Universidad de Huelva

Tema 3

Autómatas finitos

- La codificación de conjuntos infinitos numerables requiere un secuencia ilimitada de dígitos.
  - Por ejemplo, {0,1} en codificación binaria, [0-9] en codificación decimal, …

- Para modelar funciones es necesario leer la codificación del valor de entrada, que puede tener una longitud ilimitada. Esto no es posible con un circuito combinacional. Es necesario un modelo que permita leer secuencialmente la codificación de entrada.

- Supongamos que deseamos desarrollar la función suma en notación decimal.

$$f(a,b) = a + b$$

- Podemos representar las entradas como una lista de dígitos

1	0	7	8	5
2	3	0	2	6

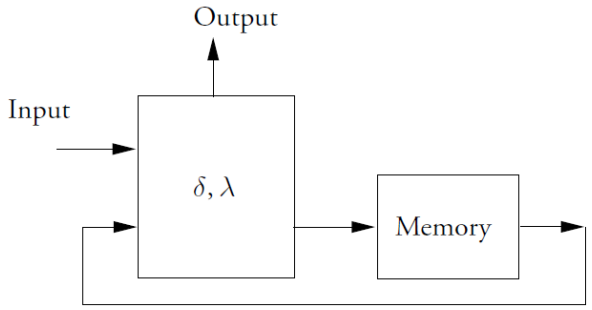
- El sistema debe ir leyendo las entradas para calcular la suma

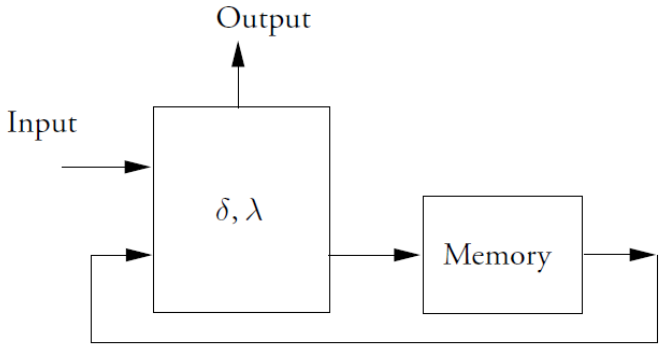
1	0	7	8	5
2	3	0	2	6
				⇓
				1

- Para calcular la siguiente cifra es necesario tener en cuenta la "llevada".

1	0	7	8	5
2	3	0	2	6
			⇓	
			1	1

- Para esto es necesario que el sistema disponga de una memoria en la que almacenar el estado en el que se encuentra el proceso de ejecución.



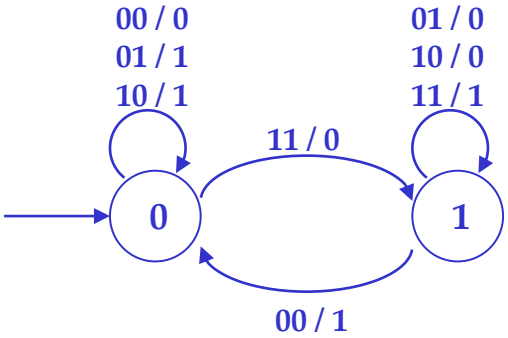


- La función λ(i,s) representa la salida a generar en función de la entrada *i* y del estado *s* (almacenado en la memoria) en el que se encuentre el sistema.
- La función δ(i,s) representa el estado a almacenar en la memoria después de cada iteración.

- Formalmente, se define un ***autómata finito*** (***finite state machine***) como una septupla $M = (\Sigma,\Psi,Q, \delta, \lambda,s,F)$, donde Σ es el alfabeto de entrada, Ψ es el alfabeto de salida, Q es el conjunto de estados del autómata, $\delta:Q\times\Sigma\rightarrow Q$ es la función de transición de estado, $\lambda :Q\times\Sigma\rightarrow\Psi$ es la función de salida, s es el estado inicial y $F\subseteq Q$ es el conjunto de estados finales del autómata.

