Ejemplo. Reducción de L_U a HP

Definición de la función de reducción.

Para pares válidos (<M>, w), se define

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

tal que M' se comporta como M, salvo que cuando M se detiene en q_R, M' no se detiene.

La función f es total computable.

Si la entrada no es una cadena válida (<M>, w), la MT M_f genera la cadena 1. En caso contrario, para generar <M'> la MT M_f modifica <M> de modo tal que M' entre en un *loop* cuando M se detiene en q_R: por ejemplo, como ya se vio en el Teorema 3.1, se puede reemplazar q_R por un estado nuevo q y definir una 5-tupla (q, x, q, x, S) por cada símbolo x del alfabeto de M.

Se cumple $(<M>, w) \in L_U \leftrightarrow f((<M>, w)) \in HP$.

$(<M>, w) \in L_U \leftrightarrow$ M acepta w $\leftrightarrow$ M' se detiene a partir de w $\leftrightarrow (<M'>, w) \in HP$.

Fin de Ejemplo

Ejemplo. Reducción de L_U^C a L_{Σ^*}

Se probará que $L_{\Sigma^*} = \{<M> \mid L(M) = \Sigma^*\} \notin RE$. Se hará $L_U^C \alpha L_{\Sigma^*}$. Como $L_U \in RE - R$, entonces $L_U^C \notin RE$, y así, con la reducción propuesta se probará que $L_{\Sigma^*} \notin RE$ (si L_{Σ^*} fuera recursivamente numerable también lo sería el lenguaje L_U^C).

Definición de la función de reducción.
Para pares válidos (<M>, w) se define

$$f((<M>, w)) = <M_w>$$

donde M_w es una MT que a partir de una entrada v simula a lo sumo |v| pasos de M a partir de w (M podría detenerse antes), y acepta si y sólo si M no acepta w. Se comprueba fácilmente que:

- Si M no acepta w, entonces $L(M_w) = \Sigma^*$
- Si M acepta w, digamos en k pasos, entonces $L(M_w) = \{v \mid |v| < k\}$ (es decir que $L(M_w) \neq \Sigma^*$, que es lo que se necesita)

La función f es total computable.
Si la entrada no es una cadena válida (<M>, w), y establecemos por convención que la misma pertenece a L_U^C, entonces M_f genera un código $<M_{\Sigma^*}>$ tal que $LM_{\Sigma^*}) = \Sigma^*$. En caso contrario, para generar $<M_w>$, la MT M_f le agrega al código <M> un fragmento que calcula el tamaño i de la entrada, decrementa i en 1 toda vez que se ejecuta un paso, detiene la ejecución cuando i = 0, y acepta si y sólo si no se alcanza al final el estado q_A.

Se cumple $(<M>, w) \in L_U^C \leftrightarrow f((<M>, w)) \in L_{\Sigma^*}$.
$(<M>, w) \in L_U^C \leftrightarrow$ M no acepta w $\leftrightarrow L(M_w) = \Sigma^* \leftrightarrow <M_w> \in L_{\Sigma^*}$.

Fin de Ejemplo

Ejemplo. Reducciones de L_U^C a L_{REC} y L_{REC}^C

Probaremos que es indecidible determinar si el lenguaje aceptado por una MT es recursivo. El lenguaje que representa el problema es $L_{REC} = \{<M> \mid L(M) \in R\}$. Demostraremos directamente que L_{REC} no es recursivamente numerable. Se hará $L_U^C \alpha L_{REC}$. Como $L_U^C \notin RE$, entonces $L_{REC} \notin RE$.

Definición de la función de reducción.
Para pares válidos (<M>, w) se define

$$f((<M>, w)) = <M_w>$$

donde M_w es una MT que, a partir de una entrada v:

1. Simula M a partir de w.
2. Si M no acepta, entonces M_w no acepta.
3. Si M_U es una MT que acepta L_U, entonces M_w simula M_U a partir de v, y acepta si y sólo si M_U acepta.

