Problemas, Autómatas y Lenguajes

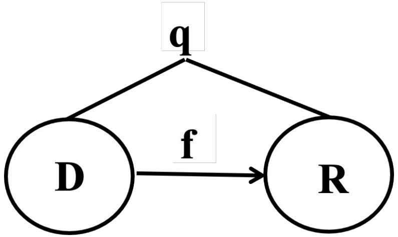
Emanuel Alvaredo, Federico Lochbaum

- Los problemas computacionales (matemáticos) y los lenguajes son herramientas que modelan problemas de la realidad

- Los autómatas son instrumentos para formular soluciones (algorítmicas) a problemas, que se reducen a reconocer lenguajes

Teoría de Problemas matemáticos y computacionales

Definimos un problema P como **P = <D, R, q>**



D = Conjunto de datos

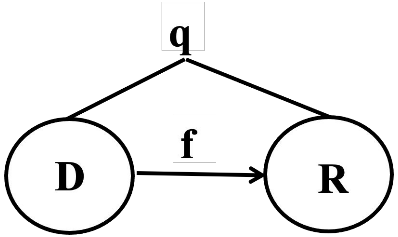
R = Conjunto de resultados

Donde D y R son subconjuntos de U (un universo) y q es una relación binaria del tipo D × R

Teoría de Problemas matemáticos y computacionales

- Un problema P es soluble si, $\forall d \in D, \exists r \in R$ tal que $(d, r) \in q$

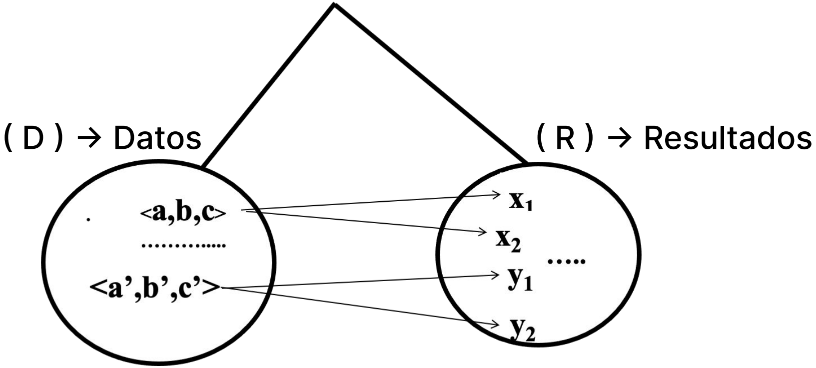
- Una solución a P es cualquier función $f \subseteq q$ definida en D



Ecuaciones de segundo grado

(f) → x1,2 = (-b +- √(b2 – 4ac))/2a

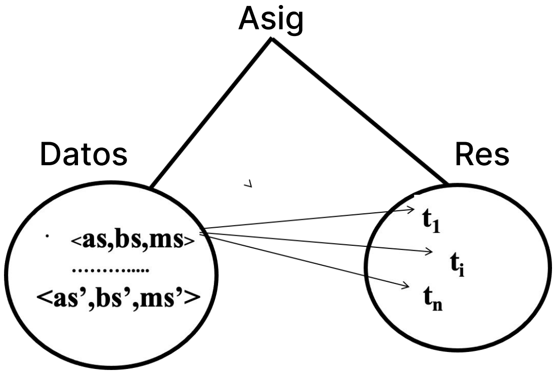
(q) → ax2 + bx + c = 0



cuad (q) = {<<a,b,c>, $x_{1,2}$ > | $x_{1,2}$ = (-b +- √ (b2 – 4ac)) / 2}

Notar que resolver este problema es equivalente a poder decidir si para cualquier par **<<r,s,t>, x> ∈ cuad**

Problema de asignación de aulas y horarios



Sea $A_1,\ldots,A_n$: Aulas; $B_1,\ldots.,B_m$: Bandas horarias y $M_1,\ldots.,M_p$: Materias

Restricciones

- R_1: Cada materia tiene que tener asignada dos bandas horarias
- R_2: Cada par (A_i, B_j) tiene asignada a lo sumo una materia M_k.
- R_3: Cada par (B_j, M_k) tiene asignada a lo sumo un aula A_i

Datos = Conjunto de triplas { (*aulas*, *bandas*, *materias*) }

Donde *aulas* es una lista de aulas, *bandas* es una lista de bandas horarias, *materias* es una lista de materias.