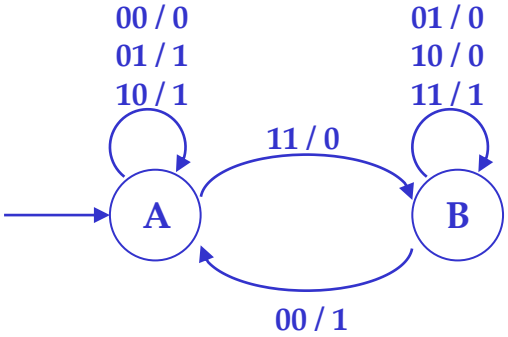
- En nuestro ejemplo,
  - $\Sigma = [0\text{-}9] \times [0\text{-}9]$ (dígitos de los números a sumar)
  - $\Psi = [0\text{-}9]$ (digitos resultado de la suma)
  - $Q = \{ q_0, q_1 \}$ (estados "me llevo 0" y "me llevo 1")
  - $\lambda = \{ [(0,0, q_0) \rightarrow 0]$, (resultados "0+0+0 es 0",
    $[(0,1, q_0) \rightarrow 1]$, "0+1+0 es 1",
    …, $[(9,9, q_1) \rightarrow 9]\}$ …, "9+9+1 es 9")
  - $\delta = \{ [(0,0, q_0) \rightarrow q_0]$, (transiciones "0+0+0 me llevo 0",
    $[(0,1, q_0) \rightarrow q_0]$, "0+1+0 me llevo 0",
    …, $[(9,9, q_1) \rightarrow q_1]\}$ …, "9+9+1 me llevo 1")
  - $s = q_0$ (estado inicial "me llevo 0")
  - $F = \{ q_0, q_1 \}$ (todos los estados son finales)

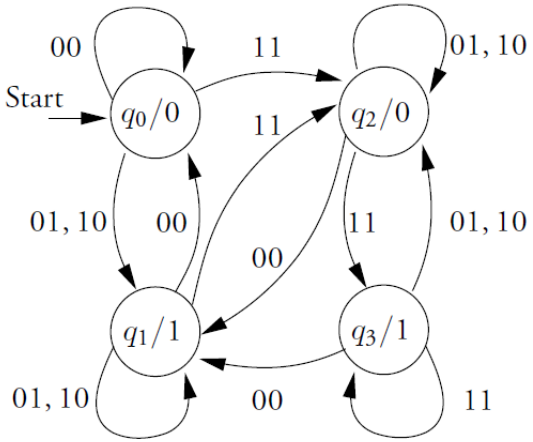
- Un ejemplo más sencillo: suma binaria



- El esquema anterior fue propuesto por G.H. Mealy en 1955 y se conoce como ***autómata de Mealy***.
- Un esquema muy parecido fue propuesto por E.F. Moore en 1956. En este modelo, conocido como ***autómata de Moore***, el valor de la salida no está asociado a la transición sino al estado de destino.
- Ambos tipos de autómatas tienen la misma capacidad de cómputo.

- Transformación de un autómata de Mealy en un autómata de Moore:
  - Cada estado del autómata de Mealy se transforma en varios estados en el autómata de Moore equivalente, uno por cada valor diferente que tengan sus transiciones de entrada.
  - Las transiciones de salida de cada estado de Moore se copian de las de su estado equivalente, colocando el destino de la transición en el estado adecuado.

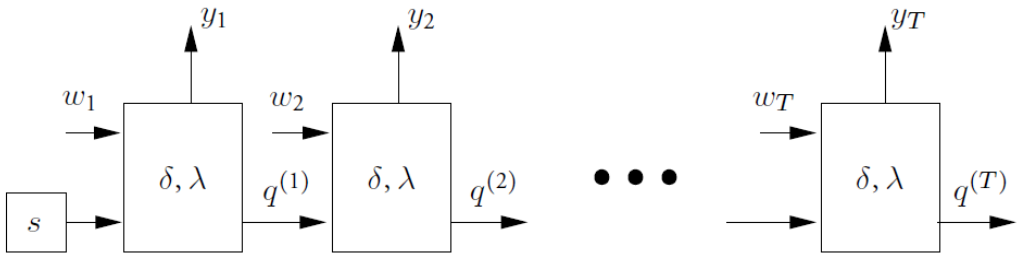




- Funciones computadas por autómatas finitos
  - Dado un estado inicial s y un conjunto de entradas externas $w_1, w_2, \ldots, w_T$, un autómata finito M genera un conjunto de salidas externas $y_1, y_2, \ldots, y_T$ y termina en un estado q^T.
  - Se dice que el autómata M computa en T pasos la función $f_M^{(T)}$

$$f_M^{(T)} : Q \times \Sigma^T \rightarrow Q \times \Psi^T$$

  - Si asumimos que Q, Σ y Ψ están codificados en binario, entonces la función $f_M^{(T)}$ es una función binaria.



- Funciones computadas por autómatas finitos
  - Cuando utilizamos un autómata en T pasos, normalmente no computa la función más general $f_M^{(T)}$, sino una subfunción de ella. Por ejemplo, se introduce que el estado inicial sea un estado concreto. También es frecuente desechar salidas (y quedarse con la última).

