Septiembre 2017

viernes, 6 de enero de 2023 20:31

EJERCICIO 1 (1,5 puntos)

(a) ¿Qué es una Autómata de Pila?

(b) ¿Qué diferencia hay entre un Automata de Pila Determinista e Indeterminista?

(c) ¿Tienen la misma capacidad? Razone la respuesta.

a) Un autómata de pila es una sextupla $(\Sigma, \Gamma, \Delta, Q, q_0, F)$ donde:

Σ: es el alfabeto de la cadena de entrada + β.

Γ: es el alfabeto de la pila + γ (pila vacía)

Δ: es el conjunto de transiciones

Q: es el conjunto de estados

q_0: es el estado inicial

F: es el conjunto de estados finales.

b) Un autómata de pila determinista es capaz de realizar una sola transición en cada estado, mientras que uno no determinista puede realizar varias transiciones sobre un mismo estado

c) Mientras que el APD tan solo tiene una pila en todo momento, el APND tiene una pila por cada estado no determinista que tenga. No podemos simular varias pilas en una sola por lo que no podemos hacer un autómata de pila determinista equivalente y, en consecuencia, no pueden tener la misma capacidad de cómputo.

EJERCICIO 2 (1,5 puntos)

Considere la siguiente gramática libre de contexto, expresada en Forma Normal de Chomsky.

S → B M	M → O B	Q → L S
S → T M	M → O T	T → **term**
S → Q R	N → A B	A → **and**
S → **term**	N → A T	O → **or**
M → N M	N → O B	L → **lparen**
M → A B	N → O T	R → **rparen**
M → A T	B → Q R	

Verifique que la cadena "**lparen term or term rparen and term**" pertenece al lenguaje definido por la gramática por medio del algoritmo de Cocke-Younger-Kasami.

(	term	or	term	)	and	term
L	Q	—	Q	S,B	—	S*
	T, S	—	S	—	—	—
		O	M, N	—	—	—
			T, S	—	—	—
				R	—	—
					A	M, N
						T, S

la cadena pertenece a la gramática porque podemos deducirla a partir del símbolo inicial.

EJERCICIO 4 (1.5 puntos)

Sea E_{TM} el lenguaje formado por las cadenas <*M*> tales que *M* es la codificación de una máquina de Turing que no reconoce ninguna entrada, es decir, cuyo lenguaje es el lenguaje vacío. Demuestre que el lenguaje E_{TM} es indecidible.

DEMOSTRACIÓN POR REDUCCIÓN:

Dada una cadena ω y la máquina M podemos construir otra máquina M_1 tal que:

$$M_1(x) = \begin{cases} \text{rechace} & \text{si } x \neq \omega \\ \text{ejecute } M(\omega) \text{ y acepte si } M \text{ acepta} & \text{si } x = \omega \end{cases}$$

Suponemos E_{TM} decidible, por lo que existirá una máquina R que lo decida. Con M_1 y R podemos construir otra máquina S tal que:

1º Construye M_1

2º Ejecuta $R(\langle M_1, \omega \rangle)$

3º Si R acepta, acepta

4º si R rechaza, rechaza.

La máquina S resuelve el problema A_{TM} que es indecidible por lo tanto, S no puede existir. S no puede existir porque usa la máquina R que tampoco puede ser construida, por tanto, el lenguaje no puede ser decidido.

EJERCICIO 5 (2 puntos)

Considere el modelo de computación de las funciones recursivas. Asuma que las siguientes funciones ya han demostrado ser recursivas primitivas: Suma(x,y), Producto(x,y), Decremento(x), RestaAcotada(x,y), Signo(x), SignoNegado(x), Min(x,y), Max(x,y), And(x,y), Or(x,y), Not(x), Igual(x,y), Mayor(x,y), Menor(x,y), MayorOIgual(x,y), MenorOIgual(x,y).

Demuestre que la función *Div*(x,y), que calcula la división entera, es primitiva recursiva.

$$Div(x, y) = x / y$$

X	y	Div(x,y)
0	3	0
1	3	0
2	3	0)+1
3	3	1
4	3	1
5	3	1)+1
6	3	2
7	3	2
8	3	2)+1
9	3	3
⋮	⋮	⋮

CASO BASE

$Div(0, y) = Z_1$

$2 = S(S(Z))$

$Eq(Z_x, Z_x)$

CASO GENERAL

$Div(S(x), y) = g(Div(x,y), x, y) =$

$= g(U_1^3, U_2^3, U_3^3)$

Podemos ver como el resultado de la función sigue un patrón de comportamiento que podemos definir mediante la función:

$$Div(S(x), y) = \begin{cases} Div(x,y) & \text{si } Producto(S(Div(x,y)), y) < S(x) \\ S(Div(x,y)) & \text{en otro caso} \end{cases}$$

Por lo tanto podemos definir la función recursiva como

$Div(x,y) = R(Z_1, C(If, C(Menor, C(Producto, C(S, U_1^3), U_3^3), C(C(S, U_2^3))),$
$, U_1^3,$
$, C(S, U_1^3)))).$

EJERCICIO 6 (1.5 puntos)

¿Qué es un problema NP-completo? Enuncie el Teorema de Cook y Levin y describa brevemente su demostración.

Un problema NP-completo es un problema que pertenece a la clase NP y que, si tuviera solución en tiempo polinomial, permitiría demostrar que todos los problemas NP tienen solución en tiempo polinomial.

TEOREMA - COOK Y LEVIN

los problemas NP tienen solución en tiempo polinomial.

TEOREMA - COOK Y LEVIN

Sea SAT el lenguaje formado por las fórmulas booleanas satisfactibles, es decir, que existan valores que hacen cierta a la fórmula.

El teorema de Cook-Levin establece que el lenguaje SAT pertenece a la clase P sí y solo si $P = NP$. es decir, SAT es un problema NP-completo y, por tanto existe un verificador que verifica SAT en tiempo polinomial. (tabla de verdad).

El teorema plantea un tablero de configuración en el que si se alcanza un estado de aceptación significaría que la máquina acepta w.

La idea de Cook y Levin era convertir el tablero en un conjunto de fórmulas booleanas de orden $O(n^{2k})$, por lo tanto se demostraría que cualquier poblema NP puede reducirse en tiempo polinomial a un problema de satisfactibilidad (SAT) y, por lo tanto, SAT sería NP-completo.