

# Parcial FTC - 2022

## Ejercicio 1

### Punto A

Un **lenguaje no es recursivo** cuando *no tiene una máquina de Turing que lo decida y pare siempre para todas las instancias positivas y negativas del problema.*

### Punto B

**No**, no puede ser, ya que si creamos una MT  $M_C$  que invierta las salidas de la máquina original  $M$  que acepta al lenguaje no recursivo, existen casos donde  $M$  loopea, por lo tanto  $w$  no pertenece a  $L$  y  $M_C$  debería aceptar, pero si  $M$  loopea,  $M_C$  también. *Esto hace que no se cumpla la definición para pertenecer a  $R$ .*

## Ejercicio 2

MT  $M_{IMPARES} = (Q, \Gamma, \delta, q_{PAR}, q_A, q_R)$

**Idea general:** La máquina  $M_{IMPARES}$  recorrerá la cadena fijándose lo siguiente:

- Si encontramos un 1 por primera vez, entonces tenemos cantidad impar de 1s.
  - Si volvemos a encontrar otro 1, entonces tenemos una cantidad par de 1s.
- Esto nos deja que ver que la máquina va intercalando los estados "par" (cero 1s), "impar" (un 1), "par" (dos 1s), etc. cuando lleguemos al Blanco si estábamos en cantidad impar, aceptamos, sino, rechazamos.

**Q:**  $(q_{par}, q_{impar}, q_A, q_R)$

- $q_{par}$ : Cantidad par de 1s.
- $q_{impar}$ : Cantidad impar de 1s.

**$\Gamma$ :**  $(0, 1, B)$

**$\delta$ :**

- $\delta(q_{par}, 0) = \{(q_{par}, 0, R)\} \rightarrow$  No encontramos algún 1 todavía.
- $\delta(q_{par}, 1) = \{(q_{impar}, 1, R)\} \rightarrow$  Primer 1.
- $\delta(q_{impar}, 0) = \{(q_{impar}, 0, R)\} \rightarrow$  Sigo en cantidad impar de 1.
- $\delta(q_{impar}, 1) = \{(q_{par}, 1, R)\} \rightarrow$  Paso a cantidad par de 1s.

- $\delta(q_{\text{impar}}, B) = \{(q_A, B, S)\} \rightarrow$  Llegué al Blanco con cantidad impar.

## Ejercicio 3

Una **máquina de Turing Universal** es una máquina de Turing capaz de ejecutar cualquier otra, lo que hace la máquina Universal con su entrada es ejecutar secuencialmente el código de máquina  $M$  codificado en su cinta de entrada a partir de la entrada  $w$  que también recibe en su cinta de entrada.

## Ejercicio 4

Podemos construir una máquina de Turing  $M_{L_1-L_2}$  que, utilizando las máquinas  $M_1$  y  $M_2$  que aceptan respectivamente a los lenguajes  $L_1$  y  $L_2$ , haga:

- A partir de la cadena  $w$  que se recibe en la cinta de entrada ejecuta secuencialmente, primero  $M_2$  y luego  $M_1$ :
  - Si  $M_2$  acepta  $w$ , entonces  $M_{L_1-L_2}$  rechaza.
  - Si  $M_2$  rechaza  $w$  y  $M_1$  también rechaza  $w$ , entonces  $M_{L_1-L_2}$  rechaza.
  - Si  $M_2$  rechaza  $w$  y  $M_1$  acepta  $w$  entonces  $M_{L_1-L_2}$  acepta.
  - Si  $M_2$  rechaza  $w$  y  $M_1$  loopea a partir de  $w$ , entonces  $M_{L_1-L_2}$  loopea.

Al ver que  $M_{L_1-L_2}$  no para siempre para todas las instancias positivas y negativas del problema, pero si para para las instancias positivas, entonces podemos decir que  $L_1 - L_2 \in RE$ .

## Ejercicio 5

La función  $f$  es **total computable**, en primer lugar porque para siempre, no hay caso donde la función se quede loopeando y la misma está definida para todo el alfabeto del problema, lo que hace  $f$  es simplemente a partir de un código de máquina  $M$ , concatenar al final del mismo el código de la máquina  $M_{\Sigma^*}$ .

