Examen de Computabilidad y Complejidad

(CMC)

12 de septiembre de 2000

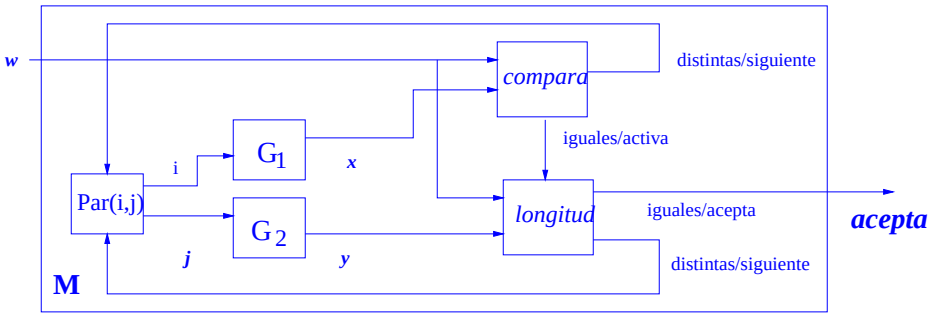
(I) Cuestiones (justifique formalmente las respuestas)

1. Sean L_1 y L_2 dos lenguajes y se define $L_1 \& L_2 = \{x : x \in L_1 \wedge \exists y \in L_2, |x| = |y|\}$. Pronúnciese acerca de la veracidad o falsedad de la siguiente afirmación: *Si L_1 y L_2 son lenguajes recursivamente enumerables entonces $L_1 \& L_2$ también lo es.*

(2 ptos)

Solución

La afirmación es cierta. Para ello, veremos que, dados dos lenguajes recursivamente enumerables L_1 y L_2, podemos construir una máquina de Turing que acepte $L_1 \& L_2$. Tomemos dos máquinas de Turing G_1 y G_2 que generan respectivamente a L_1 y L_2. A partir de ellas y utilizando otros módulos construiremos la máquina M que acepta $L_1 \& L_2$ y cuyo esquema se muestra a continuación



El funcionamiento de la máquina M es como sigue: $Par(i, j)$ es un generador de pares de enteros tal y como hemos visto en clase, *compara* es un módulo que, dadas dos cadenas de entrada, establece si son o no iguales (este módulo se puede desarrollar fácilmente utilizando una máquina con dos cintas de trabajo que explora simultáneamente las dos cadenas) y *longitud* es un módulo que, dadas dos cadenas de entrada, establece si tienen la misma longitud (de nuevo, es fácil comprobar que este módulo se puede desarrollar con una máquina con dos cintas similar a la del módulo *compara*). Inicialmente se genera un par de enteros (i, j) de forma que se obtenga la cadena i-ésima del lenguaje L_1 a partir del generador G_1 y la cadena j-ésima del lenguaje L_2 a partir de G_2. A continuación se comparan en el módulo *compara* la cadena x de L_1 con la cadena w de entrada. Si son iguales se activa el módulo *longitud* que establece si la cadena y de L_2 tiene la misma longitud que la cadena w (que por otra parte coincidirá con x). Si se supera el módulo *longitud* entonces se

acepta la cadena de entrada ya que pertenece a L_1 y existe una cadena de L_2 de igual longitud. En el caso de que no se superen alguna de las dos comprobaciones entonces se genera un nuevo par de enteros. Obsérvese que para cualquier cadena de $L_1 \& L_2$ se cumplirá que existe un par (i, j) que satisface las comprobaciones establecidas y viceversa. Dado que la máquina M acepta $L_1 \& L_2$, entonces podemos concluir que es recursivamente enumerable.

2. Sea $L \subseteq \{a, b\}^*$ un lenguaje y se define la operación $\phi(L) = \{awxa : aw, xa \in L\}$. ¿ Es ϕ una operación de cierre para la clase de los lenguajes incontextuales ?

