Tarea 5

Profesor: Jesús Rodríguez Viorato

IA & TC

Agosto-Diciembre 2021

Roberto Vásquez Martínez

Problema 1

Sea
$$ALL_{DFA} = \{\langle A \rangle \mid A \text{ es DFA y } L(A) = \Sigma^*\}$$

Muestra que ALL_{DFA} es decidible.

Solución: Construiremos una máquina de Turing que decida ALL_{DFA}.

En primer lugar recordemos que

$$E_{DFA} = \{\langle A \rangle \mid A \text{ es DFA y } L(A) = \emptyset\},$$

es decidible y sea T_1 una máquina de Turing que decide este lenguaje.

Además notamos que dado un DFA A podemos construir B tal que

$$L(B) = L(A)^c,$$

primero construyendo un DFA que acepte Σ^* y luego usando el autómata producto para hallar el autómata B que reconozca $L(A)^c = \Sigma^* \setminus L(A)$.

Sea T_2 la máquina de Turing definida de la siguiente forma:

$T_2 =$"input $\langle A \rangle$ con A un DFA

1. Construimos B tal que $L(B) = L(A)^c$.
2. Simulamos T_1 con input $\langle B \rangle$.
3. Aceptamos si T_1 acepta en el paso 2. Rechazamos si T_1 rechaza en el paso 2."

Como E_{DFA} es dedicible por T_1 entonces T_2 siempre se detiene, por lo que $L(T_2)$ es decidible, además T_2 acepta A sólo si T_1 acepta B, es decir, si $L(B) = \emptyset$ que equivale a que $L(A) = \Sigma^*$.

Concluimos que
$$ALL_{DFA} = L(T_2),$$

y ALL_{DFA} es decidible. □

Problema 2

Sea
$$A = \{\langle R, S\rangle \mid R \text{ y } S \text{ son expresiones regulares y } L(R) \subseteq L(S)\}.$$

Demuestra que A es decidible.

Solución: De manera análoga al problema anterior sea T_1 una máquina de Turing que decide E_{DFA}.

Observamos que
$$L(R) \subseteq L(S) \iff L(S)^c \cap L(R) = \emptyset. \tag{1}$$

Definimos la siguiente máquina de Turing:

T_2="input $\langle R, S\rangle$ con R y S expresiones regulares

1. Con el argumento del autómata producto construimos un DFA B tal que
$$L(B) = L(S)^c \cap L(R)$$
2. Simulamos T_1 con input $\langle B\rangle$.
3. Aceptamos si T_1 acepta en el paso 2. Rechazamos si T_1 rechaza en el paso 2."

Como T_1 decide E_{DFA} el algoritmo anterior siempre para por lo que $L(T_2)$ es decidible.

Ahora notamos que $\langle R, S\rangle$ es aceptada por T_2 si $\langle B\rangle$ es aceptada por T_1, es decir, si
$$L(S)^c \cap L(R) = \emptyset,$$

y por (1) esto equivale a que

$$L(R) \subseteq L(S).$$

Por lo tanto $A = L(T_2)$ y así A es decidible. □

Problema 3

Probar que EQ_{DFA} es decidible ejecutando los dos DFA's en todas las cadenas de cierto tamaño. Calcular un tamaño que funcione.

Solución: Consideremos $A = (P, \Sigma, \delta, p_0, F)$ y $B = (Q, \Sigma, \gamma, q_0, G)$ dos DFA's.

Sea $n = |P|$ y $m = |Q|$, denotemos por

$$SA_{nm} = \{\omega \in \Sigma^* \mid \delta^*(p_0, \omega) \in F \quad \text{y} \quad |\omega| \leq nm\},$$

y

$$SB_{nm} = \{\omega \in \Sigma^* \mid \gamma^*(q_0, \omega) \in G \quad \text{y} \quad |\omega| \leq nm\},$$

a las cadenas de caracteres aceptadas por A y B, respectivamente y que tienen longitud a lo más nm.

Vamos a probar que

$$L(A) = L(B) \iff SA_{nm} = SB_{nm} \tag{2}$$

Si $L(A) = L(B)$ es claro que $SA_{nm} = SB_{nm}$ pues los autómatas A y B aceptan las mismas cadenas de caracteres.

