Parcial FTC - 2023

Ejercicio 1

Idea General

- Podemos construir un MT que acepte el lenguaje de las cadenas con forma $(123)^N \; con \; n \geq 0$ si nuestra MT chequea que se cumpla siempre el mismo patrón, primero un 1, luego un 2 y luego un 3. A partir de esto vamos a poder darnos cuenta si no se cumple este patrón o si llegamos al Blanco final respetando el mismo.

$M_{123} = (Q, \Gamma, \delta, q_0, q_A, q_R)$

Q $= (q_0, q_A, q_R, q_1, q_2, q_3)$

- q_1 → Leímos un 1.
- q_2 → Leímos un 2.
- q_3 → Leímos un 3.

Γ $= (1, 2, 3, B)$

δ

- $\delta(q_0, B) = \{(q_A, B, S)\}$ → La cadena es vacía
- $\delta(q_0, 1) = \{(q_1, 1, R)\}$ → Leemos un 1 del patrón
- $\delta(q_1, 2) = \{(q_2, 2, R)\}$ → Leemos un 2 del patrón
- $\delta(q_2, 3) = \{(q_3, 3, R)\}$ → Leemos un 3 del patrón
- $\delta(q_3, 1) = \{(q_1, 1, R)\}$ → Se reinicia el patrón
- $\delta(q_3, B) = \{(q_A, B, S)\}$ → Terminó el patrón correctamente

Ejercicio 2

Por definición cada $L_1, L_2, \; \ldots, \; L_n$ al pertenecer a RE tienen una MT asociada que acepta su lenguaje (para siempre para todas las instancias positivas del problema), a su vez, el enunciado nos dice que son lenguajes $RE \; disjuntos$, es decir, no comparten elementos en común, una cadena w sólo pertenecerá a un L_i, y también que la unión de todos estos lenguajes es igual a $\sum^*$, por lo tanto la MT a construir tiene que estar definida para decidir cualquier cadena posible, teniendo esto en cuenta podemos construir una M_i con $1 \leq i \leq n$ de la siguiente forma:

- Dada una cadena de entrada w, M_i simulará *"en paralelo"* (de esta forma evitamos que M_i loopee) w sobre todas las $M_1, M_2, \ \ldots, \ M_n$, eventualmente una de estas la aceptará y solamente una lo hará ya que los lenguajes son disjuntos, además, nunca nos va a pasar que rechacemos una cadena ya que la unión de todas las MT forma $\sum^*$, de esta forma demostramos que cada L_i es recursivo.

Ejercicio 3

Una función es una reducción de un lenguaje L_1 a un lenguaje L_2 cuando cumple 2 propiedades:

- Es *total computable*, es decir, está definida para toda cadena y siempre para, no se queda en loop.
- Tiene *correctitud*, si $w \in L_1 \rightarrow f(w) \in L_2$ y $w \notin L_1 \rightarrow f(w) \notin L_2$.

Ejercicio 4

Al cumplirse que para todo $L_i \in RE$ existe una reducción $L_i \leq L_U$, si $L_U \in R$ p**or propiedad de las reducciones** $L_1 \leq L_2$, entonces $L_2 \in R \rightarrow L_1 \in R$ estaríamos diciendo que todo $L_i \in RE$ es recursivo ya que L_U lo es y todo lenguaje a la izquierda de una reducción es igual o menos complejo que el lenguaje de la derecha, **de esta forma se cumpliría $R = RE$.**

Ejercicio 5

Idea General

- Nuestra función de reducción utilizará la **generación de cadenas en el orden canónico** de forma que nos permita **asociar un código de máquina válido a un número natural de la siguiente forma:**
  1. Chequea que la entrada $< M >$ sea un código de MT válido (de esta forma aseguramos que no nos quedemos generando cadenas infinitamente buscando el código de máquina equivalente al recibido). Si $< M >$ no es válido, retornamos -1.
  2. Si $< M >$ es válido:
     1. Iniciamos un contador i en 0.
     2. Generamos la siguiente cadena según el orden canónico
        1. Si no es un código de MT válido, volvemos al paso 2 (el de generar).
           Si es un código de MT válido, lo comparamos comparamos con el $< M >$ recibido
           1. Si son iguales, retornamos i
              Si no son iguales, incrementamos i y volvemos al paso 2 (el de generar).

Reducción

- $f(< M >) = i$ si <M> es válido $o\ -1$ si <M> no es válido.

