Reducibilidad

- **Def.**: Sean $L_1, L_2 \subseteq \Sigma^*$ se dirá que L_1 se reduce a L_2 (L_1 α L_2) si existe una función total computable (o recursiva) $f: \Sigma^* \to \Sigma^*$ tal que

$$\forall w \in \Sigma^*, w \in L_1 \Leftrightarrow f(w) \in L_2$$

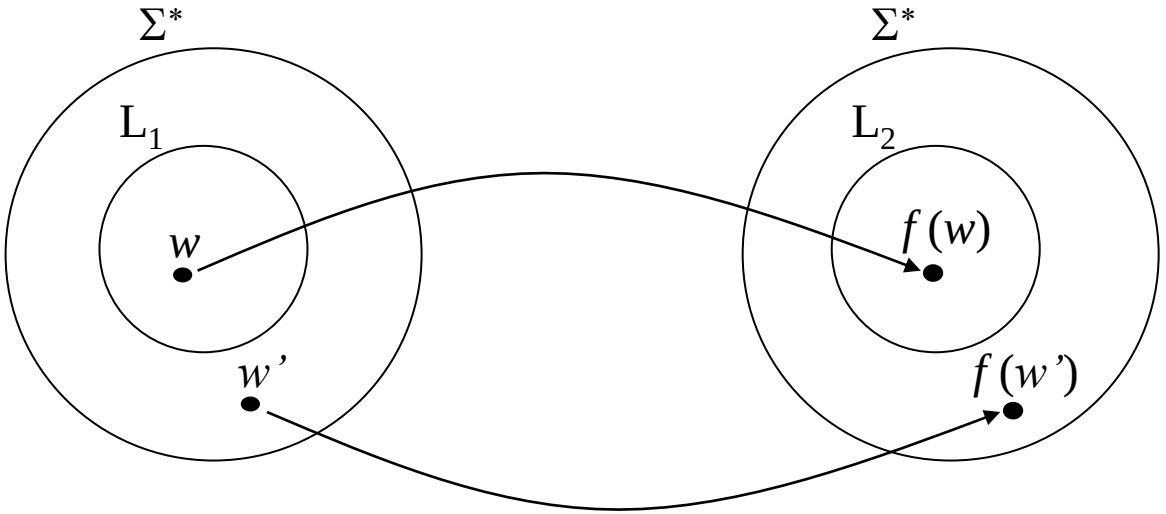
$$w \in L_1 \Rightarrow f(w) \in L_2 \quad \text{AND} \quad f(w) \in L_2 \Rightarrow w \in L_1$$

$$w \notin L_1 \Rightarrow f(w) \notin L_2$$

Reducibilidad

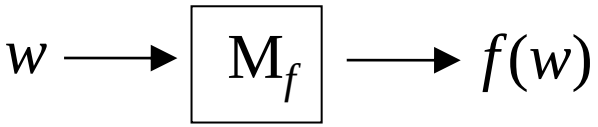
- **Def.**: Sean $L_1, L_2 \subseteq \Sigma^*$ se dirá que L_1 se reduce a L_2 ($L_1 \ \alpha \ L_2$) si existe una función total computable (o recursiva) $f: \Sigma^* \rightarrow \Sigma^*$ tal que

$$\forall w \in \Sigma^*, w \in L_1 \Leftrightarrow f(w) \in L_2$$



Reducibilidad

- Se dice que f es computable si existe una MT que la computa y que siempre se detiene. Notar la importancia de que f sea completa.



M_f nunca loopea. A veces, usaremos la expresión $M_f(w)$ para referirnos a la función computada por la MT M_f

Nota: la reducibilidad es una transformación de instancias de un problema en instancias de otro problema.