Se comprueba fácilmente que:

- Si M no acepta w, entonces $L(M_w) = \varnothing$ (que es un lenguaje recursivo).
- Si M acepta w, entonces $L(M_w) = L_U$ (que no es un lenguaje recursivo).

La función f es total computable.
Si la entrada no es una cadena válida (<M>, w), la MT M_f genera un código $<M_\varnothing>$, con $L(<M_\varnothing>) = \varnothing$. En caso contrario genera $<M_w>$, básicamente agregándole a <M> un código $<M_U>$ y un fragmento que primero reemplaza la entrada v por la cadena w y luego la restituye para continuar la ejecución.

Se cumple $(<M>, w) \in L_U^C \leftrightarrow f((<M>, w)) \in L_{REC}$.
$(<M>, w) \in L_U^C \leftrightarrow$ M no acepta w $\leftrightarrow L(M_w) = \varnothing \leftrightarrow <M_w> \in L_{REC}$.

También se prueba que $L_{REC}{}^{C}$ no es recursivamente numerable. Por lo tanto, al igual que $L_{\Sigma*}$, el lenguaje L_{REC} habita la clase más difícil de la jerarquía de la computabilidad. Se hará $L_U{}^{C}$ α $L_{REC}{}^{C}$. Como $L_U{}^{C} \notin RE$, entonces probaremos que $L_{REC}{}^{C} \notin RE$.

Definición de la función de reducción.
Para pares válidos (<M>, w) se define

$$f((<M>, w)) = <M_w>$$

donde M_w es una MT que, a partir de una entrada v, simula en paralelo M a partir de w y M_U a partir de v, aceptando si y sólo si alguna de las dos MT acepta. Se comprueba fácilmente que:

- Si M no acepta w, entonces $L(M_w) = L_U$ (que no es un lenguaje recursivo).
- Si M acepta w, entonces $L(M_w) = \Sigma^*$ (que es un lenguaje recursivo).

La función f es total computable.
Si la entrada no es una cadena válida (<M>, w), la MT M_f genera un código $<M_U>$, que pertenece a $L_{REC}{}^{C}$. En caso contrario, genera $<M_w>$ básicamente agregándole a <M>, un código $<M_U>$ más un fragmento que permite ejecutar en paralelo M a partir de w y M_U a partir de v.

Se cumple $(<M>, w) \in L_U{}^{C} \leftrightarrow f((<M>, w)) \in L_{REC}{}^{C}$.
$(<M>, w) \in L_U{}^{C} \leftrightarrow$ M no acepta w $\leftrightarrow L(M_w) = L_U \leftrightarrow <M_w> \in L_{REC}{}^{C}$.

Fin de Ejemplo

Ejemplo. Reducción de PCP a VAL

Sea VAL = {φ | φ es una fórmula válida de la lógica de primer orden}. Vamos a construir una reducción de PCP a VAL. Como PCP no es recursivo, entonces demostraremos que VAL tampoco lo es. Dada una secuencia S de pares de cadenas de unos y ceros no vacías (s_1, t_1), …, (s_k, t_k), la función de reducción le asignará a S una fórmula φ de la lógica de primer orden válida si y sólo si S tiene solución, en el sentido del problema de correspondencia de Post. Como símbolos de función utilizamos *e*, de aridad 0 (es una constante), y f_0 y f_1, de aridad 1. La idea es que e represente la cadena vacía, y f_0 y f_1 la concatenación, al final de una cadena, del dígito 0 o 1, respectivamente. De este modo, la cadena $b_1…b_m$ de dígitos binarios b_i se puede representar por el término $f_{bm}(…(f_{b1}(e))…)$, que para facilitar la notación lo abreviaremos con $f_{b1…bm}(e)$. Finalmente, como símbolo de predicado utilizamos P, de aridad 2; el significado entendido para P(s, t) es que existe una secuencia de índices i_1, …, i_n, tal que el término s representa una cadena con subcadenas s_i de unos y ceros de la forma $s_{i1}…s_{in}$, y el término t representa una cadena con subcadenas t_i de unos y ceros de la forma $t_{i1}…t_{in}$.

