Modelos Avanzados de Computación

Entrega 3

María del Mar Ruiz Martín
Doble Grado de Ingeniería Informática y Matemáticas
Universidad de Granada - UGR
18001 Granada, Spain
Curso 2016/2017

Ejercicio 3

Demostrar que P es cerrada para la unión y la intersección.
Sean $L_1, L_2 \in$P, queremos probar que $L_1 \cup L_2 \in$ P. Recordamos que, por definición, $P = \cup_{j>0} TIEMPO(n^j)$. Por tanto, existen $i, j \in \mathbb{N} \ : \ L_1 \in TIEMPO(n^i),\ L_2 \in TIEMPO(n^k)$.

Consideramos m = máx{i,k}, y claramente $L_1, L_2 \in TIEMPO(n^m)$. Sean MT_1, MT_2 las máquinas de Turing que aceptan L_1 *y* L_2 respectivamente. Si consideramos ahora la máquina de Turing MT que primero ejecuta MT_1 sobre la entrada, y, en caso de que esta rechace, ejecuta MT_2, obtenemos una máquina de Turing que claramente acepta $L_1 \cup L_2$ y por tanto es claro que $L_1 \cup L_2 \in TIEMPO(n^m)$. Por tanto, $L_1 \cup L_2 \in$ P y P es cerrada para la unión.

Para demostrar que P es cerrada para la intersección tendremos en cuenta que P = CoP junto con el álgebra de De Boole. Sean $L_1, L_2 \in$ P, entonces

$$L_1 \cap L_2 = \overline{\overline{L_1 \cap L_2}} = \overline{\overline{L_1} \cup \overline{L_2}} \in P$$

Puesto que P es cerrada para uniones y complementarios.

Ejercicio 5

Sea C una clase de funciones de los enteros no negativos en los enteros no negativos. Se dice que C es cerrada para la composición polinómica por la izquierda si $f(n) \in C \implies p(f(n)) = O(g(n))$ **para alguna función** $g(n) \in C$**, y donde p(n) es un polinomio cualquiera. Se dice que C es cerrada para la composición polinómica por la derecha si** $f(n) \in C \implies f(p(n)) = O(g(n))$ **para alguna función** $g(n) \in C$**, y donde p(n) es un polinomio cualquiera.**

Determinar cuales de las siguientes clases de funciones son cerradas para la composición polinómica por la derecha y cuales lo son por la izquierda.

Nota: en los dos últimos casos se entiende que las funciones se redondean al entero más cercano. En vez de considerar todos los polinomios, como estamos hablando de órdenes de complejidad será suficiente considerar los polinomios de la forma $p(n) = n^l$**, donde l es un número natural.**

a) A = $\{n^k : k > 0\}$
b) B = $\{nk : k > 0\}$
c) C = $\{k^n : k > 0\}$
d) D = $\{2^{n^k} : k > 0\}$
e) E = $\{log^k(n) : k > 0\}$
f) F = $\{log(n)\}$

Para comprobar si cada una de las familias anteriores es cerrada para la composición polinómica realizaremos las composiciones correspondientes y trataremos de acotar superiormente por otro elemento de la familia.

a)

- **Izquierda:** Sea $f \in A, entonces\ p(f(n)) = p(n^k) = (n^k)^l = n^{k+l} \in A$. Por tanto, A es cerrada por la izquierda.
- **Derecha:** Sea $f \in A, entonces\ f(p(n)) = f(n^l) = (n^l)^k = n^{k+l} \in A$. Por tanto, A es cerrada por la derecha.

b)

- **Izquierda:** Sea $f \in B, entonces\ p(f(n)) = p(nk) = (nk)^l = n^l k^l = O(n^l)$. Por tanto, B no puede ser cerrada por la izquierda.
- **Derecha:** Sea $f \in B, entonces\ f(p(n)) = f(n^l) = (n^l)k = O(n^l)$. Por tanto, B tampoco puede ser cerrada por la derecha.

c)

- **Izquierda:** Sea $f \in B, entonces\ p(f(n)) = p(k^n) = (k^n)^l = k^{n+l} = (k^l)^n \in C$. Por tanto, C es cerrada por la izquierda.
- **Derecha:** Sea $f \in B, entonces\ f(p(n)) = f(n^l) = O(k^{(n^l)})$. Por tanto, C no es cerrada por la derecha.

