Computabilidad y Complejidad
Ejercicios Unidad Temática: La máquina de Turing

Alumno: Stéphane Díaz-Alejo León

1.-Bosqueje una MT que reconozca el lenguaje L = {x ∈{a,b,c}*/$|x|_a=|x|_b=|x|_c$}.

Utilizaremos una MT multicinta para calcular esta función, en el que en la segunda cinta se marcará el inicio de la palabra. El funcionamiento de ésta MT es el siguiente:

Primero de todo, se buscará un símbolo 'a'. Al hallarlo, se tachará mediante un símbolo blanco, se volverá al principio y se procederá a buscar un símbolo 'b'. Cuando este se encuentre, se tachará con un símbolo blanco, se volverá al principio y se continuará buscando un símbolo 'c'. Al encontrarlo, se tachará con un símbolo blanco, se volverá al principio y entonces se volverá a buscar un símbolo 'a'.

En el caso de que no se encuentre un símbolo 'a', luego un símbolo 'b' y después un símbolo 'c', estaríamos ante el caso de una palabra que cumple la condición del lenguaje, ya que se trata de λ o tiene el mismo número de símbolos por cada símbolo, y por ende la MT se detiene aceptando. En el caso contrario, es decir, se haya un símbolo 'a' pero no un 'b' o se hayan los símbolos 'a' y 'b' pero no el 'c', la MT se pararía rechazando la palabra.

2.-Bosqueje una MT que calcule la función $g(n,m) = \sum_{n \le i \le m} i$.

Utilizaremos una MT multicinta para calcular esta función, en el que en la segunda cinta se marcará el inicio de la palabra. El funcionamiento de ésta MT es el siguiente:

Primero se copiará el valor de n en una tercera y cuarta cinta, y después se procederá a tachar n símbolos 0 de m, es decir, el procedimiento que haríamos con una resta (m - n). Con esta tachadura hemos obtenido el número de iteraciones necesarias sin contar la primera, ya que n ha sido copiado en la cinta que computa el resultado. Una vez finalizado esto, se tacharía el siguiente 0 de m si existe, se añadiría un símbolo 0 a la palabra de la tercera cinta. La palabra de la tercera cinta se concatenaría a la que hubiera en la cuarta. Finalmente se volvería a repetir el proceso hasta que no quedaran más símbolos 0 en m, se añadiría un símbolo 1 después de m, se copiaría la palabra de la cuarta cinta en la cinta de entrada y se colocaría el cabezal en el primer símbolo 0 del resultado, obteniendo el resultado deseado y deteniéndose.

3.-Sean L y L' lenguajes, se define la operación binaria ◇ como sigue

L◇L' = {z/∃x∈L,∃y∈L': z = xy ∧ |x|=|y|}

Se pregunta: Si L y L' son lenguajes recursivamente enumerables, entonces ¿lo es también L◇L'?

Puesto que L es un lenguaje recursivamente enumerable existe una máquina de Turing M(L) que lo acepta. A su vez, ya que L' también es un lenguaje recursivamente

enumerable, existe una máquina de Turing M(L') que lo acepta. En lo que sigue vamos a definir una máquina de Turing M* que acepte $L \diamond L'$.

Dada una z, a ser posible de longitud par, lo cual se puede observar utilizando una máquina de Turing con dos estados que mida la longitud, alternando entre un estado que indica que es par y otro que impar. Esta MT o módulo, se detendría para todas las entradas. Si z es par, ésta se divide en dos mitades, sino, se rechaza. Esta división se puede realizar mediante una máquina de Turing multicinta en la que se recibe en la cinta de entrada z, y se marca en una segunda las dos mitades mediante símbolos distintos. La primera parte, x, se enviará a M(L). Si ésta acepta, la segunda parte, y, se enviará a M(L'). Si ésta también acepta, nuestro reconocedor de $L \diamond L'$ acabará aceptando.

Como hemos logrado describir una máquina de Turing que acepta el lenguaje de la operación deteniéndose en los casos de aceptación, pero no teniendo garantizado la detención con rechazo, se demuestra que $L \diamond L'$ es recursivamente enumerable.

Otra forma de resolver este problema sería con dos generadores, uno del lenguaje L y otro del lenguaje L'. Deberíamos de utilizar un generador de pares (i,j), la i la introduciríamos en G(L) y la j en G(L'), por lo que podríamos generar todos los pares (x,y) posibles. Después de obtener x e y, se introducirían en una MT que midiese si su longitud es igual, si lo es, se concatenan x e y, y se da cómo salida. Finalmente se enviaría una señal de control para generar el siguiente par (i,j). Con esto habríamos obtenido un generador de $L \diamond L'$ no canónico, lo que demuestra que también es recursivamente enumerable.

