Examen de Computabilidad y Complejidad

(CMC)

13 de septiembre de 2002

(I) Cuestiones (justifique formalmente las respuestas)

1. Sea el lenguaje $L = \{a^m b^{max(m,n)} c^n : n, m > 0\}$. ¿ Es L incontextual ?

(1 pto)

Solución

El lenguaje L no es incontextual. Lo demostraremos mediante el lema de bombeo. Tomemos n como la constante del lema y la cadena $z = a^n b^n c^n$ 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 $uv^2wx^2y = a^{n+k}b^n c^n$ que no pertenece a L ya que el número de símbolos b no coincide con el máximo de símbolos a 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 $uv^0wx^0y = a^n b^{n-k} c^n$ que no pertenece a L ya que el número de símbolos b no coincide con el máximo de símbolos a y c.

(c) vx formado por símbolos c con $|vx| = k \geq 1$. En este caso tomamos el valor $i = 2$ y formamos la cadena $uv^2wx^2y = a^n b^n c^{n+k}$ que no pertenece a L ya que el número de símbolos b no coincide con el máximo de símbolos a y 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 $i = 0$ y se forma la cadena $a^{n-k} b^{n-j} c^n$ que no pertenece a L ya que el número de símbolos b no coincide con el máximo de símbolos a y c.

(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 $a^n b^{n-k} c^{n-j}$ que no pertenece a L ya que el número de símbolos b no coincide con el máximo de símbolos a y c.

Debido a las condiciones 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. Sean L_1 y L_2 dos lenguajes definidos sobre el alfabeto Σ. Se define la operación sobre lenguajes $P(L_1, L_2) = \{x \in \Sigma^* : x \notin L_1 \wedge x \notin L_2\}$.

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

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

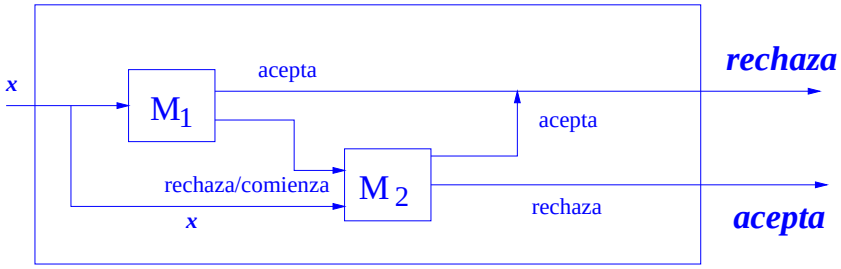
(2 ptos)

Solución

Analizaremos cada apartado por separado.

(a) La clase de los lenguajes recursivamente enumerables no es cerrada bajo P. Obsérvese que $P(L_1, L_2) = \overline{L_1} \cap \overline{L_2}$. Podemos tomar $L_2 = \emptyset$ (el lenguaje vacío) que, trivialmente, es un lenguaje recursivamente enumerable. En este caso $P(L_1, \emptyset) = \overline{L_1} \cap \overline{\emptyset} = \overline{L_1} \cap \Sigma^* = \overline{L_1}$. Es decir, la operación P obtiene, como caso particular, el complementario de un lenguaje. Como se ha visto en clase, la complementación no es una operación de cierre para la clase de los lenguajes recursivamente enumerables y, en consecuencia, P tampoco puede serlo.

(b) La clase de los lenguajes recursivos sí es cerrada bajo la operación P. Tomemos la expresión del anterior apartado para P como $P(L_1, L_2) = \overline{L_1} \cap \overline{L_2}$. Podemos obtener el siguiente esquema de una máquina de Turing que acepta las cadenas de $P(L_1, L_2)$ y siempre para. Para ello, contaremos con dos módulos M_1 y M_2 que aceptan respectivamente a L_1 y L_2 y siempre paran. El esquema para P se muestra a continuación



El funcionamiento del anterior esquema es sencillo. Una cadena de entrada se acepta si y sólo si se rechaza secuencialmente por M_1 y M_2, es decir si pertenece a $P(L_1, L_2)$. La condición de parada se garantiza siempre al quedar establecida en M_1 y M_2.

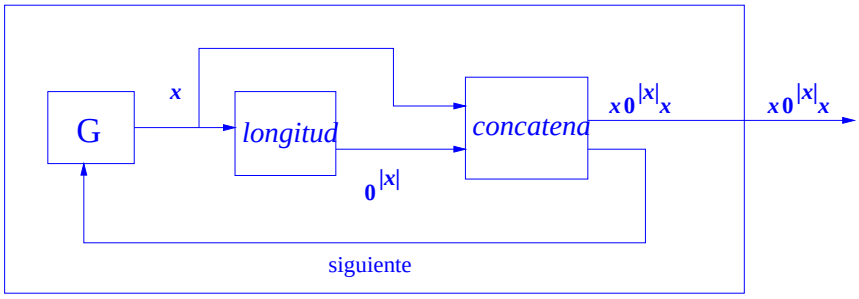
3. Sea la operación P definida sobre cadenas del alfabeto $\{0, 1\}$ como $P(x) = \{x0^{|x|}x : x \in (0 + 1)^*\}$. La operación se extiende sobre lenguajes de la forma habitual. ¿ Es la clase de los lenguajes recursivamente enumerables cerrada bajo P ? ¿ y la clase de los lenguajes recursivos ?

