Universidad Simón Bolívar
Teoría de la Computación
Tarea I

Fabiola Di Bartolo
Carnet: 09-87324

1 de febrero de 2010

Los siguientes ejercicios fueron resueltos consultando los libros [1] [2] [3]

1. **Ejercicio 3.6.** En el Teorema 3.21 mostramos que un lenguaje es Turing-reconocible ssi algún enumerador lo reconoce. ¿Por qué no usamos el siguiente algoritmo para probar la implicación en la dirección $\Rightarrow$? Como antes,$s_1, s_2, ...$ es una lista de todos los strings en $\sum^*$.

   E = Ignora la entrada.

   1. Repetir lo siguiente para $i = 1, 2, 3...$
   2. Correr M en s_i.
   3. Si acepta, imprimir s_i.

   Sea L un lenquaje Turing-reconocible y M una máquina de Turing tal que $L(M) = L$.

   Un enumerador reconoce a L, si todas las palabras en L son reconocidas por él. Sin embargo, el enumerador E que se muestra arriba, no puede ser utilizado para la demostración debido a que sólo se tiene como hipótesis que L es Turing-reconocible y esto no implica que L sea decidible, por lo tanto, M podría no ser un *decider* y en una entrada s_i tal que $s_i \in \sum^*$ y $s_i \notin L$, podría no rechazar y quedarse ciclando, por lo que las palabras siguientes: $s_{i+1}, s_{i+2}, ...$ aún cuando sean aceptadas por M no se imprimirían nunca.

   En cambio, si observamos la definición de E planteada en el Teorema 3.21:

   E = Ignora la entrada.

   1. Repetir lo siguiente para $i = 1, 2, 3...$
   2. Correr M por i pasos en cada entrada $s_1, s_2, ..., s_i$.
   3. Si con alguna de acepta, imprimir el correspondiente s_j.

M va procesando la lista de palabras colocando una cota en la cantidad de pasos que ejecuta M para evitar ciclos infinitos, comenzando con 1 y agregándole un paso en cada iteración, por lo tanto si M acepta una palabra, eventualmente será impresa por E.

2. **Ejercicio 3.12.** Una máquina de Turing con *left reset* es similar a una máquina de Turing ordinaria, pero la función de transición tiene la forma:

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, RESET\}$$

Si $\delta(q, a) = (r, b, RESET)$ entonces la maquina está en el estado q leyendo a, la cabeza de la máquina salta al extremo más izquierdo de la cinta después de escribir b en la cinta y entrar en el estado r. Notar que estas máquinas no tienen la habilidad usual de mover la cabeza un símbolo a la izquierda. Mostrar que las máquinas de Turing con *left reset* reconocen la clase de lenguajes Turing-reconocibles.

Dado un lenguaje L Turing-reconocible, existe por definición, una máquina de Turing estandar M' que lo reconoce, por lo tanto se necesita demostrar que una máquina de Turing M con *left reset* es equivalente a M', $L(M) = L(M')$ simulando una con la otra y viceversa.

Simulando M a partir de M'

La idea es colocar un marcador al inicio de cada palabra a reconocer, para de esta forma recordar donde inicia la cinta y poder simular el RESET.

- ⋆ La máquina M' es similar a M, con la diferencia que la función de transición es $\delta' : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ y el alfabeto incluye el marcador de la cinta $\Gamma \cup \{\#\}$ (suponiendo que $\# \notin \Gamma$).

- ⋆ Al recibir una palabra w: se desplazan todos los caracteres una casilla hacia la derecha y se escribe en la primera casilla de la cinta el caracter # y se mueve la cabeza una casilla a la derecha

para que quede lista para la lectura inicial.

⋆ Si M mueve la cabeza hacia la derecha: la acción es la misma en ambas máquinas, M' mueve la cabeza un caracter a la derecha.

⋆ Si M hace RESET: la acción se simula moviendo la cabeza de M' a la izquierda hasta la marca # y luego se mueve la cabeza una casilla a la derecha para situarla en el primer caracter de la palabra.

Simulando M' a partir de M

La idea es marcar la casilla donde se encuentre la cabeza, para recordar donde estaba situada antes de hacer el RESET para simular un desplazamiento a la izquierda.

