena x es aceptada por un AFD $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ cuando $\hat{\delta}(q_0, x) \in F$.
- **Lenguaje Aceptado por un AFD:**
Dado un AFD $M = \langle Q, \Sigma, \delta, q_0, F \rangle$, el lenguaje aceptado por M , $\mathcal{L}(M)$, es el conjunto de cadenas aceptadas por M y se define como:

$$\mathcal{L}(M) = \{x \in \Sigma^* : \hat{\delta}(q_0, x) \in F\}$$

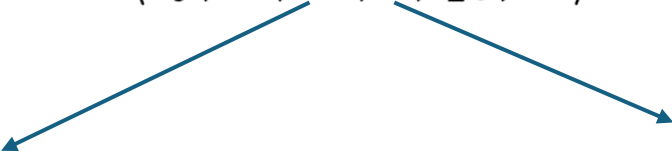


Traductor Finito

- Es un autómata finito que no solo acepta o rechaza cadenas, sino que también devuelve otra cadena cuando **acepta**.
 - “traduce” la cadena aceptada
- Un traductor finito determinístico (TFD) A es una tupla de la forma:

$$A = \langle Q, \Sigma, \Delta, \delta, q_0, F \rangle$$

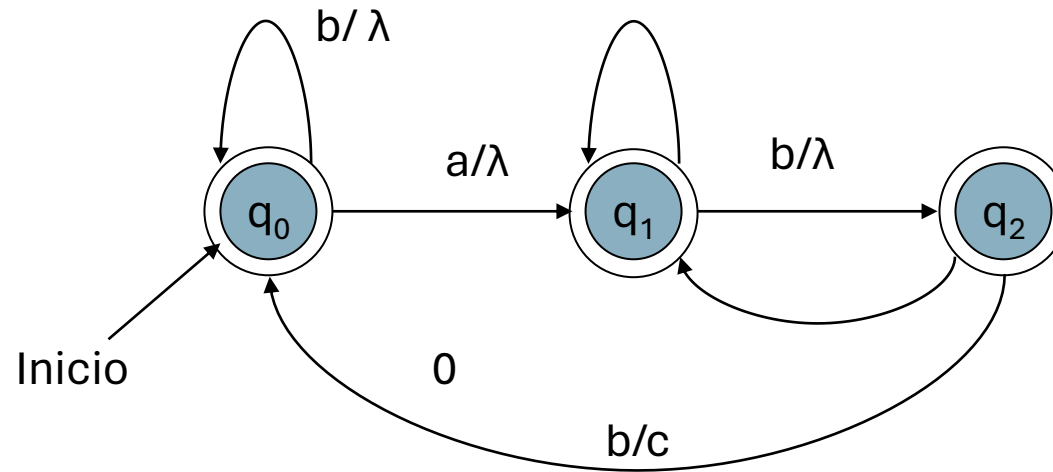
Alfabeto de las
cadenas de salida

$$\delta : Q \times \Sigma \rightarrow Q \times \Delta^*$$


Traductor Finito Determinístico

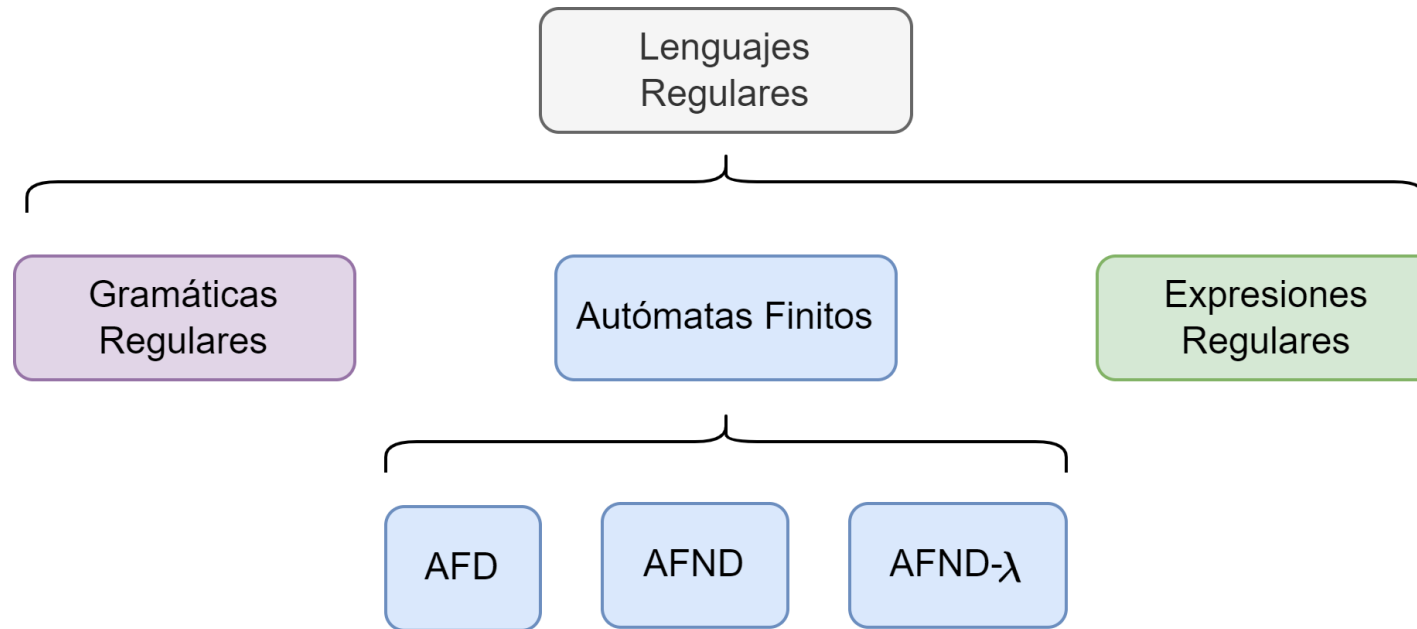
- Ejemplo:

$$\{(\omega, c^i) \mid \omega \in \{a, b\}^* \wedge i = (\text{cantidad de apariciones de } abb \text{ en } \omega)\}$$



$$A = \langle \{q_0, q_1, q_2\}, \{a, b\}, \{c\}, d, q_0, \{q_0, q_1, q_2\} \rangle$$