Resultados = { tabla | tabla: aula x banda → materia }

Relación: asig ={ ((aulas, bandas, materias), tabla): R_1 R_2 R_3 }

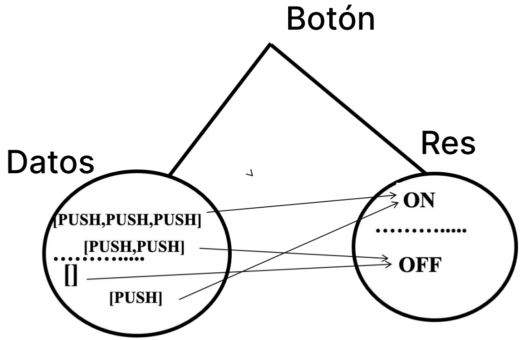
Problema del botón

Dado un dispositivo que puede estar encendido o apagado (por ejemplo, un velador, un teléfono celular, una central atómica) tenemos un botón que sirve para encenderlo o apagarlo

Una cuestión que puede ser importante es poder determinar si el botón está encendido o apagado

Problema del Botón: Dada una secuencia de operaciones *PUSH*, determinar si el botón está *ON* o está *OFF*

Botón = { <ps, x> | x = ON o bien x = OFF}



Problema del botón

Si lo definimos inductivamente

<[], *OFF*> ∈ **botón**

<PUSH : ps, *ON*> ∈ **botón** O <ps, *OFF*> ∈ **botón**

<PUSH : ps, *OFF*> ∈ **botón** O <ps, *ON*> ∈ **botón**

Vemos que

<[PUSH, PUSH, PUSH], *ON*> ∈ **botón** ⇒

<[PUSH, PUSH], *OFF*> ∈ **botón** ⇒

<[PUSH], *ON*> ∈ **botón** ⇒

< [], *OFF*> ∈ **botón**

Una posible solución

$$\mathrm{OFF} = \mathit{false} \;\; \text{y} \;\; \mathrm{ON} = \mathit{true} \qquad \beta : [PUSH] \rightarrow Boolean$$

$$\beta\,\mathrm{ps} = (|\,\mathrm{ps}\,| \bmod 2 = 1)$$

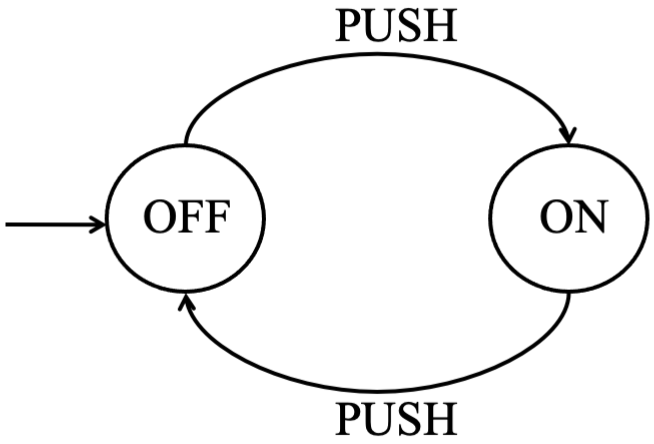
$$\mathrm{boton} = \{ < \mathrm{ps}, \mathrm{OFF} > \,|\, |\,\mathrm{ps}\,| \text{ es par} \} \cup \{ < \mathrm{ps}, \mathrm{ON} > \,|\, |\,\mathrm{ps}\,| \text{ es impar} \}$$

Notar que no es necesario guardar la historia completa de *PUSHs* para saber si está en *ON* u *OFF*

