Ma50b Complejidad Computacional	**12 de Abril de 2008**

Pauta Control 1

Prof. Cátedra: M. Kiwi	*Prof. Auxiliar: P. Camacho*

PROBLEMA 1:

(i).- Sea L un lenguaje y sea $L' = \text{mín}(L)$. Queremos probar que si L es regular entonces L' también lo es.

Consideremos el AFD $M = (Q,\Sigma,\delta,s,F)$ tal que $L = L(M)$. La observación clave es notar que $\omega \in L'$ si y sólo si M en la entrada ω entra en un estado de aceptación en el momento que lee el último caracter de ω, pero no antes. En efecto, sea $\omega = \omega_1 \dots \omega_n \in L'$. Supongamos que para algún $0 < i < |\omega|$ se tiene que $\delta(s,\omega_1 \dots \omega_i) \in F$. Sigue que $\omega_1 \dots \omega_i \in L$ y por lo tanto $\omega_1 \dots \omega_i \notin L'$. Por otro lado, si $\delta(s,\omega) \in F$ pero para todo $0 < i < |\omega|$ se tiene que $\delta(s,\omega_1 \dots \omega_i) \notin F$, entonces $\omega \in L$ y $\omega_1 \dots \omega_i \notin L$, i.e. $\omega \in L'$.

Basta entonces construir un autómata finito no-determinista M' que en la entrada ω acepta si y sólo si M en la entrada ω entra en un estado de aceptación en el momento que lee el último caracter de ω, pero no antes. Si $M = (\Sigma,Q,\delta,s,F)$ definimos el autómata $M' = (Q,\Sigma,\delta',s,F)$ cuyo diagrama de estado corresponde al de M pero eliminando todas las transiciones que salen de los estados $q \in F$. Notar que por construcción de M', si este acepta ω entonces M en la entrada ω acepta y nunca entra y sale de un estado de aceptación. El converso es obvio.

(ii).- Sea $U = (Q,\Sigma,\delta,q_0,F)$ como en el enunciado. Usaremos la técnica para simular un autómata finito no-determinista por uno determinista. Específicamente, consideramos el autómata finito determinista M sobre el alfabeto Σ, estados 2^Q, estado de partida $\{q_0\}$, estados de aceptación $\{C : C \subseteq F\}$ y función de transición $\Delta : 2^Q \times \Sigma \to 2^Q$ dada por

$$\Delta(C,\sigma) = \cup_{q \in C}\delta(q,\sigma).$$

De la forma usual se extiende Δ a una función de $2^Q \times \Sigma^* \to 2^Q$. Es fácil verificar que U acepta ω si y sólo si $\Delta(\{q_0\},\omega) \subseteq F$, i.e. cuando M acepta ω. Sigue entonces que el lenguaje reconocido por U es regular.

Una forma equivalente de abordar el problema es considerar el lenguaje L aceptado por U y notar que $\omega \notin L$ si y solo si existen $q_1,\dots,q_n \in Q$ tal que $q_{i+1} \in \delta(q_i,\omega)$ si $0 \le i < n$ y donde además $q_n \notin F$. Equivalentemente, $\omega \notin L$ si y sólo si ω es aceptada por el autómata finito no-determinista $N = (Q,\Sigma,\delta,q_0,Q \setminus F)$. Sigue que N acepta el complemento de L. Luego, dicho complemento es regular, y como los lenguajes regulares son cerrados por complementación se concluye que L es regular.

PROBLEMA 2:

(i).- Sea A un autómata apilador tal que $L = L_A$ y sea M una autómata determinista finito tal que $R = L_M$. Construiremos un autómata apilador A' para $L \cap R$ que simula M y A en paralelo. La pila de A' será utilizada para simular la pila de A. El autómata A' se ilustra en la Fig. 1. Si A no mueve su cabeza lectora, entonces A' simula una transición de A sin cambiar el estado de M. Si A mueve la cabeza lectora (i.e., lee un símbolo), entonces A' simula las transiciones de A y M en dicho símbolo. Si una vez leída toda la entrada tanto A como M se encuentran en el estado de aceptación, entonces A' acepta.

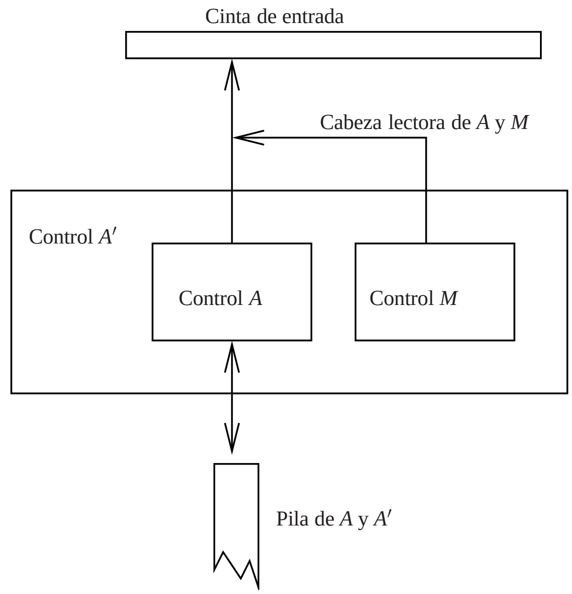

Figura 1: Autómata apilador A' que reconoce $L \cap R$.

