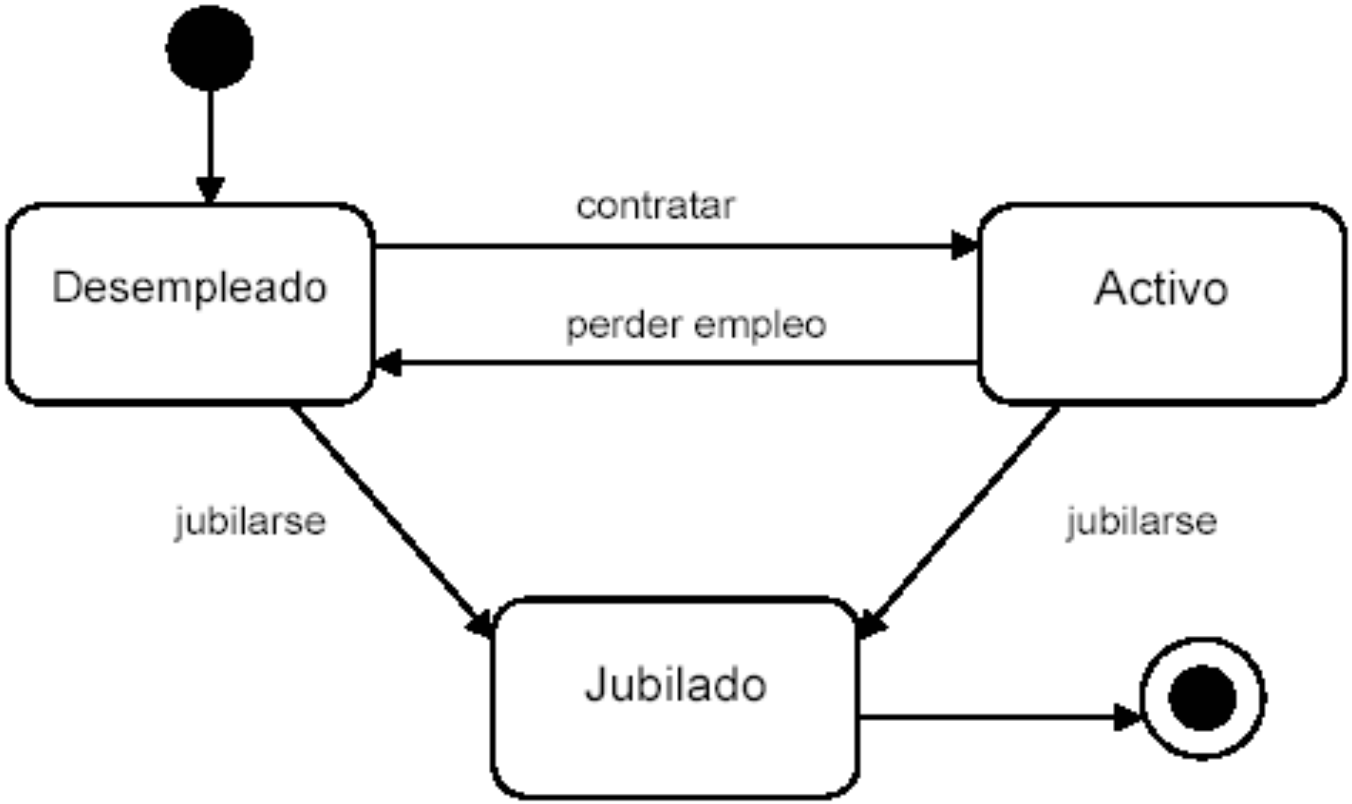
Dígrafos, Estados y Transiciones

- Los dígrafos son una representación gráfica de funciones.
- Se utilizan para representar situaciones problemáticas reales.
- Por ejemplo en Diseño de Sistemas se utilizan para realizar los DTE para ver los cambios de estado de un pedido que puede haber sido creado, cancelado, devuelto, en preparación, enviado, aceptado).