- *ON* es la **clase de equivalencia** de todas las listas de PUSH de longitud impar
- *OFF* es la **clase de equivalencia** de todas las listas de PUSH de longitud par

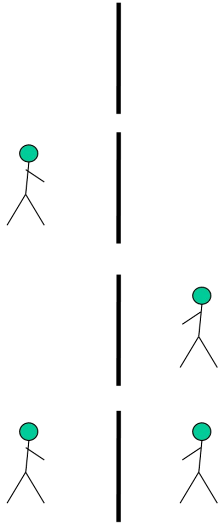
Modelado como AFD

- Estados
  - OFF (estado inicial)
  - ON
- Transiciones de estados (flechas etiquetadas)
- Eventos (etiquetas o símbolos: PUSH)

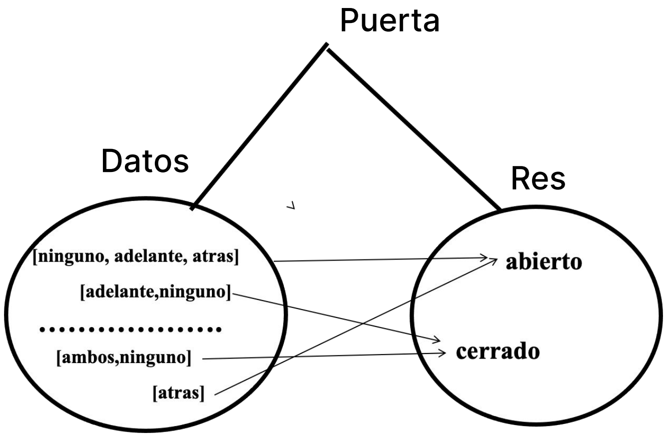


Puerta automática

El sistema control de un puerta corrediza automática



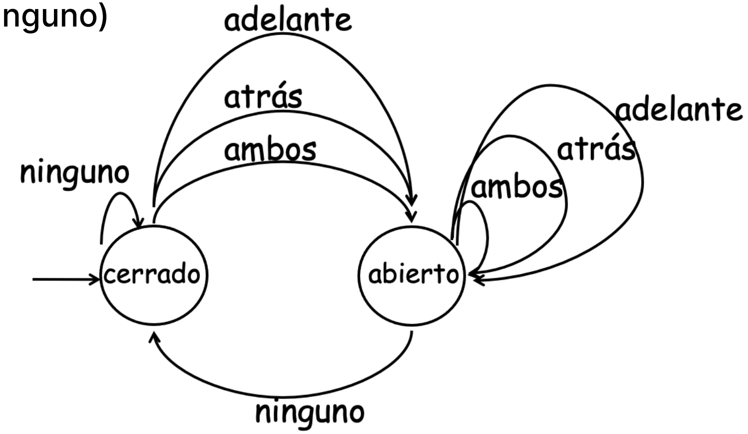
EVENTO	ESTADO
ninguno	cerrado
adelante	abierto
atrás	abierto
ambos	abierto



Puerta automática

Puede modelarse con una autómata finito

- Estados
  - cerrado (estado inicial)
  - abierto
- Transiciones de estados (flechas etiquetadas)
- Eventos (etiquetas o símbolos: adelante, atrás, ambos, ninguno)



- Vemos que hay muchos sistemas que pasan por un número finito de estados (Botón, Semáforo, Velador, Lavarropas, …)

- El propósito de un estado es **recordar** la parte relevante del historial del sistema (pushes en el botón, por ejemplo)

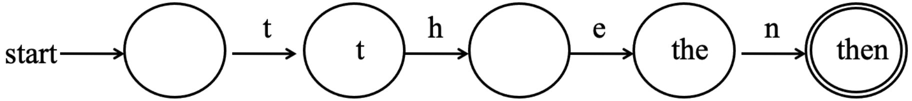
- Como hay un número finito de estados, debemos recordar lo que es **importante** y olvidar el resto

- Hay que buscar una abstracción de lo que no es **importante**

La ventaja de tener un **número finito de estados** es que podemos implementar el sistema con un **número finito de recursos**

Por ejemplo, podríamos implementar el sistema en hardware como **un circuito** o como una forma simple de programa que puede tomar decisiones mirando **solo a una limitada cantidad de datos**

Entonces qué es y qué hace un autómata finito ?