4.-Sea $M = (\Sigma, \Gamma, Q, f, B, q_0, F)$ una MT y L(M) un lenguaje recursivo. Para un estado $q \in Q$, se define el lenguaje $L_q(M) = \{x \in L(M) / \exists \alpha, \beta \in \Gamma^* : q_0 x \vdash^* \alpha q \beta\}$.

Se pregunta: Si L(M) es un lenguaje recursivo, entonces ¿lo es también $L_q(M)$?

Dada x, se introduce en una máquina de Turing que reconoce L y se detiene para todas las entradas, si ésta acepta se introduce en una versión modificada de M, a la que llamaremos M'. M' tiene de diferentes que su único estado final es el estado q, es más, para que la asunción de que los estados finales sean terminales sea correcta, haremos lo mismo con este estado, es decir, lo convertiremos en un estado de bloqueo. Si la máquina M' acepta, nuestra máquina de Turing reconocedora del lenguaje L_q acepta, en el caso de que se detenga rechazando, rechazamos.

Sabemos que la máquina M' se detendrá para todas las entradas gracias a dos hechos:

1. Sólo introducimos palabras pertenecientes al lenguaje L, por lo que en la máquina original se detenían en un estado de bloqueo que era final.
2. Al convertir el estado q en un estado de bloqueo, la máquina M' se detendrá aceptando con tan solo transicionar a este estado.

Al haber logrado diseñar una máquina de Turing que reconoce L_q y se detiene para todas las entradas aceptando o rechazando, se demuestra que el lenguaje L_q también es recursivo.

5.-Bosqueje una MT que calcule la función min(n,m) que obtiene el valor mínimo entre n y m (si n y m tienen el mismo valor entonces devuelve n).

Utilizaremos una MT multicinta para calcular esta función, en el que en la segunda cinta se marcará el inicio de la palabra. El funcionamiento de ésta MT es el siguiente:

En una tercera cinta se irán "tachando" los símbolos de n y m. Primero se buscará si aún existe un símbolo de n sin tachar y después los de m. En el caso de que no encuentre más símbolos de n, se procederá a borrar mediante símbolos blancos a partir del símbolo 1 de separación, símbolo incluido, y se pondrá el cabezal al principio de la cinta de entrada. En el caso de que no se encuentren más símbolos de m pero sí de n, se colocará el cabezal en el primer símbolo 0 de m. Con esto, en el caso de que n sea menos o igual que m, se devolverá n, y en el caso contrario, m.

6.-Sean: Σ un alfabeto, los lenguajes L y L'$\subseteq \Sigma^*$, y el símbolo # $\notin \Sigma$. Se define la operación μ(L,L') = {x#y / x$\in$L $\wedge$ y$\notin$L'}.

Se pregunta: Si L y L' son lenguajes recursivamente enumerables ¿lo es también μ(L,L')?

Puesto que L es recursivamente enumerable, existe una máquina de Turing M(L) que acepta el lenguaje L. Sin embargo, aunque exista una máquina de Turing M(L') que acepta el lenguaje, el rechazo no lo tenemos garantizado, ya que también es recursivamente enumerable. Podríamos pensar en hacer una MT que acepte el lenguaje complementario, pero esta operación no es cerrada bajo los lenguajes recursivamente enumerables. Con todo esto, podemos llegar a la conclusión que dada la entrada x#y no podemos crear una MT que reconozca la operación μ(L,L') y en el caso de perteneer al lenguaje se detenga aceptando. A su vez, tampoco podemos tratar de hacer un generador de la operación, ya que no nos es posible hacer un generador del complementario de L' con la certeza de que sea recursivamente enumerable.

En el caso de que se demostrase que L' también es recursivo, si que se podría llegar a una solución.