Definición de la función de reducción.

La fórmula que se asigna, por la función de reducción f, a una secuencia S (sintácticamente correcta) de pares (s_1, t_1), …, (s_k, t_k), es

$\varphi = \varphi_1 \wedge \varphi_2 \rightarrow \varphi_3$, con:

$\varphi_1 = \bigwedge_{i=1,k} P(f_{si}(e), f_{ti}(e))$

$\varphi_2 = \forall v \forall w: P(v, w) \rightarrow \bigwedge_{i=1,k} P(f_{si}(v), f_{ti}(w))$

$\varphi_3 = \exists z: P(z, z)$

La función f es total computable.

Claramente, existe una MT M_f que dada una secuencia S sintácticamente correcta genera la fórmula φ descripta previamente (en otro caso M_f genera la cadena 1).

Se cumple $S \in PCP \leftrightarrow f(S) \in VAL$.

a. Supongamos primero que φ es válida, lo que se denota con $\models \varphi$. Vamos a probar que S tiene solución. Más específicamente, vamos a encontrar un modelo M_0 para φ que establezca la existencia de una secuencia de índices que soluciona S. El dominio de M_0 contiene todas las cadenas finitas de unos y ceros, incluyendo la cadena vacía λ. La interpretación de *e* es λ, lo que se denota con $e^{M0} = \lambda$. La

interpretación de f_0 es la concatenación de un 0 al final de una cadena, lo que se denota con $f_0^{M0}(s) = s0$. De la misma manera se define $f_1^{M0}(s) = s1$. Finalmente, la interpretación de P es la siguiente: se cumple $P^{M0}(s, t)$ cuando existe una secuencia de índices $i_1, \ldots, i_n$, tal que $s = s_{i1}\ldots s_{in}$, $t = t_{i1}\ldots t_{in}$, y s_i y t_i son cadenas de unos y ceros de S. Como vale $\models \varphi$, se cumple en particular $M_0 \models \varphi$.

Claramente vale $M_0 \models \varphi_1$, porque se cumple $P^{M0}(s_i, t_i)$ para $i = 1, \ldots, k$. Veamos que también vale $M_0 \models \varphi_2$, que establece que cuando el par (s, t) está en P^{M0}, también lo está el par (ss_i, tt_i), para $i = 1, \ldots, k$. Si $(s, t) \in P^{M0}$, entonces existe una secuencia de índices $i_1, \ldots, i_n$, tal que $s = s_{i1}\ldots s_{in}$ y $t = t_{i1}\ldots t_{in}$. Definiendo una nueva secuencia de índices $i_1, \ldots, i_n, i$, vale $ss_i = s_{i1}\ldots s_{in}s_i$ y $tt_i = t_{i1}\ldots t_{in}t_i$, por lo que se cumple $M_0 \models \varphi_2$. De esta manera, como se cumple $M_0 \models \varphi_1 \wedge \varphi_2 \rightarrow \varphi_3$ por hipótesis y hemos demostrado recién $M_0 \models \varphi_1 \wedge \varphi_2$, vale $M_0 \models \varphi_3$. Finalmente, por la definición de φ_3 y P^{M0}, existe una solución para S.