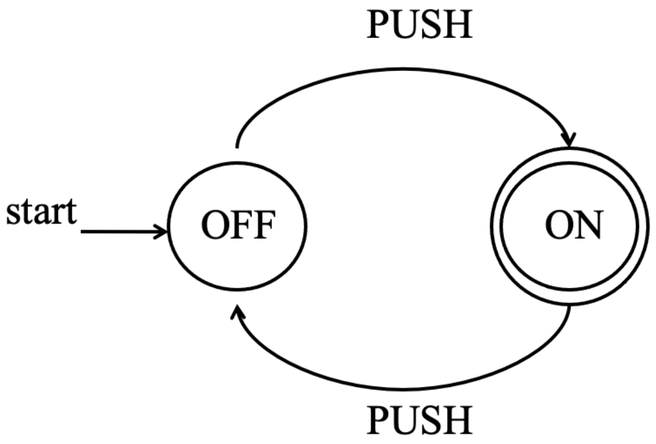
- Estados (estado inicial señalizado con *start*; *t, th, the, then*, estado final *then*)

- Transiciones de estados (flechas etiquetadas)

- Eventos (etiquetas o símbolos: t,h,e,n)

Entonces qué es y qué hace un autómata finito ?

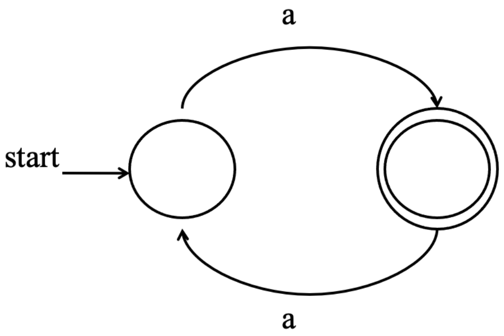
- Estados (estado inicial OFF, estado final ON)
- Transiciones de estados (flechas etiquetadas)
- Eventos (etiquetas o símbolos: PUSH)



Vemos que el autómata reconoce el lenguaje sobre el alfabeto { PUSH } formado por secuencias de longitud impar.

Podemos formular el problema del botón como un lenguaje sobre el alfabeto { PUSH }. En verdad, sobre un alfabeto de un único símbolo

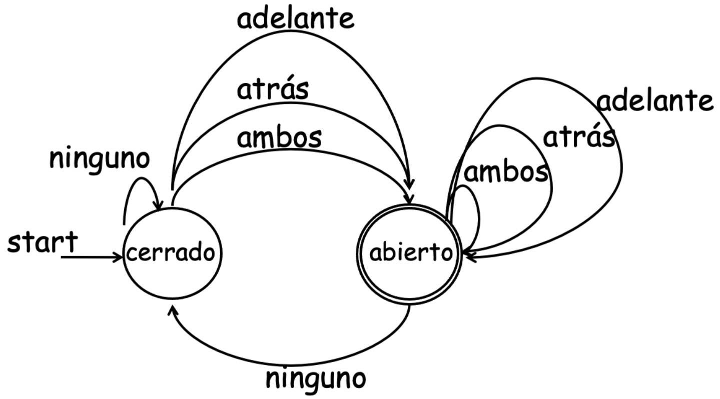
*“Si podemos **reconocer** un lenguaje formado por secuencias de longitud impar sobre un alfabeto de un solo símbolo, { w | |w| es impar} ENTONCES sabemos **resolver** el problema del botón”.*



