Examen de Computabilidad y Complejidad

(CMC)

13 de junio de 2000

(I) Cuestiones (justifique formalmente las respuestas)

1. Sea L_1 un lenguaje recursivo y L_2 un lenguaje recursivamente enumerable.

   (a) ¿ Es $L_2 - L_1$ recursivamente enumerable ?

   (b) ¿ Es $L_1 \cap L_2$ recursivo ?

   (c) ¿ Es $L_1 \cap L_2$ recursivamente enumerable ?

   (1.5 ptos)

   Solución

   Procederemos a contestar cada una de las cuestiones por separado.

   (a) $L_2 - L_1$ es recursivamente enumerable. Obsérvese que $L_2 - L_1 = L_2 \cap \overline{L_1}$. Dado que L_1 es recursivo, entonces $\overline{L_1}$ también lo es y es, por lo tanto, recursivamente enumerable. Por otra parte, sabemos que la clase de los lenguajes recursivamente enumerables es cerrada bajo intersección y, en consecuencia, $L_2 - L_1$ es recursivamente enumerable.

   (b) $L_1 \cap L_2$ no es necesariamente recursivo. Tomemos, a modo de contraejemplo, L_2 como un lenguaje recursivamente enumerable no recursivo (por ejemplo el *lenguaje universal*) y tomemos $L_1 = \Sigma^*$ que es recursivo. En este caso $L_1 \cap L_2 = L_2$ que no es recursivo.

   (c) $L_1 \cap L_2$ sí es recursivamente enumerable. Dado que L_1 es recursivo, L_1 es recursivamente enumerable. Por otra parte, la clase de los lenguajes recursivamente enumerables es cerrada bajo intersección y, por lo tanto, $L_1 \cap L_2$ es recursivamente enumerable.

2. Sea P una operación entre palabras definida como $\mathrm{P}(x) = x^2 \ \forall x \in \Sigma^*$. Se extiende a lenguajes de la manera usual ($\mathrm{P}(L) = \{ \mathrm{P}(x) : x \in L\}$)

   (a) ¿ Es la familia de los lenguajes recursivamente enumerables cerrada bajo P ?

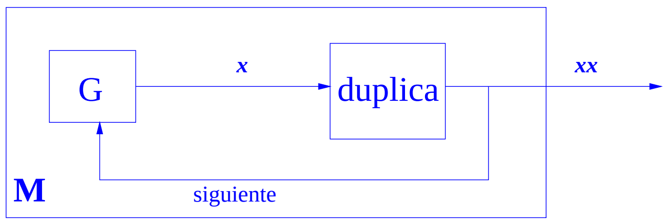
   (b) ¿ Es la familia de los lenguajes recursivos cerrada bajo P ?

   (2 ptos)

   Solución

   Procedemos a analizar cada una de las cuestiones por separado.

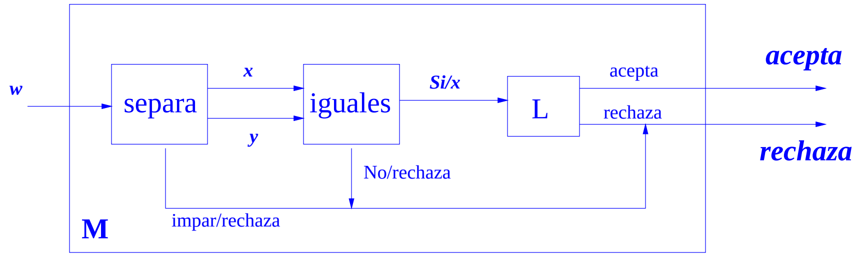
(a) La operación P es de cierre para la clase de los lenguajes recursivamente enumerables. Para ello proporcionaremos un esquema de máquina de Turing M tal que, dado el lenguaje L, genere $P(L)$. Contaremos con un módulo G que genera el lenguaje L y con un módulo *duplica* que, dada una cadena de entrada x produce como salida la cadena xx. Obsérvese que la existencia de G queda justificada al ser L un lenguaje recursivamente enumerable, mientras que el módulo *duplica* se fundamenta en una máquina de Turing multicinta que copia dos veces la cadena de entrada mediante un procedimiento trivial. A partir de ambos módulos, proporcionamos el siguiente esquema para M



El funcionamiento de M es sencillo: por cada cadena que se genera en G, el módulo *duplica* realiza la copia correspondiente y la emite como salida.