Ahora para demostrar la otra dirección procederemos por contrapositiva. Supongamos que $L(A) \neq L(B)$ queremos probar que

$$SA_{nm} \Delta SB_{nm} \neq \emptyset.$$

Como $L(A) \neq L(B)$ entonces $L(A)\Delta L(B) \neq \emptyset$. Por el principio del Buen Orden existe $t \in L(A)\Delta L(B)$ tal que

$$|t| = \text{mín}\{|\omega| \mid \omega \in L(A)\Delta L(B)\}. \tag{3}$$

Si $|t| \leq nm$ entonces $t \in SA_{nm}\Delta SB_{nm}$ y hemos terminado.

Ahora supongamos que $k := |t| > nm$.

Sea

$$p_0, p_1, \ldots, p_k \text{ los estados que se visitan en A al leer } t,$$

y

$$q_0, q_1, \ldots, q_k \text{ los estados que se visitan en B al leer } t.$$

Observamos que

$$|\{(p_i, q_i) \mid i = 0, 1, \ldots, k\}| = k + 1 > nm,$$

y como $|P| = n$ y $|Q| = m$ entonces solo hay nm parejas ordenadas (p, q) distintas con $p \in P$ y $q \in Q$.

Por el principio de las Casillas existe $i, j \in \{0, 1, \ldots, k\}$, que sin pérdida de generalidad podemos suponer $i < j$, tal que

$$(p_i, q_i) = (p_j, q_j).$$

Con esto podemos recuperar una cadena $t' \in \Sigma^*$ que ignore las transiciones por los estados p_{i+1} a p_{j-1} en A y las transiciones por los estados q_{i+1} a q_{j-1} en B.

Por lo tanto al leer la cadena t' se pasa por los estados

$$p_0, \ldots, p_i, p_{j+1}, \ldots, p_k \text{ en A,}$$

y

$$q_0, \ldots, q_i, q_{j+1}, \ldots, q_k \text{ en B,}$$

y t' es aceptada o rechazada como lo es t en A y en B, es decir, $t' \in L(A)\Delta L(B)$, pero

$$|t'| = k + i - j < k = |t|,$$

que es una contradicción a la definición de t en (3).

Por lo tanto $|t| \leq nm$ y se cumple $SA_{nm}\Delta SB_{nm} \neq \emptyset$ si $L(A) \neq L(B)$ y así tenemos el hecho en (2).

Por lo tanto para decidir si $L(A) = L(B)$ basta con simular A y B en todas las cadenas $\omega \in \Sigma^*$ con $|\omega| \leq nm$.

La máquina de Turing que simulará este algoritmo es:

$T =$“ input $\langle A, B \rangle$ con A y B DFA's

1. Contar el número de estados de A y B, que serán n y m.
2. Iterar sobre todas las cadenas $\omega \in \Sigma^*$ con $|\omega| \leq nm$.
   - Simular A con entrada ω.
   - Simular B con entrada ω.
3. Si el resultado de A y B es diferente rechazar, de lo contrario volver el paso 2.
4. Si se termina el ciclo en el paso 2 significa que $SA_{nm} = SB_{nm}$ entonces aceptar $\langle A, B \rangle$.”

Notamos que T siempre se detiene porque existe un número finito de cadenas de longitud a lo más nm, por lo que $L(T)$ es decidible, además T acepta $\langle A, B \rangle$ sólo si llega al paso 4 y esto pasa si $SA_{nm} = SB_{nm}$ que por (2) equivale a $L(A) = L(B)$, por lo que

$$L(T) = EQ_{DFA},$$

de lo que concluimos que EQ_{DFA} es decidible y el tamaño apropiado de las cadenas en las que basta simular los autómatas es nm. □

Problema 4

Probar que la clase de lenguajes decidibles no es cerrada bajo homomorfismos.

Solución: Recordemos que

$$HALT_{TM} = \{\langle M, \omega \rangle \mid M \text{ es TM y se detiene al leer} \omega\},$$

es indecidible por el *Teorema 5.1* de Sipser, 2012).

Usaremos la definición de función de condificación que aparece en la *Definición 7.33* de Martin, 2010, esto nos permite codificar $\langle M, \omega \rangle$ con elementos de $\{0, 1\}^*$.

