UNIVERSIDAD SIMON BOLIVAR **25 de marzo de 2010**
DEPARTAMENTO DE COMPUTACION
MAESTRIA EN CIENCIAS DE LA COMPUTACION **Ing. Angarita Arocha Rafael**
ASIGNATURA: CI-7541 - Teoria de la computación. **Lic. Claro Mosqueda Alejandro**

TAREA N^o 3

Ejercicio 8.8: Sea $EQ_{\text{REX}} = \{\langle R, S\rangle \mid R \text{ y } S \text{ son expresiones regulares equivalentes}\}$. Muestre que $EQ_{\text{REX}} \in PSPACE$.

Solución: En el teorema 1.54 [1, p. 66] se muestra como convertir una expresión regular en tiempo lineal en una NFA que es lineal en tamaño de la expresión regular. Por lo tanto, podemos transformar las expresiones regulares R y S de EQ_{REX} en NFAs N_R y N_S tal que,

$$EQ_{\text{NFA}} = \{\langle N_R, N_S\rangle \mid N_R \text{ y } N_S \text{ son NFAs tales que } L(N_R) = L(N_S)\}$$

Por otro lado, observe que la siguiente máquina de Turing no-determinista M,

$M =$ "En entrada $\langle N_R, N_S\rangle$ donde N_R y N_S son NFAs:

1. Colocar una marca en los estados iniciales de N_R y N_S.
2. Repetir $2^{n_r+n_s}$ veces. Donde n_r y n_s son el número de estados de N_R y N_S.
3. Seleccionar un símbolo s de forma no-determinista.
4. Mover el marcador en N_R y N_S de acuerdo con las reglas dadas por las respectivas funciones de transición y s.
5. Si el marcador de N_R llega a un estado de aceptación y el marcador de N_S no, entonces ***rechazar***.
6. Si el marcador de N_S llega a un estado de aceptación y el marcador de N_R no, entonces ***rechazar***.
7. Si el marcador de N_R llega a un estado de aceptación y el marcador de N_S también llega a un estado de aceptación, entonces ***aceptar***.
8. ***rechazar***."

decide EQ_{NFA}. El limite $2^{n_r+n_s}$ en el paso 2 se debe a que la longitud de una cadena, que es rechazada o aceptada por N_R y N_S, es a lo sumo esta cantidad, ya que sólo hay $2^{n_r+n_s}$ formas diferentes de colocar las dos marcas. Esta máquina no almacena la cadena que se va generando, en su lugar mantiene en memoria únicamente el símbolo que se va procesando durante la simulación de N_R y N_S. Por lo tanto $EQ_{\text{NFA}} \in NPSPACE$. Además, como se comentó inicialmente, la construcción de N_R y N_S requiere espacio lineal. En consecuencia, $EQ_{\text{REX}} \in NPSPACE$. Finalmente, por el teorema de Savitch conocemos que $PSPACE = NPSPACE$, lo que implica que $EQ_{\text{REX}} \in PSPACE$.

Ejercicio 8.11: Muestre que, si todo lenguaje NP-hard es también PSPACE-hard, entonces PSPACE = NP.

Solución:

Demostración. Partimos de la suposición de que todo lenguaje en NP-hard esta en PSPACE-hard. Sabemos que NP $\subseteq$ PSPACE por la explicación dada en [1, p. 308]. También sabemos que NP-completo $\subseteq$ NP-hard, por lo tanto todo lenguaje en NP-completo esta en PSPACE-hard. Además, SAT es NP-completo [1, Teorema 7.37, p. 276], en consecuencia $SAT \in$ PSPACE-hard.

Por otro lado, como todo lenguaje A en PSPACE es reducible en tiempo polinomial a un lenguaje B que esté en PSPACE-hard [1, Definicion 8.8, p. 309], entonces todo lenguaje A en PSPACE es reducible a SAT en tiempo polinomial. Pero $SAT \in$ NP, por lo cual $A \in$ NP y PSPACE $\subseteq$ NP. En conclusión, PSPACE = NP. □

Ejercicio 8.17: Sea A el lenguaje de paréntesis anidados correctamente. Por ejemplo, (()) y (() (())) () están en A, pero) (no lo esta. Muestre que A esta en L.