$f$  también cumple con la propiedad de **correctitud** ya que el concepto de equivalencia de 2 máquinas de Turing nos dice que 2 máquinas son equivalentes si aceptan el mismo lenguaje, por lo tanto:

- Si  $\langle M \rangle \in L_{\Sigma^*} \rightarrow L(M) = \Sigma^* \rightarrow \langle M \rangle = \langle M_{\Sigma^*} \rangle \rightarrow \langle M \rangle \in L_{EQ}$ .
- Si  $\langle M \rangle \notin L_{\Sigma^*} \rightarrow L(M) \neq \Sigma^* \rightarrow \langle M \rangle \neq \langle M_{\Sigma^*} \rangle \rightarrow \langle M \rangle \notin L_{EQ}$ .

## Ejercicio 6

El complemento de un lenguaje de  $P$  también está en  $P$  ya que si construimos una máquina  $M_C$  que use  $M$  que trabaja en tiempo polinomial, y lo que haga es invertir las salidas de  $M$ ,  $M_C$

también trabajará en tiempo polinomial, por lo tanto, el complemento de  $L$  también estará en  $P$ .

## Ejercicio 7

La hipótesis es incorrecta porque si la  $M'$  lo único que hace es invertir los estados de aceptación y de rechazo de la máquina  $M$  original, al ser máquinas de Turing no determinísticas que generan varias computaciones y que con que una de ellas acepte, se acepta la cadena  $w$  de entrada, nos puede pasar que exista algún caso donde hay una  $w$  que pertenece a  $L$ , y también pertenezca a  $L^C$ , algo que por definición está mal.

## Ejercicio 8

### Punto A

$L_{CdeH} = \{(G, v_1, v_m) \mid G \text{ es un grafo con un camino de Hamilton desde el vértice } v_1 \text{ al vértice } v_m\}$ .

### Punto B

Para probar que  $L_{CdeH}$  pertenece a  $NP$  debemos probar que existe un **certificado sucinto verificable en tiempo polinomial por una máquina de Turing verificadora de  $L_{CdeH}$** .

- Un **certificado sucinto** puede ser una lista de aristas que cumpla con la condición de representar un camino de Hamilton desde el vértice  $v_1$  al  $v_m$ . Este certificado es sucinto ya que en el peor de los casos tiene tamaño  $|E|$  o  $G$  que es **polinomial con respecto a la entrada**.
- La **verificación** se da en tiempo polinomial porque para verificar ese certificado tenemos que:
  - Verificar que todas las aristas del certificado existen en las aristas del grafo  $G$  de entrada, esto nos lleva un orden  $O(|E|^2) = O(G^2)$ .
  - Y verificar el camino en sí es verificar que el certificado empiece por el vértice  $v_1$  y termine por el vértice  $v_m$  y que los demás vértices del grafo  $G$  solo se visiten una única vez dentro de la lista de aristas, esto nos lleva un orden  $O(|V| \cdot |E|) = O(G^2)$ . En su totalidad, la **verificación** sería de orden  $O(G^2) + O(G^2) = O(G^2)$  que es **polinomial**.

## Ejercicio 9

### Punto A

**Un lenguaje es  $NP$  – completo cuando cumple 2 condiciones:**

- Pertenece a  $NP$  (posee un certificado sucinto verificable en tiempo polinomial por una máquina de Turing verificadora de ese lenguaje).
- Es  $NP - difícil$ , es decir, todo lenguaje  $L_i \in NP \leq_p L$ .

## Punto B

Si,  $L$  pertenecería a  $NP - completo$  ya que por enunciado se nos indica que  $L \in NP$  (primera condición) y por propiedad de transitividad de las reducciones, si existe una reducción polinomial de  $L_i \leq_p L_{CdeH}$  y otra  $L_{CdeH} \leq_p L$ , entonces existe una reducción polinomial  $L_i \leq_p L$ , esto se va a cumplir con todos los lenguajes de  $NP$  ya que  $L_{CdeH} \in NP - completo$  (segunda condición).

## Ejercicio 10

La clase temporal  $P$  está incluida en la clase espacial  $PSPACE$  ya que todo lenguaje que se decide en  $T(n)$  pasos, ocupará a los sumo  $S(n)$  celdas que es espacio polinomial, esto se justifica porque en  $T(n)$  pasos el cabezal de una máquina de Turing recorrerá como máximo  $S(n)$  celdas.