Lo anterior nos permite definir

$$D = \{xy \mid x \in \{0,1\}^*,\ x = \langle M, \omega \rangle,\ y \in \{a,b\}^* \text{ codifica a un natural } n \text{ tal que } M \text{ se detiene al leer } \omega \text{ en } n \text{ pasos}\},$$

entonces D codifica en un lenguaje la acción de M con input ω que se para en n pasos para cada $n \in \mathbb{N}$.

Notamos que D es decidible ya que para cada $n \in \mathbb{N}$, n codificado por una cadena y en $\{a, b\}^*$, se simula M con entrada ω y la máquina se detiene en a lo más n pasos. Si se llegó al estado aceptor o de rechazo durante esos n pasos la correspondiente máquina aceptará xy sino lo rechazará.

Sea $h : \{0, 1, a, b\} \to \{0, 1, \varepsilon\}$ definida por

$$\begin{aligned} h(0) =& 0 \\ h(1) =& 1 \\ h(a) =& h(b) = \varepsilon. \end{aligned}$$

Como h es una función en los caracteres entonces es un homomorfismo que se extiende como en el *Problema 1.66* de Sipser, 2012 a cadenas de caracteres.

Aplicamos el homomorfismo a lenguajes como en el *Problema 1.66* de Sipser, 2012 y obtenemos que

$$h(D) = \{h(xy) \mid xy \in D\}$$

Si $xy \in D$ entonces x codifica $\langle M, \omega \rangle$ tal que M se para con entrada ω y lo hace en n pasos para alguna $n \in \mathbb{N}$, donde el número de pasos necesarios para detenerse es la parte codificada por y.

Como $h(xy) = x$ entonces al aplicar h obtenemos x que codifican $\langle M, \omega \rangle$ tal que M se detiene al leer ω, es decir,

$$h(D) = \{x \mid x \in \{0,1\}^*, x = \langle M, \omega \rangle \text{ tal que } M \text{ se detiene al leer } \omega\} = HALT_{TM},$$

y al ser $HALT_{TM}$ indecidible se tiene que $h(D)$ es indecidible.

Por lo tanto los lenguajes decidibles no son cerrados bajo homomorfismos. □

Problema 5

Encuentra un emparejamiento en la siguiente colección de dominos del PCP

$$\left\{ \left[\frac{ab}{abab}\right], \left[\frac{b}{a}\right], \left[\frac{aba}{b}\right], \left[\frac{aa}{a}\right] \right\}$$

Solución: La colección de dominos que proporciona un emparejamiento es la siguiente

$$\left\{ \left[\frac{ab}{abab}\right], \left[\frac{ab}{abab}\right], \left[\frac{aba}{b}\right], \left[\frac{b}{a}\right], \left[\frac{b}{a}\right], \left[\frac{aa}{a}\right], \left[\frac{aa}{a}\right] \right\},$$

y la cadena de caracteres que está arriba y abajo en el emparejamiento es

$$abababab baaaa \qquad \square$$

Problema 6

Sea
$$T = \{\langle M \rangle \mid M \text{ es TM que acepta } \omega^R \text{ si acepta } \omega\}.$$
Demuestra que T es indecidible.

Solución: Sea Σ es el alfabeto de las máquinas de Turing, si $|\Sigma| = 1$, entonces para toda máquina de Turing M con dicho alfabeto se cumple
$$L(M) = L(M)^R,$$

por lo que en este caso tenemos que
$$T = \{\langle M \rangle \mid M \text{ es TM}\}$$

es decidible por la forma es que se considera la codificación $\langle M \rangle$(Ver Sipser, 2012, pp. 185).

Por lo tanto consideramos $|\Sigma| \geq 2$, sean $a, b \in \Sigma$ tal que $a \neq b$.

Por otro lado, decimos que un lenguaje $L \subseteq \Sigma^*$ es cerrado bajo la operación reversa si para todo $\omega \in L$ se tiene $\omega^R \in L$.

Por el *Teorema 4.22* de Sipser, 2012 para ver que T es indecidible basta ver que T^c, que es

$T^c = \{\langle M \rangle \mid \langle M \rangle$ no condifica a una TM o es una TM tal que $L(M)$ no es cerrado bajo la operación reversa$\}$,

es indecidible.