Solución:

Demostración. Para decidir el lenguaje A, es necesario llevar la cuenta de los paréntesis izquierdos sin par encontrados mientras se escanea la cadena de entrada de izquierda a derecha. Si en algún momento hay más paréntesis derechos que izquierdos, la cadena de entrada no puede pertenecer al lenguaje A. Como la cuenta de los paréntesis es $O(n)$, donde n es la longitud de la cadena de entrada, entonces sólo es necesario espacio $O(\log n)$ para el contador. La siguiente MT M decide a A usando espacio $O(\log n)$:

$M =$ "En entrada w donde w es una cadena de paréntesis:

1. Hacer $p \leftarrow 0$, donde p es el número de paréntesis izquierdos encontrados en w hasta el momento.
2. Para cada símbolo c en w (escaneando w de izquierda a derecha) hacer:
3. Si $c =$ '(', entonces $p \leftarrow p + 1$
4. Si no, si $c =$ ')', entonces $p \leftarrow p - 1$
5. Si $p < 0$, entonces ***rechazar*** ya que se han encontrado más paréntesis derechos que izquierdos.
6. Si $p = 0$, ***aceptar*** ya que los paréntesis están correctamente anidados. Si no, ***rechazar***."

Claramente el contador p requiere espacio $O(\log n)$, ya que puede ser incrementada a lo sumo n veces. Por lo tanto, M decide a A en espacio $O(\log n)$, lo que implica que $A \in L$.

□

Ejercicio 8.25: Un grafo no dirigido es ***bipartito*** si sus nodos pueden ser divididos en dos conjuntos tal que todas las aristas vayan de un nodo de un conjunto a un nodo del otro conjunto. Demuestre que un grafo es bipartito si y sólo si este no contiene un ciclo compuesto por un número impar de nodos.

Sea

$$BIPARTITE = \{\langle G \rangle \,|G \text{ es un grafo bipartito}\}.$$

Demuestre que $BIPARTITE \in$ NL.

Solución:

a) Demuestre que un grafo es bipartito si y sólo si este no contiene un ciclo compuesto por un número impar de nodos.

Demostración. Primero se demostrará que si un grafo es bipartito, entonces no debe contener un ciclo compuesto por un número impar de nodos. Suponga que un grafo bipartito contiene un ciclo compuesto por un número impar de nodos y sean $n_1, n_2, n_3, \ldots, n_{2k}, n_{2k+1}$ los nodos del ciclo. Si n_1 esta en algún conjunto A, entonces n_2 debe estar en el conjunto B, n_3 en el conjunto A, etc. Por inducción observamos que todos los nodos con subíndice impar deben estar en el conjunto A, y todos los nodos con subíndice par deben estar en el conjunto B. El primer nodo n_1 y el último nodo n_{2k+1} del ciclo están en A, lo cual es una contradicción porque están conectados entre si. Otra solución es dar color a cada nodo. Si un grafo es bipartito, es posible pintar los nodos usando dos colores, rojo y azul, tal que no exista una arista conectando dos nodos del mismo color. Esto es claramente imposible de hacer en un ciclo compuesto por un número impar de nodos.

Por otro lado, con la intensión de demostrar que si un grafo no contiene ciclos con un número impar de nodos, entonces el grafo es bipartito, suponga que un grafo no contiene ciclos con un número impar de nodos y realice el siguiente procedimiento:

1. Tomamos un nodo y le ponemos la etiqueta A.
2. Repetir hasta que todos los nodos estén etiquetados:
3. Etiquetamos a todos sus vecinos (sin etiquetar) como B.
4. Etiquetamos todos los vecinos (sin etiquetar) de aquellos nodos etiquetados con B como A.

