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

L1 = {0n1| n ≥ 0}

L2 = {1n0| n ≥ 0}

Los 3 incisos como que entendi mal los lenguajes y los hice mal, habria que rehacerlos

a) Demuestre que existe una reducción (L1 α L2)

Sean L1,L2 ⊆ Σ\* se dirá que L1 se reduce a L2 (L1 α L2) si existe una funcion computable (o recursiva) f: Σ\*→ Σ\* tal que:

∀w ∈ Σ\*, w ∈ L1⇔ f(w) ∈ L2

Se dice que f es computable si existe una MT que la computa y que siempre se detiene.

Nótese que de la definición de L1 y L2 podemos deducir las siguientes implicaciones:

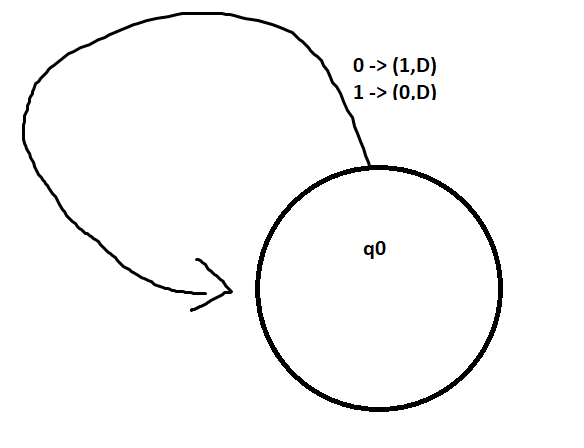
* Los w de L1 terminan en 1
* Los w de L2 terminan en 0

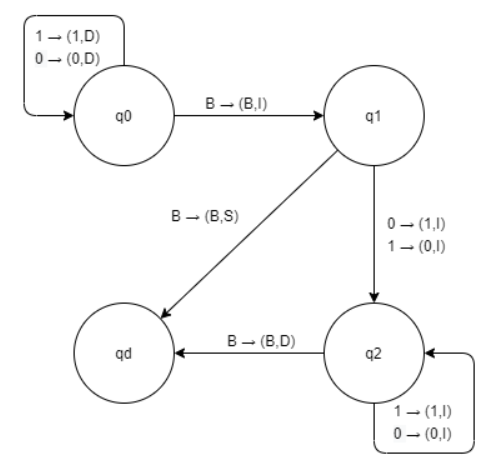
Vamos a demostrar que (L1 α L2)

Mf=<Q,Σ,Γ,δ,q0,qd> Q={q0,q1,q2} Σ={0,1} Γ={B,a,b}

Mf es una MT de/para computo

00001 ∈ L1 → f(00001) = 11110 ∈ L2

Esta que sigue esta mal:



Puede demostrarse fácilmente que:

1. Mf siempre se detiene.

Solo hay 2 bucles que podrían provocar que Mf continúe indefinidamente, ambos lo único que hacen es moverse a la derecha del todo o a la izquierda del todo respectivamente, sabemos que w es finito así que independientemente de lo largo que sea w se deberá poder llegar a los extremos en una cantidad finita de pasos.

1. ∀w ∈ Σ\*, w ∈ L1⇔ f(w) ∈ L2

Tenemos 2 situaciones principales

1. w pertenece a L1, en este caso Mf lo único que hará es moverse a derecha, cambiar el 1 por un cero, y volver al inicio de la cinta, como lo que queda en el la cinta termina en 0, entonces cumple las condiciones para pertenecer a L2

.: w ∈ L1⇒f(w) ∈ L2

1. w no pertenece a L1, esto puede ocurrir de 2 formas. El 1er caso es cuando w no termina en 1, en este caso Mf lo cambia por un 1 y vuelve al inicio de la cinta. El 2do caso se da cuando no hay nada en la cinta, si esto ocurre Mf no hace cambios. Ambos resultados dejan en la cinta frases que por la definición de L2 no pertenecerán a este.

