TCVP - TP 1: La máquina de Turing y la jerarquía de la computabilidad

Alumno	Legajo
Chilaca Castro, Sol Angela	22277/2

Ejercicio 1

1.

E = {a, b, c}
L = {a^n b^n c^n|n>= 0}

=> Σ* ⋂ L = L , pues L se encuentra contenido en Σ* por ser L un subconjunto de Σ*
=> Σ* ⋃ L = Σ* , pues todo elemento de L se encuentra en Σ*. Idem anterior.
=> L^c = Σ*- L , como L se encuentra contenido en Σ* entonces todo los elementos que pertenecen a L pertenecen también a Σ* su intersección serán todos los elementos que pertenecen a Σ* y no a L.

2. Problema de satisfactibilidad de las fórmulas booleanas

- Visión de MT calculadora -> ante la entrada de fórmula booleana la salida será una asignación de valores de verdad, si no existe la máquina rechazará.
- Visión de MT reconocedora -> ante la entrada de fórmula booleana la salida será *sí* si existe una asignación de valores que la expresión se evalúe como verdadera, por el contrario rechazará.
- Visión de MT generadora -> genera todas las fórmulas booleanas satisfactibles.

3. Tesis de Church-Turing: Todo dispositivo de computación físicamente realizable puede ser simulado por una MT.
4.

- MTs equivalentes -> aceptan el mismo lenguaje, o sea, ante la misma cadena de entrada ambas máquinas tendrán el mismo comportamiento. M1 y M2 pueden tener mecanismos internos y formas diferentes en procesar la entrada pero se comportan igual.
- Modelos de MT equivalentes -> si dada una MT de un modelo existe una MT equivalente del otro. Ejemplo: MT con una sola cinta y otro modelo con varias cintas ya que cualquier lenguaje que

pueda ser reconocido por la máquina de dos cintas entonces es también reconocible por el modelo de una cinta.

5.

- Lenguajes R (recursivos): que ante la entrada la máquina siempre se detendrá (rechace o acepte)
- Lenguajes RE (recursivos enumerables): son los lenguajes que la máquina en los casos positivas acepta la entrada o la máquina no para.
- Lenguaje no recursivamente enumerables: no existe una máquina de Turing que lo pueda reconocer, ni siquiera de forma parcial (como el caso de que acepte).

6. Probar $RE \subseteq R \subseteq \mathcal{L}$

$RE \subseteq R$ -> En R se encuentran los lenguajes que tienen una MT qe los acepte, en el caso de RE, por definición, se consideran los lenguajes que se detienen en caso de aceptar o en caso de rechazar se detienen en q_r. o loopean. Por definición entonces se prueba que todo lenguaje recursivo es también un lenguaje recursivamente enumerable.

$RE \subseteq \mathcal{L}$ -> Dado que $\mathcal{L}$ es el conjunto de todos los lenguajes y RE está compuesto por lenguajes entonces pertenecen a $\mathcal{L}$.

7. Explicar por qué (a) el lenguaje Σ* de todas las cadenas, (b) el lenguaje vacío ∅, (c) cualquier lenguaje finito, son recursivos.

Se puede definir una MT M que acepte $w \in \Sigma*$ tal que al leer el primer símbolo del alfabeto la máquina se **detenga** y lo acepte, por lo tanto Σ* pertenece al lenguaje y es recursivo.

En el caso del lenguaje vacío éste significa que no hay w que una MT M acepte, entonces siempre se **detendrá** ante cualquier cadena de entrada, por lo tanto, ∅ es recursivo.

Un lenguaje finito implica que se tiene un número finito de cadenas entonces es posible definir una MT M que siempre se detenga, es decir ante una entrada w que compare con una lista finita de palabras del lenguaje (pues el lenguaje es finito), si coincide acepta, sino rechaza, entonces el lenguaje finito es recursivo.

8. Explicar por qué no es correcta la siguiente prueba de que si $L \in RE$ también $L^c \in RE$: dada una MT M que acepta L, entonces la MT M' igual que M pero con los estados finales permutados, acepta L^c

No es correcto porque si un $w \in L \in RE$ significa que existe una máquina que se detiene o loopea entonces $R \subseteq RE$ (demostrado más antes) por lo tanto L^c no pertenece a a RE ni a R. Entonces

la cadena $w \in L^c$ no tendrá máquina que le acepte.