(ii).- Es fácil ver que $L = L_0 \cup L_1 \cup L_2$ donde

$$\begin{aligned} L_0 &= \{a^n b^m : n < m\} , \\ L_1 &= \{a^n b^m : m < n < 2m\} , \\ L_2 &= \{a^n b^m : 2m < n\} . \end{aligned}$$

Por propiedades de clausura de lenguajes libres de contexto, en particular por cerradura bajo unión, bastará probar que L_i es lenguaje libre de contexto cualquiera sea el $i \in \{1,2,3\}$.

Es fácil ver que los lenguajes L_0 y L_2 son reconocidos por los autómatas apiladores A_0 y A_2 que se describen a continuación.

El autómata A_0 hará un `push` de a por cada a y un `pop` por cada b que lea. El autómata aceptará si antes de terminar de leer la entrada su pila queda vacía. Formalmente,

A_0 = En la entrada ω,

(1) Marca el fondo de la pila con \$

(2) Lee el siguiente caracter c de la entrada

(3) Si $c = a$, entonces hace un `push` de a y regresa a (2)

  # $c \neq a$

(4) Si $c \neq b$ rechaza y en caso contrario

  Si \$ está en el tope de la pila, entonces acepta

  Hace un `pop`

  Lee el siguiente caracter c de la entrada y regresa a (4)

El autómata A_2 hará un `push` de a por cada a que lea, y un doble `pop` por cada b que lea. El autómata aceptará si al terminar la pila no esta vacía. Formalmente,

A_2 = En la entrada ω,
(1) Marca el fondo de la pila con $
(2) Lee el siguiente caracter c de la entrada
(3) Si $c = a$, entonces hace un `push` de a y regresa a (2)
$c \neq a$
(4) Si $c \neq b$ rechaza, y en caso contrario repite dos veces
Hace un `pop`
Si $ está en el tope de la pila, entonces rechaza
(5) Si no hay más caracteres de entrada, entonces acepta
(6) Lee el siguiente caracter c de la entrada, si $c \neq b$ rechaza, y en caso contrario va a (4)

El autómata apilador A_1 hará uno o dos `push` de a por cada a que lea. La decisión de hacer uno o dos `push` será realizada de forma no-determinista. No obstante, el autómata tendrá cuidado de hacer al menos un doble `push` en una de las lecturas de a, pero no en todas. Por cada b leído el autómata hará un doble `pop`, y aceptará si al momento exacto de terminar de leer la entrada la pila está vacía. Formalmente,

A_1 = En la entrada ω,
(1) Marca el fondo de la pila con $
(2) Fija $f_1 = f_2 = 0$
(3) Lee el siguiente caracter c de la entrada
(4) Si $c = a$, entonces de manera no-determinista hace $i \in \{1,2\}$
`push` de a, fija $f_i = 1$, y regresa a (3)
$c \neq a$
(5) Si $c \neq b$ rechaza, y en caso contrario hace dos `pop`
(6) Lee el siguiente caracter c de la entrada y regresa a (5)
(7) Si $ está en el tope de la pila y $f_1 = f_2 = 1$, entonces acepta

Es fácil ver que $L(A_0) = L_0$ y $L(A_2) = L_2$. Para ver que $L(A_1) = L_1$ basta notar que A_1 acepta ω si y sólo si $\omega = a^{n+k}b^m$, $1 \leq k < n$ y $n + k = 2m$, donde k es el el número de veces que A_1 hace un doble `push`.

PROBLEMA 3:

(i).- La idea central de la simulación de una mT M estándar por una de re-escritura única R es que la máquina de re-escritura va generando secuencialmente descripciones del estado de la cinta y la posición de la cabeza lectora de M en la entrada ω. Dicho estado y posición de la cabeza lectora de M se representará como una secuencia de caracteres en Σ uno de los cuales tendrá una marca especial, digamos $\widehat{\ }$, representando la posición de la cabeza lectora de M. Sean $C_0, C_1, C_2, \ldots$ las descripciones de la cinta y posición de la cabeza lectora por las que pasa M a partir de la entrada ω, siendo C_i la descripción correspondiente a la situación en la que se encuentra M después de haber realizado i transiciones. La máquina de re-escritura única R procederá en etapas. Al comienzo de la i-ésima etapa el estado de R será como el que se ilustra en la Fig. 2. Inicialmente R hace los siguientes pasos:

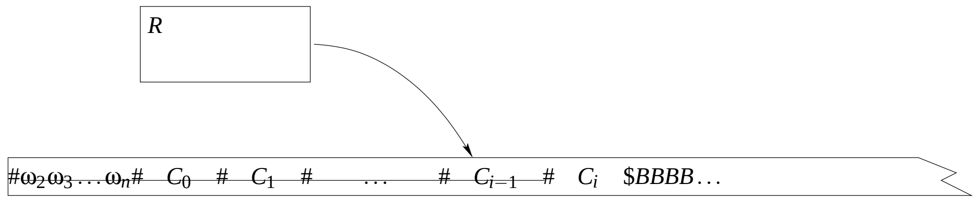

Figura 2: Estado de la cinta de R al comienzo de la i-ésima etapa. Las rayas horizontales indican cuales celdas han sido sobre-escritas.

