Máquinas de Turing

Serafín Moral García
Cristina Zuheros Montes

14 de abril de 2015

1. Veamos la equivalencia entre enumeradores y máquinas de Turing.

   Enumerador $\Rightarrow$ Máquina de Turing

   La máquina de Turing trabajará del siguiente modo:

   Supongamos que tenemos una cadena w. Ejecutamos el enumerador, comparando cada una de las cadenas que imprime con w, y si coincide, es aceptada por la máquina terminando. Está claro que las palabras que acepta la MT son las que son enumeradas

   Máquina de Turing $\Rightarrow$ Enumerador

   El enumerador funcionará de la siguiente forma:

   Sea $s_1, s_2, , , ,$ una lista ordenada de todas las posibles cadenas en A^*, siendo A el alfabeto de entrada.

   Para cada $i = 1, 2, ...$ Para cada $s_j \in \{s_1, s_2, , , s_i\}$

   Si M acepta s_j en i pasos y no ha imprimido antes s_j Imprimir s_j.

   Está claro que el enumerador acabará imprimiendo todas las palabras aceptadas por la MT.

2. Si el alfabeto de entrada es $\{0, 1\}$ se ha visto en clase que se puede asignar un número natural a cada máquina de Turing. Supongamos que tenemos ahora un alfabeto arbitrario. Nosotros vamos a considerar como equivalentes aquellos alfabetos que tienen el mismo cardinal para así poder asignar a un símbolo del alfabeto una codificación. En otro caso, no tenemos una forma de asignar una codificación a un símbolo cualquiera. Por ejemplo, si tenemos el alfabeto $\{a, b, c\}$, la a la codificamos como 0, la b como 00 y la c como 000. Según hemos visto en clase, el blanco se codifica como una sucesión 000. Así no hay forma de distinguir el blanco del tercer símbolo. Una posible solución sería poner delante de las transiciones una sucesión de n ceros, donde n es el cardinal del alfabeto de entrada. Asi el blanco y los otros símbolos del alfabeto de entrada se codifican como n+1 0s y así sucesivamente.

3. Sea $i \in \{1, 2, , , , k\}$. Por hipótesis L_j es recursivamente enumerable $\forall j = 1, , , kj \neq i$. Como la clase de los lenguajes recursivamente enumerables es cerrada para la unión la unión de todos los $L_j, j \neq i, j = 1, , , k$ es recursivamente enumerable. Pero como los L_j forman una partición de A^* este último conjunto es el complementario de L_i, por lo que el complementario de L_i es recursivamente enumerable. Por consiguiente, L_i y su complementario son recursivamente enumerables, de donde concluimos que L_i es recursivo $\forall i \in \{1, 2, , , k\}$

4. Como L no es recursivo podemos afirmar que su complementario no es recursivamente enumerable. Vamos a reducir el complementario de L a L' y después al complementario de L' para concluir que ninguno de los dos son recursivamente enumerables.

   Reducción de L^c a L':

Sea w una cadena del alfabeto $\{0,1\}$. La función de reducción f consistirá en llevarnos w a $f(w) = 1w$. De esta manera $w \in L^c \Leftrightarrow f(w) in L'$. Por lo que hemos reducido L^c a L'. Como L^c no es recursivamente enumerable, tampoco lo será L'.

Llamamos L'' al complementario de L'.

$L'' = \{1w | w \in L\} \cup \{0w | w \in L^c\}$

Ahora cada cadena w nos la llevamos a $0w$ y razonamos como en el caso anterior. Con esta reducción el complementario de L' tampoco es recursivamente enmerable.

5. *a*) Unión: La clase de LRE es cerrada bajo unión:

   Si L_1 pertenece al lenguaje L entonces existe una máquina de Turing M_1 tal que $L(M1) = L_1$ Si L_2 pertenece al lenguaje L entonces existe una máquina de Turing M_2 tal que $L(M2) = L_2$

   Luego existe una máquina de Turing M tal que $L(M) = L_1 \cap L_2$

   Dicha máquina procede del siguiente modo:

   Para cualquier entrada x, la salida de M será sí si la salida de alguna de las máquinas $M1 y M2$ es sí. Ambas máquinas, M_1 y M_2 trabajarán en paralelo para la misma entrada x.

   La clase de LR es cerrada bajo unión:

   Si L_1 pertenece al lenguaje L, entonces existe una máquina de Turing M_1 tal que $L(M1) = L_1$

   Si L_2 pertenece al lenguaje L, entonces existe una máquina de Turing M_2 tal que $L(M2) = L_2$

   Luego existe una máquina de Turing M tal que $L(M) = L_1 \cap L_2$

   Dicha máquina procede del siguiente modo:

   Para cualquier entrada x, x entra en M_1.