DTE


contratar
Desempleado
Activo
perder empleo
jubilarse
jubilarse
Jubilado

DTE II

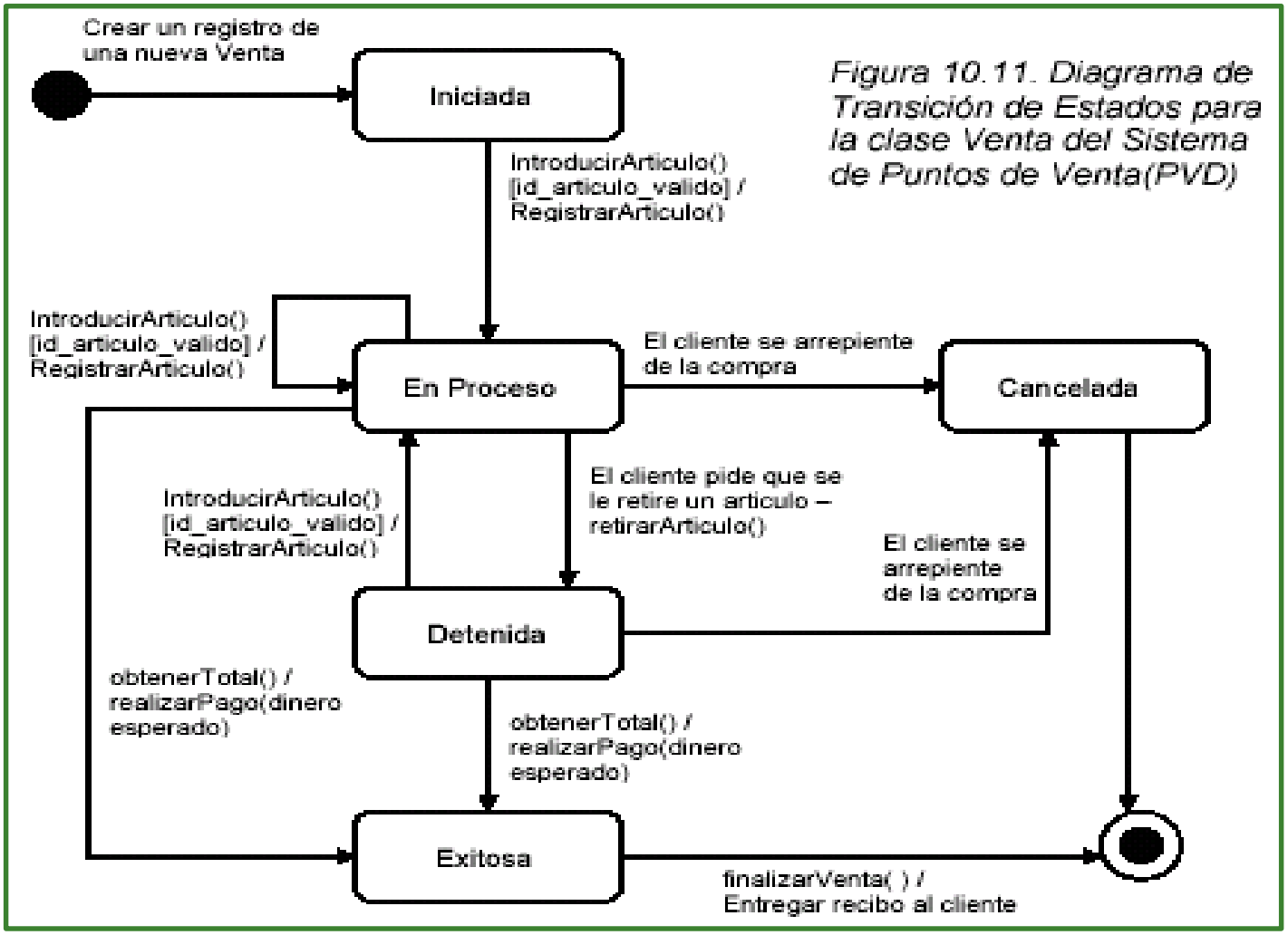
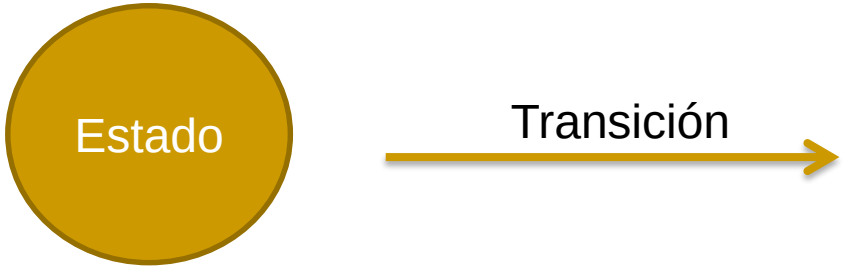


