

Práctica 4 FTC Las reducciones

Ejercicio 1. Considerando la reducción de HP a LU descripta en clase, responder:

a. Explicar por qué la función identidad, es decir la función que a toda cadena le asigna la misma cadena, no es una reducción de HP a LU.

Uno no puede reducir de HP hacia LU aplicando función de identidad por la correctitud:

- HP acepta cuando pare una maquina, LU acepta cuando acepta la maquina.
- La maquina que simula HP puede parar y rechazar pero la maquina HP aceptaría.
- Por los 2 puntos anteriores se debe modificar la maquina que le llega HP haciendo que todos los casos de parada sean aceptaciones, sino tendremos casos donde HP termina (para) y acepta y no esta en LU porque la maquina interna no acepta.

Contra-ejemplo:

- La máquina Mx siempre rechaza dada una cadena w.
- MHP resuelve Mx, puesto que siempre para, la ejecuta 1 paso y la acepta.
- Si yo reduzco con identidad a LU, al ejecutar la maquina LU no acepta Mx, por tanto no se cumple la correctitud.

b. Explicar por qué las MT M2 generadas en los pares de salida ($\langle M2 \rangle$, w), o bien paran aceptando, o bien loopean.

La reducción descrita en clase modifica las qR de la maquina interna a que sean qA, esto evita que la maquina pare y rechazo, entonces el único caso de rechazo es que loopee.

c. Explicar por qué la función utilizada para reducir HP a LU también sirve para reducir HPC a LU C

Propiedad: $L1 \leq L2$ sii $L1 C \leq L2 C$. La reducción es la misma.

d. Explicar por qué la función utilizada para reducir HP a LU no sirve para reducir LU a HP.

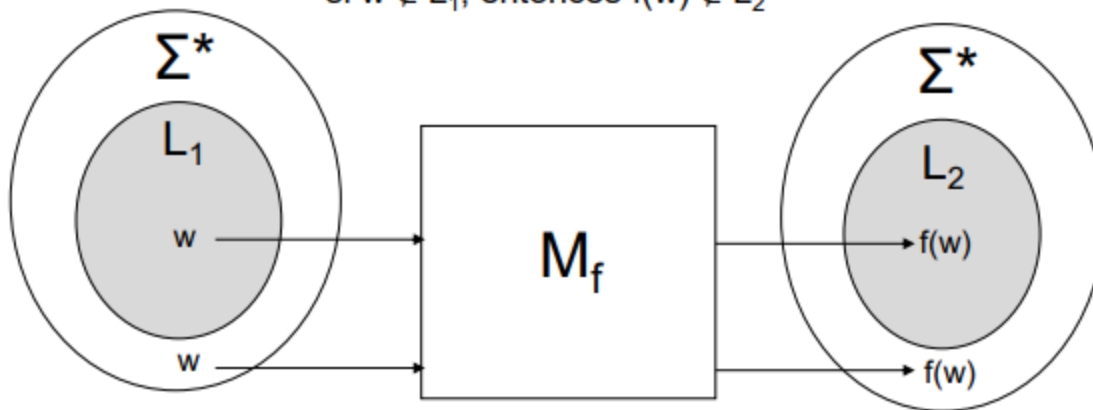
Propiedad de simetría $L1 \leq L2$ no implica $L2 \leq L1$. Ahora porque en este caso no por que:

- Lo que esta en LU \rightarrow LHP, y lo que no esta en LU no debería terminar el HP.
- Si yo hago una maquina Mx que siempre falla Y para con una cadena w cualquiera y le aplico al reducción, esta terminara en HP porque efectivamente termina, pero nunca estuvo en LU para

arrancar.

si $w \in L_1$, entonces $f(w) \in L_2$

si $w \notin L_1$, entonces $f(w) \notin L_2$



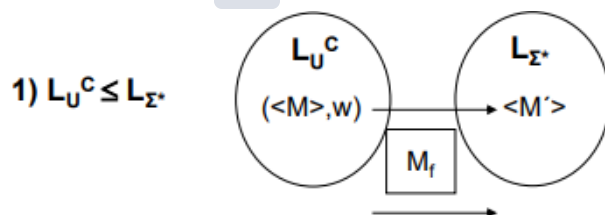
e. Explicar por qué la siguiente MT M_f no computa una reducción de HP a LU: dada una cadena válida $(\langle M \rangle, w)$, M_f ejecuta M sobre w , si M acepta entonces genera la salida $(, w)$, y si M rechaza entonces genera la cadena 1.

Una reducción no puede ejecutar nada dentro suyo ya que puede quedar colgada, POR TANTO NO ES TOTAL COMPUTABLE.