(1.5 ptos)

Solución

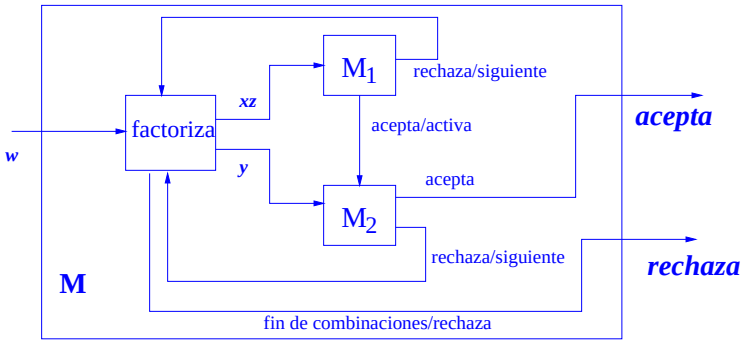
La operación ϕ es de cierre para la clase de los lenguajes incontextuales. Obsérvese que podemos expresar $\phi(L)$ como $\phi(L) = (a(a+b)^* \cap L)((a+b)^* a \cap L)$. En la expresión anterior se hace uso de la intersección entre lenguajes incontextuales y regulares y de la concatenación entre lenguajes incontextuales. Puesto que ambas operaciones son de cierre para la clase de los lenguajes incontextuales, entonces ϕ también lo es.

3. Sean L_1 y L_2 dos lenguajes y se define $L_1 \oplus L_2 = \{xyz : xz \in L_1 \wedge y \in L_2\}$. ¿ Es $\oplus$ de cierre para la clase de los lenguajes recursivos ?

(1.5 ptos)

Solución

La operación $\oplus$ es de cierre para la clase de los lenguajes recursivos. Para demostrarlo, dados L_1 y L_2 recursivos, construiremos una máquina M que acepte $L_1 \oplus L_2$ y que pare ante cualquier entrada. Tomemos dos máquinas de Turing M_1 y M_2 que aceptan respectivamente a L_1 y L_2 y que siempre paran. A partir de ellas, junto con otros módulos, construiremos una máquina de Turing M que acepta $L_1 \oplus L_2$ y que siempre para. El esquema de M se muestra a continuación.



El funcionamiento de M es como sigue: Contamos con el módulo $factoriza$ que, dada una cadena de entrada w forma todas las combinaciones posibles para subdividir ésta en xyz. Internamente, el módulo $factoriza$ puede desarrollarse a partir de una máquina multicinta que cuenta con un generador de pares (i, j), donde $0 \leq i, j \leq |w|$. El valor i marca el inicio de la subcadena y, mientras que el valor j indica su longitud. Si j toma un valor cero entonces se interpreta que $y = \lambda$, mientras que si $i + j > |w|$ entonces se rechaza esa combinación. Fácilmente se puede comprobar que a partir del generador de pares establecido se calculan todas las combinaciones que cumplan que $w = xyz$. El número de combinaciones posibles es finito ya que queda acotado por la longitud de w. Al utilizar una máquina multicinta, se puede copiar en una cinta la cadena xz y en otra cinta distinta y que es la salida que emite el módulo $factoriza$.

Inicialmente la cadena de entrada w se factoriza en el módulo anteriormente explicado. Se forman a partir de w las cadenas xz e y que se analizan respectivamente en M_1 y M_2 de forma secuencial. Si ambas máquinas aceptan, entonces se acepta la cadena de entrada w. Si alguna de las dos máquinas rechaza, entonces se procede a buscar la siguiente combinación. Si se agotan todas las combinaciones, entonces se rechaza la cadena de entrada. Así, la máquina M sólo acepta aquellas cadenas de entrada w tales que pertenecen a $L_1 \oplus L_2$. Dado que M siempre para, podemos concluir que $L_1 \oplus L_2$ es recursivo.

(II) PROBLEMAS:

4. Escriba un módulo *Mathematica* que, tomando una gramática incontextual como parámetro de entrada, devuelva *True* si en alguna de las producciones de la gramática aparecen dos símbolos auxiliares consecutivos en el consecuente y *False* en caso contrario.