Figura 10.11. Diagrama de Transición de Estados para la clase Venta del Sistema de Puntos de Venta(PVD)

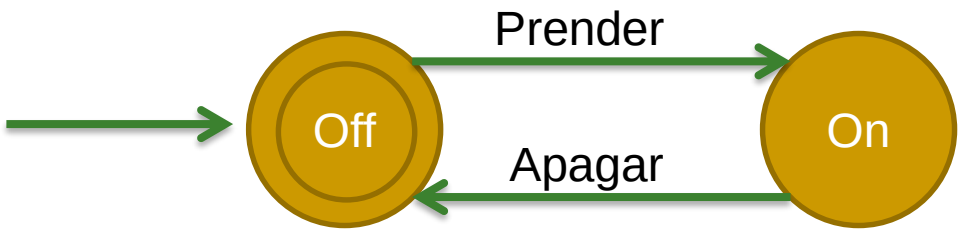
Dígrafos: Representación

- Los Dígrafos se componen de “estados” que se representan con un círculo, y de “transiciones” que se representan por una flecha que une el estado_actual y el estado_siguiente y que se etiquetan con la acción/actividad/cambio que origina esa transición.



Dígrafos: Representación (II)

- De manera abstracta se puede representar a los estados llamándolos con q0, q1, q2, … y a las transiciones con a1, a2, …. Pero tienen que representar inequívocamente un conjunto de realidades diferentes y las condiciones que hacen variar entre una y otra.



Dígrafos Representación (III)

- Conviene que en cada situación a representar con dígrafos se identifique primero una única situación inicial, porque esta será el punto de partida para poder resolver el problema y por eso es importante que sea identificado.

Diagrama de Transición de Estados

- Puede haber varios estados finales o estados de aceptación, que se representarán con un círculo de doble línea.

- Cada “estado” tiene que reunir el conjunto de todas las variables que se necesiten para describir una situación en particular. Tener en cuenta que pueden ser varias circunstancias a considerar. Por ejemplo, para saber si gané un juego de “carreras de autos” tengo que saber que:

  - llegué al final de la pista y
  - el reloj del tiempo para la llegada todavía está corriendo y
  - ningún otro jugador llegó antes que yo

Diagrama de Transición de Estados

- Existen estados “parecidos” donde por ejemplo, se cumple que llegué al final de la pista y todavía el reloj está habilitado pero hay otro jugador que llegó antes que yo, u otra situación en la que llegué primero al final de la pista pero el reloj de tiempo ya llegó a 0 (estado en el cual ningún jugador ganó la carrera)

- No sería correcto considerar “estados” que sean GANE y NO-GANE, o estados LLEGUE_AL_FINAL_DE_LA_PISTA y NO_ LLEGUE_AL_FINAL_DE_LA_PISTA porque por sí solos no describen una situación totalmente y se necesita de más información para conocer por completo el estado de la carrera de autos.

DTE

- Plantear el dígrafo que representa en forma de Estados y Acciones el comportamiento de
- La caja de cambios manual de un auto.
- El comportamiento de un semáforo.
- El Ta-Te-Ti
- Pensar en otras situaciones o dispositivos de la vida real cuyo comportamiento pueda ser representado a través de transiciones de estados.

