Examen de Computabilidad y Complejidad

(CMC)

11 de septiembre de 2003

(I) Cuestiones (justifique formalmente las respuestas)

1. ¿ Es incontextual el lenguaje $L = \{x \in \{a,b,c\}^* : |x|_a < |x|_b \wedge |x|_a < |x|_c\}$?

(1.5 ptos)

Solución

El lenguaje L no es incontextual. Lo demostraremos mediante el lema de bombeo. Tomemos n como la constante del lema y la cadena $z = a^n b^{n+1} c^{n+1}$ que pertenece a L y cumple las condiciones de partida del lema.

Factorizamos, de la forma habitual, $z = uvwxy$ y veremos las distintas posibilidades de localización de las subcadenas v y x.

(a) vx formado por símbolos a con $|vx| = k \geq 1$. En este caso tomamos el valor $i = 2$ y formamos la cadena $uv^2wx^2y = a^{n+k}b^{n+1}c^{n+1}$ que no pertenece a L ya que el número de símbolos a no es estrictamente menor que el de símbolos b y c.

(b) vx formado por símbolos b con $|vx| = k \geq 1$. En este caso tomamos el valor $i = 0$ y formamos la cadena $uv^0wx^0y = a^n b^{n+1-k} c^{n+1}$ que no pertenece a L ya que el número de símbolos a no es estrictamente menor que el de símbolos b.

(c) vx formado por símbolos c con $|vx| = k \geq 1$. En este caso tomamos el valor $i = 0$ y formamos la cadena $uv^0wx^0y = a^n b^{n+1} c^{n+1-k}$ que no pertenece a L ya que el número de símbolos a no es estrictamente menor que el de símbolos c.

(d) vx formado por símbolos a y b, con $|vx|_a = k$ y $|vx|_b = j$ donde $k, j \geq 1$. Tomamos el valor $i = 2$ y se forma la cadena $uvvwxxy$ que no pertenece a L ya que el número de símbolos a será igual a $n + k$ y el número de símbolos c igual a $n + 1$ por lo que el primero no es estrictamente menor que el segundo.

(e) vx formado por símbolos b y c, con $|vx|_b = k$ y $|vx|_c = j$ donde $k, j \geq 1$. Tomamos el valor $i = 0$ y se forma la cadena $a^n b^{n+1-k} c^{n+1-j}$ que no pertenece a L ya que el número de símbolos a no es estrictamente menor que el de símbolos b y c.

Debido a las condiciones primera y segunda del lema ya no se pueden plantear más casos y al haber demostrado, en todos los casos posibles, que el lema no se cumple en su tercera condición entonces podemos concluir que L no es incontextual.

2. Sea el alfabeto $\Sigma = \{a, b\}$ y se define la operación P sobre cadenas como $P(x, y) = 0^{2|x|}1^{|y|}$. La operación P se extiende sobre lenguajes como $P(L_1, L_2) = \{P(x, y) : x \in L_1, y \in L_2\}$. ¿ Es la clase de los lenguajes incontextuales cerrada bajo la operación P ?

(1 pto)

Solución

La operación P sí es de cierre para la clase de los lenguajes incontextuales. Para demostrarlo, expresaremos P como el resultado de la aplicación de operaciones de cierre para la citada clase. En primer lugar, definiremos el homomorfismo h tal que $h(a) = h(b) = 00$ y el homomorfismo g de forma que $g(a) = g(b) = 1$. Es fácil comprobar que $P(L_1, L_2) = h(L_1)g(L_2)$. Las operaciones de homorfismo y concatenación han sido demostradas de cierre para la clase de los lenguajes incontextuales. Dado que P es el resultado de la aplicación de las anteriores operaciones podemos concluir que P también es de cierre para la citada clase.

3. Sean L_1 y L_2 dos lenguajes definidos sobre el alfabeto Σ de forma que $L_1 \cap L_2 = \emptyset$. ¿ Son ciertas las siguientes afirmaciones ?

