PRÁCTICA 2

MAC

Máquinas de Turing

Juan Antonio Velasco Gómez
Miguel Sánchez Maldonado

1. El origen de los términos “recursivamente enumerable” y “recursivo” proviene una variante de la máquina de Turing denominada enumerador. Informalmente, un enumerador es una máquina de Turing con una impresora asociada, que puede utilizar como dispositivo de salida para imprimir cadenas de símbolos. Un enumerador E comienza con una entrada en blanco en su cinta de trabajo. Si no se detiene, podría imprimir una lista infinita de cadenas. El lenguaje enumerado por E es el conjunto de cadenas que eventualmente imprime. Un lenguaje es recursivamente enumerable si existe un enumerador que lo enumera (en cualquier orden y con posibles repeticiones). Como sucede con otros muchos modelos de cómputo, se puede demostrar la equivalencia de los enumeradores con las máquinas de Turing tradicionales.

Solución

Para demostrar la equivalencia de los enumeradores con las máquinas de Turing tradicionales estudiaremos por separado las dos implicaciones (↔)

Implicación hacia la izquierda ←

Vamos a hacer que compruebe todas las cadenas posibles para ver que encajan con las cadenas de nuestro lenguaje. Supongamos que dejamos la máquina de turing comprobando, entonces puede llegar un punto de bucle infinito. Para que se salga, basta con poner un contador para que, dado un tiempo, se salga y siga.

De forma que tendríamos el siguiente código para evitar entrar en bucle infinito.

```
i=1
For sj perteneciente a {s1,...,si}
        MT i pasos para sj
if MT acepta sj en i pasos
        print sj
i++
```

Implicación hacia la derecha →

Ejecutar el lenguaje enumerado por E. Cada vez que imprima una cadena, la vamos a comparar con w, entonces si aparece en E, aceptarla, con esto claramente vemos que acepta aquellas cadenas de E.

2. Discuta la posibilidad de asignar un número natural a cada MT con independencia del alfabeto de entrada.

Solución

Para demostrar esto lo que vamos a hacer buscar una máquina equivalente en binario, porque por la teoría sabemos que esto se convierte en un número siendo binario. Codificamos cada elemento del conjunto con un número representado en binario. Sabemos que el lenguaje que estamos utilizando es finito y por tanto deberíamos buscar que sea divisible por 2^n.

Entonces cada elemento será representado por una n-tupla formada por unos y ceros. A continuación vamos a hacer tres operaciones diferentes, lectura, interpretación y acción. Nuestra máquina tendrá en su estado 2 bloques, uno que almacena el estado anterior de la máquina (el equivalente a la máquina de la que procedemos) y otro para obtener el símbolo (del lenguaje arbitrario que hemos codificado) que toca en este instante. Cuando tenemos el símbolo tras la lectura pasamos al proceso de interpretación que nos dice que hacia la máquina anterior con ese estado y ese símbolo.

Por último aplicaremos la operación de acción que, si es escribir, escribe lo correspondiente pasado a binario, si es desplazamiento, será n-ario por lo que tendríamos que adaptar nuestra máquina.

3. Sean L1, . . . , Lk un conjunto de lenguajes sobre el alfabeto A tales que:

a) Para cada *i*=j,tenemos que:
Li∩Lj =∅.
b) Uki=1 Li = A∗.
c) ∀i ∈ {1,...,k}, el lenguaje Li es r.e.

Demostrar que ∀i ∈ {1, . . . , k}, el lenguaje Li es recursivo.

Solución

Como los L_i son disjuntos y utilizando la hipótesis b), escribimos el complementario como:

$L_i{}^c$ = U desde j=1 (j != i), hasta k de L_j .

A su vez, como por c), sabemos que los L_j son r.e. , entonces la unión también lo será. Luego, sabemos que $L_i{}^c$ es r.e. y, de nuevo por hipótesis, L_i también lo es.
Entonces, deducimos que el lenguaje L_i es recursivo, tal y como queríamos demostrar.