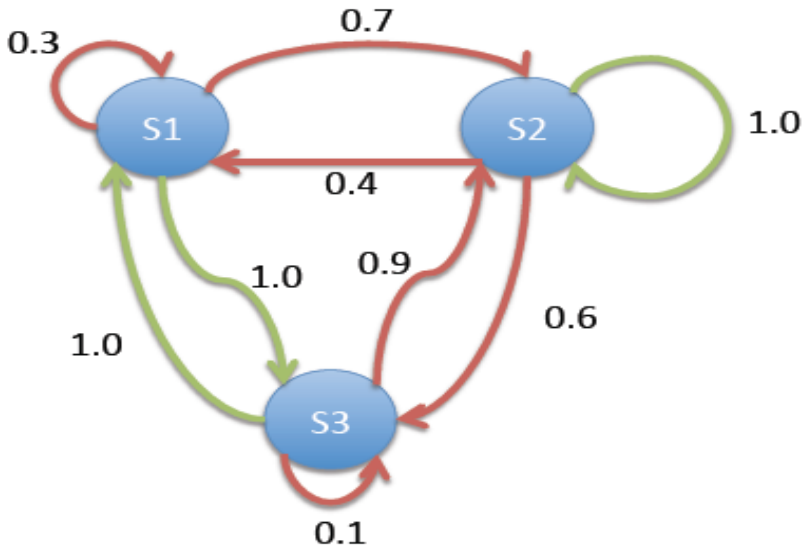
Autómatas Finitos

- Modelos computacionales abstractos. Turing, Chomsky, Cook.
- Usos: chequeo de comportamiento en circuitos, analizadores léxicos (compiladores), búsqueda de patrones, verificación de protocolos.
- Ejemplos en la vida real.
- Contraparte probabilística: Cadenas de Markov: Sistemas Inteligentes, Verificación Probabilística, Reinforcement Learning.

Autómatas: Usos

- Inteligencia Artificial

Aprendizaje por Refuerzo (RL)



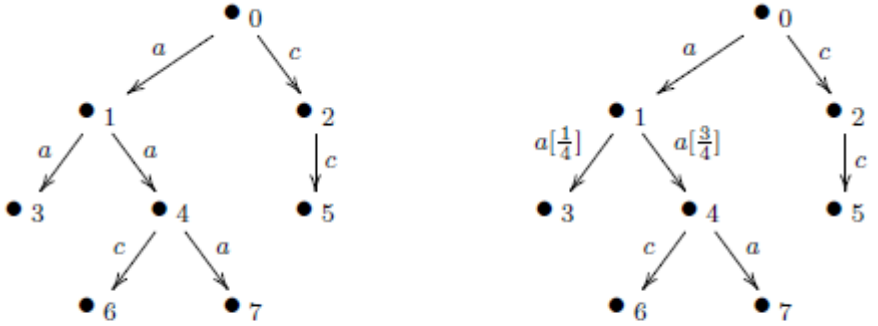
R(s1) = +1
R(s2) = 0
R(s3) = -1

Función de transición T:

S	A	S'	p
S1	R	S1	0.3
	R	S2	0.7
	V	S3	1.0
S2	R	S1	0.4
	R	S2	0.6
	V	S2	1.0
S3	V	S1	1.0
	R	S3	0.1
	R	S2	0.9

Autómatas: Usos

- Inteligencia Artificial: Divergence in Dynamic Systems



(a) LTS (b) PLTS

Figure 2.2: Labelled transition systems

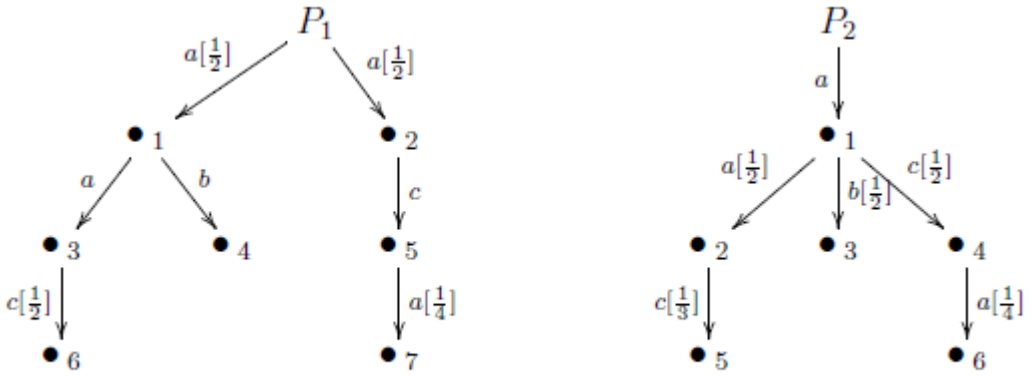


Figure 2.5: Not trace equivalent processes