Reducibilidad

Ej.: $L_1 = \{w \in \{a, b\}^* / w \text{ comienza con } a\}$

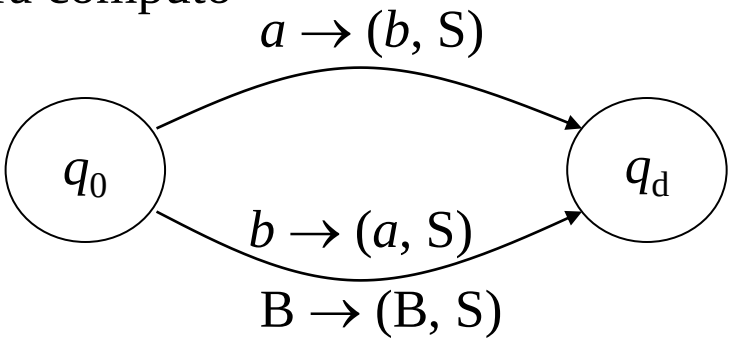
$L_2 = \{w \in \{a, b\}^* / w \text{ comienza con } b\}$

Vamos a demostrar que (L_1 α L_2)

$M_f = <Q, \Sigma, \Gamma, \delta, q_0, q_d>$ $Q = \{q_0\}$ $\Sigma = \{a, b\}$ $\Gamma = \{B, a, b\}$

M_f es una MT de/para cómputo

δ:



Puede demostrarse fácilmente que:

1. M_f siempre se detiene
2. $\forall w \in \Sigma^*, w \in L_1 \Leftrightarrow f(w) \in L_2$

Reducibilidad

Ejercicio 1:

$$L_1 = \{w \in \{0, 1\}^* / \text{cant}_1(w) \text{ es par}\}$$

$$L_2 = \{w \in \{0, 1\}^* / \text{cant}_1(w) \text{ es impar}\}$$

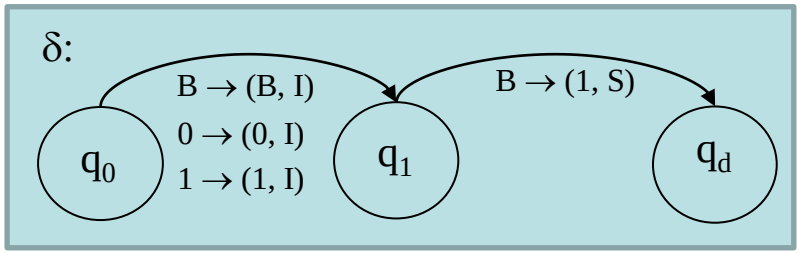
Donde $\text{cant}_1(w)$ es la cantidad de 1 que hay en w

Demostrar que $L_1 \alpha L_2$.

Reducibilidad

Se construye M_f, una MT que computa la función de reducibilidad.

$M_f = <\{q_0, q_1\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, q_d >$, M_f: MT de/para cómputo



Hay que demostrar que M_f siempre se detiene y que $w \in L_1 \Leftrightarrow M_f(w) \in L_2$

1) M_f siempre se detiene: claramente sí, pues para cualquier input solamente ejecuta dos pasos de computación y se detiene.

$L_1 = \{w \in \{0, 1\}^* / \text{cant}_1(w) \text{ es par}\}$

$L_2 = \{w \in \{0, 1\}^* / \text{cant}_1(w) \text{ es impar}\}$

Reducibilidad

2) ¿ $w \in L_1 \Leftrightarrow M_f(w) \in L_2$?

Claramente $\text{cant}_1(M_f(w)) = \text{cant}_1(w) + 1$, por lo tanto:

- si $\text{cant}_1(w)$ es par entonces $\text{cant}_1(M_f(w))$ es impar y
- si $\text{cant}_1(w)$ es impar entonces $\text{cant}_1(M_f(w))$ es par

Entonces:

a) Si $w \in L_1 \Rightarrow \text{cant}_1(w)$ es par $\Rightarrow \text{cant}_1(M_f(w))$ es impar $\Rightarrow M_f(w) \in L_2$

b) Si $w \notin L_1 \Rightarrow \text{cant}_1(w)$ es impar $\Rightarrow \text{cant}_1(M_f(w))$ es par $\Rightarrow M_f(w) \notin L_2$

De a) y b) se tiene que $w \in L_1 \Leftrightarrow M_f(w) \in L_2$.