(2 ptos)

Solución

```
Solucion[G_List]:=Module[{ P, aux, test, i, dchas, j, k },
    P=G[[3]];
    aux=G[[1]];
    test=False;
    i=1;
    While[ i ≤ Length[[P]] && !test,
        dchas=P[[i,2]];
        j=1;
        While[j ≤ Length[dchas] && !test,
            For[k=1, k < Length[dchas[[j]]], k++,
                If[MemberQ[aux,dchas[[j,k]]] && MemberQ[aux,dchas[[j,k+1]]],
                    test=True ];
            ];
            j++
        ];
        i++
    ];
    Return[test];
]
```

5. Sea G la gramática definida por las producciones $S \to 0AB \mid 10$; $A \to 0S \mid 1B$; $B \to S1 \mid 0 \mid \lambda$. Se pide una gramática incontextual para el lenguaje $(L(G)(L(G))^r)^*$.

(1.5 ptos)

Solución

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

$S_r \to B_r A_r 0 \mid 01$

$A_r \to S_r 0 \mid B_r 1$

$B_r \to 1S_r \mid 0 \mid \lambda$

A continuación una gramática para $L(G)(L(G))^r$

$S_c \to SS_r$

$S \to 0AB \mid 10$

$A \to 0S \mid 1B$

$B \to S1 \mid 0 \mid \lambda$

$S_r \to B_r A_r 0 \mid 01$

$A_r \to S_r 0 \mid B_r 1$

$B_r \to 1S_r \mid 0 \mid \lambda$

Por último la gramática que se solicita en el enunciado y que genera $(L(G)(L(G))^r)^*$.
El axioma de la gramática es S_*.

$S_* \to S_c S_* \mid \lambda$

$S_c \to SS_r$

$S \to 0AB \mid 10$

$A \to 0S \mid 1B$

$B \to S1 \mid 0 \mid \lambda$

$S_r \to B_r A_r 0 \mid 01$

$A_r \to S_r 0 \mid B_r 1$

$B_r \to 1S_r \mid 0 \mid \lambda$

6. Dar una gramática simplificada en forma normal de Chomsky equivalente a la gramática definida por las producciones:

$S \to AB \mid 0$

$A \to BBA \mid CDCD \mid \lambda$

$B \to SS \mid 01AA \mid 0C$

$C \to DC \mid CD$

$D \to SS \mid \lambda$

(1.5 ptos)

Solución

Procederemos, en primer lugar, a simplificar la gramática del enunciado

Eliminación de símbolos no generativos

Símbolos no generativos: $\{C\}$

Gramática sin símbolos no generativos

$S \to AB \mid 0$

$A \to BBA \mid \lambda$

$B \to SS \mid 01AA$

$D \to SS \mid \lambda$

Eliminación de símbolos no alcanzables

Símbolos no alcanzables: $\{D\}$

Gramática sin símbolos no alcanzables

$S \to AB \mid 0$

$A \to BBA \mid \lambda$

$B \to SS \mid 01AA$

Eliminación de producciones vacías

Símbolos anulables: $\{A\}$

Gramática sin producciones vacías

$S \to AB \mid B \mid 0$

$A \to BBA \mid BB$

$B \to SS \mid 01AA \mid 01A \mid 01$

Eliminación de producciones unitarias

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

Gramática sin producciones unitarias

$S \to AB \mid SS \mid 01AA \mid 01A \mid 01 \mid 0$

$A \to BBA \mid BB$

$B \to SS \mid 01AA \mid 01A \mid 01$

La gramática anterior ya está completamente simplificada puesto que todos sus símbolos son útiles.

Paso a Forma Normal de Chomsky

Sustitución de símbolos terminales

$S \to AB \mid SS \mid C_0C_1AA \mid C_0C_1A \mid C_0C_1 \mid 0$

$A \to BBA \mid BB$

$B \to SS \mid C_0C_1AA \mid C_0C_1A \mid C_0C_1$

$C_0 \to 0$

$C_1 \to 1$

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

$S \to AB \mid SS \mid C_0D_1 \mid C_0D_3 \mid C_0C_1 \mid 0$

$D_1 \to C_1D_2$

$D_2 \to AA$

$D_3 \to C_1A$

$A \to BD_4 \mid BB$

$D_4 \to BA$

$B \to SS \mid C_0D_1 \mid C_0D_3 \mid C_0C_1$

$C_0 \to 0$

$C_1 \to 1$