Autómatas: Usos

- Sistemas Multiagente
- Diseño de Compiladores
- Testing de Protocolos

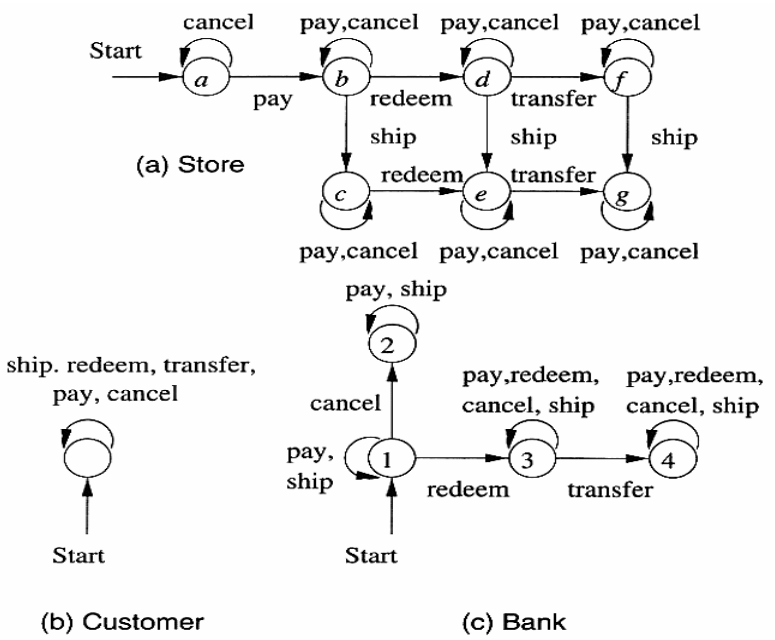


Figure 2.2: The complete sets of transitions for the three automata

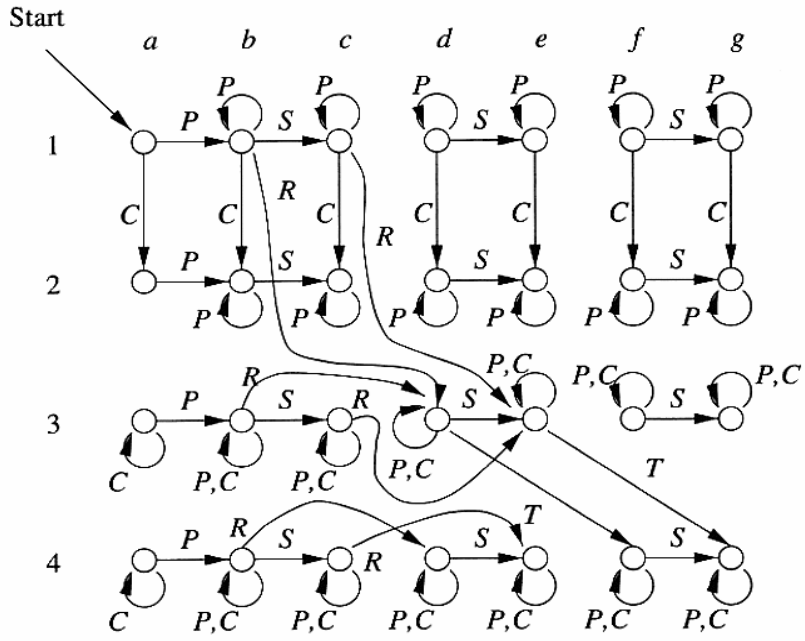
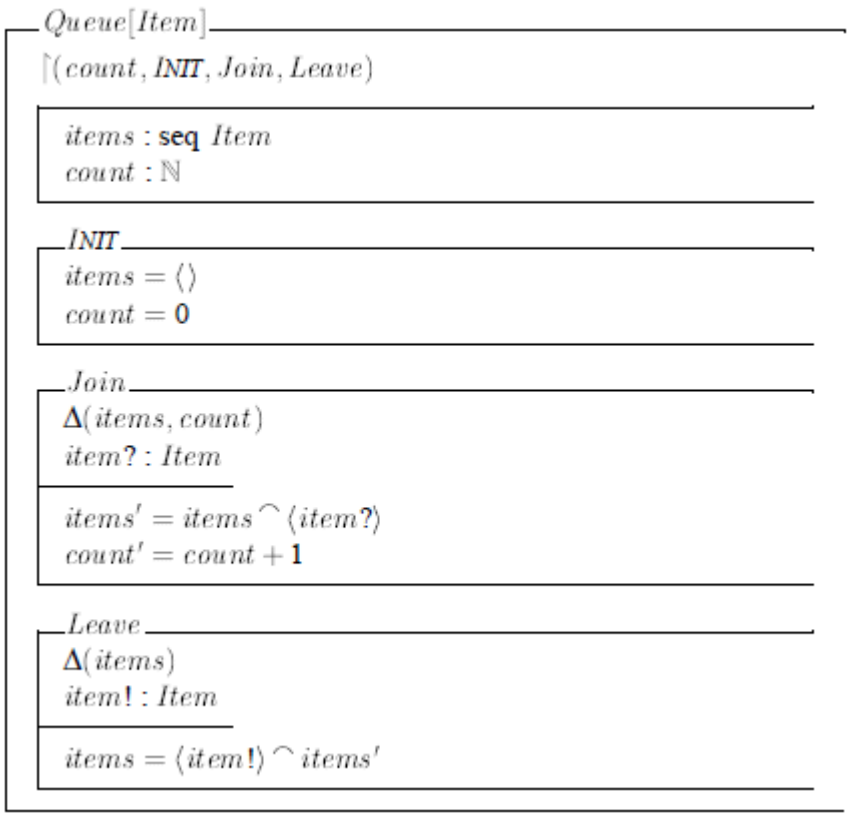
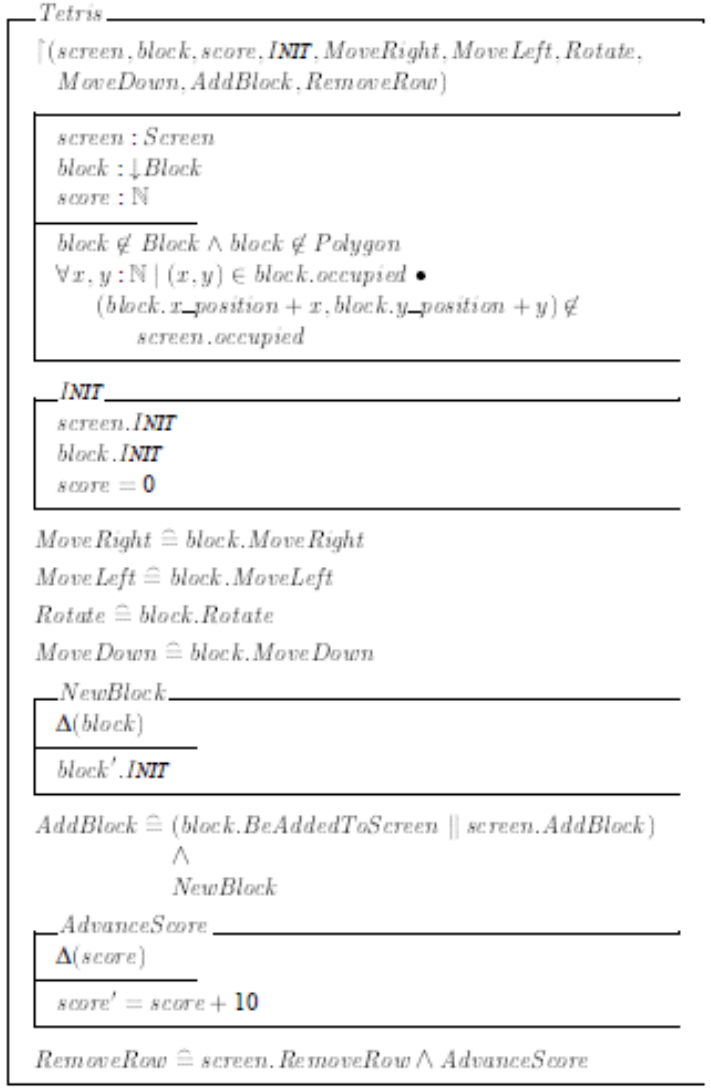