Lenguajes Regulares



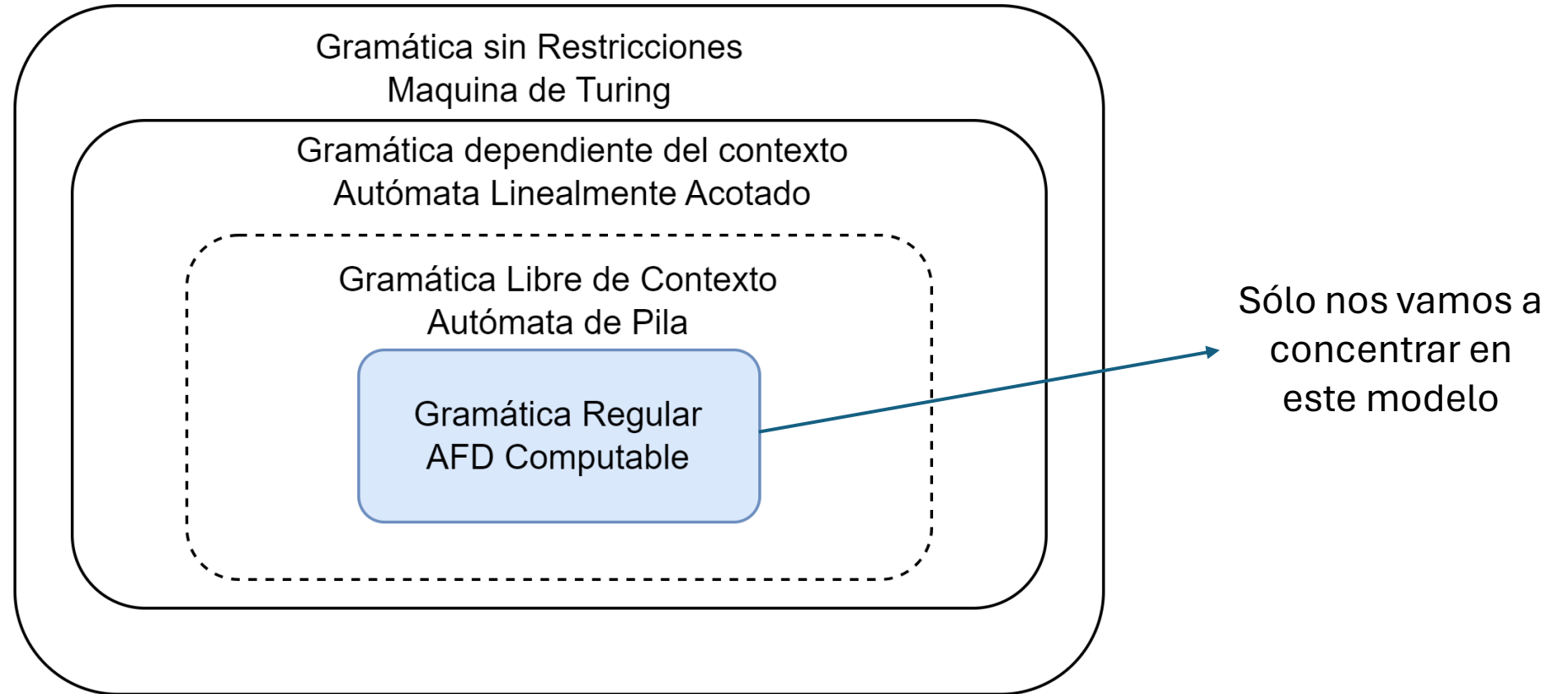
- Se demuestra que:

$$ER \Leftrightarrow AFND-\lambda \Leftrightarrow AFND \Leftrightarrow AFD \Leftrightarrow GR$$

Ejemplo de que no puedo hacer con un AFD

- $L = \{a^n b^n\} \quad n \in \mathbb{N}$, es una expression regular?
- Intuición: un AFD no puede contar a no ser que la cuenta esté acotada.
- Nota: Para este tipo de problemas se utilizan los autómatas de pila.

Jerarquía de Chomsky



Clase de Funciones Computables por AFD

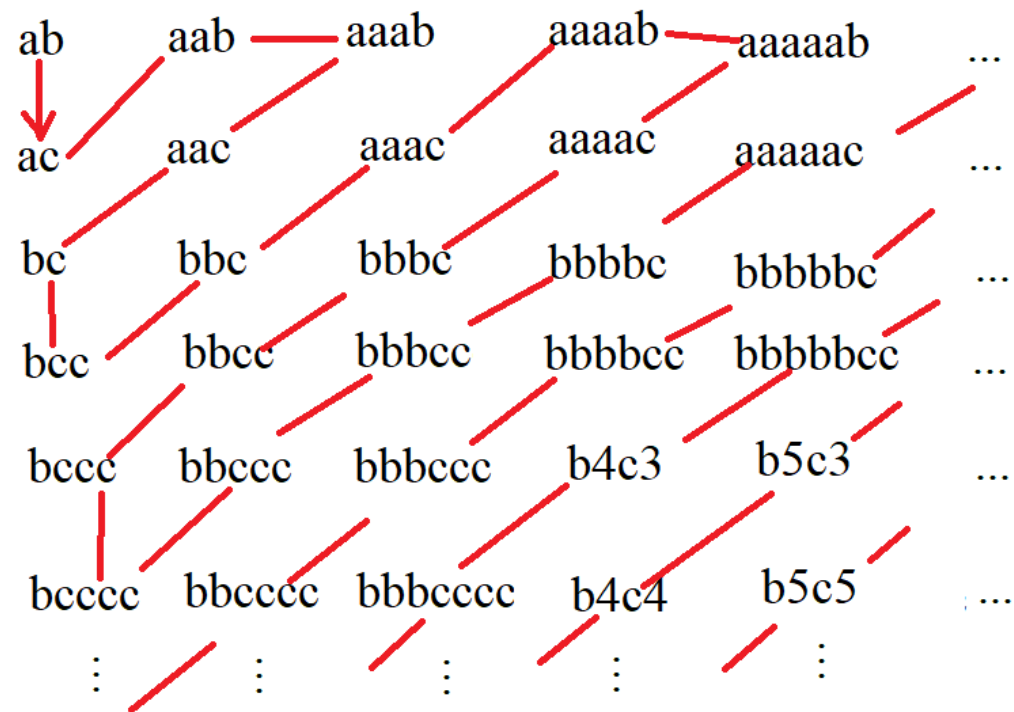
- Un traductor básicamente toma un conjunto finito de regular expressions y lo convierte en un número natural