Para probar que T^c es indecidible procederemos por contradicción. Supongamos que T^c es decidible y sea D_{T^c} una TM que lo decide.

En primer lugar, para $\omega \in \Sigma^*$ y M TM construimos la TM que denotaremos por $M'(M, \omega)$ y definimos de la siguiente forma

$M' := M'(M, \omega) =$" input x con $x \in \Sigma^*$

1. Si $x = ab$
   - Simular M en ω
   - Si M acepta ω, aceptar.
   - Si M rechaza ω, rechazar.
2. Si $x \neq ab$
   - Rechazar"

Vamos a probar que
$$\langle M, \omega \rangle \in A_{TM} \iff \langle M' \rangle \in T^c. \tag{4}$$

Si $\langle M, \omega \rangle \in A_{TM}$ entonces ω es aceptado por M, luego $L(M') = \{ab\}$, por lo que M' es una TM y
$$ba = (ab)^R \notin L(M'),$$

por lo que $\langle M' \rangle \in T^c$.

Para probar la otra dirección procedemos por contrapositiva. Si $\langle M, \omega \rangle \notin A_{TM}$ entonces ω es rechazada por M o bien M itera indefinidamente al ingresar ω, en cualquier caso $L(M') = \emptyset$, por lo que $\langle M' \rangle \notin T^c$, se tiene así que se cumple la proposición en (4).

En este punto tenemos D_{T^c} la TM que decide T^c, construimos a partir de esta máquina la siguiente TM

$D_{A_{TM}} =$" input $\langle M, \omega \rangle$ con M una TM

1. Construir M' como antes usando $\langle M, \omega \rangle$.
2. Aceptar si D_{T^c} acepta $\langle M' \rangle$. Rechazar si D_{T^c} rechaza $\langle M' \rangle$."

Como D_{T^c} decide T^c entonces $D_{A_{TM}}$ siempre se detiene por lo que $L(D_{A_{TM}})$ es decidible, y de (4) se tiene que

$$A_{TM} = L(D_{A_{TM}}),$$

lo que implica que A_{TM} es decidible lo que es una contradicción.

Concluimos así que T^c es indecidible, por lo que T es indecidible que es lo que queríamos probar. □

Problema 7

Un estado *useless* en una máquina de Turing es un estado que nunca es visitado para cualquier cadena de entrada. Considera el problema de determinar si una máquina de Turing tiene estados *useless*. Formula este problema como un lenguaje y demuestra que es indecidible.

Solución: Así como codificamos cadenas de caracteres y máquinas de Turing codificamos estados de unas máquina de Turing.

Por lo tanto formulamos el lenguaje de todas las máquinas de Turing con estados *useless* de la siguiente forma

$$UL_{TM} = \{\langle M, q\rangle \mid M \text{ es TM y } q \text{ es estado } \textit{useless} \text{ de } M\}.$$

Para demostrar que UL_{TM} es indecidible procederemos por contradicción.

Supongamos que UL_{TM} es decidible y sea $D_{UL_{TM}}$ una TM que lo decide. Definimos $D_{E_{TM}}$ de la siguiente forma

$D_{E_{TM}} =$" input $\langle M\rangle$ con M una TM

1. Simular $D_{UL_{TM}}$ con entrada $\langle M, q_{\text{accept}}\rangle$ donde q_{accept} es el estado aceptor de M.
2. Aceptar si $D_{UL_{TM}}$ acepta $\langle M, q_{\text{accept}}\rangle$. Rechazar si $D_{UL_{TM}}$ rechaza $\langle M, q_{\text{accept}}\rangle$."

Observamos que $D_{E_{TM}}$ siempre se detiene pues $D_{UL_{TM}}$ lo hace, luego $L(D_{E_{TM}})$ es decidible.

Notamos que $D_{E_{TM}}$ acepta $\langle M\rangle$ sólo si $D_{UL_{TM}}$ acepta $\langle M, q_{\text{accept}}\rangle$, es decir, si M nunca visita el estado aceptor, luego $L(M) = \emptyset$.

