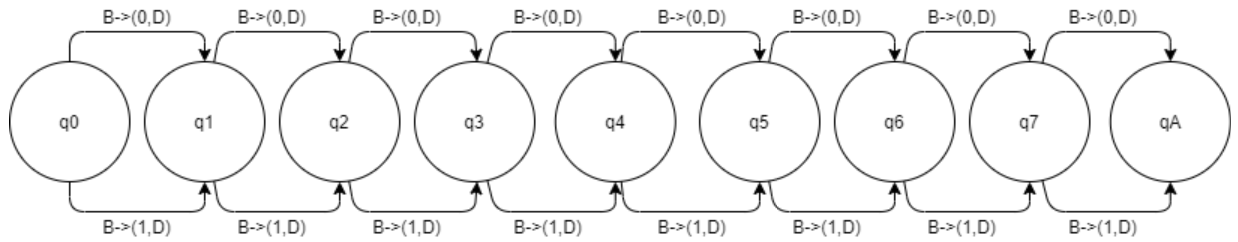
**Practica 7**

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?

Okey, la siguiente MTN debe satisfacer el enunciado:



En el gráfico falta indicar que cualquier entrada que no sea blanco será rechazada.

q0,B |- 0,q1,B |- 0,0,q2,B |- 0,0,0,q3,B |- 0,0,0,0,q4,B |- ...

|- 0,0,0,1,q4,B |- ...

|- 0,0,1,q3,B |- 0,0,1,0,q4,B |- ...

|- 0,0,1,1,q4,B |- ...

|- 0,1,q2,B |- 0,1,0,q3,B |- 0,1,0,0,q4,B |- ...

|- 0,1,0,1,q4,B |- ...

|- 0,1,1,q3,B |- 0,1,1,0,q4,B |- ...

|- 0,1,1,1,q4,B |- ...

|- 1,q1,B |- 1,0,q2,B |- 1,0,0,q3,B |- 1,0,0,0,q4,B |- ...

|- 1,0,0,1,q4,B |- ...

|- 1,0,1,q3,B |- 1,0,1,0,q4,B |- ...

|- 1,0,1,1,q4,B |- ...

|- 1,1,q2,B |- 1,1,0,q3,B |- 1,1,0,0,q4,B |- ...

|- 1,1,0,1,q4,B |- ...

|- 1,1,1,q3,B |- 1,1,1,0,q4,B |- ...

|- 1,1,1,0,q4,B |- …

Esta traza de computación seguirá hasta qA, y puede generar cualquier numero de 8 bit que se requiera. Independientemente del numero que genere siempre realizara 8 movimientos.

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

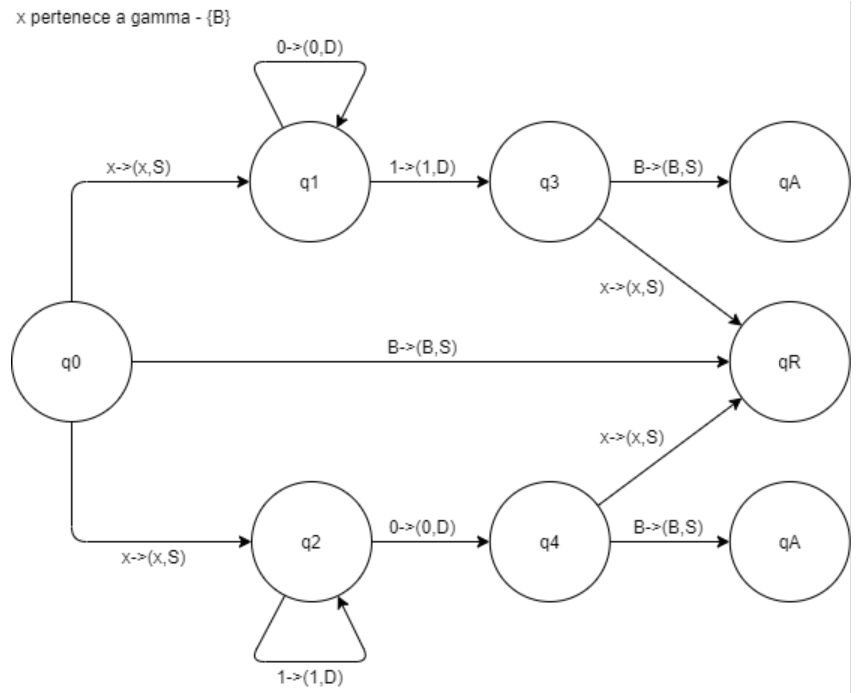
L1 = {0n1| n ≥ 0}

L2 = {1n0| n ≥ 0}

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

Uni de forma no determinística las dos máquinas que aceptan L1 y L2.