.: w ∉ L1⇒f(w) ∉ L2, o lo que es equivalente f(w) ∈ L2⇒w ∈ L1

b) Idem para L2 = {λ}

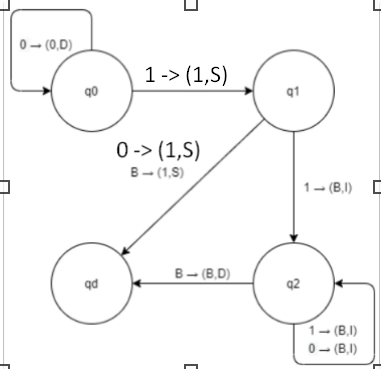
Nótese que de la definición de L1 y L2 podemos deducir las siguientes implicaciones:

* Los w de L1 terminan en 1
* Los w de L2 siempre son vacíos

Vamos a demostrar que (L1 α L2)

Mf=<Q,Σ,Γ,δ,q0,qd> Q={q0,q1,q2} Σ={0,1} Γ={B,a,b}

Mf es una MT de/para computo

Esta regular la imagen pero la idea es esta:

- Verifica que sea una cadena de ceros con un 1 al final, si lo es deja λ. Si no lo es deja un 1.

Puede demostrarse fácilmente que:

1. Mf siempre se detiene.

Solo hay 2 bucles que podrían provocar que Mf continúe indefinidamente, ambos lo único que hacen es moverse a la derecha del todo o borrar hasta la izquierda del todo respectivamente, sabemos que w es finito así que independientemente de lo largo que sea w se deberá poder llegar a los extremos en una cantidad finita de pasos.

1. ∀w ∈ Σ\*, w ∈ L1⇔ f(w) ∈ L2

Tenemos 2 situaciones principales

1. w pertenece a L1, en este caso Mf lo unico que hara es moverse a derecha, comprobar que al final haya un 1 y borrar la cinta, como no quedan símbolos en la cinta, entonces cumple las condiciones para pertenecer a L2

.: w ∈ L1⇒f(w) ∈ L2

1. w no pertenece a L1, esto puede ocurrir de 2 formas. El 1er caso es cuando w termina no termina en 1, en este caso Mf lo deja como esta y listo. El 2do caso se da cuando no hay nada en la cinta, si esto ocurre Mf escribe un uno en la cinta. Ambos resultados dejan en la cinta frases que por la definición de L2 no pertenecerán a este (lo único que pertenece a L2 es una cinta vacía).

.: w ∉ L1⇒f(w) ∉ L2, o lo que es equivalente f(w) ∈ L2⇒w ∈ L1

c) Idem para L2 = {1n0| n > 0}

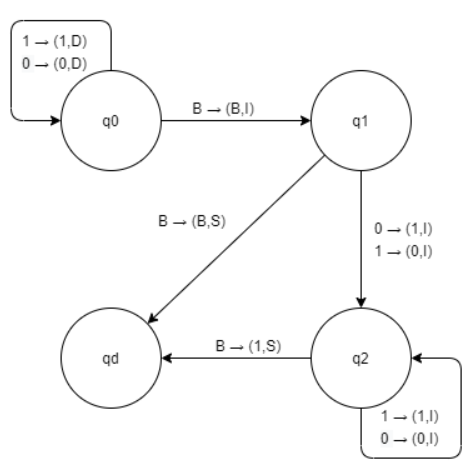
Nótese que de la definición de L1 y L2 podemos deducir las siguientes implicaciones:

* Los w de L1 terminan en 1
* Los w de L2 tienen como mínimo un 1 al inicio
* Los w de L2 terminan en 0

Vamos a demostrar que (L1 α L2)

Mf=<Q,Σ,Γ,δ,q0,qd> Q={q0,q1,q2} Σ={0,1} Γ={B,a,b}

Mf es una MT de/para computo



Puede demostrarse fácilmente que:

1. Mf siempre se detiene.

Solo hay 2 bucles que podrían provocar que Mf continúe indefinidamente, ambos lo único que hacen es moverse a la derecha del todo o a la izquierda del todo respectivamente, sabemos que w es finito así que independientemente de lo largo que sea w se deberá poder llegar a los extremos en una cantidad finita de pasos.