En general, podemos intercambiar los conceptos de problema y lenguaje

Puerta automática como Autómata finito reconocedor

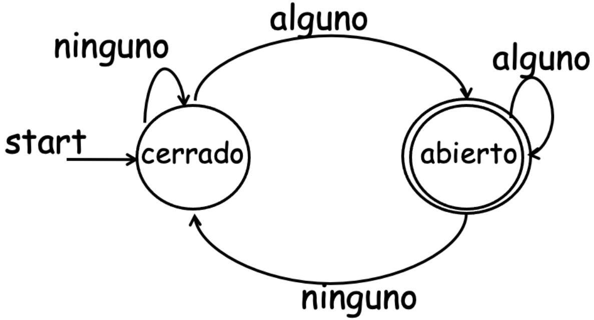
- Estados: estado inicial *cerrado*, estado final *abierto*
- Eventos: etiquetas o símbolos: *adelante, atrás, ambos, ninguno*



Puerta automática como Autómata finito reconocedor

El problema del sistema de control de la puerta puede expresarse de manera completamente abstracta como el problema de poder reconocer un lenguaje sobre un alfabeto de dos símbolos *ninguno, alguno.*

$$L = (\{ninguno\}^{*}\ \{alguno\}^{+})^{+} = \{\ (n^{i}a^{j})^{k}\ |\ i \geq 0,\ j,k \geq 1\ \}$$



Entonces, vemos que...

Los lenguajes y los problemas son realmente la misma cosa.

- Cuando solo nos interesan los strings por sí mismos, por ejemplo, en el conjunto $\{0^n1^n \mid n \geq 0\}$, entonces tendemos a pensar en el conjunto de strings como un **lenguaje**. Otras veces, tendemos a asignar semántica a las cadenas, por ejemplo, piense en los strings como codificaciones de grafos, expresiones lógicas o incluso números enteros.

- En aquellos casos en los que nos preocupamos más por lo representado por el string que por el propio string, tendemos a pensar en un conjunto de strings como un **problema**.

Automatas y Lenguajes

Sea cual sea el **problema** (computable), podemos entonces definirlo como un **lenguaje** y cualquier mecanismo (como por ejemplo, un **autómata** finito) que sea capaz de aceptar y procesar las cadenas de dicho lenguaje, es entonces una **solución** a dicho **problema**

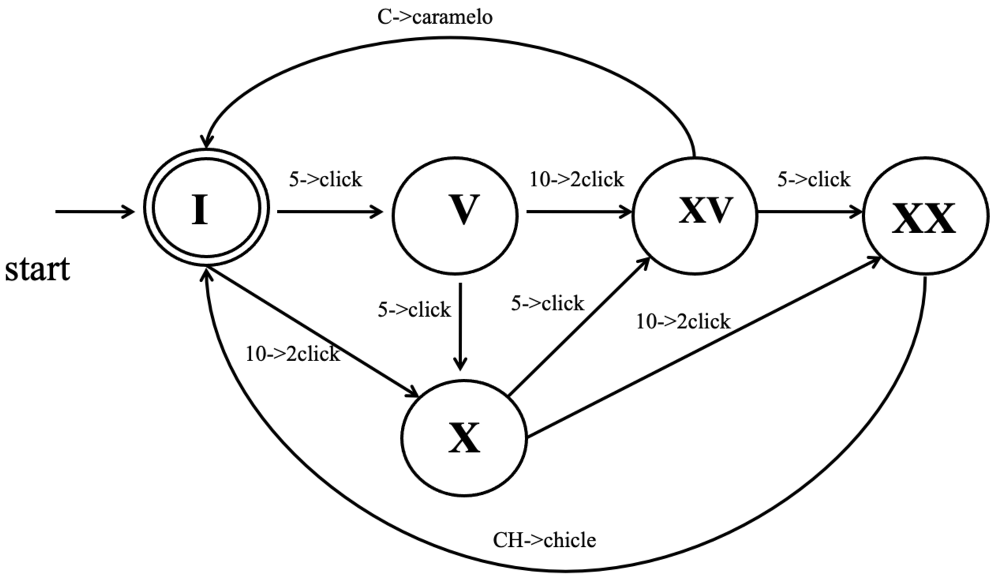
De esto derivamos que

- Existe una equivalencia entre los conceptos de problemas y lenguajes
- Un autómata que reconoce un lenguaje es una solución a problema equivalente a dicho lenguaje