- Ejemplo: $\Sigma = \{a, b, c, \}$

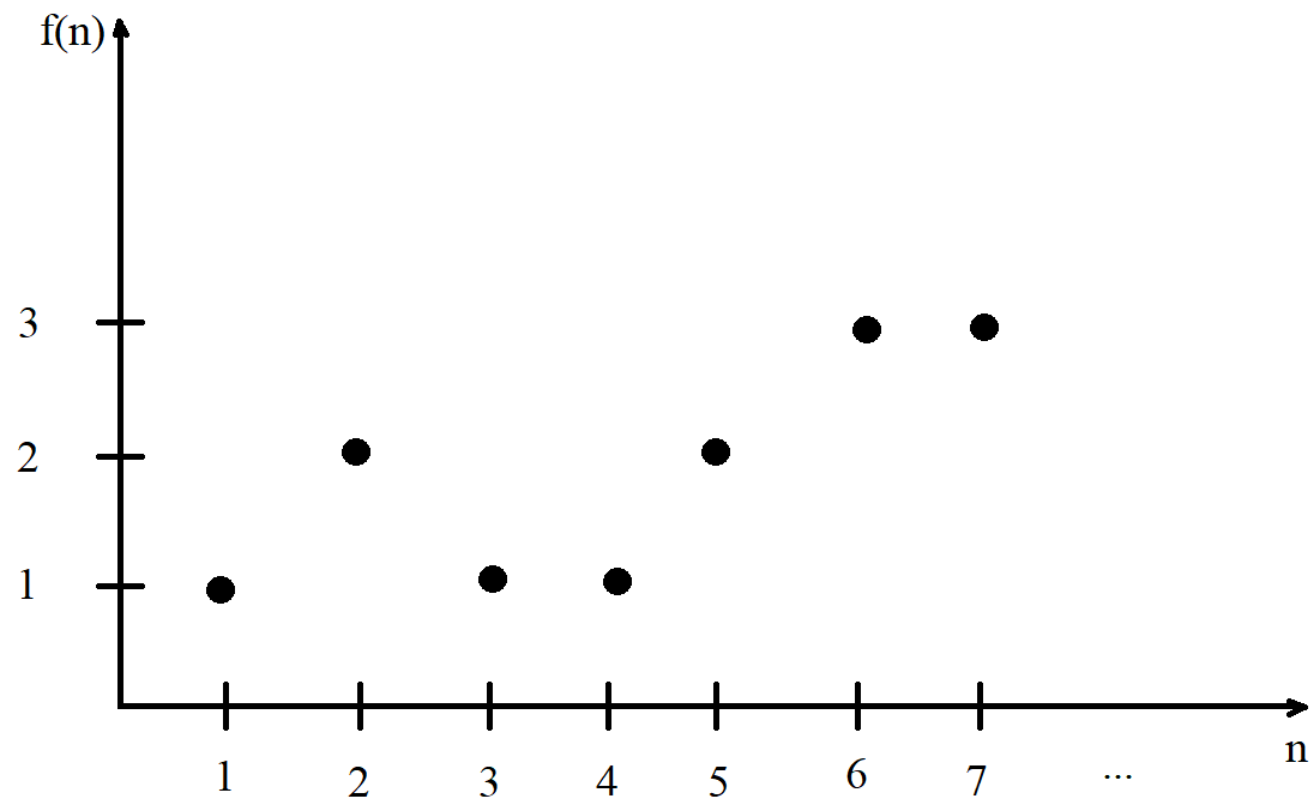
$$L = \{a^n b, a^n c, b^n c^m\}$$

L	Y
$a^n b$	1
$a^n c$	2
$b^n c^m$	3

- Quiero computar $f(X) \rightarrow Y$ donde $X = L^*$ (todos los strings que representa el language regular L)
- Observemos que X no es finito pero $\#X = \#N$



X	N	f(n)
ab	1	1
ac	2	2
aab	3	1
aaab	4	1
aac	5	2
bc	6	3
bcc	7	3
⋮	⋮	



Register Transfer Level

- Registro: FSM más “chica” posible
- Definimos una FSM como la 6-tupla: $\langle A_x, A_y, A_S, \delta, \omega, S_0 \rangle$
- Donde $S_0 \in A_S$, siendo el estado inicial de la máquina

$$A_x = \{0, 1\}$$

$$A_y = \{0, 1\}$$

$$A_S = \{0, 1\}$$

$$S_0 = \{0, 1\}$$

$$d = id$$

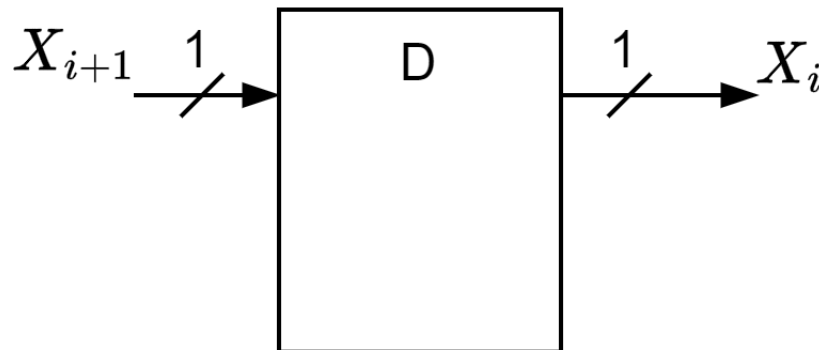
$$\omega = id$$

$$\delta(0, S_i) = 0$$

$$\delta(1, S_i) = 1$$

$$\left. \begin{array}{l} \omega(0) = 0 \\ \omega(1) = 1 \end{array} \right\} \text{ Moore}$$

$$\left. \begin{array}{l} \omega(0, x_i) = 0 \\ \omega(1, x_i) = 1 \end{array} \right\} \text{ Mealy}$$