4. Sea L r.e., pero no recursivo. Considérese el lenguaje

$L' = \{0w \mid w \in L\} \cup \{1w \mid w$ perteneciente a $L\}$

¿Puede asegurarse que L′ o su complementario sean recursivos, r.e. o no r.e.?

Solución

L es recursivamente numerable pero no recursivo. Ni L' ni el complementario son recursivamente enumerables. Para ello reducimos Lc a L'

Reducción:

Sea w perteneciente a {0,1}alpha, entonces L' no es recursivamente enumerable.

(Hemos reducido Lc a L', por tanto y como Lc no es recursivamente enumerable entonces tampoco lo será L')

Tomamos f: w ---> 1w

L'c = {1w / w pertenece a L} U {0w / w no pertenece a L}
f: w ---> 0w

entonces L'c no es recursivamente enumerable.

Y por tanto tampoco son recursivos.

5. Estudie si las clases de lenguajes recursivos y recursivamente enumerables son cerradas para las siguientes operaciones:

Solución

Son cerradas en todos los casos (unión, intersección, concatenación, clausura, homomorfismo, homomorfismo inverso). Para demostrarlo basta con probar que es lenguaje recursivamente enumerable. queda probado que es lenguaje recursivo ya que la única diferencia entre los ambos es que el lenguaje recursivamente enumerable puede no parar.

Veámoslo uno a uno:

a) Unión: Dada TM M1 , M2 que aceptan los lenguajes L1 y L2
La máquina de turing que decide la unión de ambos lenguajes L1 $\cup$ L2 , será: para la entrada k , ejecuta M1 y M2 con k, trabajando en paralelo, y acepta si y sólo si alguna lo acepta .

b) Intersección: Dada una entrada cualquiera k de M, k entra en M1.
Si la salida de M1 es sí, k entra en M2 y si la salida de M2 es no, entonces M nos da como salida no y, si la salida de M2 es sí, entonces M nos da como salida sí.

c) Concatenación: Dada una entrada cualquiera k de M, dividimos k en dos mitades, sean k1 y x2, comenzando con k1= ε y k2=n
X1 pasa a la máquina M1 y si la salida de M1 es sí, entonces k2 entra en M2.
Veamos los diferentes casos:

- Si la salida en M2 es sí, M nos da como salida sí y hemos acabado.
- Si la salida en M2 es no, entramos en el módulo A y k se divide de nuevo, en este caso k1 aumenta una unidad y k2 decrementa. Se repite el proceso.
- Si la salida de M1 es no, entramos en el módulo A, se hace la división y se repite el proceso.

d) Clausura: Dada una entrada cualquiera k de M, si k = ε M nos da como salida sí.
Si no, se hace en paralelo, para cada uno de los diferentes modos de dividir k como k1….xr, entra ki en M1 y, si M1 da como salía sí para cada ki (i=1,..,r), M nos da como salida sí.
Si para alguno de ellos, M1 nos da como salida no, M también nos dará salida negativa.

e) Homomorfismo: Dada una entrada cualquiera k de M, empieza recorriendo todas las posibles palabras w y si h(w)=k entonces w entra en M1 usando ensamblado para intercalar con otras ejecuciones de M1.

Veamos los diferentes casos:

- Si M1 nos da por salida sí, entonces M también nos da por salida la misma respuesta
- Si M1 nos da por salida no, entonces M también nos da por salida la misma respuesta

f) Homomorfismo inverso: Dada una entrada cualquiera k de M, calculamos h(k) y pasamos h(k) a la máquina M1. Entonces:

Si M1 nos da por salida sí, entonces M nos da por salida sí. Si M1 da salida negativa, entonces M2 también.

6. El problema de la parada: Demuestre que el lenguaje asociado al conjunto de parejas (M, w) tales que la MT M para cuando tiene a w como entrada es recursivamente enumerable pero no recursivo.