(2 ptos)

Solución

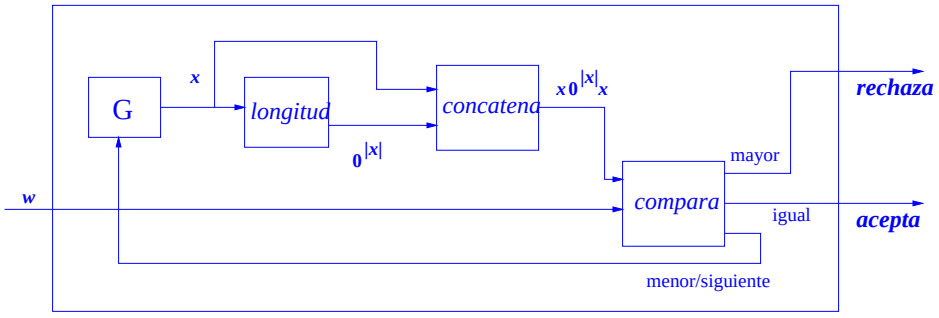
Analizaremos cada pregunta por separado.

(a) La clase de los lenguajes recursivamente enumerables es cerrada bajo P. Para demostrarlo, obtendremos una máquina de Turing que genere el lenguaje $P(L)$ siendo L un lenguaje recursivamente enumerable. Obsérvese que podemos contar con un módulo G tal que $G(M) = L$. El esquema que se propone se muestra a continuación



El funcionamiento del anterior esquema es como sigue: el módulo *longitud* calcula la longitud de la cadena de entrada y la expresa en código de ceros de la forma habitual en la que se calculan las funciones computables con alfabeto de codificación de salida unario. Esta función claramente la puede realizar una máquina de Turing que sustituya cada símbolo de entrada por un cero. El módulo *concatena* realiza la concatenación de dos cadenas de entrada x e y de la forma xyx. Este módulo se puede realizar mediante una máquina de Turing multicinta con rutinas de copia. El esquema propuesto genera el lenguaje $P(L)$. En consecuencia, este es un lenguaje recursivamente enumerable y, por lo tanto, P es de cierre para la clase bajo estudio.

(b) La clase de los lenguajes recursivos es cerrada bajo P. Propondremos una máquina de Turing que, dado un lenguaje recursivo L, acepte las cadenas de $P(L)$ y siempre pare. Para ello contaremos con un módulo G que genera L en orden canónico, los módulos *longitud* y *concatena* del apartado anterior y el módulo *compara* que establece la comparación de dos cadenas. El esquema propuesto se muestra a continuación



El funcionamiento del esquema anterior es como sigue: El módulo G genera las cadenas de L en orden canónico, los módulos *longitud* y *concatena* obtienen, a partir de una cadena x la cadena $x0^{|x|}x$. El módulo *compara* establece la comparación de la cadena de entrada w con la cadena $x0^{|x|}x$. Si las cadenas son iguales, entonces la cadena de entrada se acepta (ya que pertenece a $P(L)$). Si la cadena $x0^{|x|}x$ es menor en longitud que w entonces se solicita al módulo G la siguiente cadena de L. Si la cadena $x0^{|x|}x$ es mayor en longitud que la cadena w entonces esta se rechaza, ya que G no proporcionará ninguna cadena que, tras la transformación P, sea igual o menor en longitud que w. Puesto que G proporciona las cadenas en orden canónico y, por lo tanto, en longitud creciente, la condición de parada queda siempre garantizada.

Obsérvese que el esquema anterior pudiera no garantizar la parada en el caso de

que L sea finito. En este caso, $P(L)$ también es finito y, por lo tanto, recursivo.

(II) PROBLEMAS:

4. Se pide construir un módulo *Mathematica* que, tomando como parámetro de entrada una gramática incontextual, devuelva $True$ si la gramática tiene alguna producción en cuya parte derecha aparezcan todos los símbolos terminales de la gramática y $False$ en caso contrario.

(2 ptos)

Solución

```
Solucion[G_List]:=Module[{ P,Sigma,k, test,j },
    P=G[[3]];
    Sigma=G[[2]];
    test=False;
    For[k=1, k≤Length[P], k++,
        For[ j=1, j≤Length[P[[k,2]]], j++,
            If[Complement[Sigma,P[[k,2,j]]]=={}, test=True]
        ]
    ];
    Return[test]
]
```

5. Sea L_1 el lenguaje generado por la gramática con las reglas $S \to aSbScS \mid \lambda$. Sea el lenguaje L_2 el generado por la gramática con las reglas $S \to SA \mid AA$ y $A \to aAAb \mid cASa \mid ab$. Se define la sustitución f como $f(a) = L_1^*$, $f(b) = L_2^r$ y $f(c) = L_1 \cup L_2$. Obtenga una gramática incontextual para el lenguaje $f(L_1L_1)$.