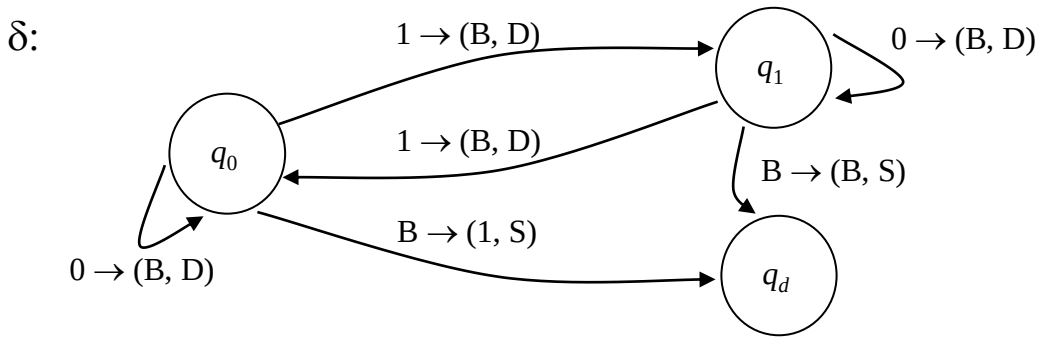
Reducibilidad

Ejercicio 2: Sean $L_1 = \{w \in \{0, 1\}^* / \text{cant}_1(w) \text{ es par}\}$

$L_2 = \{w \in \{0, 1\}^* / \text{cant}_1(w) = 1\}$

Demostrar que $L_1 \alpha L_2$.

Se construye M_f tal que $w \in L_1 \Leftrightarrow M_f(w) \in L_2$



Tarea para el lector: Demostrar que:

1) M_f se detiene para todo w

2) $w \in L_1 \Leftrightarrow M_f(w) \in L_2$.

Reducibilidad

Ejercicio 3: Sea L un lenguaje recursivo ($L \in R$)

L_1 = {<M> / <M> es un cód. válido de MT, L(M) = L y M siempre se detiene}

L_2 = {<M> / <M> es un cód. válido de MT y $L(M) = \bar{L}$}

Demostrar que $L_1 \alpha L_2$

Hay que encontrar una MT M_f que siempre se detenga para la cual sea cierto que:

$$w \in L_1 \Leftrightarrow M_f(w) \in L_2$$

Se construye M_f que trabaja de la siguiente manera: Si *w* no es un código válido de MT M_f se detiene sin hacer nada. De lo contrario (*w* es un código <M> de MT válido) recorre todas las quíntuplas de *w*, si en la 3ra posición encuentra q_A lo reemplaza por q_R, si encuentra q_R lo reemplaza por q_A.

Nuevamente hay que demostrar que:

1) M_f se detiene

2) $w \in L_1 \Leftrightarrow M_f(w) \in L_2$.

L_1 = {<M> / <M> es un cód. válido de MT, L(M) = L y M siempre se detiene}

L_2 = {<M> / <M> es un cód. válido de MT y L(M) = $\bar{L}$}

1) M_f siempre se detiene porque la entrada es finita.

2) ¿ $w \in L_1 \Leftrightarrow M_f(w) \in L_2$?

2.a) ¿ $w \in L_1 \Rightarrow M_f(w) \in L_2$?

Si $w \in L_1 \Rightarrow w$ es un código válido de una MT M que reconoce L y siempre se detiente $\Rightarrow$

- si $x \in L \Rightarrow$ M para en $q_A \Rightarrow M_f(w)$ codifica una MT que para en q_R con input x
- si $x \notin L \Rightarrow$ M para en $q_R \Rightarrow M_f(w)$ codifica una MT que para en q_A con input x

observar que M *nunca rechaza loopeando* (ver def. de L_1)

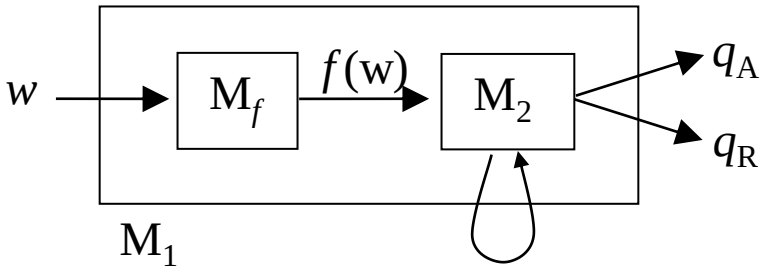
por lo tanto $M_f(w)$ acepta $\bar{L} \Rightarrow M_f(w) \in L_2$

2.b) Se demuestra similarmente que $w \notin L_1 \Rightarrow M_f(w) \notin L_2$. Además, considérese que si w es un código inválido no pertenece a L_1 y $M_f(w)$ tampoco pertenece a L_2

Por lo tanto $w \in L_1 \Leftrightarrow M_f(w) \in L_2$.