⋆ La máquina M es similar a M', con la diferencia que la función de transición es $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, RESET\}$ y el alfabeto incluye los marcadores, uno por cada letra del alfabeto $\Gamma \cup \{\dot{a} : a \in \Gamma\}$.

⋆ Si M' mueve la cabeza hacia la derecha: la acción es la misma en ambas máquinas, M mueve la cabeza un caracter a la derecha.

⋆ Si M' mueve la cabeza hacia la izquierda: M tiene que guardar la posición de la cabeza antes de hacer el RESET y eso lo hace marcando el caracter con un punto.

1. Marca el caracter actual con un punto y realiza RESET.

2a. Lee el caracter, si está marcado, significa que la cabeza estaba situada en la primera casilla y no puede moverse mas a la izquierda, así que se desmarca el caracter, termina esta simulación y continua con el resto de las operaciones.

2b. Si no está marcado, lo marca y realiza RESET.

3. Mueve la cabeza a la derecha hasta encontrar el primer caracter marcado, y se mueve la cabeza a la derecha de nuevo.

4a. Si el caracter está maracado, entonces la cabeza se encuentra en la posición original, así que lo desmarca y realiza RESET. Mueve la cabeza a la derecha hasta encontrar el primer caracter marcado, y como es el caracter a la izquierda de la posición original, se desmarca, termina esta simulación y continua con el resto de las operaciones.

4b. Si el caracter no está marcado, se marca y se realiza RESET. Se mueve la cabeza hacia la derecha hasta encontrar el primer caracter marcado, lo desmarca y realiza RESET. Se repite el proceso desde el paso 3.

Por lo tanto, hemos demostrado que ambas maquinas de Turing M (con *left reset*) y M' (estandar) son equivalentes, es decir, reconocen los mismos lenguajes.

3. **Ejercicio 3.18.** Mostrar que un lenguaje es decidible ssi algún enumerador lo reconoce en orden lexicográfico.

Si un lenguaje es decidible, es reconocible por alguna máquina de Turing, y por el Teorema 3.21, un lenguaje es Turing-reconocible ssi un enumerador lo reconoce.

($\Rightarrow$) Suponemos que M reconoce a L y es un decider, construimos un enumerador E para L. Suponemos que $s_1, s_2, ...$ es la lista de todas las posibles palabras en $\sum^*$.

E = Ignora la entrada.

1. Repetir lo siguiente para $i = 1, 2, 3...$
2. Sea s_i la i-ésima palabra en orden lexicográfico.
3. Correr M en s_i.
4. Si M acepta, IMPRIMIR s_i.

Sabemos que por ser M un decider, terminará y devolverá siempre una respuesta, así que no necesita que las palabras sean ordenadas lexicográficamente, todas las palabras reconocidas por M lo serán por E.

($\Leftarrow$) Supongamos que E reconoce a L en orden lexicográfico. Si L es finito, es evidente que es decidible. Así que suponemos que L es infinito, para esto construimos el siguiente decider.

M = En entrada w.

1. Correr E.
2. Cada vez que E imprima una palabra s_i, se compara con w.
3. Si son iguales, ACEPTAR.
4. Sino lo son y w precede lexicográficamente a s_i ($w \prec s_i$), RECHAZAR.

De esta forma, como las palabras van saliendo del enumerador en orden lexicográfico, aparecen primero las de menor tamaño, así que si w precede a la última que fue impresa por E y en la lista nunca fue impresa, entonces M rechaza porque $w \notin L$. Por lo tanto, sabemos que M parará cuando la palabra sea reconocida por el enumerador, o cuando el orden lexicográfico la palabra s_i impresa por E sea mayor que el de w.

En conclusión, $L = L(M) = L(E)$ ssi E reconoce a L en orden lexicográfico.