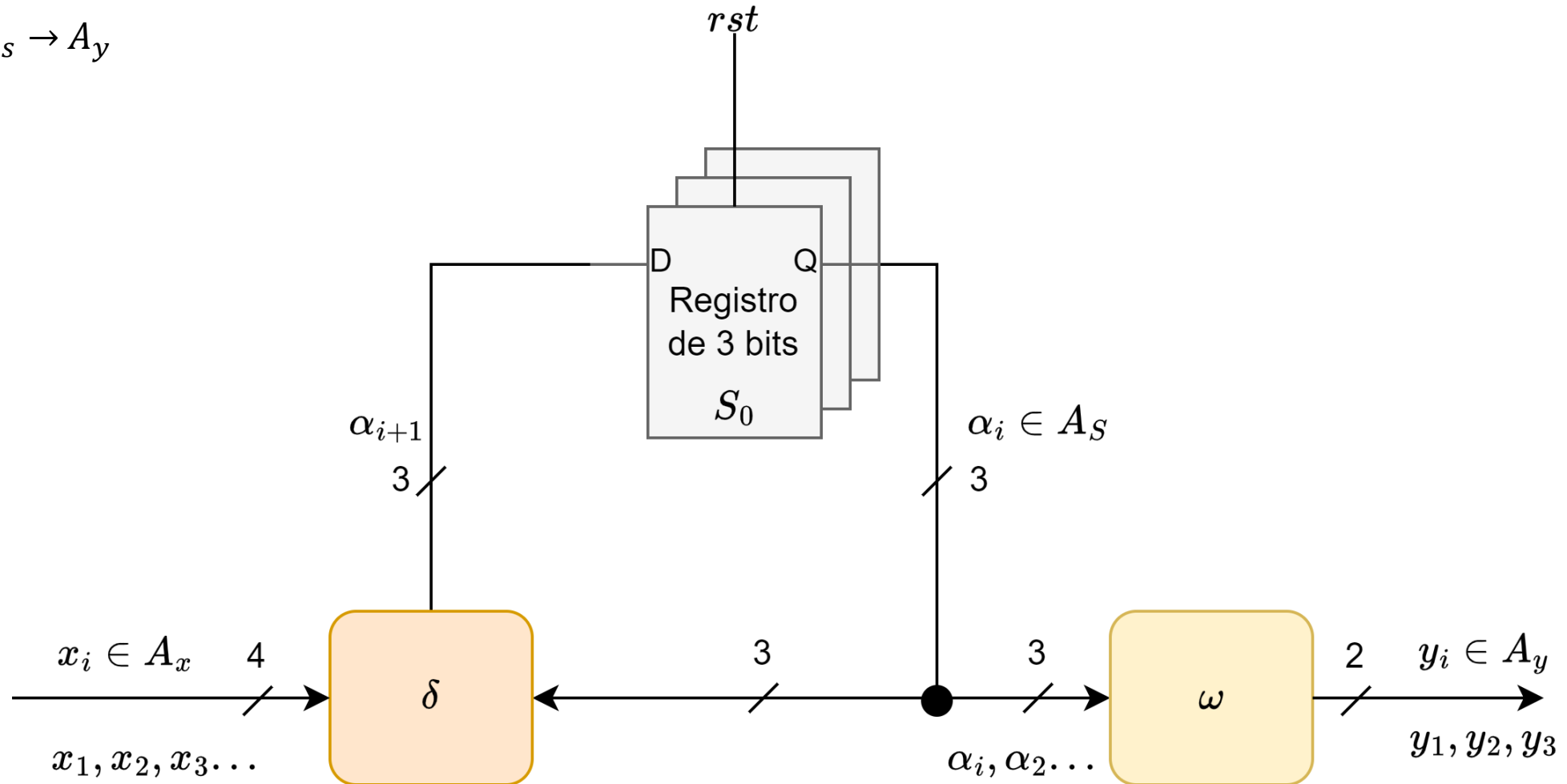


- Definición:

RTL es la especificación del FSM por medio de las expresiones de δ y ω cuando los estados del FSM se representan con registros.

RTL: Moore

$$\omega: A_S \rightarrow A_y$$



- $\delta: A_X \times A_S \rightarrow A_S$, $\delta(x_i, s_i) = s_{i+1}$
- $\omega: A_S \rightarrow A_Y$, $\omega(s_i) = y_{i+1}$
- $A_X = \{X_0, X_1, X_2, \dots, X_N\}$ es finito, entonces siempre se puede codificar en binario de forma arbitraria con $B_x = \lceil \log_2(N) \rceil$ bits.
- $A_S = \{S_0, S_1, S_2, \dots, S_M\}$ es finito, entonces siempre se puede codificar en binario con $B_S = \lceil \log_2(M) \rceil$ bits.
- $A_Y = \{Y_0, Y_1, Y_2, \dots, Y_K\}$ es finito, entonces siempre se puede codificar en binario con $B_Y = \lceil \log_2(K) \rceil$ bits.

- Se obtienen entonces las tablas de verdad de δ y ω

$\delta(x_i, s_i) = s_{i+1}$											
x_i				s_i				s_{i+1}			
$x_i[B_X - 1]$...	$x_i[1]$	$x_i[0]$	$s_i[B_S - 1]$...	$s_i[1]$	$s_i[0]$	$s_{i+1}[B_S - 1]$...	$s_{i+1}[1]$	$s_{i+1}[0]$