   Si la salida en M_1 es sí, entonces la salida de M es sí. Si la salida en M_1 es no, x entra en la M_2, y M nos da la salida de M_2.

   b) Intersección:

   Las clases de LRE es cerrada bajo intersección:

   Para una entrada x en M, x entra en M_1. Si la salida de M_1 es sí, entonces x entra en M_2. Si la salida de M_2 es sí, entonces M nos da como salida sí.

   Las clases de LR es cerrada bajo intersección:

   Para una entrada x en M, x entra en M_1:

   Si la salida de M_1 es no, entonces M nos da como salida no. Si la salida de M_1 es sí, x entra en M_2: Si la salida de M_2 es no, entonces M nos da como salida no. Si la salida de M_2 es sí, entonces M nos da como salida sí.

   c) Concatenación.

   Las clases de LRE es cerrada bajo concatenación:

Para una entrada x, hacer en paralelo, para cada una de las posibles formas de dividir x de forma x_1x_2: x_1 entra en M_1 y x_2 entra en M_2. Si tanto x_1 como x_2 nos da como salida sí, M nos da como salida sí.

Las clases de LR es cerrada bajo concatenación:

Para una entrada x, tenemos un módulo A que divide x en dos mitades, sean x_1 y x_2, comenzando con $x1 = \epsilon$ y $x2 = n$

x_1 pasa a la máquina M_1.

Si la salida de M_1 es sí, entonces x_2 entra en M_2: Si la salida en M_2 es sí, M nos da como salida sí. Si la salida en M_2 es no, entramos en el módulo A y x se divide de nuevo, en este caso x_1 aumenta una unidad y x_2 decrementa. Se repite el proceso.

Si la salida de M_1 es no, entramos en el módulo A, se hace la división y se repite el proceso.

d) Clausura.

Las clases de LRE es cerrada bajo la clausura de Klein:

Para una entrada x, si x es ϵ entonces M nos da como salida sí.

En caso contrario: haz en paralelo, para cada uno de los diferentes modos de dividir x como $x_1 \ldots . x_k$, entra x_i en M_1 y M nos da como salida sí, si M_1 da como salida sí para cada x_i donde $i = 1, .., k$.

Las clases de LR es cerrada bajo la clausura de Klein:

Para una entrada x, si x es ϵ entonces M nos da como salida sí.

En caso contrario: haz en paralelo, para cada uno de los diferentes modos de dividir x como $x_1 \ldots . x_k$, entra x_i en M_1 y M nos da como salida sí, si M_1 da como salía sí para cada x_i donde $i = 1, .., k$. Si para alguno de ellos, M_1 nos da como salida no, M nos dará como salida no.

e) Homomorfismo

Las clases de LRE es cerrada bajo homomorfismo h:

Para una entrada x, empieza recorriendo todas las posibles palabras w y si $h(w) = x$ entonces M_1 empieza a funcionar con w. Si M_1 nos da por salida sí, entonces M nos da por salida sí. La búsqueda de las posibles palabras ha de hacerse en paralelo.

Las clases de LR no es cerrada bajo homomorfismo h:

Consideramos el lenguaje L de la representación de (M, w) seguida de n símbolos c tales que M acepta w en un número de pasos menor o igual a n. Este lenguaje es recursivo. Sea ahora el homomorfismo $h(0) = 0, h(1) = 1, h(c) = \epsilon$. Se verifica que la imagen por h de $(M, w)ccccc$ es (M, w). De esta manera, una palabra pertenece a $h(L)$ si M acepta w, esto es, $h(L)$ es el lenguaje universal, que sabemos que no es recursivo.

Luego la clase de lenguajes recursivos no es cerrada bajo homomorfismos.

f) Homomorfismo inverso

Las clases de LRE es cerrada bajo homomorfismo inverso:

Para una entrada x, calculamos $h(x)$ y pasamos $h(x)$ a M_1. Si M_1 nos da por salida sí, entonces M nos da por salida sí.

Las clases de LR es cerrada bajo el homomorfismo inverso:

Para una entrada x, calculamos $h(x)$ y pasamos $h(x)$ a M_1.

Si M_1 nos da por salida sí, entonces M nos da por salida sí. Si M_1 nos da por salida no, entonces M nos da por salida no.