Solución

Sabemos que la demostración de que el problema de la parada es un problema indecidible, se basa en que el lenguaje,

$$LH = \{<M,w^*> \mid M \text{ para con } w\}$$

que sabemos es un lenguaje recursivamente enumerable, pero, sin embargo, no es recursivo.

Veamos que LH es un L.R.E:

Para ello vamos a suponer que el problema de parada es decidible; por lo tanto, la máquina de Turing que reconoce el lenguaje LH, MH, ante una entrada <M,w*> respondería si o no dependiendo de si M para o no para al trabajar sobre w.

Si esto fuera cierto, se podría construir una máquina de turing Md, que reconoce el lenguaje Ld, determinaría si una cadena w_i cualquiera es aceptada por una máquina Mi. Es decir, si la cadena w_i pertenece o no pertenece al lenguaje Ld que estamos utilizando.

Esto es imposible, MH no puede existir tal y como se ha descrito y LH no puede ser, por tanto, un lenguaje recursivo.

7. Determine si los siguientes lenguajes son recursivos, r.e. o no r.e.:

a. **Determinar si el lenguaje de una MT contiene, al menos, dos palabras distintas.**
b. **Determinar si el lenguaje de una MT es finito o infinito.**
c. **Determinar si el lenguaje de una MT es independiente del contexto.**
d. **Determinar si L(M) = (L(M))R.**
e. **Determinar si una MT con dos o más estados entrará alguna vez en un estado q.**
f. **Determinar si una MT termina escribiendo un 1 cuando comienza con una cinta completamente en blanco.**
g. **El conjunto de las MT que aceptan al menos un palindromo.**
h. **El conjunto de las MT que se paran para cualquier entrada.**
i. **El conjunto de las MT que no se paran para ninguna entrada.**
j. **El conjunto de las MT que se paran, al menos, para una entrada.**
k. **El conjunto de las MT que no se paran, al menos, para una entrada.**
l. **El lenguaje L formado por pares de cơ digos de MT mư as un entero (M1,M2,k) tales que L(M1) ∩ L(M2) contiene, al menos, k palabras.**

Solución

a) Vamos a probar que es R.E. basándonos en el siguiente algoritmo no determinista. Vamos a comparar dos palabras para ver si son iguales. Si las palabras son distintas nos basamos en la demostración de que Lne(Lenguaje de las máquinas de Turing que acepta el lenguaje no vacío) es recursivamente enumerable cuando pertenecen al lenguaje que acepta la máquina de Turing, haciendo una simulación de la máquina de Turing sobre las dos palabras. Si pertenecen al lenguaje sabemos que el lenguaje acepta como mínimo dos palabras que sean distintas y este lenguaje será recursivamente enumerable.

Tenemos que hay lenguajes recursivamente enumerables que pueden tener o no dos palabras distintas. Por tanto este lenguaje será no recursivo por el Teorema de Rice.

b. Este lenguaje es no recursivamente enumerable, lo vamos a demostrar reduciendo el complementario del lenguaje universal a comprobar si el lenguaje aceptado por una máquina de turing es finito.

Para ello consideramos M y w entrada. Vamos a construir otra M' que desconoce su entrada y funciona igual que M sobre la entrada w, es decir, escribe sobre la cinta y pasa al estado inicial como hace M. Por tanto si M acepta w entonces M' o acepta todas las palabras o acepta el vacío, por tanto el lenguaje será finito si y solo si M no acepta w. Queda probado que es recursivamente enumerable.

Ahora lo que hacemos es al contrario que antes, reducimos el complementario del lenguaje universal a comprobar si el lenguaje aceptado por una MT es infinito. Partimos como antes de M(MT), w(entrada) y M' (MT construimos). Ahora M' escribe la entrada en una segunda cinta y simula M sobre w un número de pasos que será |p|.

