Clases

jueves, 11 de marzo de 2021 11:36

Ejercicio 1. Responder breve y claramente los siguientes incisos:

1. ¿Qué es un problema computacional de decisión? ¿Es el tipo de problema más general que se puede formular?
2. ¿Qué cadenas integran el lenguaje aceptado por una MT?
3. En la clase teórica 1 se hace referencia al problema de satisfactibilidad de las fórmulas booleanas. Formular las tres formas del problema, teniendo en cuenta las tres visiones de MT consideradas: calculadora, aceptadora o reconocedora, y generadora.
4. ¿Qué postula la Tesis de Church-Turing?
5. ¿Cuándo dos MT son equivalentes? ¿Cuándo dos modelos de MT son equivalentes?
6. ¿En qué difieren los lenguajes recursivos, recursivamente numerables no recursivos, y no recursivamente numerables?
7. Probar que $R \subseteq RE \subseteq \mathfrak{L}$.
8. ¿Todo lenguaje de la clase CO-RE tiene una MT que lo acepta?
9. Justificar por qué los lenguajes Σ^* y $\emptyset$ son recursivos.
10. Si $L \subseteq \Sigma^*$, ¿se cumple que $L \in R$?
11. Justificar por qué un lenguaje finito es recursivo.
12. Justificar por qué si $L_1 \in$ CO-RE y $L_2 \in$ CO-RE, entonces $(L_1 \cap L_2) \in$ CO-RE.
13. Dados $\Sigma = \{a, b, c\}$ y $L = \{a^n b^n c^n \mid n \geq 0\}$, obtener $\Sigma^* \cap L$, $\Sigma^* \cup L$, y el complemento de L respecto de Σ^*.

Preguntar:
1) (3), 8), cambiar 7))
5) preguntar se la M1 loopea como sabemos cuando va a qR

1) Un problema computacional de decisión es un tipo de problema que para un input, da una respuesta por sí o por no. Más particularmente, determina si el string recibido pertenece al un conjunto de palabras, el de la MT que lo procesa. No es el tipo más general de problema que se puede formular, el más general es el problema computacional de búsqueda, el cuál para un input dado, nos devulve el procesamiento del mismo como un output.

2) Las cadenas que integran el lenguaje aceptado por una MT son aquellas que la máquina reconoce, es decir, cuando paran en qA.

3)

4) La Tesis de Church-Turing plantea que todo lo computable puede ser resuelto por una MT.

5) Dos MT son equivalentes cuando reconocen el mismo lenguaje. Dos modelos de MT son equivalentes cuando para cada MT de un modelo se puede encontrar una MT' equivalente del otro.

6) Los lenguajes recursivos son aquellos en los que la MT va a parar siempre.
los recursivamente numerables la MT puede parar o puede quedarse loopeando.
Los lenguajes no recursivos son aquellos que no se pueden decidir con una MT que siempre pare.
Los no recursivamente numerables son lenguajes los cuales no tienen una MT que lo acepte.

7) $R \subseteq RE$ se demuestra por definición, la definición de RE es igual que la de R pero con una restricción menos (la que permite a la MT quedarse loopeando). $RE \subseteq \mathfrak{L}$ es demostrado por definición, ya que $\mathfrak{L}$ es el conjunto de todos los lenguajes, por ende cualquier conjunto de lenguajes está incluido en él, entre otros RE.

8) No todos, solo los lenguajes que son CO-RE y a la vez pertencen a R.

9) Ambos lenguajes son recursivos porque cumplen con la definición de R, para ambos se puede conseguir una MT que los acepte y se detenga, para Σ^* tiene que ser una MT que para cualquier entrada vaya a qA, y en el caso de $\emptyset$ tiene que ser una MT que siempre se detenga en qR en cualquier entrada.

10) Falso. Todos los lenguajes $L \subseteq \Sigma^*$, pero no todos los lenguajes pertenecen a R, por ejemplo el HP.

11) Un lenguaje finito es recursivo porque siempre se puede hacer una MT que lo reconozca y pare, de la siguiente manera:
Se crea un conjunto de estados por cada carácter de un string que reconoce el lenguaje, donde cada estado puede ser por ejemplo la posición y el carácter.
Ahora se repite el procedimiento por cada string que reconoce el lenguaje.
Entonces, la MT va a probar cada conjunto de estados para el string que recibe, y si ninguno de los conjuntos acepta, la MT rechaza.

12) Se puede demostrar atendiendo a las definiciones.
Si L1 ϵ CO-RE y L2 ϵ CO-RE, entonces L1c ϵ RE y L2c ϵ RE
Si L1 ∩ L2 ϵ CO-RE entonces L1c ∩ L2c ϵ RE
Como se demostró, la intersección de lenguajes de RE es cerrada, por lo que se cumple que L1c ∩ L2c ϵ RE

13) $\Sigma^* \cap L = L$
$\Sigma^* \cup L = \Sigma^*$
El complemento de L respecto a Σ^* es Lc, o también se puede ver como $\Sigma^* - L$.

Ejercicio 2. Construir una MT, con cualquier cantidad de cintas, que acepte de la manera más eficiente posible el lenguaje $L = \{a^n b^n c^n \mid n \geq 0\}$. Plantear primero la idea general.

M tiene 2 cintas. Con el input w en la cinta 1, M hace:

1. Lee las letras "a" y a medida que lo hace las copia en la segunda cinta, así hasta encontrarse con una letra "b". En caso de encontrarse con otro símbolo que no sea una "b" al finalizar la cadena de "a"s termina en qR.

