Práctica 2

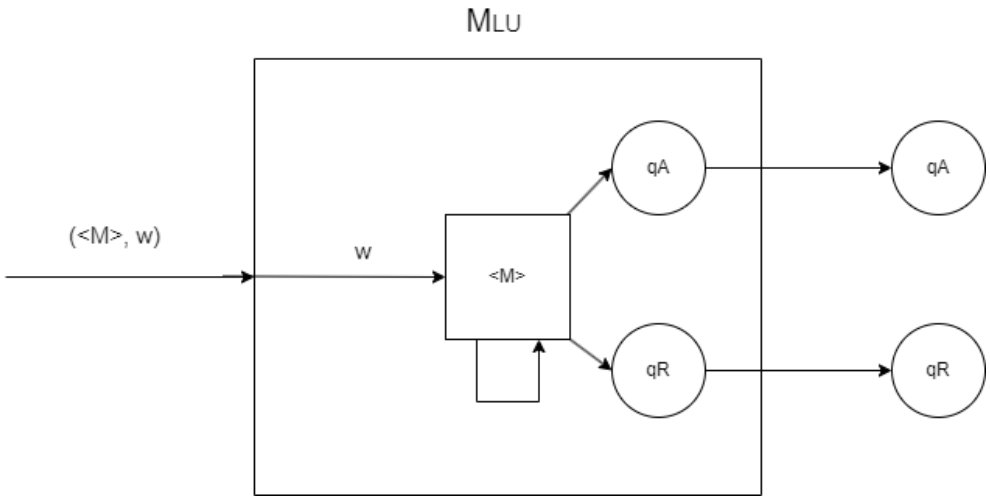
Agustina Sol Rojas y Antonio Felix Glorioso Ceretti

Ejercicio 1.

1. Probar que el lenguaje LU = {(<M>, w) | M acepta w} pertenece a la clase RE.

1) Idea General

La idea general es demostrar que para el lenguaje LU existe una Maquina de Turing que lo acepta. Para ello se construye una máquina M_{LU} que recibe una codificación de Maquina de Turing (<M>) y un input (w). Dicha maquina simulará la ejecución de <M> sobre w y podrá aceptar, rechazar o loopear.



2) Construcción

- M_{LU} tiene 2 cintas
- Dada la entrada de (<M>, w) en la cinta 1, M_{LU} hace:
    1. Verifica que el input sea un par válido (<M>,w), caso contrario rechaza.
    2. Copia <M> en la cinta 2 y cuando termina el cabezal va a estar posicionado en la cinta 1 al comienzo de w.

3. Ejecuta <M> sobre w en la cinta 1. Si <M> para en qA, entonces M_{LU} para en qA, si <M> para en qR entonces M_{LU} para en qR y si <M> loopea entonces M_{LU} loopea.

3) Prueba de correctitud de construcción

LU ∈ RE

Para que un lenguaje pertenezca a RE debe existe una máquina que lo acepte. Para el caso de LU se construyó una MT que acepta el lenguaje, dicha MT es M_{LU}, y esta en base a lo que hace <M> va a aceptar, rechazar o loopear.

Ejercicio 2.

2. Justificar o responder según el caso:
   a. Se puede decidir si una MT M, a partir de la cadena vacía λ, escribe alguna vez un símbolo no blanco. Ayuda: ¿Cuántos pasos puede hacer M antes de entrar en loop?

      Si se puede. Para ello se va a contar la cantidad de estados M. En base a esa cantidad K de estados se va a simular la ejecución de M en K pasos. Cada estado va a tener como máximo una transición desde el blanco (por definición de MT), si no se escribió nada después de ejecutar M K veces entonces la máquina, a partir de la cadena vacía, no escribe alguna vez un símbolo no blanco. Puede suceder que exista una transición en la cual luego de leer un blanco escriba algo, pero que cuando la cadena es vacía la maquina loopeara y nunca se llegara a dicha transición.

      La cantidad de pasos que puede hacer M antes de entrar en loop es a lo sumo K.

   b. Se puede decidir si a partir de una cadena w, una MT M que sólo se mueve a la derecha se detiene. Ayuda: ¿Cuántos pasos puede hacer M antes de entrar en loop?