- Si M acepta la entrada en |p| pasos -> M' No acepta w
- Si M No acepta la entrada en |p| pasos -> M' acepta w

además en el primer caso M' aceptará las palabras de tamaño menor que p y segundo caso M' aceptará todas las palabras
Si la máquina de Turing acepta un lenguaje infinito no es recursivamente enumerable.

c. Vamos a probar que es no recursivamente enumerable. Para ello reducimos el complementario de LU al lenguaje que nos dan. Como antes consideramos M(MT) y w(entrada) y nuestra M' acepta aquel lenguaje que no es el inverso y M*(MT que construimos). M* colocará w en la cinta y funciona como M. Cuando M rechaza w M* funciona y M' sabrá la cinta y aceptará el mismo lenguaje. Si M rechaza w entonces M* acepta el lenguaje vacío y si M no acepta M* acepta un lenguaje que no es igual a su inverso.

Queda probado que es no recursivamente enumerable

d. Como el apartado anterior es no recursivamente enumerable y vamos a reducir el complementario de Lu al lenguaje que nos dan.
Tenemos M(MT) y w (entrada) y M' que no aceptará lenguaje independiente de contexto. M* que construimos coloca w en la cinta y empieza a funcionar como M.
Si M no acepta la entrada , M* tampoco, por el contrario si M la acepta entonces M* pasa funcionar como M' y aceptará el lenguaje independiente de contexto, es decir el lenguaje vacío. Por tanto el lenguaje aceptado por M* es independiente del contexto si y solo si M rechaza w y por tanto queda probado que es recursivamente enumerable.

e. Un algoritmo no determinista que puede ver si una máquina de Turing con dos o mas estados entra alguna vez en un estado q será el siguiente:
Generamos una palabra w no determinista. Simulamos la MT para esa entrada. Si alcanza q se obtiene la respuesta al enunciado, en otro caso cicla de forma indefinida.
Por tanto el lenguaje es recursivamente enumerable.

Probamos que no es recursivo reduciendo el Lne al lenguaje dado. Nuestra MT solo tiene un estado de aceptación, y por consiguiente aceptar un lenguaje distinto del vacio será almacenar ese estado de aceptación. Por tanto solo tenemos que ver si el esado de aceptación es accesible.

f. El lenguaje es recursivamente enumerable. Simulamos una máquina de Turing con la cinta en blanco y si escribe un 1 ya tenemos la solución.

Probamos ahora que no es recursivo. Las Mt que aceptan la palabras vacía no son recursiva por el Teorema de Rice.

Ahora reducimos el lenguaje al que nos dan. Construimos una máquina M' que funciona como la M menos cuando M llega al estado final, que M' pasará al estado q' en el que quitará la entrada y pondrá un 1.
Por tanto lenguaje no Recursivo

g. Un algoritmo no determinista que puede ver si el lenguaje aceptado por una máquina de Turing M acepta al menos un palíndromo será el siguiente;
Construimos una entrada w no determinística. Comprobamos si es un palíndromo y si es aceptado por M. La respuesta será cuando M acepta la palabra. Por tanto el lenguaje es recursivamente enumerable.

Por el Teorema de Rice podemos afirmar que es un lenguaje no recursivo ya que hay lenguajes R.E. que admiten palíndromo y otros no H. Este lenguaje es no recursivamente enumerable, para ellos reducimos el complementario universal a 1. Consideramos M y w y construimos M' que funciona como M sin tener en cuenta su entrada. Para una entrada w' en M', si M acepta en un número de pasos entonces M' cicla. Si M no acepta en número de pasos finito entonces M' se detiene.

h. Vamos a ver que este lenguaje no es recursivamente enumerable reduciendo el complementario del lenguaje universal a l. Sea una máquina de Turing M y una entrada w. Construimos una máquina de Turing M 0 que olvida su entrada y funciona como M sobre w. Para una entrada u de M 0 , si M acepta w en |u| pasos entonces M 0 cicla indefinidamente. En caso contrario M 0 para.

i. Igual que en el caso anterior, reduciremos el complementario de Lu a este lenguaje, concluyendo que no es recursivamente enumerable. Consideramos M una máquina de Turing y w su entrada. Construimos una máquina de Turing M 0 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 sólo si la máquina M acepta w, luego este lenguaje es no recursivamente enumerable.