7.-Para cada lenguaje L se define para cada n$\geq$0 P_n(L)={x$\in$L/|x|>2n}. Sea L recursivamente enumerable:

a. **¿Es P_n(L) recursivo para cada n$\geq$0?**

Dado que L es recursivamente enumerable, no podemos utilizar una MT que reconozca L y se detenga para todas las entradas. A su vez, tampoco podemos utilizar un generador de Turing que genere L en orden canónico, por lo que tampoco nos es posible diseñar un generador canónico de P_n(L). Con todo esto, podemos concluir que el lenguaje definido por la operación no es recursivo.

b. **¿Es P_n(L) recursivamente enumerable para cada n$\geq$0?**

Puesto que L es un lenguaje recursivamente enumerable existe una máquina de Turing G(L) que lo genera. Primero, generamos x por G. A su vez, poseeremos una máquina de Turing capaz de generar 2n dado n. Esto se puede hacer mediante una máquina multicinta en la que primero se marca el final de n en una segunda cinta, y después

tacha un símbolo de n en otra cinta, y escribe un símbolo 0 a continuación de la palabra de la cinta de entrada. Siguiendo el proceso hasta que se haya acabado de tachar n, obteniendo 2n. Una vez obtenido x y 2n, éstas se introducen en una máquina M* que comprueba que la longitud de x es mayor que la de 2n. En el caso, de aceptar, nuestro generador da como salida x y se manda una señal de control al generador, en el caso contrario, solo se manda la señal. El módulo que compara las longitudes se detiene para todas las entradas, por lo que la detención está garantizada, tanto aceptando como rechazando Con esto, hemos obtenido un generador de $P_n(L)$ no canónico, por lo que queda demostrado que es recursivamente enumerable.

Otra manera de resolver este problema sería utilizando una Máquina de Turing que reconociese el lenguaje L, de manera que dada una entrada x, se comprobaría si la longitud de x es mayor que 2n con los módulos anteriores, y si acepta se introduciría en el reconocedor de L. Si éste último acepta, entonces aceptamos. En el caso de que no cumpliera la condición de la longitud se rechazaría.

c. ¿Es $P_m(L)$- $P_n(L)$ recursivo para cada $m \geq n \geq 0$?

$P_m(L)$ - $P_n(L)$ es recursivo porque en todos los casos el resultado de la operación es el conjunto vacío. Esto se puede ver, ya que, si al conjunto de todas las palabras de L que son mayores que 2m, le quitas el conjunto de todas las palabras de L que son mayores que 2n, siendo m mayor o igual que n, nos quedamos con un conjunto vacío. Explicado de otra forma, $P_n(L)$ siempre incluye a $P_m(L)$. Al ser un conjunto vacío, podríamos crear una MT que rechace todas las entradas, deteniéndose en todas ellas, por lo que queda demostrado que es recursivo.

d. ¿Es $P_m(L)$- $P_n(L)$ recursivamente enumerable para cada $m \geq n \geq 0$?

Como se ha demostrado en el apartado c, al ser recursivo, es recursivamente enumerable.

8.-Se define la operación P sobre lenguajes como sigue (considere que el alfabeto de definición es {a,b}) $P(L) = \{ a^n w b^n : w \in L \wedge |w| = n \}$. ¿Es la clase de los lenguajes recursivos cerrada bajo la operación P? ¿y la clase de los lenguajes recursivamente enumerables?

Recursivo:

Puesto que L es un lenguaje recursivo, existe una máquina de Turing que lo acepta, deteniéndose para cada entrada. Dada la palabra z, se introducirá en una máquina de Turing que compruebe que la longitud es múltiplo de 3. En el caso de que no sea así, la máquina se detiene rechazando, en el caso contrario, se divide z en tres fragmentos iguales, que denotaremos por xwy. Se introducirá x en una MT que compruebe que todos los símbolos son “a” e y en una MT que compruebe que todos los símbolos son “b”. El fragmento de w se introducirá en M(L). Si las tres MT acaban aceptando, nuestra máquina se detendrá aceptando, en cualquier otro caso, se detendrá rechazando. Esta máquina se detiene para todas las entradas, por lo que la operación es de cierre bajo los lenguajes recursivos.

Recursivamente enumerable:

Puesto que L es un lenguaje recursivamente enumerable, existe una máquina de Turing que lo acepta. Dada la palabra z, se introducirá en una máquina de Turing que compruebe que la longitud es múltiplo de 3. En el caso de que no sea así, la máquina se detiene rechazando, en el caso contrario, se divide z en tres fragmentos iguales, que denotaremos por xwy. Se introducirá x en una MT que compruebe que todos los símbolos son "a" e y en una MT que compruebe que todos los símbolos son "b". El fragmento de w se introducirá en M(L). Si las tres MT acaban aceptando, nuestra máquina se detendrá aceptando. Al no tener garantizado la detención con rechazo, se demuestra que el lenguaje descrito por la operación es recursivamente enumerable.