Si se puede decidir. Dado que la maquina solo se mueve a la derecha se puede determinar que la maquina entro en un loop si cuando luego de leer todo w, se sigue ejecutando. Como la MT solo se mueve a la derecha, una vez que lea blanco y no se detenga siempre se va a repetir las mimas configuraciones, ya que siempre se va a tener el mismo símbolo, siempre se va a mover a la derecha y cada estado va a tener a lo sumo una sola transición que parta del símbolo blanco (como es en el punto a)
Puede hacer a lo sumo K + N pasos, siendo K = |w| y N = |Q| (cantidad de estados antes que loopee.

c. Se puede decidir, dada una MT M, si existe una cadena w a partir de la cual M se detiene en a lo sumo 10 pasos. Ayuda: ¿Hasta qué tamaño de cadenas hay que chequear?

   Si se puede decidir. Para ello se van a generar todas las cadenas de longitud 0 hasta la longitud 10 y luego se van a simular 10 pasos de M sobre las mismas. No se chequea con cadenas con longitud mayor a 10 ya que se van a necesitar más de 10 pasos solo para recorrer el input.

d. Intuitivamente, ¿se puede decidir, dada una MT M, si existe una cadena w de a lo sumo 10 símbolos a partir de la cual M para?

   No, no se puede decidir ya que no podes distinguir cuando o no la maquina comienza a loopear.

Ejercicio 3.

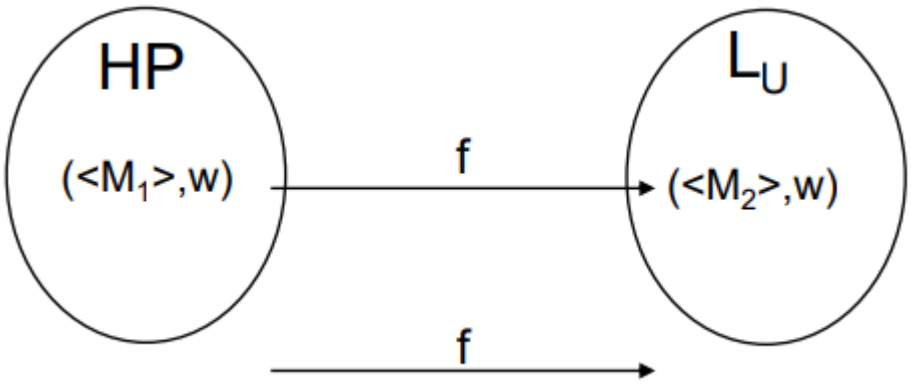
3. Considerando la reducción de HP a LU descripta en clase, responder:
   a. Explicar por qué la función identidad, es decir la función que a toda cadena le asigna la misma cadena, no es una reducción de HP a LU.

      La función de identidad no es una reducción de HP a LU ya que esta mapea cada cadena a sí misma, y en el caso en el cual una MT M para sobre un w rechazándolo este par (<M>, w) pertenece a HP (ya que HP = (<M>,w | M para sobre w), pero no a LU (ya que LU = (<M>, w) | M acepta w). Esto genera una

contradicción con la reducción ya que se espera que la función de reducción mantenga la propiedad de que si w pertenece a un lenguaje L1, entonces f(w) pertenece a L2.

b. Explicar por qué las MT M' generadas en los pares (<M'>, w), o bien se detienen aceptando, o bien loopean.

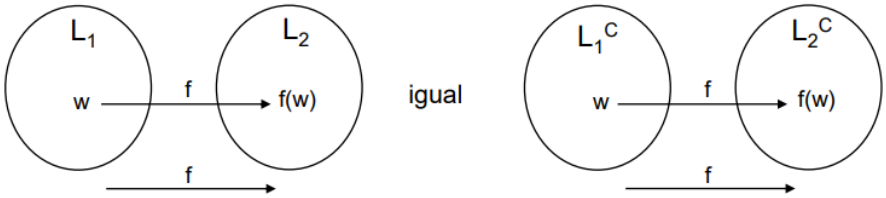
Se detienen aceptando o bien loopean porque si no no se cumpliría con la definición de reducción menciona en el punto a. Esto se realiza para que los pares (<M>, w) que pertenecen a HP pertenezcan también a LU y también para que se cumpla lo contrario.
En la reducción HP a LU si M para sobre w (ya sea en qA o qR) M' va a parar en qA, por lo que la única posibilidad es que M' loopee o acepte w.