Por lo tanto $D_{E_{TM}}$ acepta $\langle M\rangle$ si $L(M) = \emptyset$, en otro caso rechaza $\langle M\rangle$, de esto se sigue que

$$E_{TM} = L(D_{E_{TM}}),$$

luego E_{TM} es decidible lo que es una contradicción pues por el *Teorema 5.2* de Sipser, 2012 tenemos que E_{TM} es indecidible.

Concluimos así que UL_{TM} es indecidible que es lo que queríamos probar. $\square$

Problema 8

¿Cuales de las siguientes parejas de números son primos relativos?. Muestra los cálculos que te permiten dar las conclusiones

a) 1274 y 10505

b) 7289 y 8029

Solución: Para determinar si las parejas proporcionadas son de números primos relativos o no recurrimos al algoritmo presentado en el *Teorema 7.15* de Sipser, 2012.

a) Para la pareja $(10505, 1274)$ la rutina es la siguiente

1. $313 \equiv 10505 \bmod 1274$
2. $22 \equiv 1274 \bmod 313$
3. $5 \equiv 313 \bmod 22$
4. $2 \equiv 22 \bmod 5$
5. $1 \equiv 5 \bmod 2$
6. $0 \equiv 2 \bmod 1$

y en este último paso según la notación en la prueba del *Teorema 7.15* de Sipser, 2012 se tiene que $y = 0$ y $x = 1$, por lo que $\gcd(10505, 1274) = 1$, entonces la pareja de números $(10505, 1274)$ es una pareja de números primos relativos.

b) Para la pareja $(8029, 7289)$ el resultado del algoritmo de Euclides es el siguiente

1. $740 \equiv 8029 \bmod 7289$
2. $629 \equiv 7289 \bmod 740$
3. $111 \equiv 740 \bmod 629$
4. $74 \equiv 629 \bmod 111$
5. $37 \equiv 111 \bmod 74$
6. $0 \equiv 74 \bmod 37$

Por lo tanto $y = 0$ y $x = 37$ en el último paso del algoritmo, luego $\gcd(8029, 7289) = 37$ entonces en este caso la pareja de números no son primos relativos entre sí. $\square$

Problema 9

Demuestra que P es cerrado bajo unión, concatenación y complemento.

Solución:

- *Unión:* Primero probaremos la cerradura de P respecto a la unión. Sean $L_1, L_2 \in P$ y M_1, M_2 las TM que deciden en tiempo polinomial a L_1 y L_2 respectivamente.

  Construimos la siguiente TM

  $M =$"input $\omega \in \Sigma^*$

  1. Simular M_1 en ω, si se acepta, aceptar.
  2. Simular M_2 en ω, si se acepta, aceptar, en otro caso rechazar."

  Notamos que M acepta ω si y sólo si M_1 o M_2 acepta ω, por lo que $L(M) = L_1 \cup L_2$, además M es una máquina determinística de Turing porque M_1 y M_2 lo son.

  Como M_1 y M_2 deciden en tiempo polinomial entonces las fases 1 y 2 del algoritmo anterior toman un tiempo polinomial, por lo que M decide $L_1 \cup L_2$ en tiempo polinomial y así $L_1 \cup L_2 \in P$, que es lo que queríamos ver.

- *Concatenación:* De manera análoga al apartado anterior sean $L_1, L_2 \in P$ y M_1, M_2 máquinas determinísticas que deciden L_1 y L_2, respectivamente.

  En este caso construimos la siguiente TM

  $M =$"input $\omega \in \Sigma^*$

  1. Iterar sobre cada descomposición en dos strings de ω de la forma $\omega = \omega_1\omega_2$.
     - Simular $\omega_1 \in M_1$ y $\omega_2 \in M_2$. Si se acepta a ambos, aceptar.
  2. Si ω no se acepta al intentar las posibles descomposiciones en dos strings, rechazar."

  Notamos que M acepta ω si y sólo si existe una descomposición $\omega = \omega_1\omega_2$ tal que $\omega_1 \in L_1$ y $\omega_2 \in L_2$, por lo que $L(M) = L_1L_2$, es decir, M decide la concatenación de L_1 y L_2.