Otra vía para resolver este problema sería utilizar un generador de L que genere w. Pasaríamos w por dos máquinas de Turing que pasarían todos los símbolos de w a "a" y "b", obteniendo a^n y b^n. Posteriormente se concatenarían los tres fragmentos y se daría una señal de control al generador para que generase la nueva w. Si L fuese recursivo se podría crear un generador canónico. En el caso de que fuese recursivamente enumerable, solo se podría diseñar el generador sin ser canónico.

9.-Sean L1 y L2 lenguajes. Se define L1◇L2 = {xy / x∈L1, y∈L2, |x|<|y|}. Se pregunta:

a. **Si L1 y L2 son recursivos, ¿lo es también P(L)?**

Puesto que L1 es recursivo, existe una máquina de Turing M1(L1) que acepta el lenguaje y se detiene para cada entrada. Sucede lo mismo con L2, obteniendo M2(L2). Dada una palabra z y una i generado por un generador de números canónico denominado G, a una máquina de Turing, M', ésta divide z en los pares (x,y), "i" indicando dónde cortar z. El número generado por G se introducirá en una máquina que comprobará que éste no supere la longitud de z. En este caso, se reiniciará el generador y se detendrá nuestro reconocedor rechazando, puesto que habremos probado todas las divisiones posibles. Estos pares (x,y) se introducirán en una MT, M*, que acepte si la longitud de x es menor que la de y. En el caso de aceptar, se introducirá x en M1 y si ésta acepta, se introduce y en M2. Si ésta última máquina se detiene aceptando, nuestro reconocedor del lenguaje L1◇L2 también acepta. Si M*, M1 o M2 rechaza, se envía solo la señal de control que genera el siguiente número i. Debido a que este reconocedor se detiene para cada entrada, es recursivo.

b. **Si L1 y L2 son recursivamente enumerables, ¿lo es también P(L)?**

Puesto que L1 es recursivamente enumerable, existe una máquina de Turing M1(L1) que acepta el lenguaje. Sucede lo mismo con L2, obteniendo M2(L2). Dada una palabra z y un par (i,j), generado por un generador de pares (i,j) canónico denominado G, a una máquina de Turing, M', ésta divide z en los pares (x, y), el número i indicando dónde cortar z. El número i generado por G se introducirá en una máquina que comprobará que i no supera la longitud de z, en ese caso, se envía una señal de control para generar un nuevo par hasta que i vuelva a iniciarse a 0. Estos pares (x,y) se introducirán en una MT, M*, que acepte si la longitud de x es menor que la de y. En el caso de aceptar, se introducirá x en M1 y si ésta acepta, se introduce y en M2. Si ésta última máquina se detiene aceptando, nuestro reconocedor del lenguaje L1◇L2 también acepta. Si M*

rechaza, se envía la señal de control para generar un nuevo par (i,j). El número j indica el número de pasos que tienen M1 y M2 para aceptar, en caso de que no lo hagan, se envía la señal de control. Puesto que se comprueban todos los posibles cortes de x e y con diferente número de pasos, si algún par pertenece al lenguaje definido por P(L), nuestro reconocedor se detendrá aceptando, demostrando que es recursivamente enumerable.

10.-Bosqueje una MT que calcule la función $g(n) = 2^n$.

Utilizaremos una MT multicinta para calcular esta función, en el que en la segunda cinta se marcará el inicio de la palabra. También tendremos una tercera para hacer tachaduras, una cuarta para un cómputo y la quinta será la cinta en el que se obtendrá el resultado. El funcionamiento de ésta MT es el siguiente:

Primero de todo, se colocará en la quinta cinta un 0. En el caso de que n = 0, ya tendremos la solución, el resultado se escribe en la cinta de entrada después del primer blanco y se coloca el cabezal en el símbolo blanco mencionado. En el resto de los casos, se tachará un 0 de la palabra de entrada escribiendo en la tercera cinta, se pondrá a blancos toda la cuarta cinta, se copiará la quinta cinta en la cuarta, y finalmente se copiará el valor de la cuarta cinta inmediatamente después del último 0 de la quinta. Con esto habremos obtenido el valor 2^n en la quinta cinta, solo faltaría escribir un 1 inmediatamente después del último 0 de la cinta de entrada, copiar el resultado después de éste 1 y colocar el cabezal en el primer 0 del resultado.