Entrega5.pdf

patriciacorhid

Modelos Avanzados de Computacion

4º Doble Grado en Ingeniería Informática y Matemáticas

Facultad de Ciencias
Universidad de Granada

Entrega de la relación 5

1. Demostrar que NL es cerrados para la clausura de Kleene y que L lo es $\Leftrightarrow$ L = NL

Primero, vamos a probar que NL es cerrado para la clausura de Kleene:
Sea $A \in NL$, hay que ver que $A^* \in NL$. Construyo una máquina de Turing que reciba como entrada una palabra $w \in A^*$ y de forma no determinista elija una partición de la palabra $w = w_1...w_n$. A continuación, simulo el comportamiento de la máquina de Turing que acepta A, que tiene complejidad en espacio $O(log(n))$, con cada una de esas partes como entrada. Si acepta todas las partes, entonces la máquina acepta w.

En cada una de las cintas de la MT se colocará:

- En la primera cinta colocamos la entrada.
- En la segunda cinta llevamos un contador indicando la casilla a la que señala el cabezal.
- En la tercera cinta llevamos un contador indicando el inicio de w_i.
- En la cuarta cinta llevamos un contador indicando el final de w_i.
- En las sucesivas cintas se simula el comportamiento de la MT que acepta A.

La Maquina de Turing que acepta A^* es:

> Empezamos con la entrada en la primera cinta y las demás cintas contienen el cero. Si la palabra de entrada es la palabra vacía, se acepta. El inicio de la primera parte en la que dividimos la palabra es la casilla 0.
>
> Repetimos el siguiente proceso hasta llegar al final de w:
>
> > Fijado el inicio de la parte w_i que estamos considerando, en el caso de w_0 es la casilla 0, en cada paso de la máquina a la derecha se elige si la palabra w_i termina o no en esa casilla (a la vez que se incrementan los contadores de la segunda y cuarta cinta).
> >
> > Una vez que se ha elegido cual va a ser el final de la palabra w_i, se simula el comportamiento de la MT que acepta A con la entrada w_i. Como MT tiene complejidad en espacio logaritmica, no escribirá sobre la entrada, así que su entrada está en la primera cinta y controlaremos usando el contador de la segunda cinta que, si éste se mueve en una posición anterior o posterior a las especificadas en la tercera y cuarta cinta, se interprete que está leyendo un carácter # (comparar dos números se hace restándolos, y eso se consigue en complejidad en espacio logarítmica).
> >
> > Si no se acepta w_i: Rechazo.
> > Si se acepta w_i: Borro el contenido en todas las cintas posteriores a la cuarta, para reusarlas en la siguiente iteración si fuese necesario. Muevo el cabezal a la posición indicada en la cuarta cinta mas uno y si el contenido no es # copio ese número en la tercera cinta, ya que será el inicio de la siguiente parte, w_{i+1}.
>
> Acepto la palabra w.

Por tanto, se acepta la palabra si y solo si existe una descomposición de ésta en palabras de A.

Veamos la complejidad de nuestro algoritmo:
La primera cinta sólo se lee, no se escribe en ella. Respecto a la segunda, tercera y cuarta cinta, como el contenido de estas cintas son contadores que valen como mucho $|w|$, se necesitan $log(|w|)$ casillas para escribirlos. El resto de cintas simulan el comportamiento de la MT que acepta A, que es de complejidad $O(log(|w|))$. Se usa una cinta adicional para comparar el puntero con los contadores en la tercera y cuarta cinta, tarea que se hace en complejidad logarítmica. Por tanto, este algoritmo no determinista tiene complejidad en espacio $O(log(|w|))$, es decir, $A^* \in NP$.

Ahora probaremos que L es cerrado para la clausura de Kleene $\Leftrightarrow L = NL$. La demostración, que expondremos a continuación, está recogida aquí:
`https://blog.computationalcomplexity.org/2015/04/is-logarithmic-space-closed-under.html?m=1`

$\Leftarrow$ Si $L = NL$, como hemos visto anteriormente que NL es cerrado para la clausura de Kleene, entonces L también lo es.

$\Rightarrow$ Supongamos ahora que L es cerrado para la clausura de Kleene. Sabemos que $L \subseteq NL$, falta probar que $NL \subseteq L$. Para ello, tomaremos un problema NL-completo y veremos que su lenguaje asociado está en L. Como el lenguaje asociado a un problema NL-completo está en L, todos los lenguajes de NL están en L.

Consideramos el siguiente problema NL-completo: Dado una tripleta (G, s, t) con G un grafo dirigido con aristas (i, j) con $i < j$, determinar si hay un camino del nodo s al nodo t.

Definimos el lenguaje $B = \{G\#i+1\#G\#i+2\#...\#G\#j\#|$ hay una arista que une al nodo i con el nodo j en $G\}$. El lenguaje $B \in L$ y la palabra $G\#s+1\#s+2\#...\#G\#t\#$ pertenece a B^* si y solo si hay un camino que conecta al nodo s con el nodo t. Como por hipótesis L es cerrado para la clausura de Kleene, $B^* \in L$, luego el lenguaje asociado al problema NL-completo (que es B^*) está en L. Por tanto $NL \subseteq L$.

2. Demostrar que todo lenguaje regular está en L.

Un lenguaje es regular si hay un autómata finito determinista que lo reconoce. En el funcionamiento de un autómata determinista sólo necesito leer la palabra de principio a fin, no escribo sobre la entrada en ningún momento ni necesito escribir en cintas auxiliares, sólo necesito cambiar de estado.

