MEMORIA DEL EJERCICIO 42 DE SIPSER

Por: Ana Martín Conejo

ENUNCIADO

Show that P is closed under the star operation.
(Hint: Use Dynamic programming. On input $y = y_1, y_2, y_3, \ldots y_n \epsilon \Sigma$, build a table indicating for each i ≤j wether the substring $y_{i,\ldots} y_i \in A^*$ for any $A \epsilon P$.)

Demuestra que P está encerrado bajo la operación estrella (cláusula de Kleene).
(Hint: Usa programación dinámica. Con un input $y = y_1, y_2, y_3, \ldots y_n \epsilon \Sigma$, construye una tabla indicando para cada i ≤j cada subcadena de $y_{i,\ldots} y_i \in A^*$ para cualquier $A \epsilon P$.)

Definimos el problema

Dado $A \epsilon P, queremos\ demostrar\ que\ A^* \epsilon P.$

Teniendo $A \epsilon P$ *existe una máquina de Turing determinista M_A con complejidad temporal $O(n^k)$ para algunos $k \geq 0$.*

Construimos un decisor determinístico (su comportamiento está completamente determinado por las entradas que recibe), usando M_A, para A^*.

Veremos que su complejidad temporal estará ligada a la polinomial.

Inciso: Voy a denotar la cadena 'y' como 'w' por comodidad.
La observación central en nuestra construcción es que w$\in A^*$ si y solo si una de las siguientes condiciones es verdad:

- *w* = ε
- $w \in A$
- $\exists u, v : \omega = uv \ \ Y\, v \in A^* \ \ Y\, u \in A^*$

En el decisor que vamos a construir a continuación denotaremos la subcadena de $w = w_1,\ w_2,\ w_3, \ldots w_n$ como $w_{i,j}$

Se construye una tabla con valores booleanos donde tabla(i , j)= true si $w_{i,j} \in A^*$, inicialmente esta tabla quedará inicializada a false.

Hacemos esto considerando todas las subcadenas de w empezando con las de tamaño 1 y terminando con las de tamaño n.

Demostración del algoritmo:

Con entrada $w = w_1,\ w_2,\ w_3, \ldots w_n$:

```
1.  IF w = ε then accept, else
2.  FOR l := 1 to n
3.      FOR i := 1 to n – ( l – 1 )
4.      J := i + l – 1
5.      RUN M_A on w_{i,j}
6.          IF M_A accepts w_{i,j} THEN tabla( i, j) = true
7.          ELSE
8.              FOR k:= i TO j-1
9.                  IF tabla( i , k) = true AND tabla(k+1,j)=true
10.                 THEN tabla(i,j):=true
11. IF tabla (1,n)= true then accept, else reject
```

Explicación más detallada del algoritmo:

Línea 1.-) *Cuando el conjunto de subcadenas es vacío, tendríamos* $w = \varepsilon$ *lo cual entraría en la cláusula de Kleene, luego se acepta.*
Línea 2.-) *Iteramos sobre todas las longitudes posibles de subcadenas, desde la longitud 1 hasta la longitud total de la cadena w.*
Línea 3.-) *En este bucle interior, iteramos sobre todas las posiciones iniciales posibles de subcadenas de longitud l en la cadena w. Para ello vamos desde la posición inicial i=1, hasta la última posición válida desde la cual se puede formar una subcadena de longitud l.*
Línea 4.-)*Se calcula la posición final de la subcadena actual, basada en la posición inicial i y la longitud de la cadena l.*
Línea 5.-)*Se ejecuta la máquina de Turing determinista en la subcadena* $w_{i,j}$ *de la cadena w, lo que determinará si la subcadena pertenece al conjunto A.*

Línea 6.-) *Si la Máquina de Turing acepta la subcadena, se marca la entrada correspondiente en la tabla como true.*
Línea 7.) *Si la máquina rechaza la subcadena, procede con los siguientes puntos.*
Línea 8.-)*Este bucle itera sobre todas las posibles divisiones de la subcadena* $w_{i,j}$ *en dos subcadenas más pequeñas, buscando una combinación donde ambas subcadenas pertenecen al cierre de Kleene de A.*

Líneas 9/10.-)*Si se encuentra una combinación donde ambas subcadenas pertenecen al cierre de Kleene de A, entonces se marca la entrada correspondiente en la tabla como true.*
Línea 11.-) *Por último, se verifica si toda la cadena w pertenece al cierre, si la entrada correspondiente en la tabla es true, acepta la cadena, si es false, la rechaza.*
Inciso: tabla(1,n) correspondería a la subcadena que abarca todo w.

Analizamos la complejidad del algoritmo.

A continuación, vamos a analizar la complejidad de nuestro algoritmo, vemos que se utilizan tres bucles for, uno contenido en otro, de los cuales cada uno pueden recorrerse como mucho a $O(n)$.
En el segundo for corremos M_A en una entrada donde el tamaño es como máximo n.
Luego el tiempo total será como mucho:

$$\boldsymbol{O(n) \cdot O(n) \cdot \left(O\left(n^k\right) + O(n)\right)}$$

Donde:
-El primer $O(n)$ representa el primer bucle for.
-El segundo $O(n)$ representa el segundo bucle for.
-El término $O(n^k)$ representa la complejidad temporal de la ejecución de la máquina de Turing M_A en una entrada de tamaño máximo n, que se deriva de la premisa inicial de que M_A tiene una complejidad temporal de $O(n^k)$, ya que tenemos en cuenta todas las combinaciones.
-El término $O(n)$ representa la complejidad temporal de las operaciones adicionales dentro del bucle for.

Que operándolo nos queda como:

$$\boldsymbol{O\left(n^{2+(\max(k,1))}\right)}$$

Que aproximadamente queda en:

$$\boldsymbol{O(n^3)}$$

Lo cual es polinomial en n.

Esto implica entonces que P está encerrado bajo la clausula de Kleene.