Según la teoría, mientras una rama acepte la entrada entonces esta será aceptada, así que siempre habrá una rama que acepte entradas de L1 o de L2.



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

w1=001

q0,0,0,1 |- q1,0,0,1 |- 0,q1,0,1 |- 0,0,q1,1 |- 0,0,1,q3,B |- 0,0,1,qA,B

|- q2,0,0,1 |- 0,q4,0,1 |- 0,qR,0,1

w2=1101

q0,1,1,0,1 |- q1,1,1,0,1 |- 1,q3,1,0,1 |- 1,1,qR,0,1

|- q2,1,1,0,1 |- 1,q2,1,0,1 |- 1,1,q2,0,1 |- 1,1,0,q4,1 |- 1,1,0,qR,1

Se puede observar que w1 es una entrada aceptada, mientras que w2 no siempre es rechazada.

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

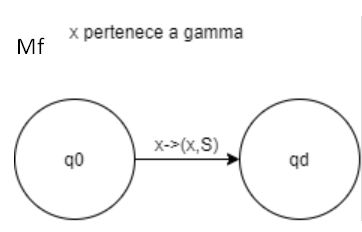
a) Reflexiva

Esta propiedad indica que L1αp L1 y además la funcion de reducción f es computada por una MTD que trabaja en tiempo polinomial. (f∈P)

Segun la teoria: Sean L1 y L2 dos lenguajes incluidos en Σ\*, decimos que L1 se reduce polinomialmente a L2 (se denota L1αp L2) sii L1α L2 y ademas la funcion de reduccion f es computada por una MTD que trabaja en tiempo polinomial (f∈P).

Primero deberemos demostrar que L1α L2 :

Llamaremos Mf a nuestra MTD que computa f , su diagrama será el siguiente:



Es evidente que Mf siempre se detiene ya que siempre realiza un único paso, ahora necesitamos saber si ∀w ∈ Σ\*, w ∈ L1 ⇔ f(w) ∈ L2 la respuesta a esto es que si, ya que en nuestro caso L2=L1 y w=f(w), así que es normal afirmar que ∀w ∈ Σ\*, w ∈ L1 ⇔ w ∈ L1.

Con esto se dan las condiciones necesarias para afirmar que L1α L2.

Segundo hay que demostrar que Mf es una MTD que trabaja en tiempo polinomial:

Mf es una MTD porque en ningún momento se presentan 2 cursos de acción distintos para el mismo símbolo en el mismo estado.

¿Mf trabaja en tiempo polinomial?

si, concretamente Mf computa f(w) en 1 paso para todo w.

∴ Queda demostrado que la reducción polinómica posee la propiedad reflexiva.

b) Simétrica

Esta propiedad implica que si L1αp L2 ⇔ L2αp L1.

Esto es falso, ya que puedo encontrar L1 y L2 tal que L1αp L2 pero no ocurra que L2αp L1.

Por ejemplo un L2 que sea NPH y un L1 que sea distinto de L1 no sea NPH pero sea NP.

∴ No se cumple la propiedad simétrica.

c) Antisimétrica

Para que esto sea cierto entonces se tiene que dar que L αp L´ ⇔ L=L1.

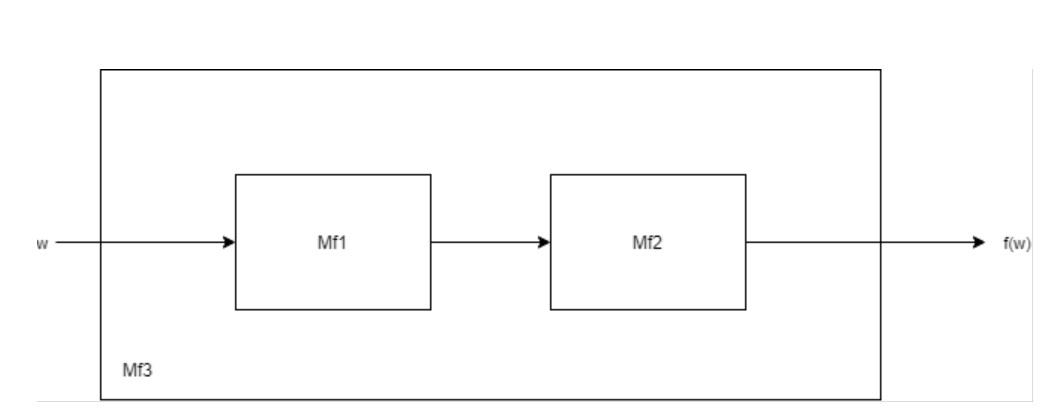
Lo cual no siempre se cumple, por ejemplo usando un L que sea NPH y un L´ que sea distinto de L1 no sea NPH pero sea NP.