6. Primero reducimos L_w a L_u

   Sea M una máquina de Turing y w una entrada. M para si y sólo si entra en una configuración para la cual no hay definida ninguna transición. Nuestra máquina M' que contruimos funciona igual que M pero para aquellas configuraciones para las que M no tiene definida transición en M' le añadimos una transición que nos lleva a un estado de aceptación. De este modo que M pare para la entrada w es lo mismo que decir que M' acepte w.

   Esto prueba que L_w es recursivamente enumerable.

   En segundo lugar reducimos L_u a L_w

   Sea M una máquina de Turing y w una entrada. Si M rechaza w entonces para la entrada w alcanzará un estado de rechazo. La máquina de Turing M' que pretendemos construir funcionará igual que M pero para los estados de rechazo se añaden transiciones que nos lleven a un estado desde el cual siempre se mueve a la derecha la cinta, dejándolo el mismo símbolo que va leyendo en la misma y ciclando así indefinidamente. De este modo, M rechaza w si, y sólo si, M' cicla indefinidamente para la entrada w. Esto concluye que L_w no es recursivo

7. *a*) Para ver que es recursivamente enumerable empleamos el siguiente algoritmo no determinista: Escribismos dos paabras de forma no determinista, comprobamos si son distintas. En caso afirmativo, comprobamos de la misma forma que en la demostración de que *Lne* es recursivamente enumerable si pertenecen al lenguaje aceptado por la máquina de Turing que nos dan, simulando la MT sobre las dos palabras. En caso afirmativo obtenemos que el lenguaje acepta al menos dos palabras distintas. Este algorito acepta las MT que aceptan dos palabras distintos y este lenguaje es recursivamente enumerable.

      Hay lenguajes recursivamente enumerables que tienen al menos dos palabras distintas y lenguajes recursivamente enumerables que no. Por tanto, se trata de una propiedad no trivial de lenguajes recursivamente enumerables. Del Teorema de Rice se sigue que este lenguaje es no recursivo

   b) Vamos a ver que este lenguaje no es recursivamente enumerable.

En primer lugar vamos a reducir el complementario del lenguaje universal al problema de ver si el lenguaje aceptado por una máquina de Turing es finito.

Sea una máquina e Turing M y una entrada w. Construimos una máquina de Turing M' que olvida su entrada y funciona como M sobre w. M' escribe w sobre la cinta de entrada y pasa al estado inicial de M y funciona como M. Si M acepta w entonces M' acepta todas las palabras, y si no aceptará el vacío, luego claramente el lenguaje aceptado por M' es finito si y solo si M rechaza w.

Luego saber si una MT acepta un lenguaje finito no es recursivamente enumerable.

Ahora vamos a reducir el complementario del lenguaje universal al problema de ver si el lenguaje aceptado por una máquina de Turing es infinito.

Sea una máquina e Turing M y una entrada w. La máquina de Turing M' que construiremos escribe w en una segunda cinta y simula M sobre w en esta cinta en un número de pasos que es igual a $|u|$ donde u es la palabra de entrada a M'. Si M acepta w en $|u|$ pasos M' rechaza la entrada y si M no acepta w en $|u|$ pasos M' acepta la entrada. Por tanto, si M no acepta w M' acepta todas las palabras, (el lenguaje aceptado por M' sería infinito), y si M acepta w en n pasos M' acepta aquellas palabras de longitus menor que n.

Por tanto saber si una MT acepta un lenguaje infinito no es recursivamente enumerable.

c) Vamos a reducir el complementario de Lu al lenguaje que nos dan.

Consideramos una máquina de Turing M y una entrada w. Vamos a considerar tambien una máquina de Turing M' cuyo lenguaje aceptado sea distinto del inverso (por ejemplo la MT que solo acepta 01). La máquina de Turing $M*$ que construiremos coloca w en la cinta y empieza funcionando como M. Si M rechaza w entonces $M*$ rechaza la entrada automáticamente. Si M acepta w, $M*$ pasa a funcionar como M' sobre la entrada u, aceptando el mismo lenguaje. Si M rechaza w entonces $M*$ acepta el lenguaje vacío (que es trivialmente igual a su inverso). Si M acepta w, entonces $M*$ acepta un lenguaje que no es igual a su inverso. Claramente el lenguaje aceptado por $M*$ es igual a su inverso si y solo si M rechaza w.

Luego saber si una MT acepta un lenguaje igual a su invers no es recursivamente enumerable.

d) Como en el caso anterior, vamos a reducir el complementario de Lu al lenguaje dado.