Ejercicio 2

Explicar como una MT que en un paso no puede simultáneamente modificar un símbolo y moverse, puede simular (ejecutar) una MT que sí lo puede hacer.
Dada una MT M que si puede modificar un paso y moverse entonces podemos definir una MT M' con el doble de pasos. Para cada estado de M que requiera escribir y moverse a una dirección se descompondrá en dos: un primer estado que modifique el símbolo y el segundo estado que implique mover el cabezal y pasar al estado que le correspondía.
Por ejemplo:
se tiene la transición en MT M: $(q, a) -> (q', A, R)$
Entonces en M' se tendrá $(q, a) -> (q*, A, S); (q*, A) -> (q', A, R)$

Ejercicio 3

Describir la idea general de una MT con varias cintas que acepte, de la manera más eficiente posible (menor cantidad de pasos), el lenguaje $L = \{a^n b^n c^n | n \geq 0\}$

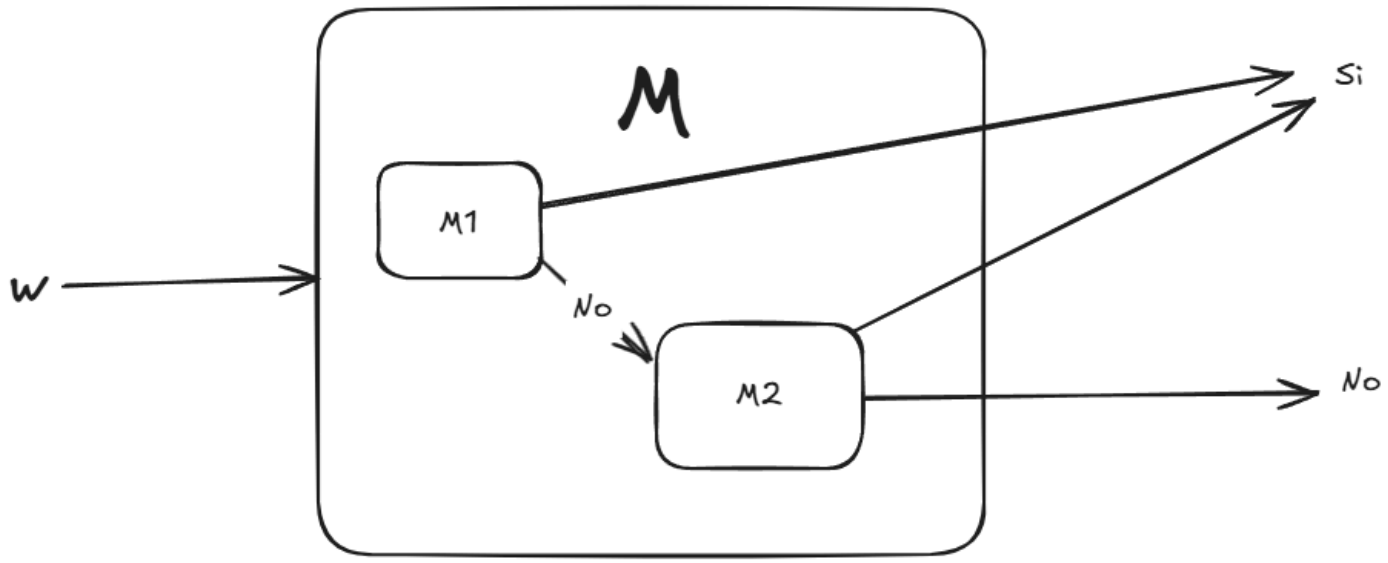
Idea general:
En la segunda cinta se copian los caracteres "a" de la primera cinta (en el caso de que en la primera cinta se encuentre un caracter diferente a "a" o "b" la máquina rechaza) hasta encontrar un caracter "b".
Luego, se recorren ambas cintas: la segunda cinta hacia la izquierda y la primera hacia la derecha. Se va comparando el contenido de las dos cintas, de cada "a" con las "b".
En el caso de que la cantidad de "a" sea igual a la cantidad de "b" entonces se comparará con los caracteres "c" de la primera cinta (caso contrario rechaza).
De la misma forma, si la cantidad de "a" es igual a la cantidad de "c" entonces la máquina acepta.