Ejercicio 2. Sabiendo que $LU \in RE$ y $LU^C \in CO-RE$:

a. Probamos en clase que existe una reducción de LU a $L\Sigma^*$. En base a esto, ¿qué se puede afirmar con respecto a la ubicación de $L\Sigma^*$ en la jerarquía de la computabilidad?

- Si $L_1 \leq L_2$ y $L_1 \notin R$, entonces $L_2 \notin R$
- Como LU se puede reducir a $L\Sigma^*$ sabemos que $L\Sigma^*$ es igual o más complejo que LU y LU están en RE, nos dejaría con que $L\Sigma^* \notin R$.
- Además, como extra, LUC se puede reducir a $L\Sigma^*$, como $LUC \notin RE$, $L\Sigma^* \notin RE$.
- Entonces como $L\Sigma^*$ no está ni en R ni en RE, este está en $L-(RE \cup CO-RE)$.



Como $L_U^C \notin RE$, entonces $L_{\Sigma^*} \notin RE$.

También se cumple $L_U \leq L_{\Sigma^*}$ (se prueba en la clase práctica).
Como $L_1 \leq L_2$ si $L_1^C \leq L_2^C$, entonces $L_U^C \leq L_{\Sigma^*}^C$, y así $L_{\Sigma^*}^C \notin RE$.

En definitiva: L_{Σ^*} y $L_{\Sigma^*}^C$ están en $\mathcal{L} - (RE \cup CO-RE)$.

Ejemplo de reducción

Sea el problema: dada una MT M , ¿acaso M acepta todas las cadenas de Σ^* ?

El lenguaje que representa el problema, sabemos, es: $L_{\Sigma^*} = \{ \langle M \rangle \mid L(M) = \Sigma^* \}$. Probaremos con una reducción que $L_{\Sigma^*} \notin R$.

Ejercicio: Intuitivamente, ¿puede ser $L_{\Sigma^*} \in R$? Más aún, ¿puede ser $L_{\Sigma^*} \in RE$?

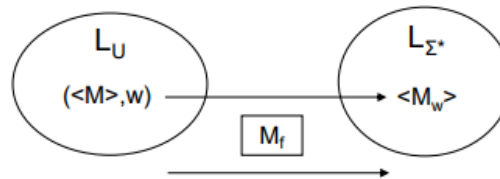
Usaremos: si $L_1 \leq L_2$ y $L_1 \notin R$, entonces $L_2 \notin R$.

Así, hay que encontrar una reducción de la forma $L_1 \leq L_2$, de modo tal que $L_1 \notin R$. Elegimos como L_1 el lenguaje L_U .

Reducción:

Definimos: $f(\langle M \rangle, w) = \langle M_w \rangle$, tal que M_w es una MT que:

- Reemplaza su entrada por w .
- Ejecuta M sobre w .
- Acepta sii M acepta.



Computabilidad:

Existe una MT M_f que computa f : genera $\langle M_w \rangle$, agregando al código $\langle M \rangle$ un fragmento inicial que borra su entrada y la reemplaza por w .

Correctitud:

$\langle M \rangle, w \in L_U \rightarrow M$ acepta $w \rightarrow M_w$ acepta todas sus entradas (¿por qué?) $\rightarrow L(M_w) = \Sigma^* \rightarrow \langle M_w \rangle \in L_{\Sigma^*}$
 $\langle M \rangle, w \notin L_U \rightarrow$ caso cadena válida: M rechaza $w \rightarrow M_w$ rechaza todas sus entradas (¿por qué?) $\rightarrow L(M_w) \neq \Sigma^* \rightarrow \langle M_w \rangle \notin L_{\Sigma^*}$
caso cadena no válida: M_w tampoco es una cadena válida $\rightarrow L(M_w) \neq \Sigma^* \rightarrow \langle M_w \rangle \notin L_{\Sigma^*}$

25

b. Se prueba que existe una reducción de L_U a L $\notin R$. En base a esto, ¿qué se puede afirmar con respecto a la ubicación de L $\notin R$ en la jerarquía de la computabilidad?

Vacío no pertenece a RE porque L_U no pertenece a RE, como el complemento está en L $\notin R$ L_U está en RE (por construcción de una máquina), entonces L $\notin R$ está en CO-RE.

Ejercicio 3. Sea el lenguaje $DHP = \{w_i \mid M_i \text{ para a partir de } w_i\}$ (considerar el orden canónico). Encontrar una reducción de DHP a HP. Comentario: hay que definir la función de reducción y probar su total computabilidad y correctitud.