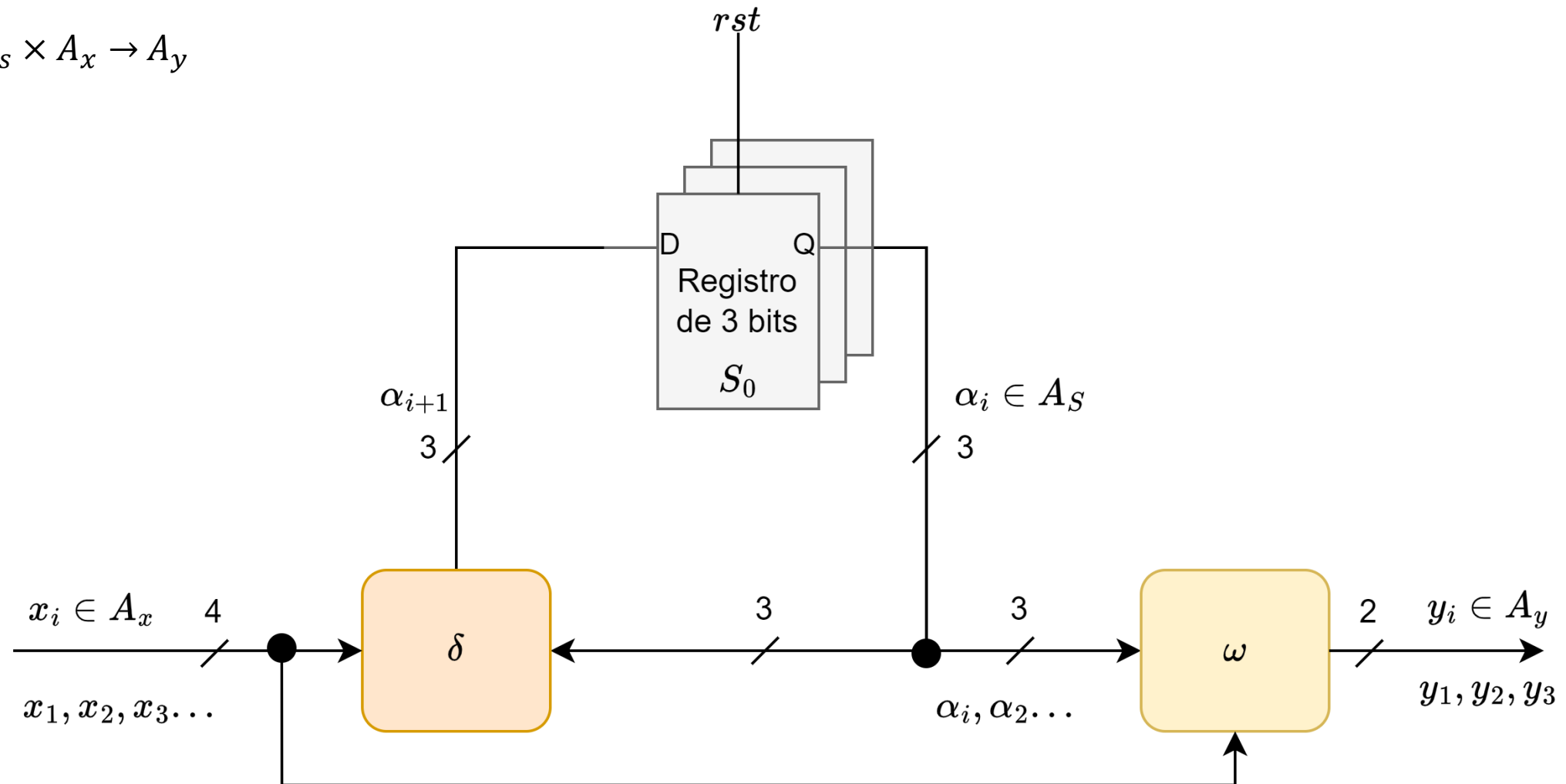
$B_X + B_S$ bits de entrada
 B_S bits de entrada

$y_i = \omega(s_i)$							
s_i				y_i			
$s_i[B_S - 1]$...	$s_i[1]$	$s_i[0]$	$y_i[B_Y - 1]$...	$y_i[1]$	$y_i[0]$

B_S bits de entrada
 B_Y bits de entrada

RTL: Mealy

$$\omega: A_S \times A_x \rightarrow A_y$$

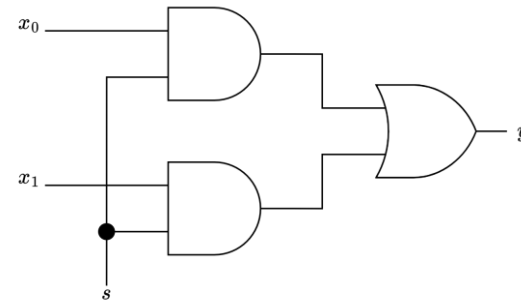
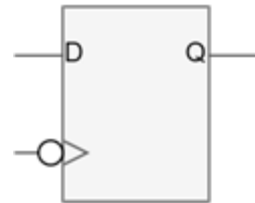
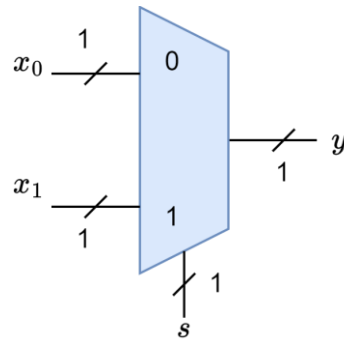


TEOREMA

- Sea una FSM que implementa un transductor sobre un lenguaje regular L implementado con un modelo de Moore, entonces existe al menos un transductor equivalente implementado con un modelo de Mealy y viceversa.
- Electrónicamente vamos a preferir el modelo de Moore.

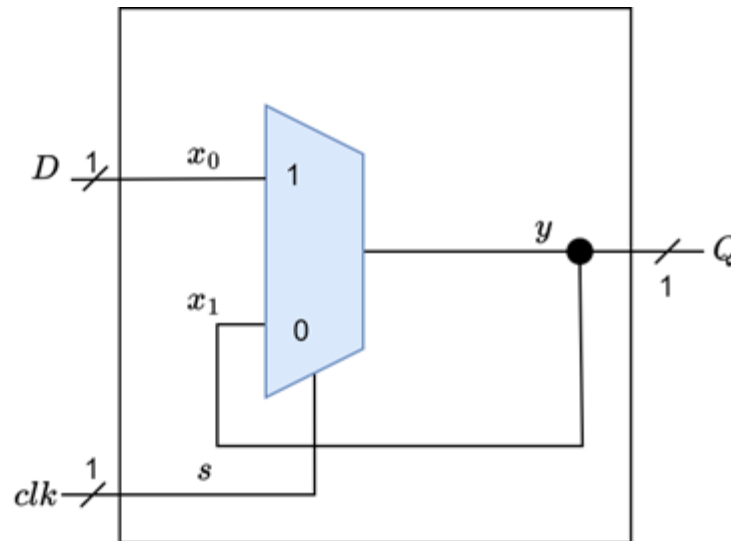
RTL: Implementación

Cómo implementamos un registro?