Suponga que quedaron dos nodos adyacentes x y y con la misma etiqueta. Esto quiere decir que los dos nodos fueron alcanzados en un número impar de pasos desde el nodo raíz, o que los dos nodos fueron alcanzados en un número par de pasos desde el nodo raíz. En ambos casos, si se excluye el nodo raíz, entonces el número total de nodos visitados para llegar de x al nodo inicial más el número total de nodos visitados para llegar de y al nodo inicial es par. Pero si también se cuenta el nodo inicial, la suma del número total de nodos visitados sera impar, lo que contradice la hipótesis inicial que afirma que el grafo no contiene ciclos con un número impar de nodos. Por lo tanto, el grafo es bipartito.

□

b) Demuestre que BIPARTITE $\in$ NL.

Demostración. La siguiente MT M,

M = "En entrada $\langle G \rangle$:

1. Hacer *contador* $\leftarrow 0$.
2. Seleccionar de forma no-determinista un nodo inicial y asignarlo a la variable *inicio*.
3. Seleccionar de forma no-determinista un nodo sucesor y asignarlo a la variable *sucesor*.
4. Mientras *contador* sea menor o igual al número de nodos en G hacer:
5. Si *sucesor* = *inicio* y *contador* es impar, entonces ***aceptar***.
6. Seleccionar de forma no-determinista un sucesor de *sucesor* y asignarlo a *sucesor*
7. Mientras *contador* es mayor que el número de nodos en G, entonces ***rechazar***."

decide a $\overline{BIPARTITE}$ en espacio $O(\log n)$ ya que sólo se requiere espacio para representar los nodos *inicio* y *sucesor*; y para la variable *contador* cuyo máximo valor es el número de nodos en G. En consecuencia $\overline{BIPARTITE} \in$ NL y $BIPARTITE \in$ coNL. Pero, es conocido que NL=coNL [1, Teorema 8.27]. Por lo tanto, $BIPARTITE \in$ NL.

□

Ejercicio 8.27: Recordemos que un grafo dirigido es ***fuertemente conexo*** si cada dos nodos están conectados por un camino dirigido en cada dirección. Sea

$$STRONGLY\text{-}CONNECTED = \{ \langle G \rangle \mid \ G \text{ es un grafo fuertemente conexo} \}.$$

Demuestre que $STRONGLY\text{-}CONNECTED$ es NL-completo.

Solución:

Demostración. Primero se debe demostrar que $STRONGLY\text{-}CONNECTED \in$ NL. Para ello, considere la siguiente MT M que decide a $\overline{STRONGLY - CONNECTED}$:

M = "En entrada $\langle G \rangle$, donde G es un grafo dirigido:

1. Seleccionar no deterministicamente dos nodos a y b.
2. Correr PATH(a,b).
3. Si rechaza, el grafo no es fuertemente conexo, entonces ***aceptar***.
4. Si no, ***rechazar***."

El almacenamiento de los números de nodos a y b sólo requiere espacio $O(\log n)$, y $PATH$ también requiere únicamente espacio $O(\log n)$, por lo tanto $\overline{STRONGLY - CONNECTED} \in$ NL. Finalmente $STRONGLY\text{-}CONNECTED \in$ coNL, pero como sabemos que NL=coNL [1, Teorema 8.27], y en consecuencia $STRONGLY\text{-}CONNECTED \int$ NL.

Por otro lado, para demostrar la completitud de $STRONGLY\text{-}CONNECTED$, basta con demostrar que PATH $\leq_L$ $STRONGLY\text{-}CONNECTED$. Para ello, considere la siguiente reducción

de PATH a $STRONGLY\text{-}CONNECTED$:

"En entrada $\langle G, s, t\rangle$:

1. Copiar G en la cinta de salida.
2. Para cada nodo i en G hacer:
3. Imprimir una arista de i a s.
4. Imprimir una arista de t a i.
5. Si rechaza, el grafo no es fuertemente conexo, entonces ***aceptar***.
6. Si no, ***rechazar***."