Total Computabilidad y Correctitud

- f es *total computable* ya que está definida para todas las cadenas posibles y para siempre.
- f es *correcta* ya que a cada código de MT válido se le asigna un número Natural unívoco, de esta forma, garantizamos que todo $< M >$ válido tenga un número Natural unívoco asociado, y todo $< M >$ inválido quede por fuera de los números Naturales al asignarle -1.

Ejercicio 6

Punto A

Un lenguaje pertenece a P cuando tiene una MT asociada que lo decide en tiempo polinomial, es decir, $O(n^K)$ con K constante.

Punto B

Un lenguaje pertenece a NP si cumple alguna de estas 2 definiciones:

1. Toda cadena w que pertenece al lenguaje posee un certificado sucinto (de tamaño polinomial con respecto a la entrada) verificable en tiempo polinomial por una máquina de Turing verificadora del lenguaje en cuestión.
2. Se puede construir una máquina de Turing no determinística que decida al lenguajes en tiempo polinomial.

Punto C

Un lenguaje pertenece a $NP - completo$ cuando cumple estas 2 condiciones:

1. El lenguaje pertenece a NP.
2. El lenguajes es $NP - difícil$, es decir, para todo $L_i \in NP$ existe una reducción polinómica $L_i \leq_p L$.

Ejercicio 7

Para demostrar que CI pertenece a NP vamos a probar que **existe un certificado sucinto verificable en tiempo polinomial para cada grafo que pertenezca al lenguaje:**

- El *certificado sucinto* será una lista de vértices de tamaño K que cumpla la condición de ser un conjunto de vértices independientes, este certificado tiene tamaño polinomial K que en

el peor de los casos será $|V| = G$.

- *La verificación se realiza en tiempo polinomial y consiste en:*
  - Verificar el tamaño del certificado sea K y que cada vértice del mismo exista dentro de nuestro grafo G de entrada, esto, en el peor de los casos, es de orden temporal $O(|V| \cdot |V|) = O(G^2)$.
  - Verificar que para todo v_i y v_j del certificado con $i \neq j$ no exista una arista dentro de las aristas del grafo G, esto, en el peor de los casos, es de orden temporal $O(|V| \cdot |E|) = O(G^2)$
    La verificación se da en orden $O(G^2) + O(G^2) = O(G^2)$ que es polinomial.

Ejercicio 8

CI no pertenecería a P ya que no se conoce una solución inteligente (en tiempo polinomial) para decidir este lenguaje, sólo nos queda explorar la exponencial cantidad de conjuntos de vértices del grafo que cumplan con la condición de ser un conjunto independiente.

Ejercicio 9

Sabemos que SAT es $NP - completo$, si existiera una reducción polinomial $SAT \leq_p CI$ por propiedad de la transitividad de las reducciones estaríamos probando que CI es un problema $NP - completo$, esto cumple la condición de ser $NP - difícil$, la condición de ser NP ya la demostramos.

Ejercicio 10

Punto A

Se trabaja con una cinta de entrada de sólo lectura y no de trabajo ya que si consideramos la cinta de entrada como una cinta de trabajo que afecte al orden espacial de la MT, ninguna máquina de Turing ocuparía un espacio menor a $poly(n)$ ya que almacenar la cadena de entrada conlleva un espacio polinomial con respecto a la longitud de esa entrada.

Punto B

Un lenguaje que pertenece a la clase $LOGSPACE$ se considera tratable ya que si sabemos que la MT asociada al lenguaje trabaja en un espacio acotado al $O(log_2(n))$, el número de configuraciones distintas que puede tener la máquina en ese espacio se termina calculando como $O(c^{log_2(n)}) = O(n^{log_2(c)})$, por lo tanto, si la máquina es determinista y termina, lo hace en tiempo polinomial, entonces, los lenguajes de $LOGSPAE$ pueden ser decididos eficientemente tanto en espacio como en tiempo, lo que los hace tratables.

Punto C

Un lenguaje de P por definición tiene una MT que lo decide polinomialmente en $T(n)$ pasos, esa misma máquina si realiza $T(n)$ pasos, entonces, a lo sumo recorrió $T(n)$ posiciones de su cinta, esto sigue siendo polinomial por lo que de esta forma se demuestra que todo lenguaje de P se decide con tiempo y espacio polinomial, lo que demuestra que $P \subseteq PSPACE$.