j. Sea M una máquina de Turing . Un algoritmo no determinista que lo verifica será el 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. El lenguaje no puede ser recursivo.

k. Vamos a reducir el complementario del lenguaje universal a este lenguaje, Sea M y w una entrada. Construiremos una máquina de Turing M' que olvida su entrada y funciona como M sobre w. Si M acepta w entonces se detiene . En caso contrario cicla. M' se detiene si y solo si M acepta w . Este lenguaje es no recursivamente enumerable.

l. Un algoritmo no determinista que resuelve este problema será el siguiente:

Veamos que para cada n palabras son disjuntas dos a dos . Si son disjuntas comprobaremos que pertenecen al lenguaje aceptado por la Máquina de Turing M_1 M,

M_2. Tenemos que $L(M_1)$ intersección $L(M_2)$ tendrá al menos n palabras si y solo si el lenguaje aceptado por M es no vacío.

8. Demuestre que las siguientes cuestiones son decidibles:

a) El conjunto de las MT M tales que, al comenzar con la cinta en blanco, en algún momento escribirán un símbolo no blanco en la cinta.

b) El conjunto de las MT que nunca se mueven a la izquierda.

c) El conjunto de los pares (M, w) tales que M, al actuar sobre la entrada w, nunca lee una casilla de la cinta más de una vez.

Solución

a) Simulamos M con la cinta en blanco. Si escribe algún símbolo, entonces, respuesta es afirmativa. Si repite estados ya sabemos que cicla indefinidamente.

b) Transiciones de M. Si no hay ninguna a la izquierda, entonces la respuesta es afirmativa. Si no, miramos si esos estados son accesibles.

De serlo, cogemos una entrada con la que lleguemos a ese estado y le añadimos el símbolo que da lugar a que nos movamos a la izquierda. Y entonces la respuesta es que no pertenece al grupo.

(Nota: No se puede simular la máquina porque es válida cualquier entrada)

c) Empezamos simulando M para w. Si se mueve a la izquierda después de moverse a la derecha entonces no pertenece al grupo. Si lee dos símbolos en blanco entonces la respuesta es afirmativa.

Se puede aplicar el mismo razonamiento si empieza moviéndose a la izquierda.

9. El castor atareado [busy beaver]: Imagine una mٞ aquina de Turing quecon una cinta completamente en blanco y, eventualmente, para. Si la máquina dejan unos en la cinta cuando para, diremos que la productividad de la máquina es n. También diremos que la productividad de una máquina de Turing que no 0. La productividad, por tanto, es una función de nٞ umeros naturales (*cٞ* odigos de m´aquinas de Turing) a números naturales. Escribiremos p(T)=n para indicar que la productividad de la mٞ aquina T es n. De entre las mٞ aquinas de Turing con un n´umero concreto de estados, existe una productividad mٞ aquina que una mٞ aquina de Turing con ese nٞ umero de tener. Esto nos permite definir otra función de n´umeros naturales (el número de estados) a números naturales (la productividad mٞ axima de una mٞ aquina con ese nٞ umero de estados). Indicaremos mediante BB(k)=n el hecho de que la productividad mٞ axima de una mٞ aquina de Turing con k estados es n. Pueden existir múltiples mٞ aquinas de Turing con k estados que alcancen dicha productividad, las que llamaremos "castores atareados" [busy beaver]. Demuestre que no existe ninguna máquina de Turing que sea capaz de calcular la función BB(k); esto es, que comenzando con una cinta con k 1's se detenga cuando en la cinta queden BB(k) 1's.

NOTA: La función BB(k) crece más rápido que cualquier función computable.

Solución

La función BB que nos dice el enunciado es no computable, entonces no puede existir ninguna máquina de Turing que pueda calcular la función. BB(k), esto es, que comenzando con una cinta con k 1's se detenga cuando en la cinta queden BB(k) 1's

(La nota que se adjunta en el ejercicio nos dice que la función BB es no computable)