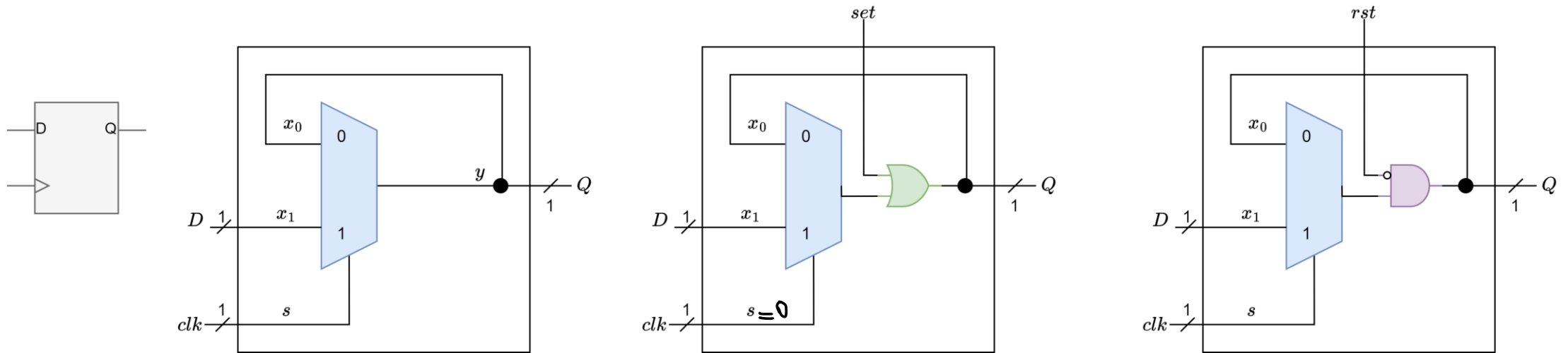


$$y = x_0 \bar{s} + x_1 s$$
$$Q_{i+1} = Q_i \bar{clk} + D_i clk$$

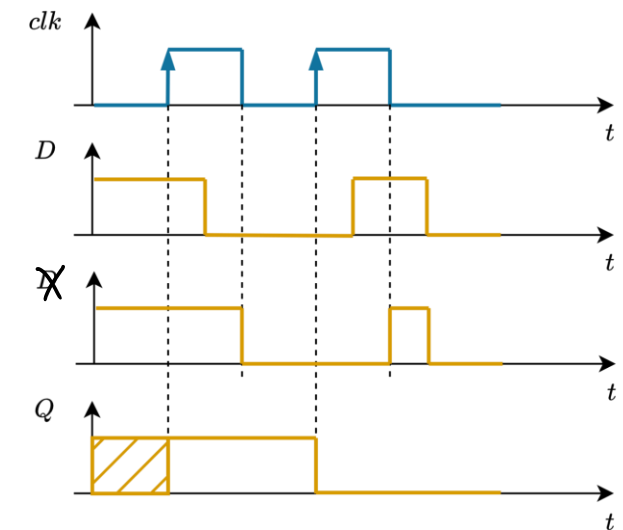
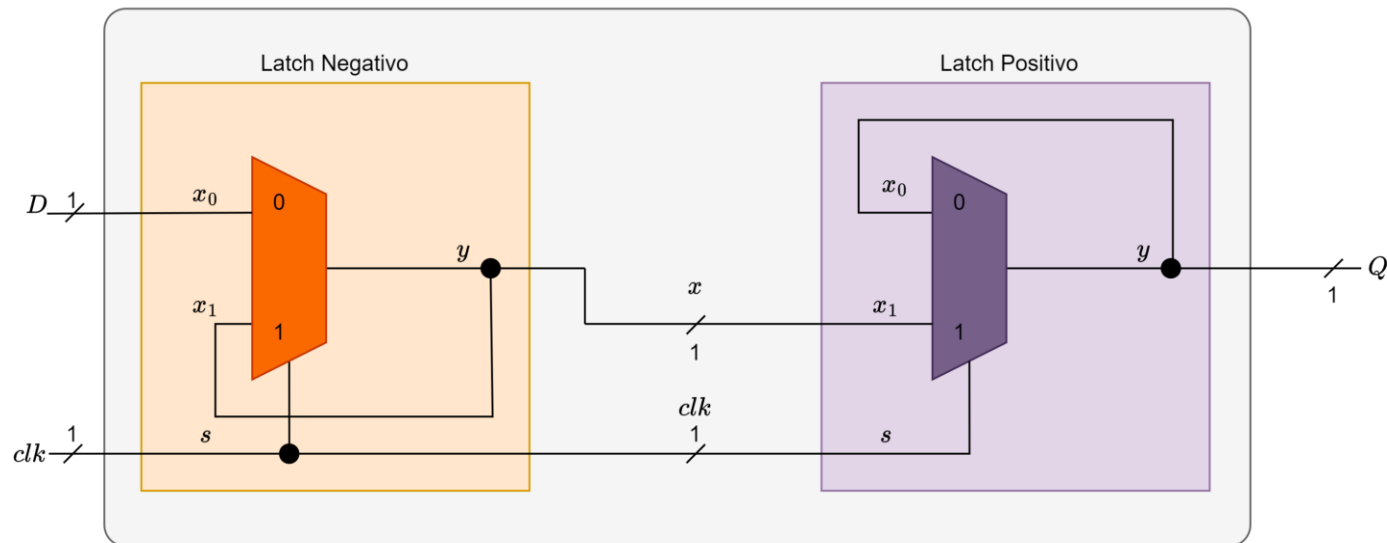
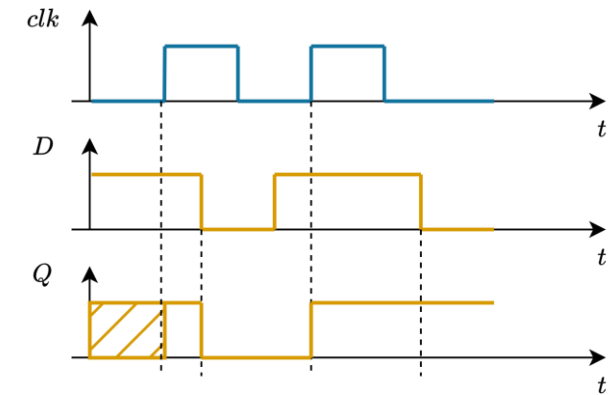
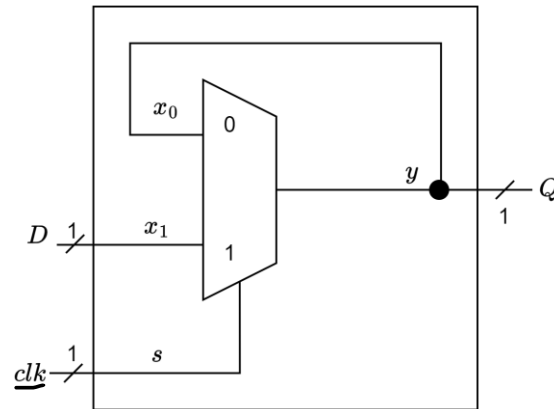
Diagram showing the implementation of the register equation $Q_{i+1} = Q_i \bar{clk} + D_i clk$ using the MUX logic. The equation is shown in a box, and the variables are mapped to the MUX inputs: Q_i is mapped to x_0 , \bar{clk} is mapped to \bar{s} , D_i is mapped to x_1 , and clk is mapped to s . The output y is mapped to Q_{i+1} .