- DHP tiene un lenguaje w_i que es resuelto con la M_i , que resuelve el Halting problem dado una máquina i para la cadena i .
- $DHP \leq HP$ en complejidad.
- $DHP = \{w_i \mid M_i \text{ para a partir de } w_i\}$ y $HP = \{ \langle M, w \rangle \mid M \text{ para con } w \}$.
- Con todo lo anterior entonces la reducción debe transformar w_i en un par (M, w_i) donde M es M_i si w_i es válido, sino generara una M que nunca se detiene por tanto no pertenece a HP.
- Computabilidad: Verifica que w_i es una codificación válida para una MT M_i
 - Si es válida (que lo sabemos mirando a la diagonal siguiendo el orden canónico) construye el par (M_i, w_i) correspondiente.
 - Si no es válida genera un par (M_{loop}, w_i) donde M_{loop} es una MT que nunca termina.

- Mf siempre termina, la validación y concatenación son operaciones finitas.
- Correctitud:
 - $w_i \in DHP \rightarrow f(w_i) = (M_i, w_i) \in HP$
 - $w_i \notin DHP$:
 - $f(w_i) = (Mloop, w_i)$ y $Mloop$ nunca para
 - $f(w_i) \notin HP$ por virtud de nunca parar

Ejercicio 4. Sean TAUT y NOSAT los lenguajes de las fórmulas booleanas sin cuantificadores tautológicas (satisfactibles por todas las asignaciones de valores de verdad) e insatisfactible (insatisfactibles por todas las asignaciones de valores de verdad). Encontrar una reducción de TAUT a NOSAT. Comentario: hay que definir la función de reducción y probar su total computabilidad y correctitud.

- TAUT a NOSAT es "fácil", si negamos cualquier elemento de TAUT nosotros obtendremos una formula insatisfactible, viceversa debería funcionar, negar la formula (esencialmente envolver la fórmula en paréntesis y poner una \neg) es la reducción.
- Computabilidad:
 - Se toma la cadena original y se la envuelve en () y luego se pone un \neg . Esto es claramente computable en tiempo finito.
- Correctitud:
 - Si una cadena $w \in TAUT$, su negación $f(w) \in NOSAT$, es decir siempre es falsa.
 - Si $w \notin TAUT$
 - Si no es una formula válida, entonces no es evaluable, lo que no la hace falsa ni verdadera, una Mt que acepte NOSAT la rechaza.
 - Si es una formula válida, al menos hay una posible solución donde no siempre es falsa.

Ejercicio 5. Se prueba que existe una reducción de $LU C$ a $L\Sigma$ (y así, como $LU C \notin RE$, entonces se cumple que $L\Sigma \notin RE$). La reducción es la siguiente. Para toda w : $f(\langle M1 \rangle, w) = \langle M2 \rangle$, tal que $M2$, a partir de su entrada v , ejecuta $|v|$ pasos de $M1$ a partir de w , y acepta sii $M1$ no acepta. Probar que la función definida es efectivamente una

reducción de $LU C$ a $L\Sigma^*$. Comentario: hay que probar su total computabilidad y correctitud.

- La reducción dice: "Para toda w : $f(\langle M1 \rangle, w) = \langle M2 \rangle$, tal que $M2$, a partir de su entrada v , ejecuta $|v|$ pasos de $M1$ a partir de w , y acepta sii $M1$ no acepta."
- La reducción es valida, pero por que:
 - $Lu = \{ \langle M, w \rangle \mid \text{la MT } M \text{ acepta la cadena } w \}$, LuC es lo mismo pero la MT rechaza o loopea.
 - $L\Sigma^* = \{ \langle M \rangle \mid L(M) = \Sigma^* \}$ es el lenguaje de las MT que aceptan todas las cadenas de su alfabeto.
 - Entonces, una cadena de LuC que es válida rechaza o queda loopeando siempre, cuando es reducida a $L\Sigma^*$ por esta reducción se transforma es una maquina que ejecuta $|v|$ pasos de $M1$ sobre w , y acepta sii $M1$ no acepta.
- Computabilidad:
 - Mf entonces recibe la tupla de entrada de $(\langle M1 \rangle, w)$ y construye un código de maquina $M2$, programado para tomar la entrada arbitraria v como contador de pasos a ejecutar de $M1$ sobre la cadena w . Todo esto es computable en tiempo finito porque es copiar, escribir, etc.
- Correctitud:
 - Si $(\langle M1 \rangle, w) \in LuC \rightarrow M2$ ejecutara $M1$ sobre w una cantidad arbitraria de v pasos, en este contexto siempre falla por lo que aceptaría (condición si $M1$ no acepta), o se ejecuta v pasos y acepta por que termino el tiempo.
 - Si $(\langle M1 \rangle, w) \notin LuC \rightarrow M2$ ejecutara $M1$ sobre w una cantidad arbitraria de pasos (v), entonces hay al menos un v donde $M1$ termina aceptando por tanto $M2$ rechaza, lo que lo hace estar fuera de $L\Sigma^*$, dado que si hay un elemento del alfabeto que lo hace rechazar entonces falla.