Último ejemplo

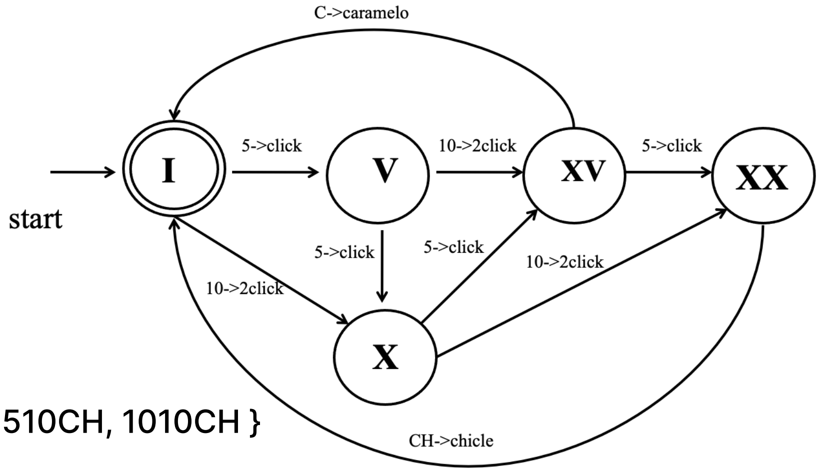
Sea el siguiente AFD M que para una máquina expendedora

"Una persona inserta monedas 5 y 10 centavos en la ranura hasta llegar a 15 o 20 centavos; en tales casos, aprieta el botón C o el botón CH en cuyos casos la máquina deja un caramelo o un chicle en la bandeja. Y sigue funcionando del mismo modo. La máquina no da vuelto. Por otro lado, cada vez que se inserta una moneda de 5 centavos por el parlante se escucha click y cuando se inserta una moneda de 10 ctvs se escucha click-click"



Qué lenguaje reconoce este autómata ?

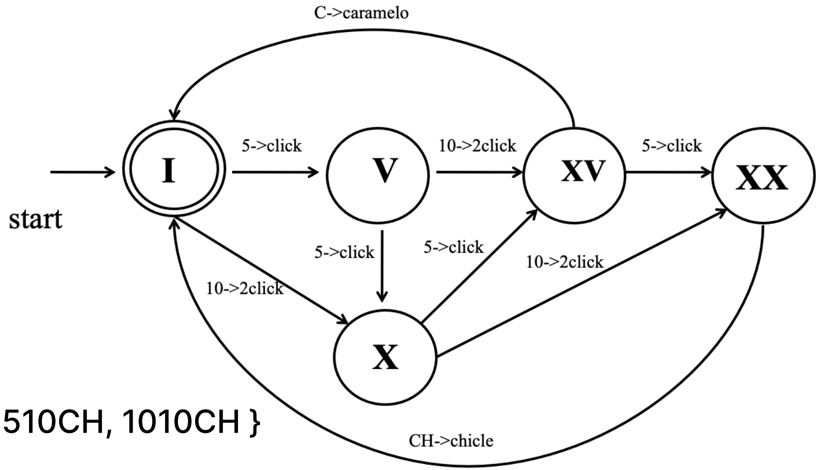
Último ejemplo



Notamos que

- $\Sigma = \{ 5, 10, C, CH \}$
- $L_0 = \{ \lambda, 510C, 555C, 105C, 5105CH, 5555CH, 1055CH, 5510CH, 1010CH \}$
- $L(M) = ?$

Último ejemplo



Notamos que

- $\Sigma = \{ 5, 10, C, CH \}$
- $L_0 = \{ \lambda, 510C, 555C, 105C, 5105CH, 5555CH, 1055CH, 5510CH, 1010CH \}$
- $L(M) = L_0^*$