# Examen septiembre 2019

jueves, 22 de diciembre de 2022 8:18

**EJERCICIO 1 (1 punto)**

Enuncie y demestre el Lema de Bombeo para Autómatas Finitos.

El lema de bombeo para autómatas finitos dice que sea $L$ un lenguaje regular sobre un alfabeto $\Sigma$ reconocido por un AFD con $m$ estados. Si $\omega \in L$ y $|\omega| \geq m$ entonces existen $r, s, t$ con $|s| \geq 1$ y $|rs| \leq m$ tal que $\omega = rst$ y para todo $n \geq 0$ $rs^n t \in L$.

Se demuestra utilizando el principio del palomar. Como hay $m$ estados y la cadena es de longitud mayor o igual que $m$, entonces existe al menos un estado repetido.

**EJERCICIO 2 (1.5 puntos)**

Considere la siguiente gramática libre de contexto, expresada en Forma Normal de Chomsky, donde L es el símbolo inicial.

| | | | |
|---|---|---|---|
| L → NM F | MT → SM T | T → **producto** | RP → **rpar** |
| L → LP LC | MT → SM L | F → ML T | NM → **num** |
| L → T MT | T → NM F | LC → L RP | SM → **plus** |
| L → **producto** | T → LP LC | LP → **lpar** | ML → **mul** |

Verifique que la cadena "**num mul lpar producto plus num mul producto rpar**" pertenece al lenguaje definido por la gramática por medio del algoritmo de Cocke-Younger-Kasami.

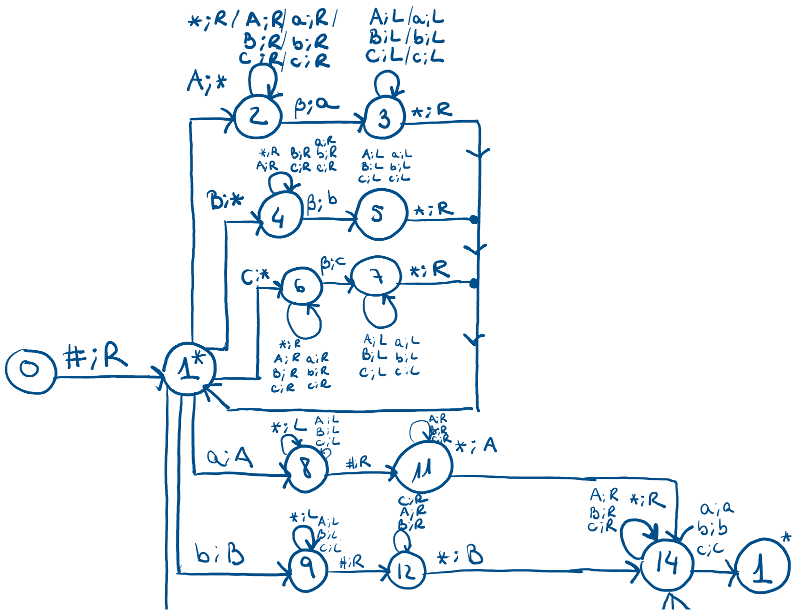
| num | mul | lpar | producto | plus | num | mul | producto | rpar |
|---|---|---|---|---|---|---|---|---|
| NM | — | — | — | — | — | — | — | L* |
| | ML | — | — | — | — | — | — | F |
| | | LP | — | — | — | — | — | T |
| | | | L, T | — | — | — | L | LC |
| | | | | SM | — | — | MT | — |
| | | | | | NM | — | T, L | LC |
| | | | | | | ML | F | — |
| | | | | | | | T, L | LC |
| | | | | | | | | RP |

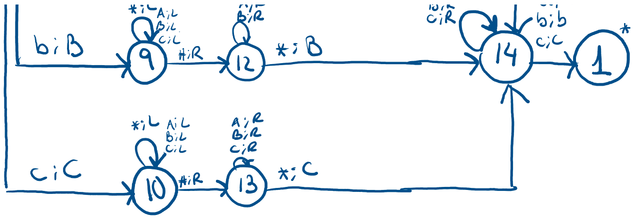
Como podemos generar la cadena a partir del símbolo inicial de la gramática, la cadena pertenece al lenguaje.

**EJERCICIO 3 (2 puntos)**

Diseñar una Máquina de Turing que haga una copia de una cadena de símbolos {A,B,C}. Por ejemplo, para la entrada "#AABCA~~bbbb~~..." devuelve en la cinta "#AABCAAABCA~~bb~~..".

NOTA: Tenga en cuenta que no existe ningún espacio entre la cadena inicial y la copia.






Primero leemos el delimitador de la cadena # en el estado 0.