Ejercicio 4

Probar:

1. La clase R es cerrada con respecto a la operación de unión.

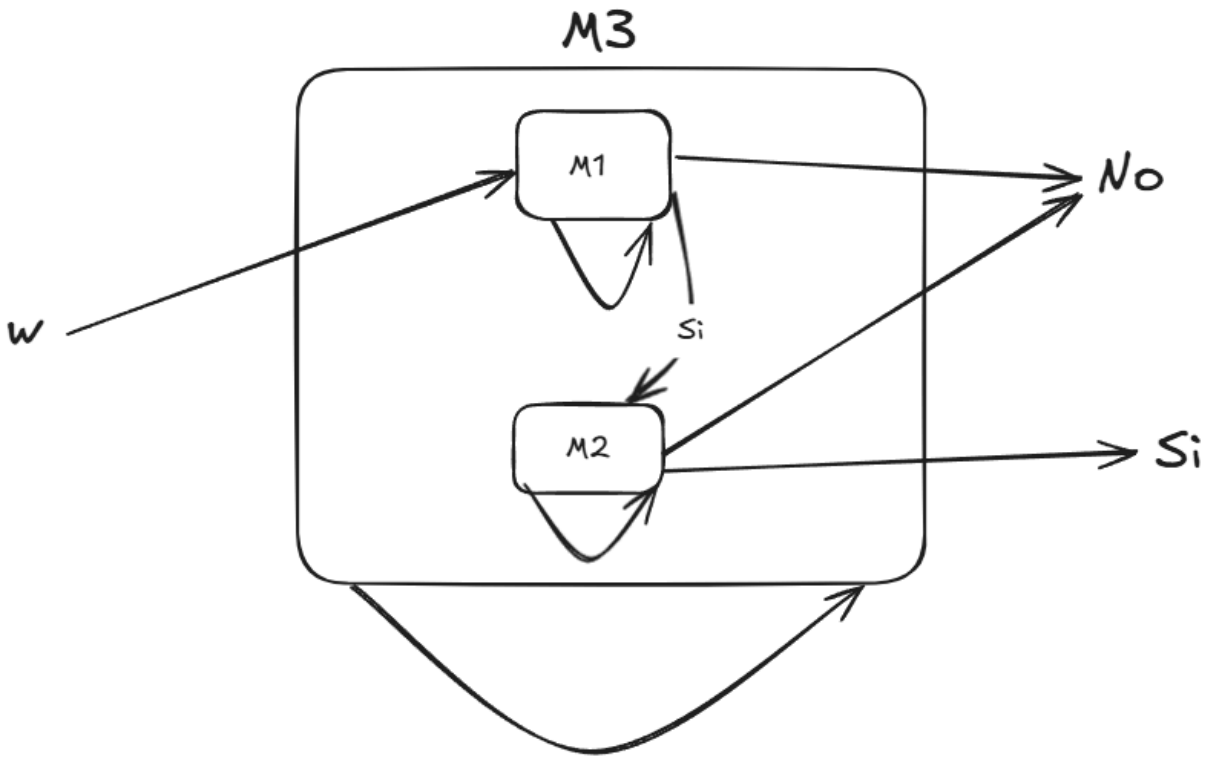
Sean $L_1, L_2 \in R$ y M_1, M_2 son MT que deciden los lenguajes correspondientes entonces existe una MT M_3 dada una entrada $w \in L_1 \cup L_2$ se detiene siempre.



Entonces, M_1 ejecuta, en caso de aceptar entonces M_3 acepta; en caso de rechazar se prueba w en M_2. En caso de aceptar entonces M_3 acepta, caso contrario M_3 rechaza. Se garantiza que M_3 siempre se detiene pues tanto M_1 como M_2 se detienen entonces M_3 reconoce $L_1 \cup L_2$

2. La clase RE es cerrada con respecto a la operación de intersección.

Sean $L_1, L_2 \in RE$ y M_1, M_2 son MT que reconocen los lenguajes correspondientes entonces existe una MT M_3 dada una entrada $w \in L_1 \cap L_2$ lo reconoce.



Entonces M_1 ejecuta, en caso de aceptar entonces se ejecuta M_2. En caso de rechazar M_3 rechaza.
M_2 ejecuta y en caso de aceptar entonces M_3 acepta, caso contrario rechaza.
Si en algún momento M_1 o M_2 no se detienen entonces M_3 no se detiene.
En síntesis, si M_1 o M_2 rechazan la entrada entonces M_3 rechaza, si es rechazada por ambas máquinas entonces M_3 rechaza, si M_1 o M_2 se quedan loopeando entonces M_3 se va a quedar loopeando.