(1 pto)

Solución

En primer lugar obtendremos una gramática para $f(a) = L_1^*$. La gramática queda definida por las siguientes reglas

$S_a \to S_1S_a \mid \lambda$

$S_1 \to aS_1bS_1cS_1 \mid \lambda$

La gramática para $f(b) = L_2^r$ queda definida por las siguientes reglas

$S_b \to A_bS_b \mid A_bA_b$

$A_b \to bA_bA_ba \mid aS_bA_cc \mid ba$

La gramática para $f(c) = L_1 \cup L_2$ queda definida por las siguientes reglas

$S_c \to S_2 \mid S_3$

$S_2 \to aS_2bS_2cS_2 \mid \lambda$

$S_3 \to S_3A_1 \mid A_1A_1$

$A_1 \to aA_1A_1b \mid cA_1S_3a \mid ab.$

Para el lenguaje L_1L_1 obtenemos la siguiente gramática

$S_4 \to SS$

$S \to aSbScS \mid \lambda$

Por último, aplicamos la sustitución f sobre la anterior gramática, obteniendo una gramática para el lenguaje $f(L_1L_1)$ que es la gramática que se pedía en el enunciado. La gramática queda definida por las siguientes reglas, siendo S_4 el axioma de la misma

$S_4 \to SS$

$S \to S_aSS_bSS_cS \mid \lambda$

$S_a \to S_1S_a \mid \lambda$

$S_1 \to aS_1bS_1cS_1 \mid \lambda$

$S_b \to A_bS_b \mid A_bA_b$

$A_b \to bA_bA_ba \mid aS_bA_cc \mid ba$

$S_c \to S_2 \mid S_3$

$S_2 \to aS_2bS_2cS_2 \mid \lambda$

$S_3 \to S_3A_1 \mid A_1A_1$

$A_1 \to aA_1A_1b \mid cA_1S_3a \mid ab.$

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

$S \to aBB \mid B \mid bC \mid CBa \quad C \to Sa \mid aS \mid bCC \mid BB$
$A \to aBB \mid aAA \mid a \quad D \to aDD \mid bE \mid Ea \mid A$
$B \to bS \mid aBB \mid \lambda \quad E \to bDb \mid Db \mid a$

(2 ptos)

Solución

En primer lugar procedemos a simplificar la gramática.

Eliminación de símbolos no generativos

Símbolos no generativos: $\{\}$

Gramática sin símbolos no generativos

$S \to aBB \mid B \mid bC \mid CBa \quad C \to Sa \mid aS \mid bCC \mid BB$
$A \to aBB \mid aAA \mid a \quad D \to aDD \mid bE \mid Ea \mid A$
$B \to bS \mid aBB \mid \lambda \quad E \to bDb \mid Db \mid a$

Eliminación de símbolos no alcanzables

Símbolos no alcanzables: $\{A, D, E\}$

Gramática sin símbolos no alcanzables

$S \to aBB \mid B \mid bC \mid CBa$
$B \to bS \mid aBB \mid \lambda$
$C \to Sa \mid aS \mid bCC \mid BB$

Eliminación de producciones vacías

Símbolos anulables: $\{S, B, C\}$

Gramática sin producciones vacías

$S \to aBB \mid aB \mid a \mid B \mid bC \mid b \mid CBa \mid Ba \mid Ca$
$B \to bS \mid b \mid aBB \mid aB \mid a$
$C \to Sa \mid a \mid aS \mid bCC \mid bC \mid b \mid BB \mid B$

Eliminación de producciones unitarias

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

Gramática sin producciones unitarias

$S \to aBB \mid aB \mid a \mid bS \mid bC \mid b \mid CBa \mid Ba \mid Ca$
$B \to bS \mid b \mid aBB \mid aB \mid a$
$C \to Sa \mid a \mid aS \mid bCC \mid bC \mid b \mid BB \mid bS \mid aBB \mid aB$

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

$S \to C_a BB \mid C_a B \mid a \mid C_b S \mid C_b C \mid b \mid CBC_a \mid BC_a \mid CC_a$
$B \to C_b S \mid b \mid C_a BB \mid C_a B \mid a$
$C \to SC_a \mid a \mid C_a S \mid C_b CC \mid C_b C \mid b \mid BB \mid C_b S \mid C_a BB \mid C_a B$
$C_a \to a$
$C_b \to b$

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

$S \to C_a D_1 \mid C_a B \mid a \mid C_b S \mid C_b C \mid b \mid CD_2 \mid BC_a \mid CC_a$
$B \to C_b S \mid b \mid C_a D_1 \mid C_a B \mid a$
$C \to SC_a \mid a \mid C_a S \mid C_b D_3 \mid C_b C \mid b \mid BB \mid C_b S \mid C_a D_1 \mid C_a B$
$D_1 \to BB$
$D_2 \to BC_a$
$D_3 \to CC$
$C_a \to a$
$C_b \to b$