Figure 2.3: The product automaton for the store and bank

Autómatas: Usos

- Object-Z

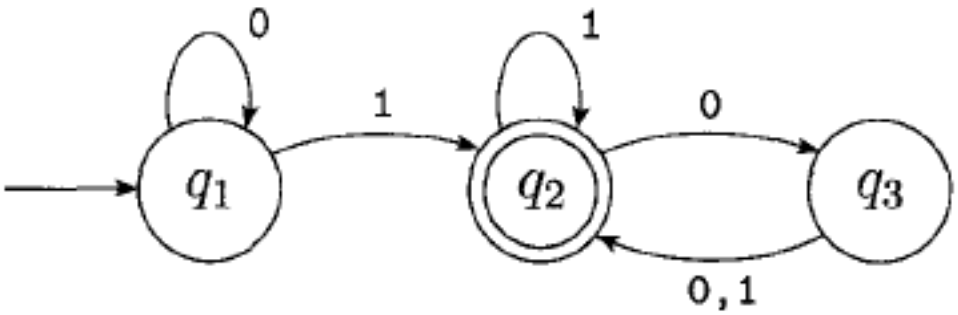




Autómatas y complejidad

- **Decidibilidad** (Puede ser resuelto?)
- **Eficiencia** (Tratabilidad)

Autómata M1



- La figura representa el diagrama de Estados del autómata

Autómata

- Estados: q1, q2, q3.
- Estado Inicial: q1
- Estado final o de aceptación: q2
- Transiciones: Flechas

- El autómata recibe inputs formados por strings, los procesa, y determina si el string ha sido aceptado o rechazado.

Autómata

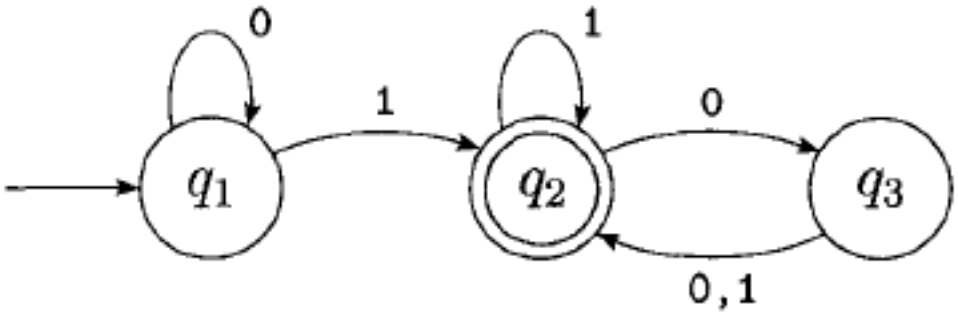
- El autómata produce la salida luego de recibir el último símbolo.
- Ejemplo: entrada 1101

Definición formal

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***,[1]
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.[2]

Definición formal



We can describe M_1 formally by writing $M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

4. q_1 is the start state, and
5. $F = \{q_2\}$.

automata-lib

- Librería en Python para implementar y testear el comportamiento de autómatas (DFA, NFA, PA, TM, etc.)
- https://github.com/caleb531/automata
- Requiere Python 3. Se recomienda como IDE PyCharm o Anaconda

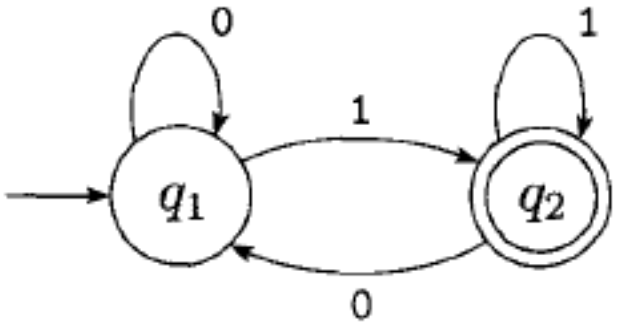
Autómatas

- Si A es el lenguaje formado por todos los strings que M acepta, decimos que A es lenguaje de la máquina M, o que M reconoce A.