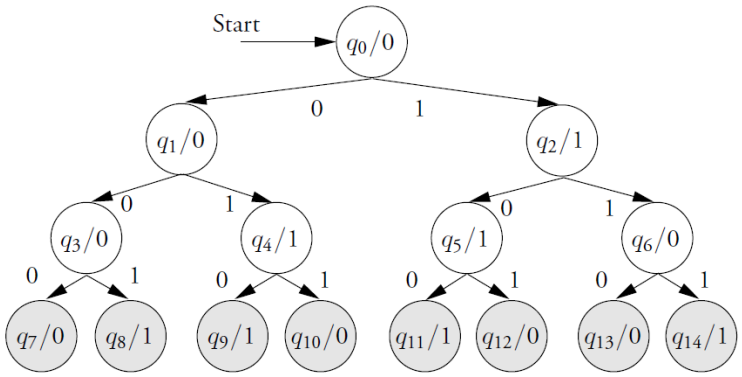
  - Sea $f_M^{(T)}$ la función computada por un autómata $M = (\Sigma,\Psi,Q, \delta, \lambda,s,F)$ en T pasos. El tamaño de circuito y la profundidad de circuito de cualquier función f computada por M en T pasos satisface las siguientes ecuaciones:

$$C_\Omega(f) \leq C_\Omega(f_M^{(T)}) \leq T \cdot C_\Omega(\delta, \lambda)$$

$$D_\Omega(f) \leq D_\Omega(f_M^{(T)}) \leq T \cdot D_\Omega(\delta, \lambda)$$

- Funciones computadas por autómatas finitos
  - Se denomina ***potencia*** (***power***) de un autómata finito M a $C_\Omega(\delta, \lambda)$, es decir, al número de operaciones lógicas que desarrolla M en cada paso.
  - Se denomina ***trabajo computacional*** desarrollado por un autómata finito M al valor $T \cdot C_\Omega(\delta, \lambda)$, es decir, al número de operaciones lógicas totales.
  - Corolario: Es imposible computar funciones f para las que su tamaño y profundiad sean mayores que los límites $T \cdot C_\Omega(\delta, \lambda)$ y $T \cdot D_\Omega(\delta, \lambda)$, respectivamente.

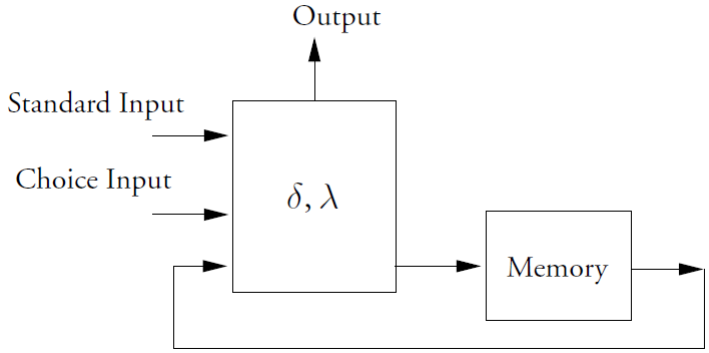
- Funciones computadas por autómatas finitos
  - Teorema: Toda función booleana de *n* entradas puede ser computada mediante un autómata finito de 2^{n+1}-1 estados.
  - Demostración: A partir del estado inicial se puede construir un árbol binario ramificando cada estado en función del valor 0 o 1 de la siguiente entrada, hasta llegar a la entrada n-ésima. La salida del autómata es la correspondiente a las hojas del árbol, desechando las salidas internas.



- Autómatas finitos no deterministas
  - Los autómatas finitos presentados anterioriormente son ***deterministas***, en el sentido de que, dado un estado y una entrada, el siguiente estado del autómata está unívocamente determinado.
  - Se denomina ***autómata finito no determinista*** a un autómata finito en el que, dado un estado y una entrada, existen varias transiciones posibles y, por tanto, varios siguientes estados posibles.
  - Formalmente, un autómata finito no determinista viene dado por una función de transición $\delta:Q\times\Sigma\rightarrow 2^Q$, es decir, el resultado de una transición no es un único estado de Q sino un subconjunto de estados de Q.
  - El resultado de una transición podría ser un subconjunto vacío, indicando que no existe ningún sucesor del estado origen para esa entrada.

- Autómatas finitos no deterministas
  - Los autómatas finitos no deterministas se utilizan como reconocedores de lenguajes.
  - Dada una cadena de entrada w, se considera aceptada por el autómata si existe un encadenamiento de transiciones que mueva el autómata desde el estado inicial a un estado final.
  - Los autómatas finitos no deterministas no pueden utilizarse como generadores de funciones ya que para una misma entrada el autómata podría generar salidas diferentes, lo que va en contra del concepto de función.

- Autómatas finitos no deterministas
  - Un autómata finito no determinsta se puede modelar como un autómata finito determinista con una entrada adicional (***choice input***) utilizada como selector para elegir la trayectoria correcta.
  - Se asume que el ***agente selector*** es capaz de suministrar los valores adecuados para esta entrada a partir del valor de la cadena.



