Práctica 4

Lógica Computacional, 2018-2
Facultad de Ciencias, UNAM

Noé Salomón Hernández Sánchez
no.hernan@gmail.com

María del Carmen Sánchez Almanza
carmensanchez@ciencias.unam.mx

Albert Manuel Orozco Camacho
alorozco53@ciencias.unam.mx

25 de mayo de 2018

1. Ejercicios

Los ejercicios siguientes serán realizados en `Prolog`.

1.1. Árboles binarios

Considere a los términos `btnil/0` y `btbranch/3` que codifican un árbol binario (vacío y un rama con dos hijos, respectivamente). El siguiente predicado asegura la construcción de un árbol binario de acuerdo a la definición acostumbrada:

```
binTree(btnil).
binTree(btbranch(_, LChild, RChild)) :-
        binTree(LChild),
        binTree(RChild).
```

Realice los siguientes predicados de acuerdo a lo antes presentado.[1].

1. `infoTree/3`. Dado un árbol binario en el primer argumento, unifica el número de nodos internos que éste posee en el segundo y el número de hojas en el tercero.

[1] Para mayor información sobre recorridos en árboles, consultar `https://es.wikipedia.org/wiki/Recorrido_de_árboles#Recorrido_en_profundidad-primero`.

2. `binSearchTree/2`. Dada una lista (posiblemente, de enteros) en su primer argumento, unifica el *árbol binario de búsqueda* en su segundo argumento, construído a partir del primero.

3. `preOrder/2`. Dado un árbol binario binario (*¡de búsqueda!*) pasado como primer argumento, unifica la lista con su recorrido **preorden** en el segundo argumento.

4. `postOrder/2`. Dado un árbol binario binario (*¡de búsqueda!*) pasado como primer argumento, unifica la lista con su recorrido **postorden** en el segundo argumento.

5. `inOrder/2`. Dado un árbol binario binario (*¡de búsqueda!*) pasado como primer argumento, unifica la lista con su recorrido **en orden** en el segundo argumento.

1.2. Un autómata compresor

Un algoritmo moderno de compresión de cadenas ha detectado que la existencia de subcadenas *palíndromas* implica una redundancia innecesaria sintáctica, cuya supresión es inherente a la semántica del enunciado. Para evitar mayores pérdidas, se propone un método en el cual se recorra únicamente una vez una cadena de longitud n dada, de izquierda a derecha, detectando la existencia de palíndomos de longitud mayor a 1.

Una vez que se encontró un palíndromo, deberán suprimirse los últimos $\lceil \frac{n}{2} \rceil$ símbolos del palíndromo dado.

Consideremos el predicado `delPal/2` que unifica una cadena con su versión compresa, de acuerdo a los párrafos anteriores. Si α es una cadena de la forma

$$\alpha = \beta\omega\omega^R\gamma$$

(donde β y ω también son cadenas) entonces, el algoritmo propuesto deberá de procesar α de la siquiente manera:

$$\text{delPal}(\alpha) = \text{delPal}(\beta)\omega\text{delPal}(\gamma).$$

En el caso de que tengamos un palíndromo de longitud *impar*, α sería de la forma

$$\alpha = \beta\omega c\omega^R\gamma$$

(donde c es un símbolo) y su procesamiento sería

$$\text{delPal}(\alpha) = \text{delPal}(\beta)c\omega\text{delPal}(\gamma).$$

Realice la implementación de `delPal/2` en `Prolog`, considerando que se tiene como entrada una lista binaria (cadenas de 0s y 1s).

2. Entrega

La fecha de entrega es el próximo **sábado 02 de junio de 2018** por la plataforma de *Google Classroom* del curso y siguiendo los lineamientos del laboratorio.