Reducibilidad

Nota: si $L_1 \alpha L_2$ entonces se puede construir una MT que acepte L_1 a partir de la MT que acepta L_2 (si existe).
Debe quedar claro que "en cierto sentido" L_1 no puede ser más difícil computacionalmente que L_2 porque se puede utilizar L_2 para resolver L_1.



Intuitiva y coloquialmente, la reducción establece una relación de "≤" grado de dificultad computacional entre lenguajes/problemas…

Reducibilidad

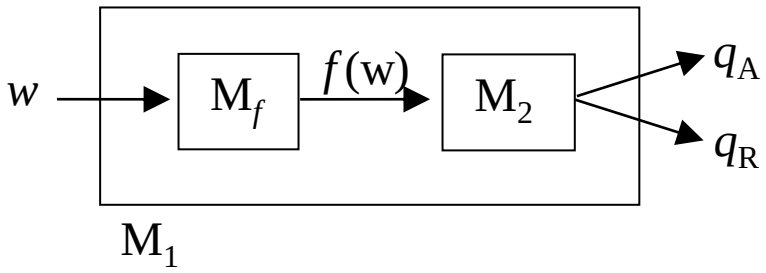
Teorema 1: $L_1, L_2 \subseteq \Sigma^*$ y existe la reducción $L_1 \alpha L_2$, entonces:

$$\text{si } L_2 \in R \Rightarrow L_1 \in R$$

Dem.: $L_2 \in R \Rightarrow$ existe una máquina de Turing M_2 tal que $L_2 = L(M_2)$ y M_2 siempre se detiene. Se construye M_1 que hace:

1) Simula M_f sobre w y obtiene $f(w)$

2) Simula M_2 sobre $f(w)$ y acepta sii M_2 acepta

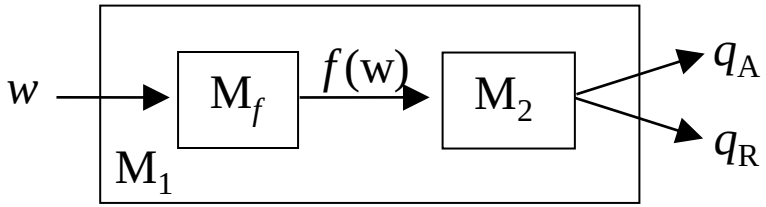


Reducibilidad

Se debe demostrar que:

a) $L_1 = L(M_1)$

b) M_1 siempre de detiene.



a.1) $w \in L_1 \Rightarrow f(w) \in L_2$ (porque L_1 α L_2)

$\Rightarrow M_2$ para en q_A con input $f(w)$

$\Rightarrow M_1$ para en q_A con input w

$\Rightarrow w \in L(M_1)$

a.2) $w \notin L_1 \Rightarrow f(w) \notin L_2$ (porque L_1 α L_2)

$\Rightarrow M_2$ para en q_R con input $f(w)$

$\Rightarrow M_1$ para en q_R con input w

$\Rightarrow w \notin L(M_1)$

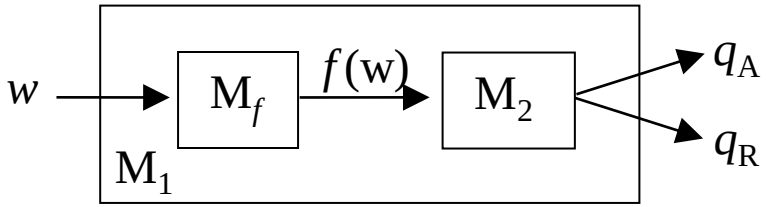
Por a.1) y a.2) se tiene que $L_1 = L(M_1)$ ✓

Reducibilidad

Se debe demostrar que:

a) $L_1 = L(M_1)$ ✓

b) M_1 siempre de detiene.



