CyC - Practica 7

Facundo Tomatis

(Ejercicio 1)

Consigna:
Construya una MTN que genere de manera no determinística todos los números de 8 bits. Es decir que, dado cualquier número, alguna computación de la máquina lo generará. ¿Cuántos movimientos hace la maquina?
Respuesta:

$$M =< Q, \Sigma, \Gamma, \Delta, q_0, q_A, q_R >, \Delta : Q \times \Gamma \rightarrow \rho((Q \cup \{q_A, q_R\}) \times \Gamma \times \{D, I, S\})$$

$$\Delta(q_0, B) = \{(q_1, 0, D), (q_1, 1, D)\}$$

$$\Delta(q_1, B) = \{(q_2, 0, D), (q_2, 1, D)\}$$

$$\Delta(q_2, B) = \{(q_3, 0, D), (q_3, 1, D)\}$$

$$\Delta(q_3, B) = \{(q_4, 0, D), (q_4, 1, D)\}$$

$$\Delta(q_4, B) = \{(q_5, 0, D), (q_5, 1, D)\}$$

$$\Delta(q_5, B) = \{(q_6, 0, D), (q_6, 1, D)\}$$

$$\Delta(q_6, B) = \{(q_7, 0, D), (q_7, 1, D)\}$$

$$\Delta(q_7, B) = \{(q_A, 0, D), (q_A, 1, D)\}$$

La maquina hace 8 movimientos por la anchura del arbol, el resto de transiciones van hacia q_R

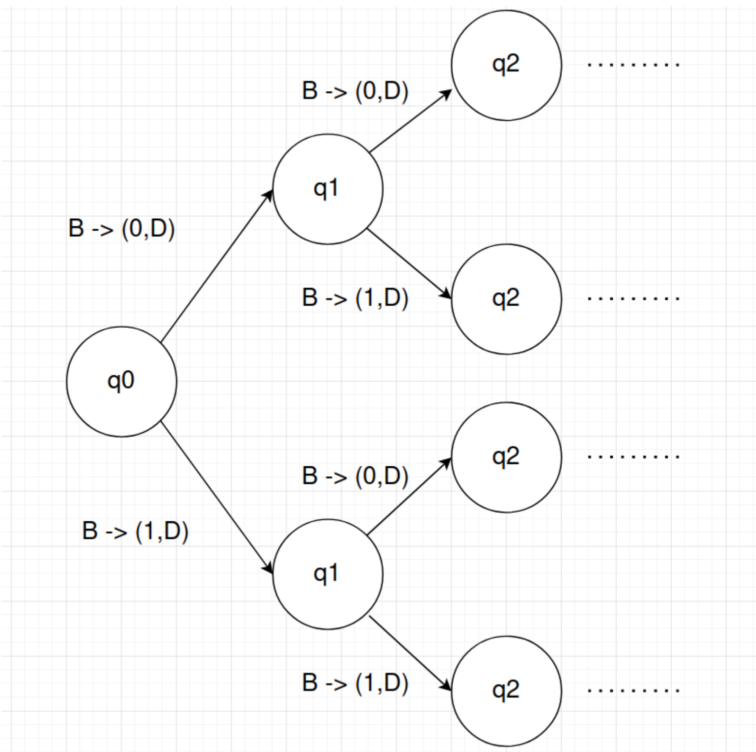

Fig 1.1 Idea de como funciona la maquina M

(Ejercicio 2)

Consigna:
Sean L_1 y L_2, dos lenguajes definidos sobre $\{0,1\}^*$

- $L_1 = \{0^n 1 \mid n \geq 0\}$
- $L_2 = \{1^n 0 \mid n \geq 0\}$

a) Construya una MTN M tal que $L(M) = L_1 \cup L_2$

b) describa la traza de ejecucion para las entradas $w_1 = 001$ y $w_2 = 1101$

Respuesta:

a)

$$\Delta(q_0, r) = \{(q_1, r, S), (q_2, r, S)\}, r \in \Gamma$$
$$\Delta(q_1, 0) = \{(q_1, 0, D)\}$$
$$\Delta(q_1, 1) = \{(q_3, 1, D)\}$$
$$\Delta(q_2, 1) = \{(q_2, 1, D)\}$$
$$\Delta(q_2, 0) = \{(q_3, 0, D)\}$$
$$\Delta(q_3, B) = \{(q_A, B, S)\}$$

q_R para el resto de transiciones

b) $w_1 = 001$ ☑
$q_0001 \vdash q_1001 \vdash 0q_101 \vdash 00q_11B \vdash 001q_3B \vdash 001q_AB$
. $\vdash q_2001 \vdash 0q_301 \vdash 0q_R01$

c) $w_2 = 1101$ ☒
$q_01101 \vdash q_11101 \vdash 1q_3101 \vdash 1q_R101$
. $\vdash q_21101 \vdash 1q_2101 \vdash 11q_201 \vdash 110q_31 \vdash 110q_R1$

(Ejercicio 3)