1. ∀w ∈ Σ\*, w ∈ L1⇔ f(w) ∈ L2

Tenemos 2 situaciones principales

1. w pertenece a L1, en este caso Mf lo unico que hara es moverse a derecha, cambiar el 1 por un cero, y volver al inicio de la cinta y agregar un 1 delante, como lo que queda en el la cinta termina en 0 y tiene un 1, entonces cumple las condiciones para pertenecer a L2

.: w ∈ L1⇒f(w) ∈ L2

1. w no pertenece a L1, esto puede ocurrir de 2 formas. El 1er caso es cuando w no termina en 1, en este caso Mf lo cambia por un 1 y vuelve al inicio de la cinta y agrega un 1 delante. El 2do caso se da cuando no hay nada en la cinta, si esto ocurre Mf no hace cambios. Ambos resultados dejan en la cinta frases que por la definición de L2 no pertenecerán a este.

.: w ∉ L1⇒f(w) ∉ L2, o lo que es equivalente f(w) ∈ L2⇒w ∈ L1

2) Sean L1 y L2, dos lenguajes tales que existe una reducción (L1 α L2)

a) Qué se puede afirmar de L1 si se sabe que L2 ∈ R

En la filmina 12 de la teoría de reducibilidad se plantea el teorema

L1,L2⊆Σ\* y existe la reducción L1 α L2 entonces: L2 ∈ R ⇒ L1 ∈ R

así que puedo decir en el inciso que si L2 ∈ R ⇒ L1 ∈ R

b) Qué se puede afirmar de L1 si se sabe que L2 ∈ (CO-RE - RE)

Primero analicemos que significa L2 ∈ (CO-RE - RE)

De esto deduzco que:

* L2 ∈ CO-RE
* L2 ∈ RE
* L2 ∉ RE
* L2 ∉ R

L1 α L2

L1 <= L2

L2 ∈ R -> L1 ∈ R

L1 ∉ R -> L2 ∉ R

En la teoría se plantea en la filmina 11 que:

La reducción establece una relación de “≤” grado de dificultad computacional entre lenguajes.

L1 ∈ (CORE - RE) (por def de -)

L1 ∈ CORE ^ L1 ∉ RE (POR DEF DE RE)

L1 NO ∉ RE (POR INCLUSO)

L1 ∉ R ni RE

c) Qué se puede afirmar de L2 si se sabe que L1 ∈ R

Puede ser cualquier cosa

d) Qué se puede afirmar de L2 si se sabe que L1 ∈ (CO-RE - RE)

En la filmina 16 de la teoría se nos dice que por contrarrecíproca del teorema 2 sabemos que

L1 ∉ RE ⇒ L2 ∉ RE

Esto se adecua a nuestro caso ya que sabemos (gracias al enunciado) que L1 ∈ (CO-RE - RE), de esto podemos deducir que L1 ∉ RE, así que podemos afirmar que L2 ∉ RE.

Como extra podemos afirmar que la complejidad computacional de L2 es como mínimo tan difícil que la de L1.

3) Sean HP y Lu los lenguajes Halting Problem y Lenguaje Universal respectivamente.

HP = {(<M>,w) / M se detiene con input w}

Lu = {(<M>,w) / M acepta w}

Demuestre que existe una reducción HP α Lu.

Primero unos datos interesantes:

* HP ∈ (RE - R)
* Lu ∈ (RE - R)

Aprovechare esta MT de la teoría:

Usaremos la convención de que si <M> es un código invalido entonces representa el código de una MT que rechaza cualquier entrada.

Y que si (<M>,w) no es un par valido entonces representa una MT que no se detiene.

Vamos a demostrar que (HPα Lu)

Para esto debe existir una función computable (o recursiva) f: Σ\*→ Σ\* tal que:

* f debe ser computable
* ∀(<M>,w) ∈ Σ\*, (<M>,w) ∈ HP⇔ (<M’>,w) ∈ Lu.

f((<M>,w))=(<M’>,w)

Construiremos Mf para demostrar estos puntos

Mf trabaja de la siguiente manera:

Primero chequea que (<M>,w) sea un par válido de una MT, si no lo es, escribe un par válido en la cinta en el que <M> es la codificación de una MT MR y un w, tal que MR rechaza w, si era un par valido pero <M> no es un código válido escribe un par válido en la cinta en el que <M> es la codificación de una MT MA y un w, tal que MA acepta w (ya que asumimos que representa un MT que acepta lambda y rechaza en qR caso contrario, por ende se detiene siempre, por ende pertenece a HP)

La parte importante viene si ambas cosas son válidas, en este caso busca en las tuplas de <M> el estado qR y lo reemplaza por un nuevo estado q. Luego agrega las quintuplas (q,x,qA,x,S) por cada símbolo x del alfabeto de la cinta de M. Así la M’ construida siempre termina en qA en caso de que termine.