4. **Ejercicio 4.6.** Sea B el conjunto de todas las secuencias infinitas sobre $\{0, 1\}$. Mostrar que B no es contable, usando diagonalización

   Por contradicción, supongamos que B es contable y utilizando el método de diagonalización existe una función biyectiva $f : N \to B$, a continuación se muestra una tabla con algunas correspondencias:

n	f(n)
1	$\underline{1}$010101...
2	0$\underline{1}$01010...
3	11$\underline{0}$1101...
.	.
.	.
.	.
i	x

   Construcción de la secuencia x

   La secuencia infinita binaria x se construye colocando el bit i opuesto al bit i de la secuencia $f(i)$ con $i = 1, 2, 3...$, así $x = 001...b....$, pero el problema ocurre con el valor de b, ya que debería ser opuesto al iesimo bit de la secuencia $f(i)$, y esto debe ocurrir con todas las demás secuencias, por lo que x difiere de todas las secuencias listadas en al menos un bit, el bit de la posición i, para todo i, pero B contiene todas las secuencias, por lo tanto es una contradicción. B no es contable.

5. **Ejercicio 4.12.** Sea $A = \{\langle R, S\rangle : R \text{ y } S \text{ son expresiones regulares y } L(R) \subseteq L(S)\}$. Mostrar que A es decidible.

   Los Teoremas 4.1, 4.2 y 4.3 muestran que para propósitos de decibilidad, presentar una máquina de Turing con un DFA, NFA o expresión regular es equivalente porque la máquina puede convertir una codificación en otra, por lo tanto utilizaremos DFAs para esta prueba, utilizaremos el Teorema 4.5 (EQ_{DFA} es decidible) y la siguiente propiedad:

$L(R) \subseteq L(S)$ ssi $L(R) \cup L(S) = L(S)$.

Sea M el *decider* para EQ_{DFA}. Construimos el *decider* F para A.

$F =$ En entrada $\langle R, S \rangle$

1. Chequear que R y S son expresiones regulares, sino RECHAZAR.
2. Traducir R y S en sus equivalentes DFAs U y V (utilizando el Teorema 1.54)
3. Construir un DFA T que reconozca $L(U) \cup L(V)$, esto es posible porque los lenguajes regulares son cerrados bajo la unión (Teorema 1.25).
4. Correr M en entrada $\langle T, V \rangle$ (determina si son iguales).
5. Si M acepta, ACEPTAR. Sino, RECHAZAR.

Sabemos que el paso 4 siempre terminará porque EQ_{DFA} es decidible, si M acepta es porque R es subconjunto de S, si rechaza es porque no lo son, por lo tanto F es un *decider* para A.

6. **Ejercicio 4.28.** Sea A un lenguaje Turing-reconocible que consiste en las descripciones de máquinas de Turing $\{\langle M_1 \rangle, \langle M_2 \rangle, ...\}$
donde cada M_i es un *decider*. Probar que algún lenguaje decidible D no es decidido por ningún decider M_i cuya descripción aparezca en A.

Sea $A = \{\langle M_i \rangle : M_i \text{ es un decider } \}$ y A Turing-reconocible, lo que se desea probar es que para un lenguaje D tal que $D = L(M)$ y M es un *decider*, $\langle M \rangle \notin A$.

Por el Teorema 3.21 existe un enumerador E que reconoce A.

Asumiendo que $\langle M_1 \rangle, \langle M_2 \rangle, ...$ es la salida del enumerador E y $\sum = \{s_1, s_2, ...\}$ la lista de todas las palabras del alfabeto Γ de E. Definimos el *decider* N para D de la siguiente forma:

N = En entrada w

1. Chequear que $w \in A$, sino RECHAZAR.
2. w es igual a s_i para algún i.
3. Correr E hasta que imprima $\langle M_i \rangle$.
4. Correr M_i en entrada w.
5. Si M acepta, RECHAZAR. Sino, ACEPTAR.

Debido a que cada M_i es un *decider*, N se parará en todas las entradas, por lo tanto N es un *decider* también, así que $\langle N \rangle \in A$, pero $\langle N \rangle$ es distinto a cualquier $\langle M_i \rangle$ que imprima E, ya que si existe alguno, N hace lo contrario a lo que M_i haría en entrada s_i. Contradicción, N no es un decider para D.

Bibliografía

[1] M. Sipser, *Introduction to the Theory of Computation; 2nd ed.* Cambridge: Thomson Course Technology, 2006.

[2] J. D. U. John E. Hopcroft, Rajeev Motwani, *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, second ed., 2001.

[3] N. Cutland, *Computability: An Introduction to Recursive Function Theory.* Cambridge: Cambridge Univ. Press, 1980.