Teoría de la Computación y Verificación de Programas

Ulises J. Cornejo Fandos[1], Lucas Di Cunzolo[2]

[1] 13566/6, Licenciatura en Informática, Facultad de Informática, UNLP
[2] 13572/5, Licenciatura en Informática, Facultad de Informática, UNLP,

1 Ejercicio 1

Probar que el lenguaje $L_u = \{(< M >, w) |\ M \text{ acepta } w\ \}$ *pertenece a la clase RE. Ayuda: la prueba es similar a la desarrollada en la clase 3 para demostrar que* $D = \{w_i | M_i \text{ acepta } w_i\} \in RE$.

La siguiente MT M_{L_u} acepta el lenguaje L_u. Dado un input de la forma $(< M >, w)$, la MT M_{L_u} hace lo siguiente:

1. Toma la MT $< M >$ y la ejecuta con el input w
2. La MT M_{L_u} acepta si y solo si $< M >$ acepta w. De lo contrario, la misma rechaza o entra en estado de *loop*.

Luego, el mismo está en RE. A su vez, suponer que $L_u \in R$ es ilógico dado que eso significaría que $D = \{w : M \text{ acepta } w\} \in R$, el cual ya se demostró pertenece a $RE - R$, entonces sería un absurdo. Por lo tanto, se demuestra que $L_u \in RE - R$.

2 Ejercicio 2

Explicar informal pero claramente cómo trabajaría una MT que genera la n-ésima fórmula booleana satisfactible (es decir que existe una asignación de valores de verdad que la hace verdadera), cuya sintaxis contiene variables de la forma x_i*, los operadores lógicos del conjunto* $\{\neg, \wedge, \vee\}$ *y paréntesis.*

Se construye una MT M que dado un input n,

1. Genera el primer w_i en orden canónico.
2. Ejecuta una MT que compruebe que la fórmula booleana sea satisfacible.
   - Si es satisfacible entonces aumenta un contador m. Si m=n entonces acepta.
3. Genera el siguiente w_i en orden canónico y vuelve al paso 2.

3 Ejercicio 3

Justificar informal pero claramente cada uno de los incisos siguientes

1. *Se puede decidir si una MT M, a partir de la cadena vacía λ, escribe alguna vez un símbolo no blanco. Ayuda: ¿Cuántos pasos puede hacer M antes de entrar en loop?*

   Sea L = {M : M, a partir de la cadena vacía λ, escribe alguna vez un símbolo no blanco} y dada una MT M_L, la misma reconoce L, es decir $L(M_L) = L$, de la siguiente forma. Dado una MT $< M >$, se ejecuta M con λ como input acotando la cantidad de pasos que esta puede hacer. Para acotar la cantidad de pasos, se tiene en cuenta que la maquina no debería escribir en la cinta y a su vez, no debería pasar por el mismo estado dos veces. Luego, teniendo en cuenta que en la cinta solo habrá blancos, decimos que la cantidad de pasos posibles antes de entrar en loop no debería exceder la cantidad de estados, $|Q|$. Durante la ejecución de M, M_L observa si la maquina M escribe o no en la cinta. Si la misma escribe, M_L para en su estado q_a. Si la misma, excede la cantidad de pasos, planteada anteriormente, M_L para en estado q_r. Luego, dado que M_L nunca loopea, $L \in R$.

2. *Se puede decidir si a partir de un input w, una MT M que sólo se mueve a la derecha para. Ayuda: ¿Cuántos pasos puede hacer M antes de entrar en loop?*

   Sea L = {M : M, a partir de un input w, M se mueve solamente a la derecha} y dada una MT M_L, la misma reconoce L, es decir $L(M_L) = L$, de la siguiente forma. Dado una MT $< M >$, se ejecuta M con w como input acotando la cantidad de pasos que esta puede hacer. Para acotar la cantidad de pasos, se tiene en cuenta que la maquina no debería exceder una cantidad de pasos $N = |w|$ si solo se mueve hacia la derecha. Durante la ejecución de M, M_L observa si la maquina M excede, o no, esta cantidad de pasos. Si la misma no excede, M_L para en su estado q_a. En caso contrario, para en q_r. De igual forma, M_L para siempre, por lo que $L \in R$.

3. *Se puede decidir, dada una MT M, si existe un input w a partir del cual M para en a lo sumo 10 pasos. Ayuda: ¿Hasta qué tamaño de cadenas hay que chequear?*

   Sea L = {M : M, a partir de un input w, M para en a lo sumo 10 pasos} y dada una MT M_L, la misma reconoce L, es decir $L(M_L) = L$, de la siguiente forma. Dado una MT $< M >$, se ejecuta M con w como input observando la ejecución del mismo. Para acotar la cantidad de pasos, se tiene en cuenta que la maquina no debería exceder una cantidad de pasos $N = |w|.|Q| < 10$. Durante la ejecución de M, M_L observa si la maquina M para o no. En caso contrario, M_L evalúa por cada paso de M si la misma paró o no. Si la misma

para, M_L para en su estado q_a. Si M hizo más de N pasos, M_L para en estado q_r. Luego, es decidible y no loopea, entonces L es un lenguaje recursivo, es decir, $L \in R$.

4 Ejercicio 4

Considerando la reducción de HP a L_U descripta en clase, responder:

1. Para hacer una reducción de HP a L_u, es decir, $HP \alpha L_u$, debería utilizarse un función f computable en la cual al ejecutar un input w sobre una MT M, la misma pare siempre pero nunca en un estado q_r. Por otro lado, las tuplas que componen HP están dadas por MT M y un input w tales que M para siempre sobre w, pero no necesariamente acepta.