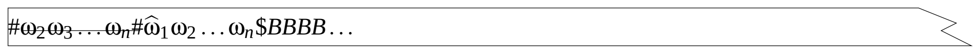

Figura 3: Estado de la cinta de R terminada la etapa inicial. Las rayas horizontales indican cuales celdas han sido sobre-escritas.

1. Lee el símbolo ω_1 que está en la primera celda de la cinta, escribe # en dicha celda y mueve su cabeza lectora hasta la primera celda en blanco a la derecha de la entrada. Escribe \$ en dicha celda, mueve la cabeza lectora una celda a la derecha, escribe $\widehat{\omega}_1$ y mueve su cabeza lectora a la izquierda hasta la celda que contiene # (la primera celda de la cinta).

2. Repite hasta que no hayan símbolos sobre-escritos a la izquierda de \$:

   - Se mueve de izquierda a derecha hasta encontrar la primera celda C con un símbolo σ que no ha sido sobre-escrito con una X.
   - Re-escribe C con una X si $\sigma \neq \$$, y # en caso contrario.
   - Desplaza la cabeza lectora hasta la primera celda de la derecha que contiene un blanco, en dicha celda escribe σ.
   - Vuelve la cabeza lectora a la izquierda hasta encontrar el símbolo #.

Terminada la etapa inicial el estado de la cinta de R es el que se muestra en la Figura 3.

En la etapa i-ésima la máquina de re-escritura única R hace esencialmente lo mismo que durante la etapa inicial, es decir “copia” C_i a la derecha del símbolo \$ cuidándose de actualizar C_i y reemplazarla por C_{i+1}. Esto implica copiar todos los símbolos de C_i salvo quizás aquel marcado con $\widehat{\ }$ y el inmediatamente a la izquierda si la cabeza lectora se mueve a la izquierda en la $(i+1)$-ava transición, o el inmediatamente a la derecha en caso contrario. Detalles omitidos.

(ii).- Sea $L = \{\omega \in \Sigma^* : \exists \pi, \omega\#\pi \in D\}$ donde D es decidible. Sea M una mT determinista M que decide D. Sin pérdida de generalidad podemos asumir que M tiene una sola cinta. Sea S la mT no–determinista tal que

S = En la entrada ω,

(1) Mueve su cabeza a la derecha hasta la primera celda en blanco

(2) Escribe # y mueve la cabeza a la derecha

(3) Usando no–determinismo

– adivina un bit y lo escribe en su cinta,

– mueve su cabeza a la derecha,

– adivina si pasa a (3) o a (4)

(4) Mueve su cabeza a la izquierda hasta el comienzo de su cinta.

(5) Simula M tomando como entrada el contenido de la cinta.

(6) Acepta si M acepta, y rechaza si M rechaza.

Claramente, S acepta ω si y sólo si $\exists\pi$ tal que $\omega\#\pi$ es aceptada por M. Esto último equivale a decir que existe π tal que $\omega\#\pi \in D$, i.e. que $\omega \in L$. En resumen S reconoce L.

Supongamos ahora que L es reconocible. Sigue que existe una mT M tal que $L = L(M)$. Sea D el lenguaje formado por las palabras de la forma

$$\omega\#C_0\#C_1\#\ldots\#C_m\,,$$

donde cada C_i corresponde a una posible descripción instantánea de M. En particular C_0 y C_m son descripciones instantáneas de M al partir en la entrada ω y al aceptar respectivamente. Más aun, C_{i+1} es una configuración que M puede alcanzar a partir de C_i en una transición. Observar que $L = \{\omega : \exists\pi, \omega\#\pi \in D\}$. La conclusión deseada se obtiene una vez que se comprueba que D es efectivamente decidible (omitido).

Otra alternativa es considerar D como el lenguaje formado por las palabras de la forma $\omega\#\langle N_\omega\rangle$ donde $\langle N_\omega\rangle$ es la codificación de un entero no-negativo N_ω que representa el número de transiciones que requiere la mT M para aceptar ω. Es fácil ver que $L = \{\omega : \exists N_\omega \in \mathbb{N}, \omega\#\langle N_\omega\rangle \in D\}$. Por otro lado, veamos que D es decidible. En efecto, sea M' una mT que en la entrada $\omega\#\alpha$ verifica que α es una codificación en binario de un entero N (si no lo es la máquina para), simula M en la entrada ω durante N transiciones, acepta cuando M acepta y rechaza si M no acepta en N transiciones. Claramente M' es una mT que para independiente de la entrada y tal que $L(M') = D$, i.e. D es decidible.