$$L(M) = A.$$

$$A = \{w \mid w \text{ contains at least one } 1 \text{ and an even number of } 0\text{s follow the last } 1\}.$$

Autómata M2



$$M_2 = (\{q_1, q_2\}, \{0,1\}, \delta, q_1, \{q_2\})$$

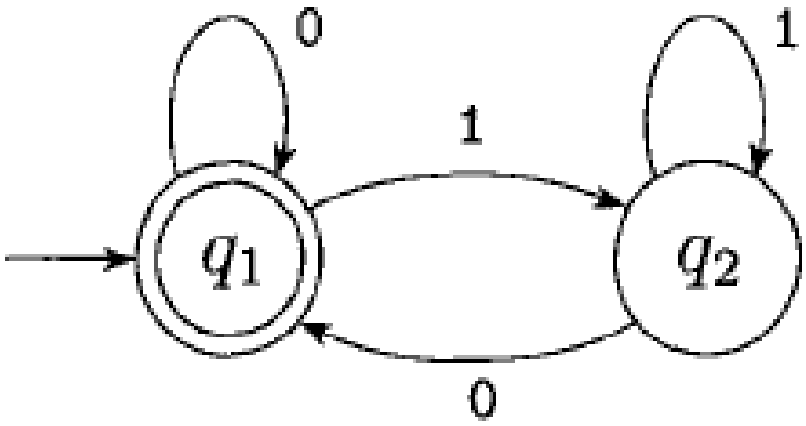
	0	1
q_1	q_1	q_2
q_2	q_1	q_2

Autómata M2

- ¿Qué lenguaje reconoce?

$$L(M_2) = \{w | \, w \text{ ends in a } 1\}.$$

Autómata M3



Autómata M3

- ¿Qué lenguaje reconoce?

$$L(M_3) = \{w | \ w \text{ is the empty string } \varepsilon \text{ or ends in a } 0\}.$$

Lenguaje Regular

- Decimos que un lenguaje es regular si es reconocido por un Autómata Finito
- Relación con la propiedad de Markov

Ejemplo: detección de subcadenas

- Diseñar un autómata que reconozca el lenguaje formado por todas las cadenas que contengan 001 como subcadena.

- Ej.+ 0010, 1001, 11111001111.
- Ej.- 0000, 11

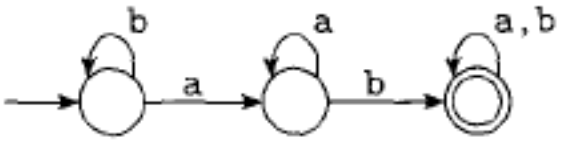
Ejemplo

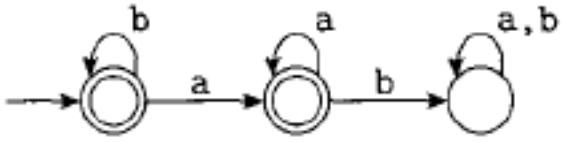
- Primer punto: salteamos todos los 1, hasta detectar un 0. (El primer componente de la subcadena buscada)
- Si luego detectamos un 1 volvemos al estado inicial.
- Sino, tenemos 00, también parte del patron que buscamos
- Continuamos el scanning hasta que detectemos el siguiente 1.

Ejemplo II

- Problema: Autómata que reconoce el lenguaje formado por todas las cadenas que NO contienen “ab”

- Solución: Construimos el autómata que reconoce el leng. Formado por todas las cadenas que contienen “ab”, e intercambiamos los estados no finales por estados finales, y viceversa.





Preguntas

- ¿Cuántos Estados necesito?
- ¿Cómo asigno las transiciones?

- Tener en cuenta:
- No he visto ningún símbolo del patrón que busco.
- He visto uno/dos símbolos.
- He visto el patrón completo