  Por otra parte, M es una máquina determinística de Turing pues M_1 y M_2 lo son, además el ciclo en el paso 1 es del orden $O(n)$ pues hay $n+1$ descomposiciones de ω distintas. Como la subrutina del ciclo en 1 toma tiempo polinomial pues M_1 y M_2 deciden en tiempo polinomioal entonces M decide en tiempo polinomial, luego $L_1L_2 \in P$ que es lo que queríamos probar.

- *Complemento:* Sea $L \in P$ y M una TM determinística tal que $L = L(M)$ y M decide en tiempo polinomial.

  Construimos la siguiente TM

  $M' =$" input $\omega \in \Sigma^*$

  1. Simular M en ω.
  2. Si M acepta ω rechazar. Si M rechaza ω aceptar."

  Observamos que $\omega \in L(M')$ si ω es rechazada por M, es decir, $\omega \notin L(M)$ lo que equivale a $\omega \in L(M)^c$, de manera similar si $\omega \notin L(M')$ entonces ω es rechazada por M', luego $\omega \in L(M)$ que es equivalente a $\omega \notin L(M)^c$.

  Por lo tanto
  $$L(M') = L^c.$$

  Como en la construcción de M' sólo simulamos M que corre en tiempo polinomial entonces M' decide en tiempo polinomial, concluimos así que $L^c \in P$, que es lo que queríamos verificar. □

Problema 10

Demuestra que si $P = NP$, entonces todo lenguaje $A \in P$, excepto $A = \emptyset$ y $A = \Sigma^*$, es NP-completo.

Solución: Vamos a demostrar que todo lenguaje $A \in P$, excepto $A = \emptyset$ y $A = \Sigma^*$, es tal que

(i) $A \in NP$

(ii) Para todo $B \in NP$ se cumple $B \leq_P A$.

La condición *(i)* es clara pues $A \in P$ y por hipótesis $P = NP$.

Para demostrar la condición *(ii)* consideremos $x, y \in \Sigma^*$ tales que

$$x \in A \quad \text{y} \quad y \notin A,$$

esto lo podemos hacer pues $A \neq \emptyset$ y $A \neq \Sigma^*$.

Como $P = NP$ entonces para todo $B \in NP$ existe M_B una máquina determinística de Turing tal que $L(M_B) = B$ y M_B lo hace en tiempo polinomial.

Construimos la siguiente TM

$M =$" input $\omega \in \Sigma^*$

1. Simular M_B con entrada ω
2. Si M_B acepta ω, escribir x en la cinta y aceptar.
3. Si M_B rechaza ω, escribit y en la cinta y rechazar."

Como M_B decide B en tiempo polinomial entonces M decide B también en tiempo polinomial y termina con x o y en la cinta, según si la cadena de entrada está o no en B.

Por lo tanto la función $f : \Sigma^* \to \Sigma^*$ definida como

$$f(\omega) = \begin{cases} x & \text{si } \omega \in B \\ y & \text{si } \omega \notin B \end{cases}$$

es una función calculable en tiempo polinomial (Ver Sipser, 2012, Definición 7.28), además notamos que para todo $\omega \in B$ se tiene que M acepta ω y $f(\omega) = x \in A$, por lo que

$$\omega \in B \;\Rightarrow\; f(\omega) = x \in A.$$

Por otro lado, si $\omega \notin B$ entonces M rechaza ω y escribe y en la cita, esto es $f(\omega) = y \notin A$, luego

$$\omega \notin B \;\Rightarrow\; f(\omega) = y \notin A,$$

que al tomar contrapositiva es equivalente a

$$f(\omega) \in A \;\Rightarrow\; \omega \in B.$$

Se sigue de lo anterior que para todo $\omega \in \Sigma^*$

$$\omega \in B \iff f(\omega) \in A,$$

por lo que $B \leq_P A$.

Concluimos que todo $B \in NP$ es reducible en tiempo polinomial a A, por lo que todo lenguaje $A \in P$ es NP-completo si $A \neq \emptyset$ y $A \neq \Sigma^*$, que es precisamente lo que queríamos probar. $\square$

Referencias

Martin, J. (2010). *Introduction to Languages and the Theory of Computation* (4.ª ed.). McGraw-Hill Science/Engineering/Math.

Sipser, M. (2012). *Introduction to the Theory of Computation* (3.ª ed.). Thomson South-Western.