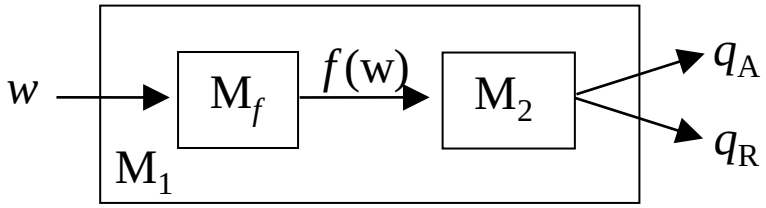
Claramente M_1 siempre de detiene pues M_f y M_2 se detienen siempre por hipótesis. ✓

Reducibilidad

Se debe demostrar que:

a) $L_1 = L(M_1)$

b) M_1 siempre de detiene.



Por lo tanto, de a) y b) se tiene que $\mathbf{L_1 \in R}$.

Reducibilidad

Teorema 2: $L_1, L_2 \subseteq \Sigma^*$ y existe la reducción $L_1 \alpha L_2$, entonces:

$$L_2 \in RE \Rightarrow L_1 \in RE$$

Dem.: Se demuestra de manera similar a la demostración del teorema anterior

Corolario: Sean $L_1, L_2 \subseteq \sum^*$ y la reducción $L_1 \alpha L_2$, por las respectivas contrarecíprocas del teorema 1 y del teorema 2, se cumple que:

$$L_1 \notin R \Rightarrow L_2 \notin R$$

$$L_1 \notin RE \Rightarrow L_2 \notin RE$$

Halting Problem

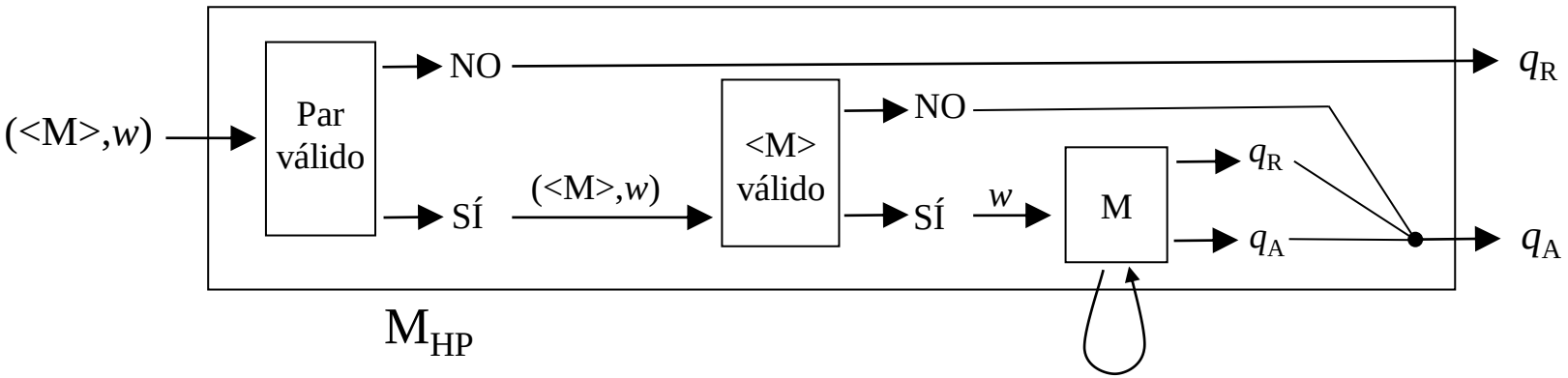
El Lenguaje *Halting Problem* se define como:

$$HP = \{(<M>, w) \text{ tal que M se detiene con input } w\}$$

Ejercicio: demostrar que HP $\in$ (RE-R).

I) HP $\in$ RE.

Se construye una MT M_{HP} tal que $L(M_{HP}) = HP$. M_{HP} chequea sintácticamente la entrada (<M>, *w*), si es un par inválido para en q_R, si es un par válido pero <M> es un código de MT inválido para en q_A, si ambos son válidos simula M sobre *w*, si M para (en q_A o en q_R) M_{HP} para en q_A, si M no para, M_{HP} no para.



Halting Problem

II) HP $\notin$ R.

Se puede probar que existe la reducción L_u α HP, y como $L_u \notin$ R será cierto HP $\notin$ R.