Los estados 2, 3, 4, 5, 6 y 7 marcan la cadena de entrada y la copian en minúsculas a la derecha de la cinta. En la cinta quedaría lo siguiente después de leer la cadena completa:

"# * * * * * a a b c a β β"

Los estados 8, 9, 10, 11, 12, 13 y 14 leen la copia generada anteriormente y la transforman a mayúscula a la vez que la vuelven a copiar al comienzo de la cinta.

**EJERCICIO 4 (1.5 puntos)**

Sea $HALT_{TM}$ el lenguaje formado por las cadenas <*M*,*w*> tales que *M* es la codificación de una máquina de Turing y *w* es una cadena que hace que dicha máquina termine (ya sea aceptando o rechazando). Demuestre que el lenguaje $HALT_{TM}$ es indecidible.

Demostración por reducción

Suponemos que el lenguaje $HALT_{TM}$ es decidible entonces debe existir una máquina, R, que lo decida.

$$R(\langle M, w\rangle) = \begin{cases} \text{acepta} & \text{si } M \text{ para con } w \\ \text{rechaza} & \text{si } M \text{ no para con } w \end{cases}$$

Podemos construir otra máquina S tal que:

1º Ejecute $R(\langle M, w\rangle)$

2º Si R rechaza, es decir, no para con w entonces rechaza.

3º Si R acepta, es decir, para con w simula M sobre w

4º Si M acepta entonces acepta

5º Si M rechaza entonces rechaza

La máquina S soluciona el problema de aceptación

La máquina S soluciona el problema de aceptación sobre máquinas de Turing que es un problema indecidible por tanto no podemos crear la máquina S y, en consecuencia tampoco R. Por lo tanto, $HALT_{TM}$ no se puede decidir.

**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), Potencia(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), If(x,y,z)*.

Demuestre que la función *Log*($x+1$,$n$), que calcula el logaritmo en base $n$ de un número entero, es una función primitiva recursiva.

NOTA: El logaritmo está definido para números mayores o iguales a 1. Al utilizar el argumento ($x$+1) el caso base de la recursión es $x$=0.

$$Log(x+1, n) = y \mid n^y \leq x+1 < n^{y+1}$$

| $x+1$ | $n$ | $Log(x+1, n)$ |
|---|---|---|
| 1 | 2 | 0 |
| 2 | 2 | 1 |
| 3 | 2 | 1 |
| 4 | 2 | 2 → |
| 5 | 2 | 2 |
| 6 | 2 | 2 |
| 7 | 2 | 2 |
| 8 | 2 | 3 |
| 9 | 2 | 3 |
| 10 | 2 | 3 |
| 11 | 2 | 3 |
| . | ⋮ | ⋮ |

CASO BASE

$Log(x+1, n) = Z_1$

CASO GENERAL

$Log(S(x+1), n) = g(Log(x+1), x+1, n) =$
$g(U_1^3, U_2^3, U_3^3)$.

Observamos que los valores de la función siguen un patrón:

$$Log(S(x+1), n) = \begin{cases} S(Log(x+1, n)) & \text{si } Potencia(n, S(Log(x+1,n))) \geq (x+1) \\ Log(x+1, n) & \text{en otro caso.} \end{cases}$$

Por tanto, podemos definir la función como:

$Log(S(x+1), n) = R(Z_1, C(If, C(MayorIgual, C(Potencia, U_3^3, C(S, U_1^3)), U_2^3),$
$, C(S, U_1^3),$
$, U_1^3))$

**EJERCICIO 6 (1 punto)**

(a) ¿Qué es un lenguaje NP?

(b) ¿Qué es un verificador de un lenguaje?

(c) Demuestre que un lenguaje es NP si y solo si es verificable polinomialmente.

a) Un lenguaje NP es un lenguaje que es decidible en tiempo polinomial mediante una máquina de Turing no determinista.

b) Un verificador de un lenguaje A es un algoritmo que permite conocer si una cadena w pertenece al lenguaje. Si su complejidad temporal depende polinómicamente de la longitud de la cadena, se dice que es un verificador en tiempo polinomial.

c) Si un lenguaje, A, es NP entonces existe una máquina de Turing no determinista que lo decide en un tiempo polinomial. Podemos construir un verificador en tiempo polinomial de la siguiente forma:

- Dado <w, c>

  1- Ejecutar la máquina sobre w tratando cada símbolo de c como la elección de cada paso indeterminista.

  2- Si el camino seleccionado por c acepta w entonces acepta, rechaza en caso contrario.

Si un lenguaje es verificable polinomialmente entonces existe un verificador V en tiempo polinomial. Podemos construir una máquina de Turing indeterminista de la siguiente forma:

1- Generar de forma indeterminista una cadena c de tamaño $n^a$.

2- Ejecutar V sobre <w, c>

3- Si V acepta → acepta
si no → rechaza.