Practica 7

1) 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 máquina?

   M=<Q, Σ, Γ, Δ, q0, qA, qR>, Δ: Q x Γ → r ((Q U {qA,qR}) x Γ x {D, I, S})

   Δ (q0, B) = {(q1, 0, D), (q1, 1, D)}
   Δ (q1, B) = {(q2, 0, D), (q2, 1, D)}
   Δ (q2, B) = {(q3, 0, D), (q3, 1, D)}
   Δ (q3, B) = {(q4, 0, D), (q4, 1, D)}
   Δ (q4, B) = {(q5, 0, D), (q5, 1, D)}
   Δ (q5, B) = {(q6, 0, D), (q6, 1, D)}
   Δ (q6, B) = {(q7, 0, D), (q7, 1, D)}
   Δ (q7, B) = {(qA, B, S), (qA, B, S)}

   Esta maquina realiza siempre 8 movimientos. Todas las otras combinaciones de estado símbolo que faltan llevan al estado qR

2) Sean L1 y L2, dos lenguajes definidos sobre {0,1}*

   $L1 = \{0^n 1 \mid n \geq 0\}$
   $L2 = \{1^n 0 \mid n \geq 0\}$

   a) Construya una MTN M tal que L(M)= L1 ∪ L2

   M=<Q, Σ, Γ, Δ, q0, qA, qR>, Δ: Q x Γ → r ((Q U {qA,qR}) x Γ x {D, I, S})

   Δ (q0, x) = {(q1, x, S), (q2, x, S)} para todo x ∈ Γ

   Δ (q1, 0) = {q1, 0, D)}
   Δ (q1, 1) = {q3, 1, D)}

   Δ (q2, 1) = {q2, 1, D)}
   Δ (q2, 0) = {q3, 0, D)}

   Δ (q3, B) = {qA, B, S)}

   Todas las otras combinaciones de estado símbolo que faltan llevan al estado qR

   b) Describa la traza de ejecución para las entradas w1=001 y w2= 1101

   w1 = 001

   $q_0001 \vdash q_1001 \vdash 0q_101 \vdash 00\,q_11 \vdash 001q_3B \vdash 001q_AB$
   $\vdash q_2001 \vdash 0\,q_301 \vdash 0\,q_R01$

   w2= 1101

$q_0 1101 \vdash q_1 1101 \vdash 1q_3 101 \vdash 1q_R 101$
$\vdash q_2 1101 \vdash 1q_2 101 \vdash 11q_2 01 \vdash 110q_3 1 \vdash 110q_R 1$

3) ¿La reducción polinomial posee las siguientes propiedades? Justifique
   a) Reflexiva

      ¿ L1 αp L1?

      Si, se puede construir una MTD Mf que computa la función de reduccion de identidad en un tiempo polinomial. Este Mf básicamente no hace nada.

   b) Simétrica

      ¿L1 αp L2 $\Rightarrow$ L2 αp L1?

      No, o al menos no por ahora. Contraejemplo: L1 $\in$ P y L2 $\in$ NP, se puede construir una MTD que compute la función de reduccion tiempo polinomial de L1 a L2, pero no se puede construir una MTD que realice lo mismo pero de L2 a L1, porque esto significaría que P = NP (ya que dado que L1 $\in$ P, por teorema L2 también $\in$ P) y eso todavía aun no se ha demostrado.

   c) Antisimétrica

      ¿L1 αp L2 $\wedge$ L2 αp L1 $\Rightarrow$ L1 = L2?

      No. Contraejemplo: L1 = { 0^n 1 / n >= 0 } y L2 = { 1^n 0 / n >= 0 } se puede hacer L1 αp L2 y L2 αp L1 pero sabemos por definición que L1 $\neq$ L2.

   d) Transitiva

      ¿L1 αp L2 $\wedge$ L2 αp L3 $\Rightarrow$ L1 αp L3?

      Si, por enunciado sabemos que se puede realizar una reducción de L1 a L2 con una MTD en tiempo polinomial y una reducción de L2 a L3 con una MTD en tiempo polinomial. Se puede realizar una MTD que junte ambas MTD que realizan las reducciones de L1 αp L2 y L2 αp L3. Esta MTD va a trabajar en tiempo polinomial también, porque la suma de polinomios da otro polinomio.