Puesto que hemos podido construir una máquina de Turing que genera $P(L)$, podemos concluir que $P(L)$ es recursivamente enumerable.

(b) La operación P también es de cierre para la clase de los lenguajes recursivos. En este caso, construiremos una máquina de Turing M que garantice la parada ante cualquier cadena de entrada y que acepte el lenguaje $P(L)$. Para construir la anterior máquina contaremos con un módulo *separa* que, dada una cadena w de entrada establece en primer lugar si es de longitud par y, en caso afirmativo, proporciona las dos mitades de la cadena con una longitud idéntica x e y. Contaremos también con un módulo *iguales* que, dadas dos cadenas de entrada, establece si son iguales o no. Por último, contaremos con un módulo L que, dada una cadena de entrada, establece si pertenece a L o no. Obsérvese que los módulos *separa* e *iguales* se fundamentan en máquinas de Turing multicintas con una operativa bastante simple, mientras que el módulo L existe por ser L un lenguaje recursivo. El esquema de M lo mostramos a continuación



El funcionamiento de M se explica a continuación. Inicialmente, para una cadena de entrada w, se somete ésta al módulo *separa*. Si la cadena es de longitud impar directamente se rechaza (ya que en $P(L)$ sólo pueden haber cadenas pares). Si la cadena w es par, entonces el módulo da como salida las dos mitades de w que consideramos las cadenas x e y. Las cadenas x e y se proporcionan como entrada al módulo *iguales* que establece si las dos cadenas son iguales o no. Si no son iguales, la cadena de entrada se rechaza (ya que en $P(L)$ las dos mitades deben ser iguales), mientras que, en caso contrario, se proporciona como salida cualquiera de las dos mitades x. Finalmente, la cadena x que ha proporcionado el módulo anterior, se analiza en el módulo L. Si el módulo L acepta entonces la cadena de entrada w se acepta ya que tiene dos mitades iguales y la mitad x es una cadena de L, mientras que en caso contrario se rechaza. Fácilmente se observa que la máquina M únicamente acepta aquellas cadenas de $P(L)$ y que para ante cualquier entrada. Por lo tanto, $P(L)$ es recursivo.

3. Se define sobre el alfabeto $\{a, b\}$ el lenguaje $L = \{xyz : |x| = |y| = |z| \wedge |x|_a = |y|_a = |z|_a\}$. ¿ Es L incontextual ?

(1.5 ptos)

Solución

El lenguaje L no es incontextual. Partiremos de L y, mediante operaciones de cierre para la clase de los lenguajes incontextuales, llegaremos a un lenguaje no incontextual. En primer lugar, tomemos la intersección $L \cap ab^*aab^*aab^*a = L_1 = \{ab^naab^naab^na \mid n \geq 0\}$. Tomemos ahora el homomorfismo g tal que $g(a) = b$, $g(b) = b$, $g(c) = b$ y $g(d) = a$. Se cumple que $g^{-1}(L_1) = L_2 = \{d\{a,b,c\}^n dd\{a,b,c\}^n dd\{a,b,c\}^n d \mid n \geq 0\}$. A continuación, formamos el lenguaje $L_3 = L_2 \cap da^*ddb^*ddc^*d = \{da^nddb^nddc^nd \mid n \geq 0\}$. Por último, definimos el homomorfismo h tal que $h(a) = a$, $h(b) = b$, $h(c) = c$ y $h(d) = \lambda$. Aplicando $h(L_3)$ obtenemos $\{a^nb^nc^n \mid n \geq 0\}$ que, como se ha visto en clase, no es incontextual. Por lo tanto, hemos partido de L y le hemos aplicado sucesivas operaciones de cierre (intersecciones con lenguajes regulares, homomorfismos y homomorfismos inversos) y hemos obtenido un lenguaje no incontextual. Como conclusión podemos afirmar que L no es incontextual.