Puede demostrarse fácilmente que:

1. f es computable?

Claramente f es computable pues Mf siempre se detiene pues la entrada es finita y luego de recorrerla agrega un numero finito de quintuplas y se detiene.

1. ¿∀(<M>,w) ∈ Σ\*, (<M>,w) ∈ HP⇔ (<M’>,w) ∈ Lu?
   1. (<M>,w) ∈ HP ⇒ f((<M>,w)) ∈ Lu.
      1. Si <M> no es un código válido de MT, por convención representa una MT que solo acepta lambda y siempre se detiene (osea que (<M>,w) ∈ HP), en este caso se deja un par termina en qA asi que (<M’>,w) ∈ Lu.
      2. (<M>,w) ∈ HP ⇒ M se detiene con entrada w.

⇒ M para en qA o en qR con input w.

⇒ M’ para en qA con input w. (por construcción)

⇒ (<M’>,w) ∈ Lu.

* 1. (<M>,w) ∉ HP ⇒ f((<M>,w)) ∉ Lu.
     1. Si (<M>,w) no es un par valido entonces de deja un par que termina en qR, así que (<M’>,w) ∉ Lu.
     2. (<M>,w) ∉ HP ⇒ M no se detiene con w   
         ⇒ M loopeara con w   
         ⇒ M’ rechazara a w loopeando (por construcción)  
         ⇒ (<M’>,w) ∉ Lu.

4) Sea HPλ el problema de detención a partir de la cinta en blanco

HP = {(<M>,w) / M se detiene con input w}

HPλ = {<M> / M se detiene con input λ }

Demuestre que existe una reducción HP α HPλ.

Vamos a demostrar que (HPα HPλ)

Para esto debe existir una funcion computable (o recursiva) f: Σ\*→ Σ\* tal que:

* f debe ser computable
* ∀(<M>,w) ∈ Σ\*, (<M>,w) ∈ HP⇔ <M’> ∈ HPλ.

f((<M>,w))=<M’>

Construiremos Mf para demostrar estos puntos

Mf trabaja de la siguiente manera:

Primero chequea que (<M>,w) sea un par válido de una MT, si no lo es, borra la cinta y escribe un código válido en la cinta en el que <M’> es la codificación de una MT Mloop y un w, tal que Mloop siempre loopea. Si era un par valido pero <M> no es un código válido entonces borramos w y dejamos en la cinta <M> (ya que asumimos que representa un MT que acepta λ y va a qR caso contrario, por ende se detiene siempre)

La parte importante viene si ambas cosas son válidas, en este caso construye <M’> escribiendo las quintuplas necesarias para que M’ borre la entrada y escriba w en la cinta, posicione el cabezal y simula M sobre w. Así M’ se detiene en qA o en qR ⇔ M se detiene en qA o en qR con input w. En esencia, obliga a M’ a terminar con λ o cualquier otra entrada si M termina con entrada w, en caso de que esto no ocurra entonces significa que M loopeo, nótese que en este caso nos dará igual si con λ M termina o no ya que si (<M>,w) ∉ HP ⇒<M’> ∉ HPλ .

Puede demostrarse fácilmente que:

1. f es computable?

Claramente si pues Mf se detiene luego de realizar una cantidad finita de acciones.

1. ¿∀(<M>,w) ∈ Σ\*, (<M>,w) ∈ HP⇔ <M’> ∈ HPλ?
2. (<M>,w) ∈ HP ⇒ f((<M>,w)) ∈ HPλ .
   * 1. Si <M> no es un código válido de MT, por convención representa una MT que rechaza cualquier entrada (osea que (<M>,w) ∈ HP), en este caso se deja solo el código de M ya que sabemos que M terminará con λ y por ende (<M’>,w) ∈ HPλ.
     2. (<M>,w) ∈ HP ⇒ M se detiene con entrada w.