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

Sirve para la reducción de los complementos ya que lo que no pertenece a HP (lo que sería HP^C) tampoco va a pertenecer a LU (va a pertenecer a LU^C). A su vez se cumple que lo que pertenece a HP (por lo tanto no pertenece a HP^C) pertenece a LU (por lo tanto no pertenece a LU^C).
En simples palabras lo que no va a pertenecer a HP no va a pertenecer a LU y esto básicamente son sus respectivos complementos.
Esto es una propiedad de la reducción que se vio en clase

- **Otra propiedad:** $L_1 \leq L_2$ sii $L_1^C \leq L_2^C$. Es la misma función de reducción.



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

No funciona porque lo que realiza la función de reducción es cambiar los estados que terminan en qR en M1 por qA en M2. Esto va a hacer que algo que no pertenece a LU (cuando M rechaza w) pertenezca a HP (M va a aceptar a w, y por lo tanto va a parar). Esto último no cumple con la definición de reducción.

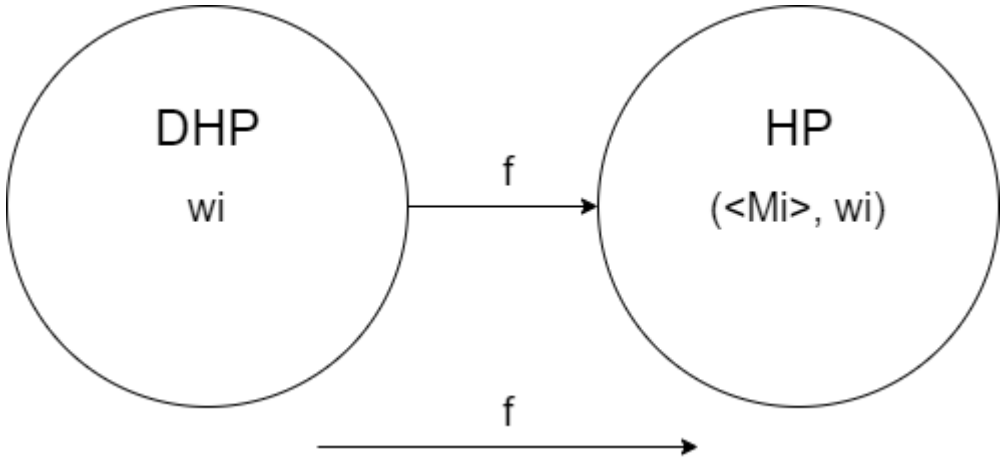
e. Explicar por qué la siguiente MT Mf no computa una reducción de HP a LU: dada una cadena válida (<M>, w), Mf ejecuta M sobre w, si M acepta entonces genera el par de salida (<M>, w), y si M rechaza entonces genera la cadena 1.

El input (<M>, w) va a pertenecer a HP puesto que M rechaza a w y por lo tanto para. Esto va a ser transformado por la Mf en la cadena 1, la cual no es un par (<M>, w) valido, por lo tanto no pertenece a LU. Nuevamente no se cumple con la definición de reducción mencionada en el punto a.

Ejercicio 4.

4. Sea el lenguaje DHP = {wi | Mi para desde wi, según el orden canónico}. Encontrar una reducción de DHP a HP.

DHP = {wi | Mi para desde wi, según el orden canónico}
HP = {<M>, w | M para sobre w}
Vamos a probar DHP <= HP:



Definición de reducción

f(wi) = ((<Mi>,wi))

Computabilidad

Existe una MT Mf que va a calcular el índice i generando todas las cadenas en orden canónico hasta llegar a la cadena que se corresponda con el input recibido. De esta forma se obtiene el índice i.

Luego de esto se va a generar Mi, formando los strings en orden canónico incrementando un contador en 1, cuando se trata de un código de maquina valido así hasta que el contador sea igual a i y por lo tanto se obtenga Mi.

De esta forma se obtiene el par (<Mi>, wi) perteneciente a HP.