Ejercicio 5

Sean L_1 y L_2 dos lenguajes recursivamente numerables de números naturales codificados en unario. Probar que también es recursivamente numerable el lenguaje $L = \{x | x\ es un número natural codificado en unario y existen y, z / x + z = x, y \in L_1, z \in L_2\}$

Dado que una cadena $w = x_1 \times x_2 \in L$ con $x_1 \in L_1$ y $x_2 \in L_2$ particiones del lenguaje respectivo se prueba que existe una MT M que acepte el lenguaje $L_1 \times L_2$
Sea n la longitud de la cadena w, M debe ejecutar M_1 con los primeros k-n símbolos de w y M_2 con los últimos n símbolos de w. Si las dos máquinas aceptan entonces M acepta. En el caso de que se rechace tendrá que volver a realizarse lo mismo pero ahora M_1 deberá tomar los primeros k+1 símbolos de w y M_2 los últimos n-1. De nuevo, si ambas máquinas aceptan entonces M acepta.
Los pasos anteriores deberán repetirse hasta que M_1 reciba los n símbolos de w y M_2 reciba 0. En el caso de que en ninguna de las iteraciones se acepte entonces M rechaza la cadena w.

Cabe destacar que L_1 y $L_2 \in RE$ entonces se corre riesgo que las máquinas que las reconozcan (M_1 y M_2 respectivamente) loopen. Es necesario entonces que la ejecución de los pasos de cada máquina se realice de forma paralela o alternada para evitar la no detención y no impedir que se siga computando otra partición de w.

Dado que M se detiene una vez que reconozca las cadenas w si sus subcadenas son reconocidas por M_1 y M_2 entonces M reconoce $L_1 \times L_2$ y en si se quedan loopeando entonces M va a loopear entonces también $L_1 \times L_2 \in RE$

Ejercicio 6

Dada una MT M_1 con alfabeto $\Gamma = \{0, 1\}$:

1. Construir una MT M_2 utilizando la MT M_1 que acepte cualquiera sea su cadena de entrada sii la MT M_1 acepta al menos una cadena.

Para que M_2 acepte cualquiera sea la entrada sii M_1 acepta al menos una cadena entonces M_2 deberá ignorar su propia entrada y deberá probar las posibles cadenas $w \in L = L(M_1)$ para que M_1 acepte y así M_2 acepte todas las cadenas que ingresen o M_2 rechace todas las cadenas que pruebe M_2 y rechace todas las cadenas.

Entonces una vez que ingrese una cadena w_2 a M_2 ésta deberá ignorar la entrada y probbar las posibles cadenas w que M_1 acepte. Sea $n >= 0$,M_2 irá probando generando las cadenas posibles sujetas al alfabeto indicado al incio de longitud n hasta que M_1 acepte. Por ejemplo en la iteración de $n = 0$ se probará λ en M_1 si la acepta entonces M_2 acepta todas las cadenas que se ingresen, caso contrario n se incrementa +1 y M_2 probará ingresar 0 o 1, si rechaza se probará con "00", "11", "10", "01" y así repetidos pasos.
Ya que en el enunciado no se especifica que el lenguaje que acepta M_1 pertenezca a R no podemos asumir que la máquina se detendrá, entonces es necesario que se lleve un control. M_2 puede hacer que se ejecuten una cantidad de pasos limitados por cada cadena ingresada y llevarlo de forma paralela, es decir, se simula la ejecución de k pasos para todas las cadenas de longitud n y si M_1 acepta entonces M_2 acepta. En caso contrario, se repetirá el proceso incrementando la cantidad de k pasos y la longitud de n en esa iteración.

2. ¿Se puede construir además una MT M_3 utilizando la MT M_1 que acepte, cualquiera su cadena de entrada, sii la MT M_1 acepta a lo sumo una cadena? Justificar.
   En el caso de que $\#L(M_1) = 1$, M_3 buscará la cadena que sea aceptada por M_1 y una vez que se encuentre $w \in L(M_1)$ seguirá buscando una segunda cadena para constatar que esa sea la única y lo hará de forma infinita (dado que el conjunto de cadenas generadas a partir de un alfabeto son infinitas) y la máquina M_3 loopeará cuando debería aceptarlo.