(II) PROBLEMAS:

4. Se pide una obtener una gramática lo más simplificada posible, sin producciones vacías y sin reglas unitarias, que genere $L(G) - \{\lambda\}$, donde G está definida por las reglas:

$$\begin{array}{lll} S \to AB \mid SA & A \to aA \mid aBB \mid D \mid \lambda & B \to bA \mid DD \mid a \\ C \to bA \mid bBA & D \to aD \mid aE & E \to bD \end{array}$$

(1 pto)

Solución

Procedemos, en primer lugar, a simplificar la gramática G.

Eliminación de símbolos no generativos

Símbolos no generativos: $\{D, E\}$

Gramática sin símbolos no generativos

$S \to AB \mid SA$

$A \to aA \mid aBB \mid \lambda$

$B \to bA \mid a$

$C \to bA \mid bBA$

Eliminación de símbolos no alcanzables

Símbolos no alcanzables: $\{C\}$

Gramática sin símbolos no alcanzables

$S \to AB \mid SA$

$A \to aA \mid aBB \mid \lambda$

$B \to bA \mid a$

Eliminación de producciones vacías

Símbolos anulables: $\{A\}$

Gramática sin producciones vacías

$S \to AB \mid B \mid SA \mid S$

$A \to aA \mid a \mid aBB$

$B \to bA \mid b \mid a$

Eliminación de producciones unitarias

$\mathcal{C}(S) = \{S, B\}$ $\mathcal{C}(A) = \{A\}$ $\mathcal{C}(C) = \{C\}$

Gramática sin producciones unitarias

$S \to AB \mid bA \mid b \mid a \mid SA$

$A \to aA \mid a \mid aBB$

$B \to bA \mid b \mid a$

La anterior gramática ya está totalmente simplificada puesto que todos sus símbolos son útiles. Es, por lo tanto, la gramática que se nos pedía en el enunciado del problema.

5. Sea G la gramática definida por las reglas $S \to 0A1 \mid 1B0$; $A \to 0AS \mid 1$; $B \to 1B \mid \lambda$. Sea h un homomorfismo definido como $h(0) = 01$ y $h(1) = 0$. Se pide dar una gramática para el lenguaje $(L(G))^r h(L(G))$.

(2 ptos)

Solución

Calculamos, en primer lugar, una gramática que genere $(L(G))^r$

$S_r \to 1A_r0 \mid 0B_r1$

$A_r \to S_rA_r0 \mid 1$

$B_r \to B_r1 \mid \lambda$

A continuación una gramática que genera $h(L(G))$

$S_h \to 01A_h0 \mid 0B_h01;$

$A_h \to 01A_hS_h \mid 0;$

$B_h \to 0B_h \mid \lambda$

Por último, una gramática que genera el lenguaje $(L(G))^r h(L(G))$. Definimos S_c como el axioma de la gramática

$S_c \to S_rS_h$

$S_r \to 1A_r0 \mid 0B_r1$

$A_r \to S_rA_r0 \mid 1$

$B_r \to B_r1 \mid \lambda$

$S_h \to 01A_h0 \mid 0B_h01;$

$A_h \to 01A_hS_h \mid 0;$

$B_h \to 0B_h \mid \lambda$

6. Construir un módulo *Mathematica* que, dada una gramática independiente del contexto, devuelva la cantidad de reglas recursivas que contiene. (Una regla se dice recursiva si el antecedente aparece en el consecuente)

(2 ptos)

Solución

```
Solucion[G_List]:=Module[{ P, contador, k, j },
    P=G[[3]];
    contador=0;
    For[k=1, k≤Length[P], k++,
        For[ j=1, j≤Length[P[[k,2]]], j++,
            If[MemberQ[P[[k,2,j]],P[[k,1]]], contador++]
        ]
    ];
    Return[contador]
]
```