4) ¿Es cierto que si dos lenguajes L1 y L2 son NPC entonces L1 αp L2, y también L2 αp L1? Justifique su respuesta.

   Sabemos que como L1 $\in$ NPC (NP-Completo) entonces, por definición, L1 $\in$ NPH y L1 $\in$ NP.

   Que L1 $\in$ NPH significa que para todo L $\in$ NP se cumple L αp L1.

   Como se sabe que L2 $\in$ NPC, se sabe que L2 $\in$ NP (por definición de NPC), por lo tanto, por lo anterior dicho se puede afirmar que L2 αp L1 (definición de NPH). Se puede demostrar análogamente para L1 αp L2

5) Sean L1 y L2 tales que L1 αp L2, ¿Qué se puede inferir?

a) Si L1 está en P entonces L2 está en P

No necesariamente. L2 podría estar en P pero podría también estar en NP o en R o en RE u otras clases de complejidad. No hay ningún teorema que nos permita saber que si L1 esta en P, L2 también lo está. Contraejemplo: Se podría escribir una MTD Mf que compute la función de reduccion en tiempo polinomial de L1 siendo L1 = MCD a L2 siendo L2 = HP. No se puede computar una función de reducción de L2 a L1.

b) Si L2 está en P entonces L1 está en P

Si, es cierto. El teorema 3 de la clase 12 dice que L1 αp L2 y L2 $\in$ P $\Rightarrow$ L1 $\in$ P

c) Si L2 está en NPC entonces L1 está en NPC

No necesariamente. No hay ningún teorema que nos permita saber que si L2 está en NPC L1 también lo está.

d) Si L2 está en NPC entonces L1 está en NP

Si, esto es cierto. Que L2 este en NPC implica que L2 esta en NPH, si está ahí significa que cualquier problema NP se puede reducir polinómicamente a L2. Dado que hay una reducción polinómica de L1 a L2 entonces L1 también debe estar en NP (o en P ya que está incluido en NP).

e) Si L1 está en NPC entonces L2 está en NPC

Esto es cierto solo si L2 $\in$ NP (o a P porque P está incluido en NP). El teorema 6 de la clase 12 dice que si L1 $\in$ NPC y L1 αp L2 entonces L2 $\in$ NPC. Como no se sabe si L2 $\in$ NP, no se puede afirmar lo anterior. Contraejemplo: Se podría escribir una MTD Mf que compute la función de reducción en tiempo polinomial de L1 = TSP a L2 = HP. L2 no esta en NPC (ni si quiera esta en NP)

(cuando reducis lo que esta a la derecha puede ser muy complicado)

f) Si L1 está en NPC y L2 está en NP entonces L2 está en NPC

Si es cierto, por lo anterior dicho.

6) Decir si las siguientes afirmaciones son verdaderas o falsas y justificar

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

Verdadero, ya que si un lenguaje NPC por definición está en NP

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

No necesariamente. Para que esto se cumpla el lenguaje NPH debe pertenecer a NP y esto no siempre es así. De hecho, si el lenguaje pertenece a NPH y a NP entonces se trata de un lenguaje que pertenece a NPC.

7) ¿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

Falso. Si existen algoritmos que resuelven instancias de TSP, pero no lo hacen en un tiempo polinomial

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

Verdadero. Si bien puede que exista un algoritmo que eficientemente resuelva instancias de TSP con una pequeña cantidad de entradas, es seguro que no existe es un algoritmo que eficientemente resuelva instancias de TSP con gran cantidad de entradas, ya que no se tiene un tiempo polinomial. Si existiese, entonces TSP seria P.

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

Puede que exista como puede que no. Como nadie lo ha encontrado, no se puede afirmar que existe. Si existiese entonces se podría probar que P = NP

d) TSP no está en P

Verdadero. Por el teorema 5 si $(L \in NPC)$ AND $(L \in P) \Rightarrow P = NP$. Esto se contradice con lo que se asume en un principio de que $P \neq NP$. Por lo que TSP no está en P.