Dem. Sea la siguiente MT M_f que computa la función *f* de reducibilidad

$$M_f((<M>,w)) = (<M'>,w)$$

M_f trabaja de la siguiente manera:

Si (<M>,*w*) no es un par válido o <M> no es un código válido de MT borra la cinta (deja λ como salida), de lo contrario busca en las quíntuplas de <M> el estado q_R y lo reemplaza por un nuevo estado *q*. Luego agrega las quíntuplas (q, x, q, x, S) por cada símbolo *x* del alfabeto de la cinta de M. Así la máquina M' construida por M_f entra en loop cuando M para en q_R.

Halting Problem

Hay que demostrar que M_f es una máquina que computa la función de reducibilidad, es decir se debe probar que:

1) *f* es computable ?

2) $(<M>,w) \in L_u \Leftrightarrow (<M'>,w) \in HP$?

Claramente *f* es computable pues M_f siempre se detiene pues la entrada es finita y luego de recorrerla agrega un número finito de quíntuplas y se detiene 

Halting Problem

Hay que demostrar que M_f es una máquina que computa la función de reducibilidad, es decir se debe probar que:

1) *f* es computable ✓

2) $(<M>,w) \in L_u \Leftrightarrow (<M'>,w) \in HP$?

2.a) $(<M>,w) \in L_u \Rightarrow (<M'>,w) \in HP$?

2.b) $(<M>,w) \notin L_u \Rightarrow (<M'>,w) \notin HP$?

$(<M>,w) \in L_u \Rightarrow$ M acepta *w*

$\Rightarrow$ M para en q_A

$\Rightarrow$ M' para en q_A (por construcción)

$\Rightarrow$ M' se detiene con input *w*

$\Rightarrow (<M'>,w) \in HP$ ✓

Halting Problem

Hay que demostrar que M_f es una máquina que computa la función de reducibilidad, es decir se debe probar que:

1) *f* es computable ✓

2) $(<M>,w) \in L_u \Leftrightarrow (<M'>,w) \in HP$?

2.a) $(<M>,w) \in L_u \Rightarrow (<M'>,w) \in HP$? ✓

2.b) $(<M>,w) \notin L_u \Rightarrow (<M'>,w) \notin HP$?

i) Si $(<M>,w)$ no es un par válido o $<M>$ no es un cód. válido de máquina de Turing $\Rightarrow (<M'>,w) = \lambda \Rightarrow (<M'>,w) \notin HP$ ✓

ii) M rechaza *w* $\Rightarrow$ M *loopea* o para en q_R con input *w* $\Rightarrow$ M' *loopea* con input *w* $\Rightarrow (<M'>,w) \notin HP$ ✓

Halting Problem

Hay que demostrar que M_f es una máquina que computa la función de reducibilidad, es decir se debe probar que:

1) *f* es computable ✓

2) $(<M>,w) \in L_u \Leftrightarrow (<M'>,w) \in HP$?

2.a) $(<M>,w) \in L_u \Rightarrow (<M'>,w) \in HP$? ✓

2.b) $(<M>,w) \notin L_u \Rightarrow (<M'>,w) \notin HP$? ✓

Halting Problem

Hay que demostrar que M_f es una máquina que computa la función de reducibilidad, es decir se debe probar que:

1) *f* es computable ✓

2) $(<M>,w) \in L_u \Leftrightarrow (<M'>,w) \in HP$ ✓

De 1) y 2) se tiene que M_f computa la función de reducibilidad buscada y por lo tanto queda demostrado que $L_u \alpha HP$.

Como $L_u \notin R \Rightarrow HP \notin R$

De I) y II) $HP \in (RE - R)$

Lenguaje L_{Σ^*}

Ejercicio: Probar que $L_{\Sigma^*} = \{<M> / L(M) = \Sigma^*\}$ no es recursivo.