Si existe un camino de s a t, el grafo construido es fuertemente conexo. Si no existe un camino de s a t, entonces el grafo construido no es fuertemente conexo ya que las aristas adicionales van a s y salen de t, por lo que no habría un camino de s a t. La salida es $O(n)$, por lo cual el único espacio requerido es aquel para el contador i en el ciclo de la reducción. En conclusión, $STRONGLY\text{-}CONNECTED$ es NL-completo. □

Ejercicio 9.12: Describa cual es el error en la siguiente "demostración" engañosa de $P \neq NP$. Asuma que $P = NP$ y obtenga una contradicción. Si $P = NP$, entonces $SAT \in P$ y para algún k, $SAT \in TIME(n^k)$. Debido a que cada lenguaje en NP es polinomialmente reducible a SAT, se obtiene que $NP \subseteq TIME(n^k)$. Por lo tanto $P \subseteq TIME(n^k)$. Pero, por el teorema de las jerarquía en tiempo, $TIME(n^{k+1})$ contiene un lenguaje que no esta en $TIME(n^k)$, lo cual contradice $P \subseteq TIME(n^k)$. Por lo tanto $P \neq NP$.

Solución:

El error en la demostración se encuentra en la afirmación $NP \subseteq TIME(n^k)$ si $SAT \in TIME(n^k)$, ya que la reducción polinomial de los problemas en NP a SAT no está acotada en tiempo por $O(n^k)$. Por lo tanto, no es posible asegurar que $NP \subseteq TIME(n^k)$ y mucho menos que $P \subseteq TIME(n^k)$. En consecuencia, no es posible demostrar por contradicción que $P \neq NP$ con este argumento.

Ejercicio 9.19: Sea el problema **UNIQUE-SAT**,

UNIQUE-SAT $= \{\varphi \mid \varphi$ es una fórmula booleana que tiene una sola asignación que la satisface$\}$.

Demuestre que **UNIQUE-SAT** $\in \mathrm{P}^{SAT}$.

Solución:

Demostración. Para demostrar que **UNIQUE-SAT** $\in \mathrm{P}^{SAT}$ es conveniente formular el problema de una forma ligeramente diferente,

$$\textbf{UNIQUE-SAT} = \{\varphi \mid \exists x \forall z : (x \text{ satiface } \varphi) \wedge (z \neq x \rightarrow z \text{ no satisface } \varphi)\}.$$

donde $x = x_1 x_2 \ldots x_n$ y $z = z_1 z_2 \ldots z_n$ son asignaciones de φ [2]. La idea es construir una máquina de Turing con un oráculo de SAT que escoja una asignación x que satisfaga φ y verifique universalmente que ninguna otra asignación z es satisfactoria. La MT M,

$M =$ "En la entrada φ donde φ es una formula booleana:

1. Consultar al oráculo de SAT con φ como entrada.
2. Si el oráculo rechaza, ***rechazar***.
3. Construir $\rho \leftarrow$ TRUE
4. Para $i \leftarrow 1$ hasta n realizar: (donde n es el número de variables en φ)
5. Consultar el oráculo SAT con $\varphi \wedge \rho \wedge x_i$ como entrada.
6. Si el oráculo acepta, simbólicamente $\rho \leftarrow \rho \wedge x_i$. Si no, $\rho \leftarrow \rho \wedge \neg x_i$.
7. Consulta al oráculo de SAT con $\varphi \wedge \neg\rho$ como entrada.
8. Si el oráculo rechaza, ***aceptar***. Si no, ***rechazar***."

reconoce **UNIQUE-SAT** en tiempo polinomial siguiendo la idea antes expuesta. Los pasos del 4 al 6, construyen una asignación que satisface tanto a φ como a ρ. Por lo tanto la única manera que $\varphi \wedge \neg\rho$ sea satifasible es debido a la existencia de otra asignación que también satisface φ pero no a ρ. Finalmente, de esto se concluye que **UNIQUE-SAT** $\in \mathrm{P}^{SAT}$.