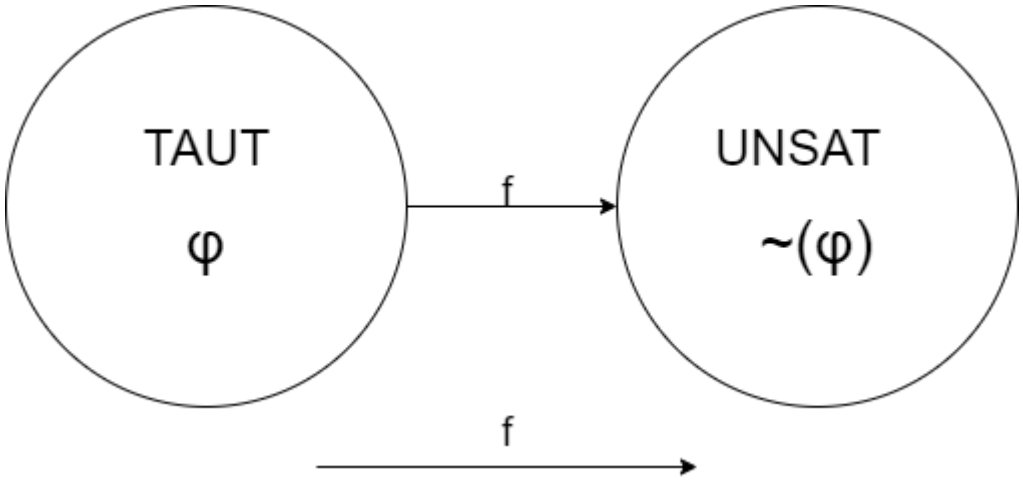
Correctitud

wi ∈ DHP → Mf genera un Mi que para sobre wi → (<Mi>, wi) ∈ HP

wi ∉ DHP → Mf genera un Mi que no para sobre wi → (<Mi>, wi) ∉ HP

Ejercicio 5.

5. Sean TAUT y NOSAT los lenguajes de las fórmulas booleanas sin cuantificadores, respectivamente, tautológicas (satisfactibles por todas las asignaciones de valores de verdad), e insatisfactibles (ninguna asignación de valores de verdad las satisface). Encontrar una reducción de TAUT a NOSAT.



Definición de reducción

f(ϕ) = ~(ϕ) siendo este la negación de ϕ

Computabilidad

Existe una MT Mf que recibe una cadena y la niega. Para esto se escribirá sobre el input un símbolo de negación y a su lado un paréntesis. Finalmente se colocará un paréntesis de cierre al final del input, dejando así como output su negación.

Correctitud

ϕ ∈ TAUT → ϕ es una tautología → ~(ϕ) es una contradicción → ~(ϕ) ∈ UNSAT

ϕ ∉ TAUT → ϕ formula booleana valida: ϕ no es una tautología → ~(ϕ) no es una contradiccion → ~(ϕ) ∉ UNSAT

ϕ formula booleana invalida: ϕ no es una tautología → ~(ϕ) no es una contradiccion → ~(ϕ) ∉ UNSAT

Ejercicio 6.

6. Un autómata linealmente acotado (ALA) es una MT con una sola cinta, con la restricción de que su cabezal sólo puede moverse a lo largo de las celdas ocupadas por la cadena de entrada. Probar que el lenguaje aceptado por un ALA es recursivo. Ayuda: ¿en cuántos pasos se puede detectar que el ALA entra en loop?

   El lenguaje aceptado por un ALA es recursivo ya que existe una maquina que lo acepta y siempre para. Si bien ALA puede loopear (no hay ninguna restricción en el

enunciado al respecto), se puede detectar cuando comienza a hacerlo y así construir una maquina equivalente que siempre se detenga.

Se puede detectar que ALA esta loopeando luego de una cierta cantidad de pasos, calculados de la siguiente manera:

$P = |w|.|Q|.|\Sigma|^{|w|}$ cantidad máxima de pasos antes de que se repita una configuración.

ALA' (MT equivalente a ALA) una vez que detecta que se superó P va a rechazar con qR. Por lo tanto $L(ALA') \in R \rightarrow L(ALA') = L(ALA) \rightarrow L(ALA) \in R$.

Ejercicio 7.

7. Construir una MT que genere todos los índices i tales que (<Mi>, wi) ∈ HP, según el orden canónico.

   Solución:

   1) M_{GHP} tiene un contador que parte en i=0.
   2) Genera y ejecuta (i pasos de M sobre w) sobre todos los w y M hasta la wi y Mi.
   3) Si Mi para con wi significa que está en HP por lo que se escribe en la salida el par (<Mi>, wi).
   4) Se borra la salida (para respetar el orden canónico y evitar repeticiones en el output), se hace i+= 1 y se vuelve al paso 2).