CC3102 — Teoría de la Computación

Solución Control 3

29 de julio de 2013

1. Hay una infinidad de formas de construir una MT M que realice lo pedido. Una de ellas elimina todos los ceros, recordando el valor módulo 3 de la cantidad de ceros borrados, y una vez hecho esto, escribe el valor recordado, dejando el cabezal de M en el primer cero de la respuesta (si es que tiene ceros).

   Definimos $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ con

   - $Q = \{q_0, q_1, q_2, h\}$
   - $\Sigma = \{0\}$
   - $\Gamma = \{0, B\}$
   - $F = \{h\}$
   - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$

   $$\delta(q_0, 0) = (q_1, B, \rightarrow)\ ,\ \delta(q_0, B) = (h, B, \rightarrow)$$
   $$\delta(q_1, 0) = (q_2, B, \rightarrow)\ ,\ \delta(q_1, B) = (q_0, 0, \leftarrow)$$
   $$\delta(q_2, 0) = (q_0, B, \rightarrow)\ ,\ \delta(q_2, B) = (q_1, 0, \leftarrow)$$

   la cual computa el resto de dividir un número por 3.

2. La idea es utilizar el hecho de que la computación de las máquinas de Turing trabaja localmente sobre la cinta, para simularla con una M2S que sólo puede ver lo que se encuentra en el tope de los stacks. Ocuparemos el segundo stack para representar lo que se encuentra a la derecha del cabezal. De esta forma el cabezal lee lo que se encuentra en el tope de éste. Análogamente el primer stack guarda lo que se encuentra a la izquierda del cabezal.

   Formalmente sea L un lenguaje RE y $M = \langle Q, \Sigma, \Gamma, \delta, q_0, B, F\rangle$ una MT que lo reconoce. Asumiremos sin pérdida de generalidad que M nunca escribe el símbolo B. Construiremos a partir de M una máquina de doble stack $M' = \langle Q, \Sigma, \Gamma, \delta', q_0, B, F\rangle$ que acepta L. Definimos la función δ' adaptando la definición de δ.

   - Por cada definición $\delta(q, a) = (p, b, \rightarrow)$, creamos $\delta'(q, c, a) = (p, bc, \epsilon)$ si $a \neq B$ y $\delta'(q, c, a) = (p, bc, B)$ en caso contrario
   - Por cada definición $\delta(q, a) = (p, b, \leftarrow)$, creamos $\delta'(q, c, a) = (p, \epsilon, cb)$ si $c \neq B$ y $\delta'(q, c, a) = (p, B, cb)$ en caso contrario

   Podemos definir la configuración instantanea de una M2S como $s_1^R q s_2$. Donde q es el estado actual, s_1 es el contenido del primer stack y s_2 el del segundo. Es fácil notar que con esta definición las configuraciones de M y M' se mantienen iguales luego de cada aplicación de la función de transición. No es difícil a partir de esto notar que M' acepta L.

   Ahora bien, la hipótesis de Church estable una equivalencia entre función computable y máquina de Turing. De este modo las M2S no pueden aceptar lenguajes que no recursivamente enumerables, pues estos no son computables.

3. *a*) Como L es recursivo existe una máquina de Turing M que lo reconoce. En este caso, basta probar para todas las particiones $w = w_1 w_2 \cdots w_m$ si $w_1, w_2 \cdots, w_m \in L$. Para esto sólo hay que simular M en cada uno de estos strings. Como L es recursivo y la cantidad de particiones es finita para un input dado, este procedimiento se detendrá en algun momento.

   Para probar las particiones se puede hacer de forma no determinista o bien probando en orden las particiones de 1 string, de 2 strings, $\cdots$, $|w|$ strings.

   b) Es fácil encontrar un contraejemplo a esta afirmación. Definiendo $L_1 = \Sigma^*$ obtenemos que $L_1 \backslash L_2 = \Sigma^* \cap \overline{L_2} = \overline{L_2}$. Basta entonces tomar L_2 como cualquier lenguaje RE pero no recursivo pues sabremos que su complemento no será RE.