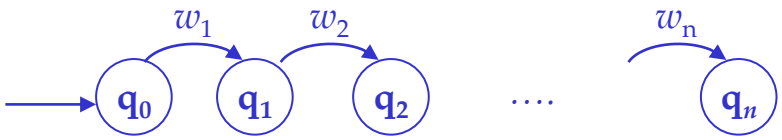
- Autómatas finitos no deterministas
  - Dado un autómata finito no determinista NFSM es posible construir un autómata finito determinista equivalente DFSM. Los estados de este autómata finito determinista DFSM se generan como representación de subconjuntos de estados del autómata finito no determinista. Por tanto, potencialmente el conjunto de estados del autómata finito determinista equivalente puede ser hasta 2^Q, siendo Q el conjunto de estados del NFSM.
  - El teorema anterior indica que la capacidad de cómputo de los autómatas deterministas y no deterministas es la misma. Sin embargo, este resultado está basado en que el conjunto de estados de ambos es finito.
  - En sistemas como las máquinas de Turing (con una capacidad de memoria infinita) no sabemos si el conjunto de lenguajes aceptados en un tiempo polinomial por una máquina determinista (clase P) es el mismo que el de los lenguajes aceptados en tiempo polinomial por una máquina indeterminista (clase NP).

- Lema de bombeo:
  - Sea L un lenguaje regular sobre el alfabeto Σ reconocido por un DFSM con m estados.
  - Si $w \in L$ y $|w| \geq m$, entonces existen las cadenas r, s y t con $|s| \geq 1$ y $|rs| \leq m$ tales que $w = rst$ y para todo $n \geq 0$, $rs^n t$ también pertenece a L.

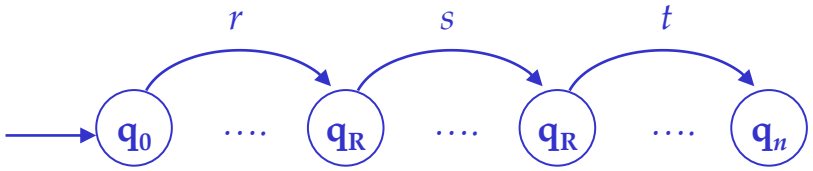
- En término de funciones, la función reconocida por un DFSM de m estados f: $\Sigma^* \rightarrow \Psi^*$ cumple que,
  - $\forall w \in \Sigma^*$, $|w| \geq m$ y $f(w)=a$,
  - $\exists\, r, s, t, b, c, d$ con $|s| \geq 1$, $|rs| \leq m$,
  - tales que $w = rst$, $a=bcd$
  - y $\forall n \geq 0$, $f(rs^n t) = bc^n d$

- Identifica el tipo de funciones que se pueden definir por medio de autómatas finitos. Las funciones que no lo cumplan no pueden ser descritas por autómatas finitos.

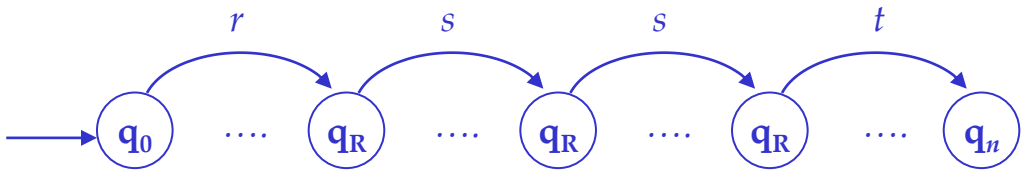
- Demostración:
  - Siguiendo el comportamiento del DFSM, cada símbolo de la cadena de entrada w genera una transición entre estados:



  - Teniendo en cuenta que existen m estados diferentes y que $|w| \geq m$, entonces debe existir al menos un estado q_R repetido en el conjunto de transiciones. Esto se conoce como "*pigeonhole principle*" (principio del palomar): Si en un palomar hay n huecos y n+1 palomas, en algún hueco hay más de una paloma.

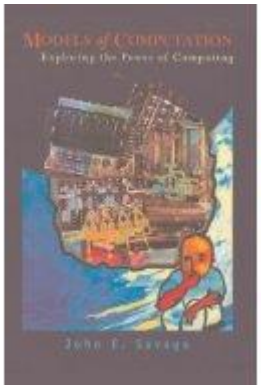


- Demostración:
  - Esto implica que la subcadena *s* se puede "bombear", es decir, que el autómata también reconocería las cadenas *rt*, *rst*, *rsst*, *rssst*, …



  - Por tanto, la expresión rs^nt debe ser reconocida por el autómata.

Bibliografía



- Savage, John E. (1998). "Models Of Computation: Exploring the Power of Computing". Capítulo 4