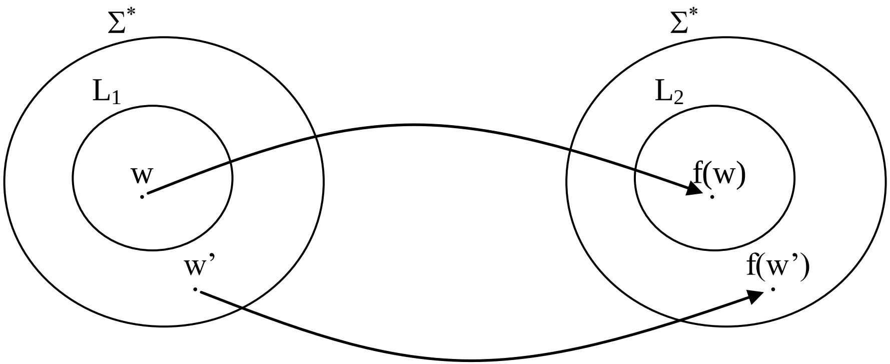
Reducibilidad

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



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

$$w \longrightarrow \boxed{M_f} \longrightarrow f(w)$$

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

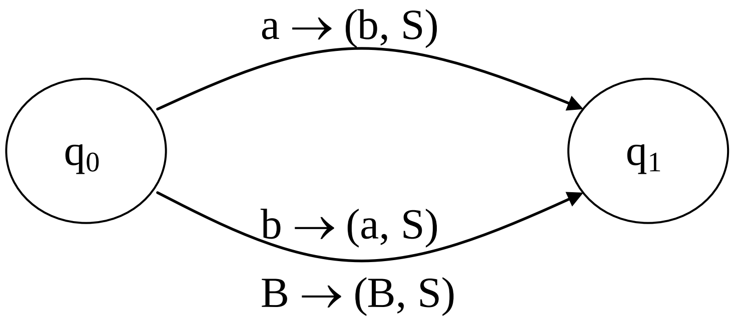
Nota: la reducibilidad es una transformación de instancias de un problema en instancias de otro problema.

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

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

$M_f = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$; $Q = \{q_0, q_1\}$; $\Sigma = \{a, b\}$; $\Gamma = \{B, a, b\}$; $F = \varnothing$

δ:



Puede demostrarse fácilmente que M_f siempre se detiene y que

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

Ejercicio 1: $\Sigma = \{0, 1\}$

$L_1 = \{w \in \Sigma^* \text{ tq } cant_1(w) \text{ es par}\}$

$L_2 = \{w \in \Sigma^* \text{ tq } cant_1(w) \text{ es impar}\}$

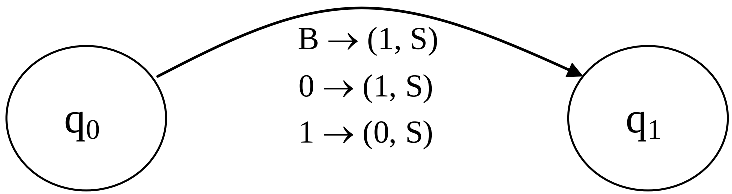
Donde $cant_1(w)$ es la cantidad de 1 que hay en w

Demostrar que $L_1 \alpha L_2$.

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

$M_f = <\{q_0, q_1\}, \{a, b\}, \{a, b, B\}, \delta, q_0, \varnothing>$

δ:



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 solamente ejecuta un movimiento.

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

Claramente si w comienza con 1 entonces $cant_1(M_f(w)) = cant_1(w) - 1$, y si w no comienza con 1 $cant_1(M_f(w)) = cant_1(w) + 1$. Por lo tanto

a) Si $w \in L_1 \Rightarrow cant_1(w)$ es par $\Rightarrow cant_1(M_f(w))$ es impar $\Rightarrow M_f(w) \in L_2$

b) Si $w \notin L_1 \Rightarrow cant_1(w)$ es impar $\Rightarrow cant_1(M_f(w))$ es par $\Rightarrow M_f(w) \notin L_2$

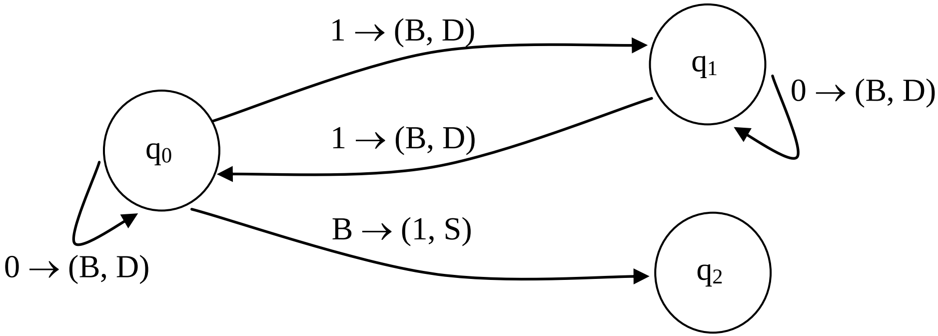
De a) y b) se tiene que $w \in L_1 \Leftrightarrow M_f(w) \in L_2$.

Ejercicio 2: Sean $L_1 = \{w \in \{0, 1\}^* \text{ tq } cant_1(w) \text{ es par}\}$

$L_2 = \{w \in \{0, 1\}^* \text{ tq } cant_1(w) = 1\}$

Demostrar que $L_1 \; \alpha \; L_2$. Es decir construir $M_f(w)$ tal que

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



Habría que demostrar que M_f se detiene y que $w \in L_1 \Leftrightarrow M_f(w) \in L_2$.

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

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

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

Demostrar que $L_1 \; \alpha \; L_2$

Para demostrar que $L_1 \; \alpha \; L_2$ hay que encontrar una MT M_f que siempre se detenga para la cual sea cierto que $\langle M \rangle \in L_1 \Leftrightarrow M_f(\langle M \rangle) \in L_2$

Se construye M_f que trabaja de la siguiente manera: Si <M> no es un código válido de MT M_f se detiene sin hacer nada. De lo contrario recorre todas las quíntuplas de <M>, 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 M_f se detiene y que:

$$\langle M \rangle \in L_1 \Leftrightarrow M_f(\langle M \rangle) \in L_2.$$

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

2) $<M> \in L_1 \Leftrightarrow M_f(<M>) \in L_2$.

a) $<M> \in L_1 \Rightarrow M_f(<M>) \in L_2$? Si $<M> \in L_1 \Rightarrow L(M) = L \Rightarrow$

$$\begin{cases} \text{si } w \in L \Rightarrow M \text{ para en } q_A \Rightarrow M_f(<M>) \text{ para en } q_R \text{ para la misma entrada} \\ \text{si } w \notin L \Rightarrow M \text{ para en } q_R \Rightarrow M_f(<M>) \text{ para en } q_A \text{ para la misma entrada} \end{cases}$$

por lo tanto $M_f(<M>)$ acepta $\overline{L} \Rightarrow M_f(<M>) \in L_2$.

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

Por lo tanto $<M> \in L_1 \Leftrightarrow M_f(<M>) \in L_2$.

Nota: si $L_1 \alpha L_2$ significa que se puede construir una MT que acepte L_1 a partir de la MT que acepta L_2.

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.