2. Al encontrarse con la letra "b", primero hace un movimiento para acomodar el cabezal de la cinta 2 al final del string, arriba de la última "a", mientras que en la cinta 1 no se mueve. Una vez preparado, la primera cinta se desplaza a la derecha, mientras que la segunda cinta desplaza a la izquierda. La idea es hacer la misma cantidad de movimientos sobre la cadena de "a"s de la segunda cinta y de "b"s en la primera. Estos movimientos continúan hasta que la primer cadena se encuentre con una "c" y la segunda con un blanco. Cualquier otro evento finaliza la MT en qR (por ejemplo, leer un blanco en la segunda cadena mientras se lee "b" en la primera).

3. Se repite el procedimiento parecido al paso 2. Primero se acomoda el cabezal de la segunda cinta un lugar a la derecha, sobre la primer "a", mientras que en la primer cinta no se realizan movimientos. Luego, se avanza en simultaneo en la cinta 1 y 2, leyendo "c" y "a" respectivamente, y se continúa hasta leer blanco en simultaneo en las dos cintas. Cualquier otra lectura que no sea "B" en la cinta 1 mientras que se lee "B" en la cinta 2 resultaría un error, por lo que la máquina se detendría en qR.

Ejercicio 3. Explicar (informal pero claramente) cómo simular una MT por otra que en un paso no pueda simultáneamente modificar un símbolo y moverse.

Basicamente habría que separar cada "movimiento y modificación" en dos pasos que sean continuos.

Por cada transición genérica, del tipo:
δ'(qi, ak) = (qj, ai, X)

Se deberían generar 2, del tipo:
δ'(qi, ak) = (qjX, ai, S)
δ'(qjX, x) = (qj, x, X)

Con x como cualquier símbolo que pueda escribir la máquina, y X como cualquier movimiento que pueda hacer la máquina. qi y qj dos estados cualquiera que pueda tener la máquina.
qjX representa un estado auxiliar para terminar la operación original.

Por dar un ejemplo:
δ(q0, 0) = (q1, 1, D)

Pasaría a:
δ(q0, 0) = (q1D, 1, S)
δ(q1D, 1) = (q1, 1, D)

Es obvio recalcar que las transiciones en donde ak = ai o el movimiento X = S no necesitan modificación, ya que originalmente no incumplen la regla de "movimiento y modificación".

Ejercicio 4. Sean L_1 y L_2 dos lenguajes recursivamente numerables de números naturales representados en notación unaria (por ejemplo, el número 5 se representa con 11111). Probar que también es recursivamente numerable el lenguaje $L = \{x \mid x$ es un número natural representado en notación unaria, y existen y, z, tales que $y + z = x$, con $y \in L_1, z \in L_2\}$.
Ayuda: la prueba es similar a la vista en clase de la clausura de RE respecto de la concatenación.

Una suma unaria de dos strings es idéntica a la concatenación de los mismos, por ejemplo:

11 (2) + 111 (3) es igual a 11111 (5)

Símbolos útiles:
ⱵλϵδΣΓ∀⊂⊆⇒ℒαΩ
ΘΠ

Por lo tanto, y+z = x es lo mismo que y.z = x

Entonces, L = L1.L2, siendo L1 ϵ RE y L2 ϵ RE, L ϵ RE, ya que sabemos que la concatenación de lenguajes de RE es cerrada.
Se puede utilizar la definición:
Si L1 ϵ RE y L2 ϵ RE, entonces L1 . L2 ϵ RE, siendo L1 . L2 el lenguaje concatenación (o producto) de L1 con L2 , es decir:
L1 . L2 = {w | w = w1w2 , con w1 ϵ L1 y w2 ϵ L2 }, en donde w1 es x, w2 es z y w es x en nuestro problema.

Ejercicio 5. Dada una MT M_1 con $\Sigma = \{0, 1\}$:
1. Construir una MT M_2 que determine si $L(M_1)$ tiene al menos una cadena.
2. ¿Se puede construir además una MT M_3 para determinar si $L(M_1)$ tiene a lo sumo una cadena? Justificar.

Ayuda para la parte (1): Si $L(M_1)$ tiene al menos una cadena, entonces existe al menos una cadena w de unos y ceros, de tamaño n, tal que M_1 a partir de w acepta en k pasos. Teniendo en cuenta esto, pensar cómo M_2 podría simular M_1 considerando todas las cadenas de unos y ceros hasta encontrar eventualmente una que M_1 acepte (¡cuidándose de los posibles loops de M_1!).

1) Lo que va a hacer la MT M2 es probar todos los strings de Σ* en M1, por ejemplo, ordenados canónicamente, y cuando M1 acepte, M2 acepta, cuando M1 rechaza, M2 vuelve a ejecutar el M1 con el próximo string. En caso de que M1 no acepte nunca, M2 estaría rechazando por loop.
Como M1 no necesariamente rechaza en qR, se considera que la maquina 1 loopea cuando se realizan una cantidad k de pasos.
Para esto, se podría tener una máquina de 5 cintas:
En la primera está la entrada w que le llega a M2, que es indiferente.
Sobre la cinta 2 la M2 va a crear los strings que va a probar en M1 de forma canónica.
En la cinta 3 se va a guardar el número "k", que es la cantidad de pasos máxima que puede hacer M1 antes de que se considere que entró en loop.
Sobre la cinta 4 va a ir copiando el string que está en la cinta 2 y sobre el que ejecuta M1
Sobre la cinta 5 se va a ir copiando "k" y decrementando por cada paso que haga M1.

2) No, esta máquina M3 no se puede crear, ya que para afirmar que L(M1) tiene a lo sumo una cadena se deberían probar todas las cadenas de Σ*, y después determinar cuantas acepta M1, lo que es imposible ya que la cantidad de cadenas de Σ* es infinita.

Pregunta de parcial: Para probar pertenencia hay que construir, para probar no-pertenecencia hay que hacer otra cosa (como reducciones)