Consigna:
¿La reducción polinomial posee las siguientes propiedades? Justifique

a) Reflexiva

b) Simétrica

c) Antisimétrica

d) Transitiva

Respuesta:

a) Si, $L \ \alpha_p \ L$ ya que existe una funcion de reduccion (identidad) que reduce de forma constante (un paso) de L a L.

b) No, $L_1 \ \alpha_p \ L_2 = L_2 \ \alpha_p \ L_1$
Es falso por *Contraejemplo:* $L_1 \in P, L_2 = HP$ se cumple que L_1 es reducible polinomialmente a HP ya que hay k pasos, siendo k un polinomio, para convertirlo a una maquina que acepte HP pero es imposible realizarlo al reves (por lo menos no se sabe hasta ahora).

c) No, para que ocurriese debe cumplirse $L_1 \ \alpha_p \ L_2 \wedge L_2 \ \alpha_p \ L_1 \implies L_1 = L_2$
Contraejemplo: Defino $L_1 = \{\text{acepta } w \text{ si es } 0\}$, $L_2 = \{\text{acepta } w \text{ si es } 1\}$, L_1 reduce polinomialmente a L_2 y viceversa pero por definicion sabemos que $L_1 \neq L_2$

d) Si, $L_1 \ \alpha_p \ L_2 \wedge L_2 \ \alpha_p \ L_3 \implies L_1 \ \alpha_p \ L_3$
Existen n pasos (polinomial) para reducir L_1 a L_2, j pasos para L_2 a L_3, para reducir de L_1 a L_3 se requieren $n * j$ pasos. (reducirlo primero a L_2 y luego a L_3)

(Ejercicio 4)

Consigna:
¿Es cierto que si dos lenguajes L_1 y L_2 son NPC entonces $L_1 \ \alpha_p \ L_2$, y también $L_2 \ \alpha_p \ L_1$? Justifique su respuesta.
Respuesta:
Si, por el teorema que dice $L_1, L_2 \in \text{NP}, L_1 \in \text{NPC} \wedge L_1 \ \alpha_p \ L_2 \implies L_2 \in \text{NPC}$
Por lo que si $L_2 \in$ NPC, $L_1 \in$ NPC se cumple $L_1 \ \alpha_p \ L_2$ y también $L_2 \ \alpha_p \ L_1$

(Ejercicio 5)

Consigna:
Sean L_1 y L_2 tales que $L_1 \ \alpha_p \ L_2$, ¿Qué se puede inferir?

a) Si L_1 está en P entonces L_2 está en P

b) Si L_2 está en P entonces L_1 está en P

c) Si L_2 está en NPC entonces L_1 está en NPC

d) Si L_2 está en NPC entonces L_1 está en NP

e) Si L_1 está en NPC entonces L_2 está en NPC

f) Si L_1 está en NPC y L_2 está en NP entonces L_2 está en NPC

Respuesta:

a) Si L_1 está en P entonces L_2 está en P ☒ L_2 puede ser infinitamente grande

b) Si L_2 está en P entonces L_1 está en P ☑ por propiedad de reduccion (existen n pasos para llevar L_1 a L_2)

c) Si L_2 está en NPC entonces L_1 está en NPC ☒ no se puede saber si L_1 esta en NPC, puede ser que L_1 sea muy chico (por ejemplo P) y no se cumpla NPH.

d) Si L_2 está en NPC entonces L_1 está en NP ☑ ya que si L_2 esta en NPC entonces L_1 esta en P o en NP, si esta en P por definicion esta en NP

e) Si L_1 está en NPC entonces L_2 está en NPC ☒ no se sabe si L_2 esta en NP

f) Si L_1 está en NPC y L_2 está en NP entonces L_2 está en NPC ☑ por teorema

---

(Ejercicio 6)

Consigna:
Decir si las siguientes afirmaciones son verdaderas o falsas y justificar

a) Si P=NP entonces todo lenguaje de NPC pertenece a P

b) Si P=NP entonces todo lenguaje de NPH pertenece a P

Respuesta:

a) Si, ya que si P=NP significaria que el lenguaje perteneceria a NPC y al mismo tiempo a P

b) No necesariamente ya que si un leanguje pertenece a NPH no necesariamente esta en NP.

(Ejercicio 7)

Consigna:
¿Qué se puede decir respecto del problema del viajante de comercio (TSP) si se sabe que es NPC, y se asume que P $\neq$ NP?

a) No existe un algoritmo que resuelva instancias de TSP

b) No existe un algoritmo que eficientemente resuelva instancias de TSP

c) Existe un algoritmo que eficientemente resuelve instancias de TSP, pero nadie lo ha encontrado

Respuesta:

a) Existen instancias que resuelvan TSP pero no en un tiempo polinomial.

b) Es relativo, puede ser que una gran cantidad de entradas sea ineficiente pero sea eficiente con una pequeña cantidad de entradas y siga sin ser P. *Por ejemplo:* un algoritmo que resuelva TSP con $\Theta(n^{loglogn})$

c) No, si alguien lo hubiese encontrado se podria probar que $P = NP$.