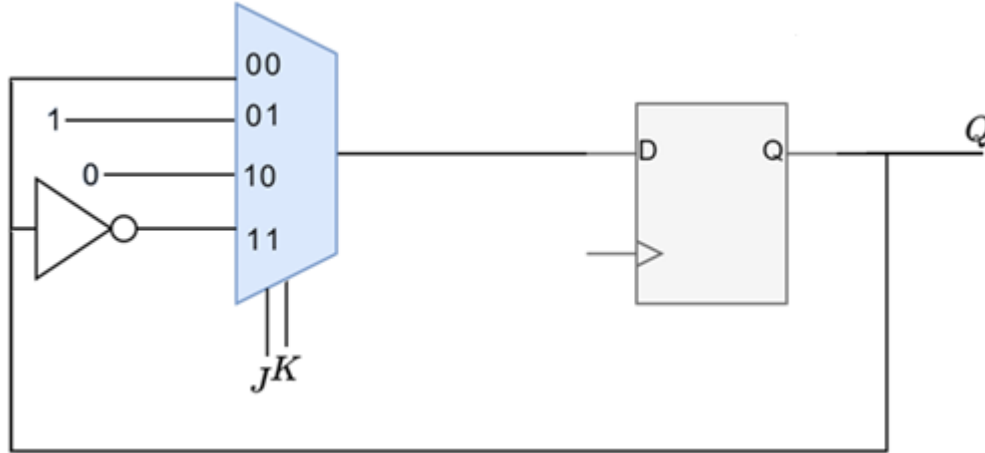
RTL: Implementación



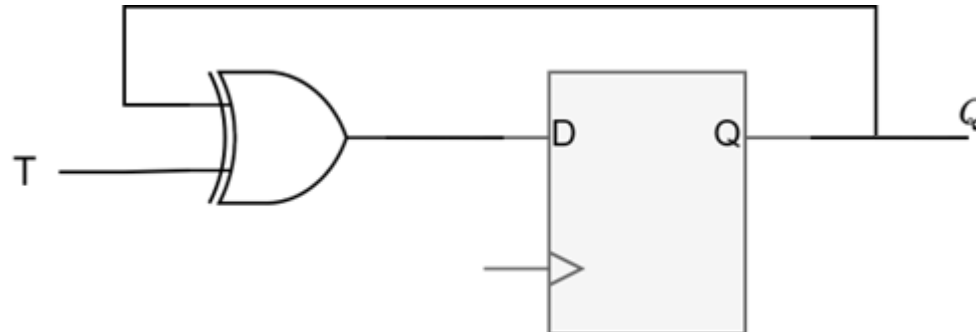
RTL: Implementación



Existe otro registro que no sea el flip-flop D?



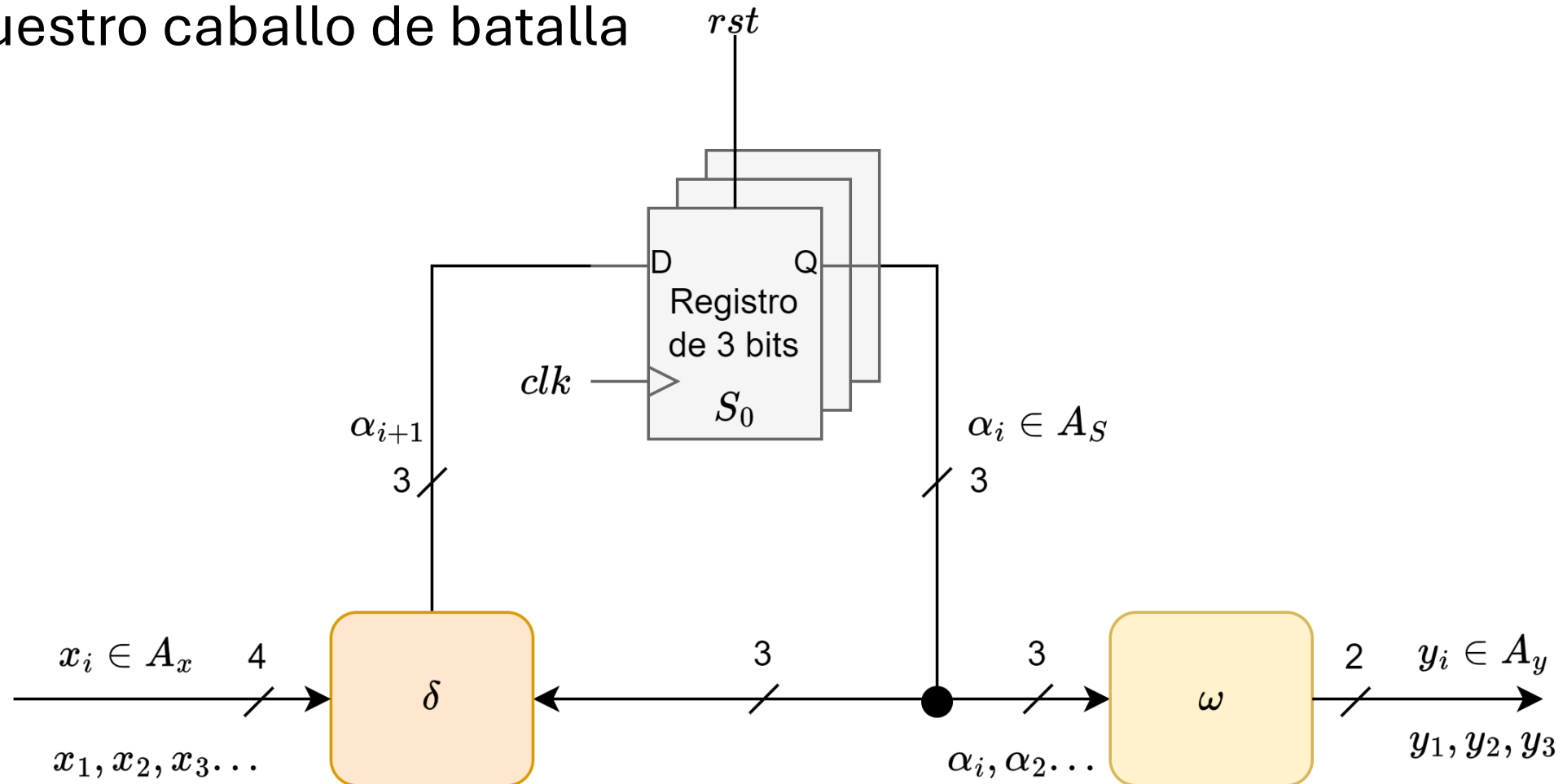
J	K	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	$\overline{Q_n}$