∴ No se cumple la propiedad antisimétrica.

d) Transitiva

Esta propiedad implica que si L1αp L2 y L2αp L3 ⇒ L1αp L3.

Se cumple ya que supongamos que existe Mf1 y Mf2 que hacen L1αp L2 y L2αp L3 respectivamente, entonces podríamos armar Mf3 de esta forma:



Ya que Mf1 y Mf2 son MTD que funcionan en tiempos polinómicos entonces Mf3 al simplemente hilar ambas también será una MTD, en cuanto a su tiempo ya consiste en la suma del tiempo de Mf1 y de Mf2, asi que tambien seria polinómico.

∴ Si se cumple la propiedad transitiva.

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

Si. Por que si dos lenguajes son NPC entonces son NP y son capaces de reducir cualquier otro lenguaje de NP, así que por definición L1 αp L2 ya que L2 es NP, y también L2 αp L1 ya que L1 es NP.

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

Falso, tenemos que L1 αp L2 y que L1 está en P.

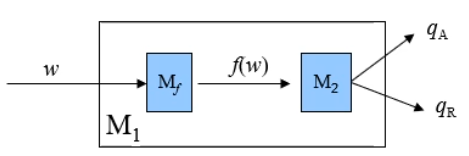
Esto nos dice que L1 será como máximo tan complejo como L2 pero esto no implica que L2 esté en P, L2 puede estar en NP y la reduccion seguiria cumpliendoce sin problema.

∴ No se puede inferir.

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

Verdadero, tenemos que L1 αp L2 y que L2 está en P.

Dem: sean Mf la MTD que computa f en tiempo polinomial. Construimos M1 una MTD tal que L1=L(M1) de la siguiente manera:



L1=L(M1)?

Dado que w ∈ L1 sii f(w) ∈ L2 se tiene que L1=L(M1)

M1 trabaja en tiempo polinomial?

Sean n=|w|

Mf computa f(w) en a lo sumo cnk pasos,

por lo tanto f(w) ≤ cnk

por lo tanto M1 hará a lo sumo cnk + c2(cnk)k2

por lo tanto M1 trabaja en tiempo O(nk3)

∴ Se puede inferir.

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

Falso, tenemos que L1 αp L2 y que L2 está en NPC.

Esto implica que L2 está en NP.

L1 podría estar en NP o en NPC, pero no tenemos forma de saberlo, solo sabemos que es como mucho tan complejo como L2.

∴ No se puede inferir.

d) Si L2 está en NPC entonces L1 está en NP (Verdadero, porque L1 será a lo sumo tan difícil como L2)

Verdadero, porque L1 será a lo sumo tan difícil como L2.

∴ Se puede inferir.

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

Falso, tenemos que L1 αp L2 y que L1 está en NPC.

Esto nos indica que L2 para poder reducir a L1 tendrá que ser como mínimo tan complejo como L1, osea estar en NPC, pero no necesariamente se tiene que quedar ahí, puede ser mas complejo e ir a rangos superiores de complejidad alejándose mas y mas a NPC.

∴ No se puede inferir.

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

Verdadero,tenemos que L1 αp L2, que L1 está en NPC y que L2 está en NP.

Esto nos indica que L2 para poder reducir a L1 tendrá que ser como mínimo tan complejo como L1, osea estar en NPC, además también sabemos que como mucho podra estar en NPC ya que se encuentra en NP, asi que acotando por arriba y por abajo nos quedamos con que L2 estará en NPC

∴ Se puede inferir.

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, porque NPC está incluido en NP, si P=NP entonces NPC está incluido en P.

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

Falso, porque NPH no está incluido en NP, si P=NP entonces NPH sigue sin estar incluido en P, puede darse que algún NPH lo esté, pero esto sin duda no es la norma.

7) ¿Qué se puede decir respecto del problema del viajante de comercio (TSP) si se sabe que es NPC, y se asume que P ≠ NP?

Sabemos que:

* TSP está en NP

Nos dicen que:

* TSP está en NPC
* P ≠ NP

a) No existe un algoritmo que resuelva instancias de TSP

Falso, si está en NP implica que existe pero no es eficiente.

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

Verdadero, si está en NPC y P ≠ NP esto implica que no está en P, osea que es intratable el problema por ahora.

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

Falso por que tenemos la certeza de que TSP no está en P

d) TSP no está en P

Verdadero, si está en NPC y P ≠ NP esto implica que no está en P