⇒ M para en qA o en qR con input w.

⇒ M’ para en qA o en qR con cualquier input (por construcción)

⇒ <M’> ∈ HPλ.

b. (<M>,w) ∉ HP ⇒ f((<M>,w)) ∉ HPλ .

* + 1. Si (<M>,w) no es un par valido entonces <M’> = al código de una MT que siempre loopea, así que <M’> ∉ HPλ.
    2. (<M>,w) ∉ HP ⇒ M no se detiene con w   
        ⇒ M loopeara con w   
        ⇒ M’ rechazara cualquier input loopeando   
        ⇒ <M’> ∉ HPλ.

5) Demuestre que LV ∉ RE

LV={(<M>)/L(M) = ∅}.

Considere que si <M> es un código inválido de máquina de Turing también pertenece a LV (ya que no reconoce ningún string). Así LV es el complemento del lenguaje LNV={(<M>)/L(M)≠∅}

(Ayuda: puede utilizar el complemento de Lu para encontrar una reducción)

LV={(<M>)/L(M) = ∅}

LNV=LV={(<M>)/L(M) ≠ ∅}

Lu = {(<M>,w) / M acepta w}

Lnu =Lu= {(<M>,w) / M no acepta w}

Podemos demostrar que existe una reducción Lnu α LV y como Lnu ∉ RE ⇒ LV ∉ RE

Vamos a demostrar que (Lnu α LV)

Para esto debe existir una funcion computable (o recursiva) f: Σ\*→ Σ\* tal que:

* f debe ser computable
* ∀(<M>,w) ∈ Σ\*, (<M>,w) ∈ Lnu⇔ <M’> ∈ LV.

f((<M>,w))=<M’>

Construiremos Mf para demostrar estos puntos

Mf trabaja de la siguiente manera:

Primero chequea que (<M>,w) sea un par válido, si no lo es entonces (<M>,w) ∉ Lnu por ende deberemos dejar una salida en la cinta tal que <M’> ∉ Lv, para esto borraremos la cinta y escribiremos el código correspondiente a una MT que acepta todo.

Si es un par valido, chequeamos que <M> sea un código válido de MT, si no lo es, lo dejaremos en la cinta y borramos w todo lo que no pertenezca a este, ya que por convención asumimos que corresponde a una MT M’ tal que <M> no acepta ningún input ((<M>,w) ∈ Lnu) y por ende L(M’) = ∅ (<M’> ∈ LV).

La parte importante viene si ambas cosas son válidas, en este caso Mf construye <M’> escribiendo las quintuplas necesarias para que M’ borre la entrada y escriba w en la cinta, posicione el cabezal y simula M sobre w. Así M’ para en qR o loopea para cualquier input ⇔ M no acepta w.

Puede demostrarse fácilmente que:

1. f es computable?

Mf se detiene luego de realizar una cantidad finita de acciones

1. ¿∀(<M>,w) ∈ Σ\*, (<M>,w) ∈ Lnu ⇔ (<M’>,w) ∈ LV?
2. (<M>,w) ∈ Lnu ⇒ f((<M>,w)) ∈ LV.
   1. Si <M> no es un código válido de MT, por convención representa una MT que rechaza cualquier entrada (osea que (<M>,w) ∈ Lnu), en este caso se deja solo el código de M ya que sabemos que L(M)= ∅ (osea <M> ∈ LV).
   2. (<M>,w) ∈ Lnu ⇒ M rechaza w.

⇒ M para qR o loopea con input w.

⇒ M’ para en qR o loopea siempre con cualquier input

(por construcción)

⇒ <M’> ∈ LV.

b. (<M>,w) ∉ Lnu ⇒ f((<M>,w)) ∉ LV.

i. Si (<M>,w) no es un par valido entonces escribimos una MT

que acepta todo, por ende <M’> ∉ Lnu.

ii. (<M>,w) ∉ Lnu ⇒ M acepta w.

⇒ M para qA con input w.

⇒ M’ para en qA siempre con cualquier input

(por construcción)

⇒ <M’> ∉ LV.