4. Para demostrar indecibilidad del problema, reduciremos un problema indecidible conocido al problema $E_{\cap GLC}$ de determinar vacuidad de la intersección de dos gramáticas libres de contexto.

   En efecto, si el problema $E_{\cap GLC}$ es decidible, entonces existe una MT M_E que lo decide. Mostraremos que usando M_E, es posible construir una MT M_{PCP} que recibe una instancia del problema de correspondencia de Post (PCP) de la forma $\langle (u_1, v_1), \cdots, (u_n, v_n) \rangle$ y decide si esta instancia tiene solución.

   Sea $\langle (u_1, v_1), \cdots, (u_n, v_n) \rangle$ una instancia del PCP, donde cada par ordenado $(u_i, v_i), u_i, v_i \in \Sigma^*$ representa una ficha con u_i en la mitad superior, y v_i en la mitad inferior. Es posible generar una gramática que genera uno o más palíndromes para una instancia cualquiera del PCP *ssi* la instancia tiene solución. La idea es que la gramática genera todas las posiciones válidas de fichas de la instancia del PCP, de forma que la lectura de la mitad superior de las fichas aparece tal cual, seguida de un separador, para luego agregar finalmente el contenido de la mitad inferior de las fichas pero invertido. La inversión es necesaria debido a que la gramática genera las mitades superior e inferior de cada ficha de forma anidada, por ejemplo, dada la secuencia de k fichas $i_1, i_2, \cdots, i_k$ que se representa por la cadena $u_{i_1} u_{i_2} \cdots u_{i_k} \# v_{i_k} \cdots v_{i_2} v_{i_1}$, si v_{i_j} tiene largo mayor que 1 para algún j, entonces al leer lo que está a la derecha del separador de derecha a izquierda, no se obtendría correctamente la cadena de la mitad inferior de las fichas. El separador es necesario para rechazar casos donde una mitad (sin pérdida de generalidad, la superior) de la secuencia de fichas contiene lo mismo que la otra mitad (la inferior) pero concatenado con un palíndrome.

   Luego, construimos una GLC G_1 de un solo no-terminal S, y por cada ficha (u_i, v_i) de la instancia de PCP se genera una regla

   $$S \to u_i S v_i^R$$

   donde x^R es el string x escrito en forma reversa. Además, agregamos una regla base

   $$S \to u_i \# v_i$$

   pues esto garantiza que se usa al menos una ficha del PCP en la representación en forma de GLC. Observemos que todas las cadenas generadas corresponden a exactamente a una configuración válida de fichas en el PCP. Notar que como # aparece exactamente una vez, todos los palíndromes generados deben tener largo impar y deben tener # justo en el centro. Estos palíndromes sólo se generan si la configuración correspondiente en el PCP es una solución del PCP.

   Finalmente, se construye una GLC G_2 que genera todos los palíndromes posibles. Esto es sencillo, basta tomar cada símbolo s que aparece en los pares de la instancia de PCP anterior, y el símbolo # y construir reglas de la forma

   $$S \to sSs$$

   $$S \to s$$

   además de la regla

   $$S \to \epsilon$$

Las GLC anteriores son computables por una MT, por lo que es posible realizar esta conversión de la instancia de PCP $\langle (u_1, v_1), \cdots, (u_n, v_n) \rangle$ a las representaciones $\rho(G_1)$ y $\rho(G_2)$ respectivas de G_1 y G_2 que se describen en los párrafos anteriores.

Finalmente, se entrega a la máquina M_E el input $\langle \rho(G_1), \rho(G_2) \rangle$ y la salida de ésta se utiliza como salida de la M_{PCP} que estamos construyendo. Así, la M_{PCP} descrita decide el PCP, lo cual es una contradicción.

Luego, nuestro supuesto inicial es falso; $E_{\cap GLC}$ es indecidible.