   (a) Si L_1 y L_2 son recursivos entonces $L_1 \cup L_2$ no es recursivo

   (b) Si L_1 es recursivo y L_2 no es recursivo entonces $L_1 \cup L_2$ no es recursivo

   (c) Si L_1 y L_2 no son recursivos entonces $L_1 \cup L_2$ no es recursivo

(1.5 ptos)

Solución

Nos pronunciaremos acerca de cada afirmación por separado.

(a) La afirmación es falsa. Dado que la unión es una operación de cierre para la clase de los lenguajes recursivos, queda garantizado que si L_1 y L_2 son recursivos entonces $L_1 \cup L_2$ también lo es.

(b) La afirmación es cierta. Realicemos una demostración por reducción al absurdo suponiendo que L_1 es recursivo, L_2 no es recursivo y $L_1 \cup L_2$ sí es recursivo. Puede observarse que al ser $L_1 \cap L_2 = \emptyset$ se cumple que $L_2 = (L_1 \cup L_2) - L_1$. La anterior expresión es equivalente a la igualdad $L_2 = (L_1 \cup L_2) \cap \overline{L_1}$. Dado que L_1 es recursivo, entonces $\overline{L_1}$ también lo es. Por otra parte, suponemos que $L_1 \cup L_2$ es recursivo y, en consecuencia, $(L_1 \cup L_2) \cap \overline{L_1} = L_2$ también lo es, lo cual no se puede cumplir ya que en el enunciado se afirma que L_2 no es recursivo. En conclusión, suponiendo que el enunciado es falso llegamos a una contradicción por lo que el enunciado es cierto.

(c) La afirmación es falsa. A modo de contraejemplo, tomemos un lenguaje arbitrario L_1 no recursivo definido sobre el alfabeto Σ y $L_2 = \overline{L_1}$. Obviamente, L_2 tampoco es recursivo ya que, en caso contrario, L_1 sería recursivo. Trivialmente, $L_1 \cup L_2 = L_1 \cup \overline{L_1} = \Sigma^*$ que es recursivo ya que se puede definir una máquina de Turing que acepte cualquier cadena de entrada y siempre pare.

4. Sea la operación P definida sobre lenguajes de Σ de la forma $P(L_1, L_2) = \{x \in \Sigma^* : x = yz, y \in L_1 \vee z \in L_2\}$. ¿ Es la clase de los lenguajes recursivamente enumerables cerrada bajo P ?

(1 pto)

Solución

La operación P sí es de cierre para la clase de los lenguajes recursivamente enumerables. Obsérvese que podemos expresar P de la siguiente forma, $P(L_1, L_2) = L_1\Sigma^* \cup \Sigma^* L_2$. El lenguaje Σ^* es recursivo (véase la cuestión anterior en su apartado (c)) y, por lo tanto, recursivamente enumerable. Por otra parte, es sabido que la clase de los lenguajes recursivamente enumerables es cerrada bajo operaciones de producto (concatenación) y unión. Por lo tanto, P es el resultado de aplicar operaciones de cierre para la clase de los lenguajes recursivamente enumerables y, en consecuencia, P es de cierre para la citada clase.

(II) PROBLEMAS:

5. Se pide construir un módulo *Mathematica* que, tomando como parámetro de entrada una gramática incontextual y dos símbolos auxiliares de la misma (en total tres parámetros), devuelva *True* si cada auxiliar de entrada aparece en algún consecuente de alguna producción del otro y *False* en caso contrario.

(2 ptos)

Solución

```
Solucion[G_List, A_, B_ ]:=Module[{ P, k, test, l, dA, dB },
    P=G[[3]];
    test=False;
    l=Cases[P,{A,_}];
    dA = l[[1,2]];
```

```
    k=1;
    While[ k ≤Length[dA] && !test,
        If[MemberQ[dA[[k]],B], test=True, k++]
    ];
    If[!test, Return[False]];
    test=False;
    l=Cases[P,{B,_}];
    dB = l[[1,2]];
    k=1;
    While[ k ≤Length[dB] && !test,
        If[MemberQ[dB[[k]],A], test=True, k++]
    ];
    Return[test]
]
```