Mostraremos que existe una reducción $L_u \ \alpha \ L_{\Sigma^*}$

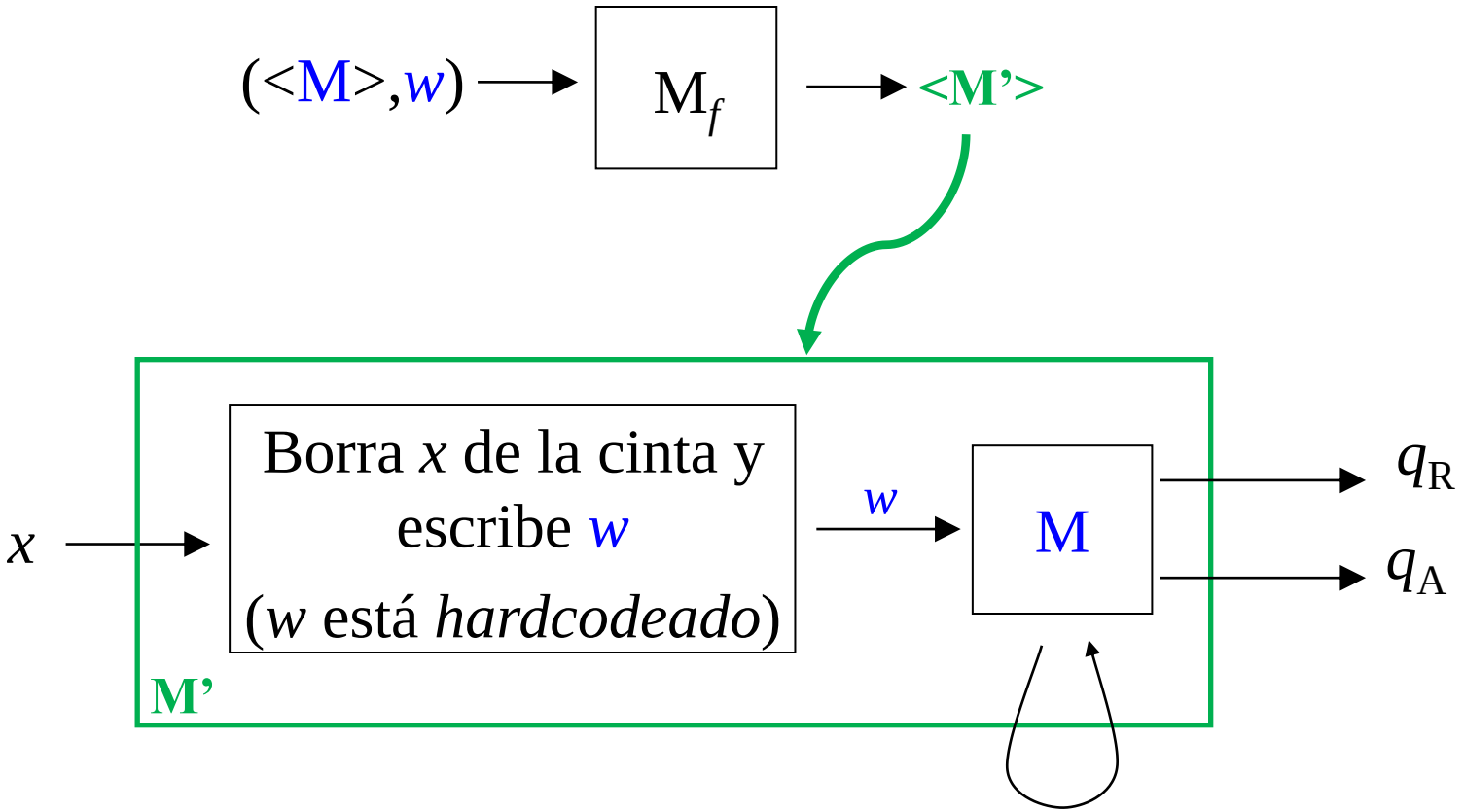
Hay que encontrar una función total computable tal que

$$(<M>,w) \in L_u \ \Leftrightarrow f\ (<M>,w) \in L_{\Sigma^*}$$

Sea M_f la máquina de Turing que computa la función f (<M>,w) = <M'> y trabaja de la siguiente manera:

Si (<M>,w) no es un par válido o <M> no es un código de MT válido, M_f borra la cinta (deja λ como salida), de lo contrario M_f construye <M'> escribiendo las quíntuplas necesarias para que M' borre la entrada y escriba w en la cinta, posicione el cabezal y simule M sobre w. Así M' para en q_A para cualquier input $\Leftrightarrow$ M acepta w

Lenguaje L_{Σ^*}



Para cualquier x de Σ^* la máquina M' ejecuta M sobre w

Lenguaje L_{Σ^*}

1) f ((<M>,w)) es computable? Claramente sí, pues M_f se detiene luego de realizar una cantidad finita de acciones.