d)

- **Izquierda:** Sea $f \in B, entonces\ p(f(n)) = p(2^{n^k}) = (2^{n^k})^l = 2^{ln^k}$. Claramente el valor hallado no puede pertenece a la clase D. Por tanto, D no es cerrada por la izquierda.
- **Derecha:** Sea $f \in B, entonces\ f(p(n)) = f(n^l) = 2^{(n^l)^k} = 2^{n^{lk}} \in D$. Por tanto, D es cerrada por la derecha.

e)

- **Izquierda:** Sea $f \in B, entonces\ p(f(n)) = p(log^k(n)) = (log^k(n))^l = log^{k+l}(n) \in E$. Por tanto, E es cerrada por la izquierda.
- **Derecha:** Sea $f \in B, entonces\ f(p(n)) = f(n^l) = log^k(n^l) = l^k log^k(n) = O(log^k(n)) \in E$. Por tanto, E es cerrada por la derecha.

f)

- **Izquierda:** Sea $f \in B, entonces\ p(f(n)) = p(log(n)) = log^l(n)$. Claramente F no es cerrada por la izquierda.
- **Derecha:** Sea $f \in B, entonces\ f(p(n)) = f(n^l) = log(n^l) = llog(n) = O(log(n)) \in F$. Por tanto, F es cerrada por la derecha.

Ejercicio 6

Sea L =$\{wcw : w \in \{0,1\}^*$ **}**

Demostrar que L se puede reconocer en espacio log(n).

Construiremos una máquina de Turing MT que usará dos cintas con contadores binarios además de la cinta de entrada. En primer lugar cuenta el número de casillas ocupadas por w en el primer contador, cuando encuentre una c pasa a contar el número de casillas ocupadas por la palabra tras la c. Si nos encontramos otra c o al terminar de leer wcw los contadores no coinciden se rechaza. En caso contrario, pasamos a comprobar que las palabras separadas por la c son iguales. Para esto comprobaremos que coinciden símbolo a símbolo haciendo uso de los dos contadores auxiliares, donde uno de ellos indicará la posición actual dentro de la palabra y el otro la posición a comprobar. Es claro que MT acepta L, y puesto que los contadores ocupan un número de casillas igual a log(n), donde $n = |w|$ y nunca escribimos en la entrada podemos concluir que MT reconoce L en espacio log(n).

Ejercicio 7

Demostrar que si L está en P, entonces L* también está en P.

En primer lugar realizaremos unas consideraciones iniciales. Si $w \in L^*$, entonces w está formado por subcadenas de L, por tanto, vamos a estudiar todas las posibles subcadenas de w. Para ello notaremos como $w_{i,j}(i, j \in 1, 2, ..., |w|, i \leq j)$ a la subcadena de w formado por los elementos que ocupan las posiciones de la i a la j, ambas incluidas. Para comprobar si $w \in L^*$ iremos comprobando de forma recursiva si lo están las subcadenas de la forma $w_{i,j}$.

La forma de comprobarlo será la siguiente:

- Comprobamos las cadenas de longitud 1: $w_{i,i} \in L^* \Leftrightarrow w_{i,i} \in L$
- Para cada $j \leq n$ comprobamos las cadenas de longitud j. Es fácil comprobar que:
  $w_{i,i+j} \in L^* \Leftrightarrow w_{i,i+j} \in L$ o $(w_{i+1,i+j} \in L^*$ y $w_{i,i} \in L)$o $(w_{i,i+j-1} \in L^*$ y $w_{i+j,i+j} \in L)$

Es claro que en el último paso obtenemos si $w_{1,|w|} \in L^*$ o no. Observamos que el procedimiento anterior se realiza con dos bucles anidados, lo que supone un número de órden polinómico de comprobaciones de si una subcadena pertenece a L. Puesto que L está en P, sabemos que existe una MT que realiza esta tarea en órden polinómico, y, por tanto, el algoritmo descrito resuelve el problema de si $w \in L^*$ en tiempo polinómico. Finalmente, considerando la tésis fuerte de Church-Turing podemos garantizar la existencia de una máquina de Turing MT' que acepta L^* en tiempo de órden polinómico, y por tanto $L^* \in P$.