Sea una máquina de Turing M y una entrada w. Consideramos tambien una máquina de Turing M' tal que el lenguaje que acepta no es independiente del contexto (por ejemplo el lenguaje $\{0^n1^n0^n|n \geq 1\}$). La máquina de Turing $M*$ que construiremos coloca w en la cinta y

empieza funcionando como M. Si M rechaza w entonces $M*$ rechaza la entrada original. Si la acepta $M*$ pasa a funcionar como M' y acepta el mismo lenguaje. Si M rechaza w entonces $M*$ acepta el lenguaje vacío (independiente del contexto). Si M acepta w, entonces $M*$ acepta un lenguaje que no es independiente del contexto. Claramente el lenguaje aceptado por $M*$ es independiente del contexto si y solo si M rechaza w

Luego saber si el lenguaje aceptado por una MT es independiente del contexto no es recursivamente enumerable.

e) Un algoritmo no determinista que puede ver si una máquina de Turing con dos o más estados entra alguna vez en un estado q sería e siguiente:

Primero se genera una palabra w de forma no determinista. Entonces simulamos la máquina de Turing para la entrada w. Si se alcanza el estado q obtenemos la respuesta sí a problema dado. En caso contrario se rechaza o se cicla de forma indefinida. Por consiguiente, el lenguaje es recursivamente enumerable.

Para ver que no es recursivo reducimos *Lne* al lenguaje dado. Para ello vamos a suponer máquinas de Turing con un sólo estado de aceptación. Entonces que acepten un lenguaje distinto del vacío es lo mismo que decir que alcancen ese estado de aceptación. La reducción consiste en dada una MT, preguntar si el estado de aceptación es accesible.

f) El lenguaje que nos dan es claramente recursivamente enumerable. Para verlo simplemente empezamos a simular la máquina con la cinta en blanco y si en algún momento escribe un 1 obtenemos la respuesta sí al problema que nos dan. Vamos a ver que no es recursivo.

Claramente, el lenguaje del conjunto de las máquinas de Turing que aceptan la palabra vacía es no recursivo, como consecuencia del teorema de Rice. Reducimos este lenguaje al que nos dan. Sea una MT M, entonces construimos una MT M' que funciona igual que M, excepto que cuando M llega a un estado final, M' pasa a un estado q' en el que borra la entrada y escribe un 1. Está claro que M acepta la palabra vacía si y solo si M' escribe un 1 cuando empieza con la palabra vacía como entrada.

Por lo tanto, este lenguaje no es recursivo.

g) Un algoritmo no determinista que puede ver si el lenguaje aceptado por una máquina de Turing M acepta al menos un palíndromo sería el siguiente:

Escribe una apalabra cualquiera w de forma no determinística. Comprueba entonces si es un palíndromo y si es aceptado por M y en caso afirmativo, obtenemos la respuesta sí al problema original. Por tanto, el lenguaje es recursivamente enumerable.

Hay lenguajes recursivamente enumerables que admiten al menos un palíndromo y otros que no, luego se trata de una propiedad no trivial de los lenguajes recursivamente enumerables. El Teorema de Rice nos permite concluir que este lenguaje no es recursivo.

h) Vamos a ver que este lenguaje no es recursivamente enumerable reduciendo el complementario del lenguaje universal a él. Sea una máquina de Turing M y una entrada w. Construimos una máquina de Turing M' que olvida su entrada y funciona como M sobre w. Para una entrada u de M', si M acepta w en $|u|$ pasos entonces M' cicla indefinidamente. En caso contrario M' para.

Esto es una reducción del complementario del lenguaje universal al lenguaje de las MTs que se paran para cualquier entrada.

i) Igual que en el caso anterior, reduciremos el complementario de Lu a este lenguaje, concluyendo así que no es recursivamente enumerable. Consideramos M una máquina de Turing y w su entrada. Construimos una máquina de Turing M' que olvida su entrada y funciona como M sobre w. Si M acepta w entonces para. En caso contrario entra en un estado en el que cicla. Es claro que M' para si y solo si M acepta w, esto es cicla para alguna entrada si y solo si M no acepta w.

Como consecuencia de esta reducción, este lenguaje es no r.e.

j) Sea M una máquina de Turing. Un algoritmo no determinista que acepta este lenguaje es la siguiente: Escribe de forma no determinista una palabra w, y por el ejercicio 6 sabemos que existe un algoritmo que en caso de que M pare para esa entrada nos dice si. Por lo que este lenguaje es recursivamente enumerable.

Este lenguaje no puede ser recursivo, pues en tal caso su complementario sería recursivamente enumerable, y ya hemos visto en el caso anterior que su complementario no es recursivamente enumerable.

k) Vamos a reducir el complementario del lenguaje universal a este lenguaje.