6. Sea G la gramática con las reglas $S \to AB \mid 0$; $A \to 0BS \mid 1$ y $B \to S10 \mid \lambda$ y sea h el homomorfismo definido como $h(0) = 01$ y $h(1) = 0$. Obtenga una gramática incontextual para el lenguaje $h(L(G)) \cup (L(G))^r$

(1 pto)

Solución

En primer lugar, obtendremos una gramática para $h(L(G))$ definida por las siguientes reglas

$S_h \to A_h B_h \mid 01$

$A_h \to 01 B_h S_h \mid 0$

$B_h \to S_h 001 \mid \lambda$

A continuación, una gramática para $(L(G))^r$ que queda definida por las siguientes reglas

$S_r \to B_r A_r \mid 0$

$A_r \to S_r B_r 0 \mid 1$

$B_r \to 01 S_r \mid \lambda$

Por último, la gramática para $h(L(G)) \cup (L(G))^r$ queda definida por las siguientes reglas, donde S_u es el axioma

$S_u \to S_h \mid S_r$

$S_h \to A_h B_h \mid 01$

$A_h \to 01 B_h S_h \mid 0$

$B_h \to S_h 001 \mid \lambda$

$S_r \to B_r A_r \mid 0$

$A_r \to S_r B_r 0 \mid 1$

$B_r \to 01 S_r \mid \lambda$

7. Dada la gramática G definida por las siguientes producciones se pide obtener una gramática simplificada y en Forma Normal de Chomsky que genere $L(G) - \{\lambda\}$

$S \to CDC \mid DAD \mid 0S1S$
$A \to 0 \mid DD$
$B \to AS \mid 0A \mid \lambda$
$C \to 0C \mid 1BC$
$D \to 0C1 \mid AA \mid \lambda$

(2 ptos)

Solución

En primer lugar procedemos a simplificar la gramática.

Eliminación de símbolos no generativos

Símbolos no generativos: $\{C\}$

Gramática sin símbolos no generativos

$S \to DAD \mid 0S1S$
$A \to 0 \mid DD$
$B \to AS \mid 0A \mid \lambda$
$D \to AA \mid \lambda$

Eliminación de símbolos no alcanzables

Símbolos no alcanzables: $\{B\}$

Gramática sin símbolos no alcanzables

$S \to DAD \mid 0S1S$
$A \to 0 \mid DD$
$D \to AA \mid \lambda$

Eliminación de producciones vacías

Símbolos anulables: $\{S, A, D\}$

Gramática sin producciones vacías

$S \to DAD \mid DA \mid DD \mid AD \mid D \mid A \mid 0S1S \mid 01S \mid 0S1 \mid 01$
$A \to 0 \mid DD \mid D$
$D \to AA \mid A$

Eliminación de producciones unitarias

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

Gramática sin producciones unitarias

$S \to DAD \mid DA \mid DD \mid AD \mid AA \mid 0 \mid 0S1S \mid 01S \mid 0S1 \mid 01$
$A \to 0 \mid DD \mid AA$
$D \to AA \mid 0 \mid DD$

La gramática anterior ya está totalmente simplificada ya que todos sus símbolos son útiles (generativos y alcanzables).

Paso a Forma Normal de Chomsky

Sustitución de símbolos terminales

$S \to DAD \mid DA \mid DD \mid AD \mid AA \mid 0 \mid C_0SC_1S \mid C_0C_1S \mid C_0SC_1 \mid C_0C_1$
$A \to 0 \mid DD \mid AA$
$D \to AA \mid 0 \mid DD$
$C_0 \to 0$
$C_1 \to 1$

Factorización de las producciones y obtención de la gramática definitiva en FNC

$S \to DD_1 \mid DA \mid DD \mid AD \mid AA \mid 0 \mid C_0D_2 \mid C_0D_3 \mid C_0D_4 \mid C_0C_1$
$A \to 0 \mid DD \mid AA$
$D \to AA \mid 0 \mid DD$
$D_1 \to AD$
$D_2 \to SD_3$
$D_3 \to C_1S$
$D_4 \to SC_1$
$C_0 \to 0$
$C_1 \to 1$