b. Supongamos ahora que S tiene solución, digamos la secuencia de índices $i_1, \ldots, i_n$. Vamos a probar que cualquiera sea el modelo M, con una constante e^M, dos funciones unarias f_0^M y f_1^M, y un predicado binario P^M, entonces M satisface φ, es decir $M \models \varphi$. Dada la fórmula $\varphi = \varphi_1 \wedge \varphi_2 \rightarrow \varphi_3$, vamos a asumir $M \models \varphi_1 \wedge \varphi_2$ y demostraremos $M \models \varphi_3$. Para la interpretación de las cadenas finitas de unos y ceros en el dominio de M definimos inductivamente una función denominada *interpret*, de la siguiente manera: $interpret(\lambda) = e^M$, $interpret(s0) = f_0^M(interpret(s))$, e $interpret(s1) = f_1^M(interpret(s))$. Por ejemplo, a la cadena 110 se le asigna $f_0^M(f_1^M(f_1^M(e^M)))$. Más genéricamente, si una cadena s de dígitos binarios b_i tiene la forma $b_1\ldots b_m$, entonces $interpret(b_1\ldots b_m) = f_{bm}^M(\ldots(f_{b1}^M(e^M))\ldots)$, abreviado con $f_s^M(e)$ como indicamos antes. Entonces, como vale $M \models \varphi_1$, se cumple $(interpret(s_i), interpret(t_i)) \in P^M$, para $i = 1, \ldots, k$. Como también vale $M \models \varphi_2$, entonces para todo par $(s, t) \in P^M$ se cumple $(interpret(ss_i), interpret(tt_i)) \in P^M$, para $i = 1, \ldots, k$. De esta manera, comenzando con $(s, t) = (s_{i1}, t_i)$, si se considera repetidamente la última observación se obtiene $(interpret(s_{i1}\ldots s_{in}), interpret(t_{i1}\ldots t_{in})) \in P^M$, y dado que las cadenas $s_{i1}\ldots s_{in}$ y $t_{i1}\ldots t_{in}$ son iguales porque $i_1, \ldots, i_n$ es una solución de S, entonces $interpret(s_{i1}\ldots s_{in}) = interpret(t_{i1}\ldots t_{in})$. De este modo se cumple la fórmula $\exists z$: P(z, z), y así $M \models \varphi_3$.

Fin de ejemplo

Ejemplo. L_{imp0} no es recursivo

Sea L_{imp0} = {<M> | M es una MT que a partir de toda entrada escribe alguna vez el símbolo 0}. Como los códigos <M> no se definen en términos de L(M), tampoco en este caso puede aplicarse el Teorema de Rice. Probaremos que $L_{imp0} \notin R$, mediante una reducción de problemas de HP a L_{imp0}.

Definición de la función de reducción.
Para pares válidos (<M>, w) se define

$$f((<M>, w)) = <M_w>$$

tal que $<M_w>$ tiene las siguientes características:

- Un primer fragmento borra la entrada y la reemplaza por la cadena w', tal que w' es como w salvo que en lugar del símbolo 0 tiene un símbolo x que no existe en el alfabeto de M.
- Un segundo fragmento tiene las mismas tuplas que M, salvo que en lugar del símbolo 0 utiliza el símbolo x, y en lugar de los estados finales q_A y q_R utiliza, respectivamente, nuevos estados no finales q_A' y q_R'.
- Un último fragmento tiene nuevas tuplas definidas a partir de los estados q_A' y q_R' y todos los símbolos z del alfabeto de M_w menos el símbolo 0, de la forma $\delta(q_A', z) = (q_A, 0, S)$ y $\delta(q_R', z) = (q_A, 0, S)$.

La idea es que M_w replique los pasos de M sin escribir nunca el símbolo 0, y sólo en el caso de que M se detenga, M_w haga un paso más escribiendo el 0.

La función f es total computable.
La función de reducción es claramente total computable (si la entrada no es una cadena válida (<M>, w), la MT M_f genera la cadena 1).

Se cumple (<M> , w) $\in$ HP $\leftrightarrow$ $<M_w> \in L_{imp0}$.
(<M>, w) $\in$ HP $\leftrightarrow$ M se detiene a partir de w $\leftrightarrow$ M_w a partir de toda entrada escribe el símbolo 0 $\leftrightarrow$ $<M_w> \in L_{imp0}$.

Fin de Ejemplo