Sea M una máquina de Turing y w una entrada. Construimos una máquina de Turing M' que olvida su entrada y funciona como M sobre w. Si M acepta w entonces para. En caso contrario entra en un estado en el que cicla. Es claro que M' para si y solo si M acepta w. Con esto queda verificado que este lenguaje no es recursivamente enumerable.

l) Un algoritmo no determinista que resuelve este problema sería el siguiente: Primero se escriben de forma no determinista k palabras. Entonces comprobamos que son disjuntas dos a dos. En caso afirmativo, comprobamos si pertenecen al lenguaje aceptado por la máquina de Turing M_1 y después comprobamos si pertenecen al lenguaje aceptado por la máquina M_2. Si todas pertenecen al lenguaje aceptado por ambas máquinas obtenemos la respuesta sí.

Este algoritmo no determinista acepta todas las parejas de MT que aceptan al menos k palabras en común. Luego es recursivamente enumerable.

Para ver que no es recursivo vamos a reducir Lne al lenguaje que nos dan. Sea M una máquina de Turing. Tomamos $M_1 = M$, M_2 una máquina que acepte todas las palabras y $k = 1$. Claramente se tiene que $L(M_1) \cap L(M_1)$ contiene al menos k palabras si y solo si el lenguaje aceptado por M es no vacío.

Como Lne no es recursivo, este lenguaje no es recursivo.

8. *a*) El algoritmo sería el siguiente:

   Empezamos simulando la máquina de Turing. En el momento en el que se escribe un símbolo en blanco en la cinta obtenemos la respuesta Sí. En el momento en el que se repiten estados obtenemos la respuesta no. Como el conjunto de estados de la máquina es finito el algoritmo termina siempre.

   b) El algoritmo funcionaría del siguiente modo:

   Miramos las transiciones de la máquina. Si no hay ninguna en la que se mueva a la izquierda obtenemos la respuesta sí. Si hay alguna en la que sí se mueva a la izquierda miramos di dicho estado es accesible. En caso afirmativo podemos encontrar una cadena que nos lleve a ese estado. A dicha cadena le añadimos el símbolo que da lugar a que la máquina se mueva a la izquierda para obtener la respuesta no.

   c) El algoritmo funcionaría así:

   Empezamos simulando la máquina de Turing para esa entrada. Si emppieza moviéndose a la derecha y en algún momento se mueve a la izquierda obtenemos la respuesta no. En el momento en el que lee dos blancos seguidos con el mismo estado obtenemos la respuesta sí. Si lee todos los blancos con todos las combinaciones de estados posibles obtenemos también la respuesta sí.

9. Vamos a probar que si f(n) es una función computable cualquiera, entonces existe un número natural n0, a partir del cual BB(n)>f(n). Trabajaremos con máquinas que, al recibir f(n), dispongan de una cinta con n 1's a la derecha de primer blanco, y para después de un número finito de movimientos con un total de f(n) 1's.

   Defino $F(x) = \sum(f(i)+i^2)$. Si f es computable, entonces F es computable.

   Podremos considerar una máquina de Turing MF que cuando empieza con una cinta con $x1's$, escribe $F(x)1's$ a su derecha, separados por al menos un blanco. Asumimos que MF tiene n estados.

   Considero una máquina de Turing M que primero escribe x 1's en una cinta inicialmente blanca y luego para en el último 1 (x estados). Luego la máquina imita a MF (n estados más) y por último escribe F(F(x)) 1's

a la derecha del ultimo 1 que habíamos escrito, separado al menos por un blanco. Por tanto nuestra máquina tienen x+2n estados.

Ahora cualquier máquina de Turing del castor atareado con x+2n estados devolverá al menos tantos 1's como M para una entrada de x 1's. Por tanto tenemos

$BB(x+2n) >= x + F(x) + F(F(x))$

Pero por definición, $F(x) \geq x^2$. Existe una constante C tal que $x^2 > x+2n$ para todo $x \geq C$.

Por tanto, $F(x) > x + 2 * n$ para todo $x \geq C$.

Además tenemos que $F(x) > F(y)$ si $x > y$, luego $F(F(x)) > F(x+2n)$ para todo $x \geq C$. Luego

$BB(x+2n) \geq x + F(x) + F(F(x)) > F(F(x)) > F(x+2n) \geq f(x+2n)$ para todo $x \geq C$.

Luego BB es mayor que f a partir de cierto C y como f era arbitrario, entonces $BB(k)$ es no computable. Por ser $BB(k)$ no computable, no puede existir ninguna máquina de Turing que sea capaz de calcular dicha función.

Bibliografía: grail.cba.csuohio.edu/ somos/beaver.ps