T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

Circuito de Moore sincrónico

- Nuestro caballo de batalla

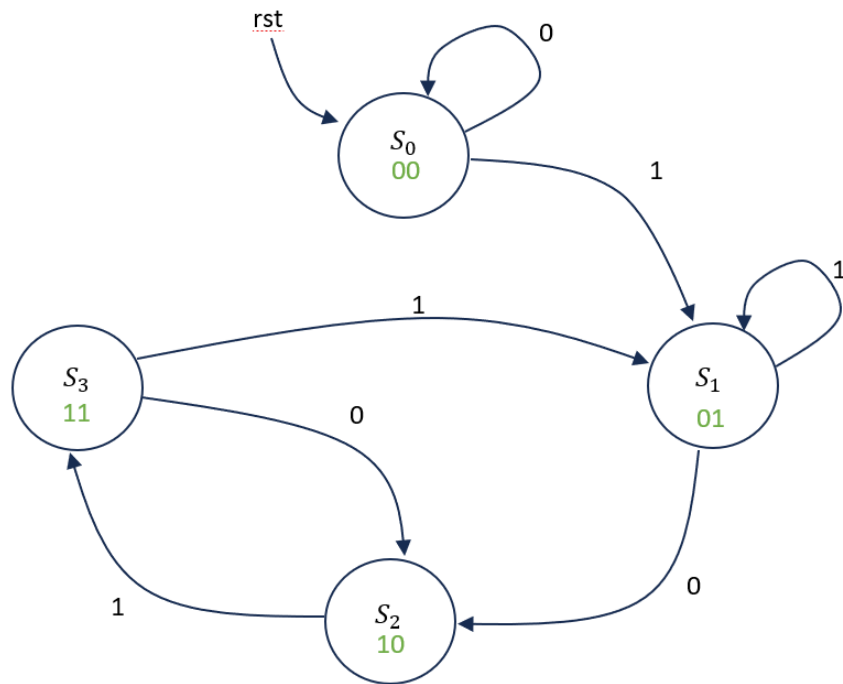


Condición fundamental del sincronismo

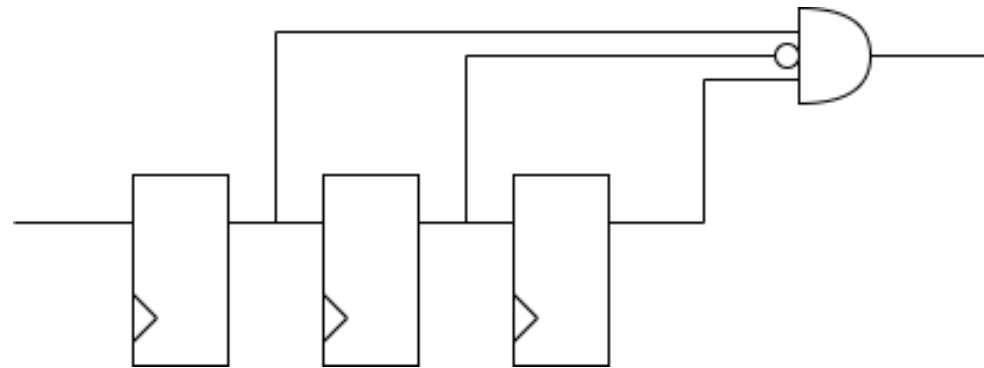
- Todos los registros operan con el mismo flanco de reloj (subida o bajada, es arbitrario pero uno solo)
- Todos los registros operan con el mismo estado de reset (alto o bajo, es arbitrario pero uno solo).

Indistinguibilidad

- Quiero detector esta secuencia: 101 en una entrada de 1 bit



4 estados \Rightarrow 2 registros



3 registros \Rightarrow 8 estados

Conclusiones



Ejercicio para entregar de función computada por un AFD

- De un texto del idioma castellano de máximo 1.000 caracteres, se quiere reconocer cuántas veces aparece la palabra “casa”.
- Diseñe el AFD (cuántos registros de 1 bit se necesita, y halle la función de traducción $f: N \rightarrow N$ que calcula el traductor, halle las funciones ω y δ).
- TIP: se considerará que solo pueden aparecer los siguientes símbolos en el texto: a, b, c, d, e, f, g, h, i, j, k, l, m, n, ñ, o, p, q, r, s, t, u, v, w, x, y, z, ‘ ‘ (espacio en blanco para separar). Es decir no hay signos de puntuación ni upper cases.