2) ¿(<M>,w)$\in L_u \Leftrightarrow$ <M'> $\in L_{\Sigma^*}$?

a) Sea (<M>,w)$\in L_u \Rightarrow$ M para en q_A con entrada $w \Rightarrow$M' para en q_A con cualquier entrada $\Rightarrow$<M'> $\in L_{\Sigma^*}$

b) Sea (<M>,w) $\notin L_u \Rightarrow$

i. Si (<M>,w) no es un par válido o <M> no es un código de MT válido $\Rightarrow$ <M'> = $\lambda \Rightarrow$ <M'> $\notin L_{\Sigma^*}$

ii. M rechaza $w \Rightarrow$ M' rechaza toda entrada $\Rightarrow$<M'> $\notin L_{\Sigma^*}$

De a) y b) se tiene que (<M>,w)$\in L_u \Leftrightarrow$ <M'> $\in L_{\Sigma^*}$

De 1) y 2) se tiene que $L_u \alpha L_{\Sigma^*}$

Por lo tanto $L_{\Sigma^*} \notin$ a R (porque $L_u \notin$ a R)

Lenguaje L_{Σ^*}

Nota: Puede demostrarse también que existe una reducción $\bar{L}_u \alpha L_{\Sigma^*}$ y como $\bar{L}_u \notin RE$ se tiene que $L_{\Sigma^*} \notin$ a RE

Para practicar. Demostrar que existe una reducción de $\bar{L}_u \alpha L_{\varnothing}$ con $L_{\varnothing} = \{<M> / L(M) = \varnothing\}$

Lenguaje L_{EQ}

Teorema. $L_{EQ} = \{(<M_1>,<M_2>) / L(M_1)=L(M_2)\} \notin RE$

Prueba: $L_{\Sigma*} \ \alpha \ L_{EQ}$

La función *f* de reducibilidad que computa M_f es

f (<M>)=(<M>,$<M_{\Sigma*}>$) Siendo $<M_{\Sigma*}>$ el código de una máquina de Turing que acepta $L_{\Sigma*}$

Por ejemplo si $\Sigma=\{a,b\}$, la δ de transición de $M_{\Sigma*}$ puede ser la siguiente:

$\delta(q_0,a)=(q_A,a,S)$; $\delta(q_0,b)=(q_A,b,S)$; $\delta(q_0,B)=(q_A,B,S)$

Claramente M_f se detiene, por lo tanto *f* es computable

Además:

$<M> \in L_{\Sigma*} \Leftrightarrow L(M) = \Sigma* \Leftrightarrow L(M) = L(M_{\Sigma*}) \Leftrightarrow (<M>,<M_{\Sigma*}>) \in L_{EQ}$

Lenguaje L_{EQ}

Teorema. $L_{EQ} = \{(<M_1>,<M_2>) / L(M_1)=L(M_2)\} \notin RE$

Prueba: $L_{\Sigma^*} \ \alpha \ L_{EQ}$

La función *f* de reducibilidad que computa M_f es

f (<M>)=(<M>,$<M_{\Sigma^*}>$) Siendo $<M_{\Sigma^*}>$ el código de una máquina de Turing que acepta Σ*

Por ejemplo si Σ={a,b}, la δ de transición de M_{Σ^*} puede ser la siguiente:

$\delta(q_0,a)=(q_A,a,S)$; $\delta(q_0,b)=(q_A,b,S)$; $\delta(q_0,B)=(q_A,B,S)$

Claramente M_f se detiene, por lo tanto *f* es computable

Además:

$<M> \in L_{\Sigma^*} \Leftrightarrow L(M) = \Sigma^* \Leftrightarrow L(M) = L(M_{\Sigma^*}) \Leftrightarrow (<M>,<M_{\Sigma^*}>) \in L_{EQ}$

Por lo tanto, *f* es una función de reducibilidad y queda probado que $L_{\Sigma^*} \ \alpha \ L_{EQ}$

Entonces $L_{EQ} \notin RE$