□

Ejercicio 9.21: Una máquina de Turing con oráculo de k-consultas es una TM con oráculo que permite realizar a lo sumo k consultas en cada entrada. Una máquina de Turing M con un oráculo de k-consultas para A es representado por $M^{A,k}$ y $P^{A,k}$ es la colección de lenguajes que son decidibles en tiempo polinomial por esta clase de máquinas.

a Demuestre que $\mathrm{NP} \cup \mathrm{coNP} \subseteq \mathrm{P}^{SAT,1}$.

b Asuma que NP $\neq$ coNP. Demuestre que $\mathrm{P} \cup \mathrm{coNP} \subsetneq \mathrm{P}^{SAT,1}$.

Solución:

a) Demuestre que $\mathrm{NP} \cup \mathrm{coNP} \subseteq \mathrm{P}^{SAT,1}$.

Demostración. Primero se mostrara que $NP \subseteq \mathrm{P}^{SAT,1}$. Para esto basta con demostrar que,

$$\forall A \in \mathrm{NP} \rightarrow A \in \mathrm{P}^{SAT,1}.$$

Sabemos que SAT es NP-completo, y por lo tanto es posible reducir A a SAT ($A <_p SAT$) por cierta función $f(w)$ computable en tiempo polinomial. Conocida la función $f(w)$ es entonces posible construir una máquina de Turing con oráculo de 1-consulta $M^{SAT,1}$,

$M^{SAT,1} =$ "En la entrada w:

1. Computar la reducción $f(w)$.
2. Consulta al oráculo de SAT con $f(w)$ como entrada.
3. Si el oráculo rechaza, ***rechazar***. Si no, ***aceptar***."

que decide el lenguaje $A \in$ NP. Claramente los pasos 1 y 2 son computables en tiempo polinomial, y en consecuencia queda demostrado que $NP \subseteq \mathrm{P}^{SAT,1}$.

Lo segundo que se debe demostrar es que $coNP \subseteq \mathrm{P}^{SAT,1}$. Para esto basta con demostrar que,

$$\forall \overline{A} \in \mathrm{NP} \rightarrow A \in \mathrm{P}^{SAT,1}.$$

Esta última afirmación es sencilla de demostrar con una pequeña alteración a la máquina $M^{SAT,1}$ mostrada anteriormente,

$Q^{SAT,1}$ = "En la entrada w:

1. Construir la reducción $f(w)$.
2. Consulta al oráculo de SAT con $f(w)$ como entrada.
3. Si el oráculo rechaza, ***aceptar***. Si no, ***rechazar***."

Este cambio en el funcionamiento implica que $Q^{SAT,1}$ decide a los lenguajes que no se encuentran en NP en tiempo polinomial con una sola consulta al oráculo de SAT. En consecuencia $coNP \subseteq \mathrm{P}^{SAT,1}$.

Finalmente como tanto NP como coNP están contenidos en $\mathrm{P}^{SAT,1}$, entonces su unión claramente también lo está.

□

b) Asuma que NP $\neq$ coNP. Demuestre que P $\cup$ coNP $\subsetneq \mathrm{P}^{SAT,1}$.

Demostración. La suposición NP $\neq$ coNP implica que P $\neq$ NP ya que es conocido que P es cerrado bajo complementación. En consecuencia, $P \subsetneq NP$ y por lo tanto $P \subsetneq \mathrm{P}^{SAT,1}$.

Por otro lado, como quedo demostrado en el apartado anterior, $coNP \subseteq \mathrm{P}^{SAT,1}$ y NP $\cup$ coNP $\subseteq \mathrm{P}^{SAT,1}$. Esto implica que si $P \subsetneq NP$, entonces P $\cup$ coNP $\subsetneq$ NP $\cup$ coNP y por lo tanto P $\cup$ coNP $\subsetneq \mathrm{P}^{SAT,1}$.

□

Referencias

[1] Sipser, Michael. *Introduction to the Theory of Computation.* International Thomson Publishing, 2nd Edition, 2006.

[2] Papadimitriou, Christos. *On the complexity of unique solutions.* J. ACM, p.392-400, 1984.