Sea $M = (Q, A, \delta, q_0, F)$ un autómata finito determinista que reconoce el lenguaje. Sin pérdida de generalidad podemos suponer que $F = q_f$. Vamos a simular el comportamiento de M con la máquina de Turing $N = (\bar{Q}, A, B, \bar{\delta}, q_0, \#, \bar{q}_f)$, con $\bar{Q} = Q \cup \{\bar{q}_f\}$ de manera que:

Por cada transición $\delta(q_i, a) = q_j$ de M, añadimos a N la transición $\bar{\delta}(q_i, a) = (q_j, a, D)$. Además, añadimos a N la transición $\delta(q_f, \#) = (\bar{q}_f, \#, S)$. De esta forma, N simula el comportamiento de M y sólo acepta si la palabra termina en un estado final.

Como no hemos añadido ningún contenido a la cinta, la complejidad en espacio es $O(log(n))$, por tanto pertenece a L.

3. Sea MULT = {a * b * c tales que a, b y c son números naturales en binario y a $\times$ b = c}. Demostrar que MULT $\in$ L.

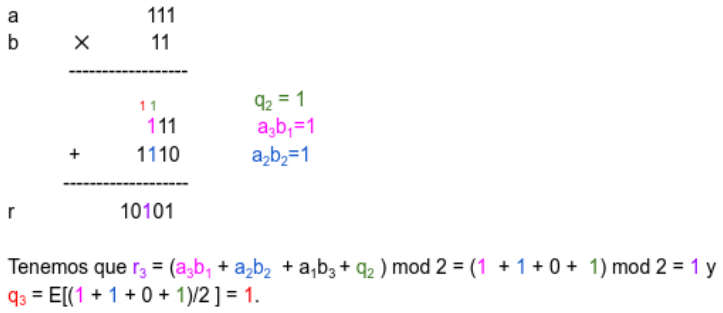
Para ver si una palabra pertenece a MULT crearé una MT que multiplica a y b símbolo a símbolo e irá comprobando si el símbolo resultante está en la posición correspondiente de c. Para resolver este ejercicio, usaré el algoritmo explicado en la sección "Optimizing space complexity" del enlace: `https://en.wikipedia.org/wiki/Multiplication_algorithm#Usage_in_computers`.

Sea $a = a_m...a_1$ y $b = b_k...b_1$, tomamos $n = max(m, k)$. En el algoritmo explicado en la página se ajusta la longitud de ambos a n para facilitar las demostraciones siguientes. Como a nosotros no nos interesa tocar la entrada, no lo haremos y asumiremos que si $b_r = 0$ con $r > k$ y $a_s = 0$ con $s > m$.
Llamaremos r_i al símbolo i-ésimo del resultado de multiplicar a por b, que habrá que comprobar que coincide

con c_i y q_i al acarreo producido al calcular r_i. Según la página, tenemos que:

$$r_i = (q_{i-1} + \sum_{j+k=i+1} a_j b_k) \bmod 2 \text{ y } q_i = E[(q_{i-1} + \sum_{j+k=i+1} a_j b_k)/2], q_0 = 0.$$

Veamos esto en un ejemplo:



(a) Ejemplo

Nuestra MT alculará r_i y q_i en cada iteración. Se comprobará que r_i es igual a c_i y se guarda q_i para usarlo en la siguiente iteración de la multiplicación, machacando el valor de q_{i-1}. La longitud de r puede ser como mucho $2n$.
La máquina de Turing funciona así:

Para $i = 0$ hasta $i = 2n$:
Calculamos r_i y q_i y los guardamos en la segunda y tercera cinta respectivamente, machacando los valores anteriores.
Comprobamos que existe c_i.
Si existe c_i:
Comprobamos $r_i = c_i$.
Si no lo es: Rechazo.
Si no existe c_i:
Comprobamos $r_i = 0$.
Si no lo es: Rechazo.
Aceptamos.

Veamos la complejidad en espacio que supone este algoritmo. La entrada la recorremos varias veces y no escribimos nada en una cinta de salida, así que la complejidad en espacio depende de las casillas usadas para escribir r_i y q_i.
Vamos a demostrar que $q_i \leq n \quad \forall i \in \{1, \ldots, 2n\}$ con $q_i = E[(q_{i-1} + \sum_{j+k=i+1} a_j b_k)/2]$.
Por inducción:

- Caso $n = 0$: $q_0 = 0 < n$
- Supongo que $q_{i-1} \leq n$:
  $\sum_{j+k=i+1} a_j b_k$ vale como mucho n, ya que a lo más tendré n unos en la columna a sumar. Por hipótesis de inducción $q_{i-1} + \sum_{j+k=i+1} a_j b_k \leq n + n = 2n$, por tanto, tenemos que $(q_{i-1} + \sum_{j+k=i+1} a_j b_k)/2 \leq n$ y por consiguiente $q_i = E[(q_{i-1} + \sum_{j+k=i+1} a_j b_k)/2] \leq n$.

Para escribir un valor menor o igual que n necesito a lo más $log_2(n)$ casillas.
Sabemos que $r_i \in \{0, 1\}$. Veamos su cálculo: $r_i = (q_{i-1} + \sum_{j+k=i+1} a_j b_k) \bmod 2$.

Por lo visto anteriormente $q_{i-1} + \sum_{j+k=i+1} a_j b_k \leq n+n = 2n$. Hacer módulo 2 es quedarnos con el último dígito del resultado. Para guardar el valor $2n$ necesito $log_2(2n)$ casillas, que es igual a $log_2(2) + log_2(n) = 1 + log_2(n)$ casillas.
Guardar r_i y q_i se consigue usando menos de $1 + 2log_2(n)$, así que nuestra máquina de Turing tiene una complejidad de espacio de $O(log_2(n))$, luego tenemos que $MULT \in L$.