   El problema de aplicar la función identidad es que no me garantiza que la MT M pare siempre en estado de aceptación por lo que no sería una reducción válida de HP a L_u.

2. La MT M' loopea, o para en estado q_a, pero nunca rechaza dada la definición de f. En la misma, dado un input de la forma (M, w), la función modifica M cambiando todos los estado q_r por q_a. Luego, M' resulta en una maquina donde q_r no pertenece al conjunto de estados finales, por lo que M' nunca para en q_r para cualquier input w.

3. La función de reducción planteada sirve para hallar $HP^c \alpha L_u^c$ dado que existe una propiedad de reducción en la cual se enuncia que $\exists L_1 \alpha L_2 \leftrightarrow \exists L_1^c \alpha L_2^c$ aplicando la mismo función computable. Luego, como existe $HP \alpha L_u$ se cumple que existe $HP^c \alpha L_u^c$ y en ambos casos es la misma.

4. La función de transición planteada no sirve para hallar la reducción $L_u \alpha HP$, y esto se debe a que la misma hace que una MT M' pare en q_a cuando M para en q_r. Ahora, necesitamos que dado un input w la maquina M' no se detenga cuando M para en q_r. Es por esto que la reducción no serviría para este caso.

5. Esto se debe a que la reducción iría de fuera de HP a dentro de L_u. Ya que el v no tiene forma $(< M >, w)$, entonces no pertenece a HP, si le cambiamos la forma a $(< M\Sigma* >, v)$, el mismos pertenece a L_u ya que $M\Sigma*$ acepta todas las cadenas.

6. Es incorrecto, ya que si M rechaza w, entonces $(< M >, w)$ pertenece a HP, pero con la función dada, se genera el output 1, por lo que $v \notin L_u$, lo que la vuelve una función de reducción invalida.

5 Ejercicio 5

Considerando la reducción de L_u a L_Σ descripta en clase, responder:

1. *Explicar por qué no sirve como función de reducción la función siguiente: a todo input le asigna como output el código $< M\Sigma >$.*

   No funciona ya que, dado un input $(< M >, w)$, M no acepta w entonces $(< M >, w) \notin L_u$, pero al realizar la función de reducción $M\Sigma$ pertenece a L_Σ, lo que resulta absurdo.

2. *Explicar por qué la reducción descrita en clase no sirve para probar que $L_\Sigma \notin RE$.*

   No sirve ya que $LU \in RE$, y al reducir decimos que es tan o más difícil, por lo que L_Σ sería tan o más difícil que LU, lo cual no sirve para probar $L_\Sigma \notin RE$, para eso deberíamos reducir $L_1 \alpha L_\Sigma$ con $L1 \notin RE$.

6 Ejercicio 6

Probar formalmente que las funciones de reducción gozan de la propiedad transitiva. Ayuda: revisar la idea general comentada en clase.

Sean L_a, L_b, L_c lenguajes, $(\forall a, b) : (a \in L_a, b \in L_b)$ y, sean f_a, f_b dos funciones tales que $f_a(a) \in L_b, f_b(b) \in L_c$. Se prueba la propiedad transitiva de la reducción de lenguajes.

- $(\forall a \in L_a)(\forall f_a)$ se cumple que $f_a : L_a \to L_b \leftrightarrow f_a(a) \in L_b$, por definición de reducción.
- $(\forall b \in L_b)(\forall f_b)$ se cumple que $f_b : L_b \to L_c \leftrightarrow f_b(b) \in L_c$, por definición de reducción.
- Luego, sea h una función tal que $h(a) = f_b(f_a(a))$, tenemos por definición de composición que h es una función de la forma $h : L_a \to L_c$. Esto quiere decir que $\forall a \in L_a, h(a) \in L_c$

7 Ejercicio 7

Sea el lenguaje $D_{HP} = \{w_i | M_i$ para desde w_i, según el orden canónico$\}$. Encontrar una reducción de D_{HP} a HP.

La idea es que la funcion de reduccion recibe un input y genera un output. No tiene la nocion de aceptar o rechazar. El output, de ser valido, deberia pertenecer a HP, es decir, ser de la forma (M, w). Se define a continuación la función de reducción.

– Dado un input inválido, retorna algun caracter simbolizando un error como por ejemplo "1".
– Dado un input válido y sea una MT M, $f(w_i) = (< M_i >, w_i)$, retorna una tupla perteneciente a HP donde M_i es una maquina generada la cual para con w_i.

8 Ejercicio 8

Sean VAL y UNSAT los lenguajes de las fórmulas booleanas válidas e insatisfactibles (todas y ninguna asignación de valores de verdad las hace verdaderas, respectivamente). Encontrar una reducción de VAL a UNSAT.

Dado un input γ, se define a continuación la función de reducción.

– Dado un γ, se define que $f(\gamma) = \gamma'$, donde γ' es una formula booleana de la forma $\neg(\gamma)$. Luego,
  – Si γ es una fórmula booleana valida (todas las asignaciones de verdad la hacen verdadera) al negarla todas las asignaciones de verdad darían falso.
  – Si γ no es una formula booleana valida, entonces al negarla existiría una asignación de valores de verdad que no daría *false*, por lo que γ' no estaría en UNSAT.