INF-253 Lenguajes de Programación
Tarea 5: Prolog

20 de junio de 2022

1. Mecánica

Se les entregarán dos problemas a resolver utilizando el lenguaje de programación Prolog.

Para poder realizar la tarea, se debe utilizar SWI-Prolog, el cual se puede encontrar en: https://www.swi-prolog.org/download/stable.

2. El final de una aventura

Patode Hule, después de visitar cuatro lugares los cuales le dejaron muy agotado, solo quiere descansar, para ello se acerca a un lugar llamada el retiro lógico. Patode en este lugar comienza a recordar toda la aventura que tuvo, con una sonrisa aún sabiendo que este es el final de su historia, debido a que siento que logro impactar por dónde paso e hizo su mayor esfuerzo por realizar un buen trabajo. Para poder irse a descansar en su retiro lógico debe resolver una secuencia de consultas que lo llevarán a sentir que fue un viaje completo.

3. Gablema - Separación de eventos

El primer paso para relajarse es poder separar un grupo de eventos de tu vida del resto, entonces a Patode le gustaría poder tener una lista de eventos y sacar los eventos dentro del intervalo i hasta j. Patode para simplificar esto decide cuantificar los eventos con un entero. Entonces, él quiere realizar una consulta que reciba una lista de enteros, un número i, un número j y logre sacar los elementos de la lista dentro de ese intervalo.

Por ejemplo, si se tiene la lista de números $[1, 4, 23, 10, 2, 5]$ y se pide sacar los números entre el intervalo 3 hasta 5. El resultado de esta consulta sería: $[10, 2, 5]$

```
?- separacion([1, 4, 23, 10, 2, 5], 3, 5, L).

L = [10 2 5]
```

4. Pantema - División de eventos

Otra cosa muy importante para relajarse es poder comparar una secuencia de eventos dentro de tu vida, al igual que el problema anterior los eventos están cuantificados por enteros. Teniendo la lista de enteros, él debe separar esta lista en dos mitades, la parte de la izquierda y la derecha. Luego, cada mitad lo guarda dentro de una misma lista.

Por ejemplo, si la consulta recibe $[1, 4, 23, 10, 2, 5]$, la consulta debe entregar $[[1, 4, 23], [10, 2, 5]]$.

Nota: Siendo n la cantidad de elementos, si n es impar, la mitad de la izquierda será de largo $\lfloor n/2 \rfloor + 1$ y la mitad de de la derecha será de largo $\lfloor n/2 \rfloor$.

```
?- division([1, 4, 23, 10, 2, 5], LL).

LL = [[1, 4, 23], [10, 2, 5]]
```

5. PantLema - Pasado o Futuro, quién fue mejor

Ahora, a Patode no solo le gustaría poder dividir los eventos en dos mitades, si no que le gustaría poder comparar estas mitades, saber si es que dentro de una lista de eventos, cuales fueron los más predominantes: lo del pasado (mitad izquierda) o lo del futuro (mitad derecha). Para determinar cual fue más predominante, se suma todos los elementos de la mitad correspondiente. Si la mitad izquierda tiene un valor más alto o igual, entonces predominó el pasado, lo cual deberá responder con el número 1; si no significa que predominó el futuro, lo cual deberá responderse con el número 0.

Por ejemplo, si la consulta recibe $[1, 4, 23, 10, 2, 5]$. La mitad izquierda corresponde a $[1, 4, 23]$, dando una suma total de 28. La mitad derecha corresponde a $[10, 2, 5]$, dando una suma total de 17. Como la suma total de la mitad izquierda es mayor, entonces se debería responder con 1.

```
?- pasadofuturo([1, 4, 23, 10, 2, 5], RES).

RES = 1
```

6. GabPantema - Pasado o Futuro, el árbol acumulado

Por último, Patode se da cuenta que su vida ha estado regida por diversas listas de eventos, las cuales curiosamente están organizadas según un árbol binario

Recordar, que un árbol binario consiste en una representación de un nodo tiene dos nodos hijos, uno a la izquierda y otro a la derecha.

Para el caso de esta tarea, los nodos del árbol serán representados de la siguiente forma:

```
[lista_eventos, hijo_izquierdo, hijo_derecho]
```

Y un nodo hoja tendrá la siguiente forma:

```
[lista_eventos, [], []]
```

Para este problema se asume que cada nodo o puede tener dos hijos o dos listas vacías.

Esta consulta recibirá un árbol que seguirá la descripción previa. Este árbol contendrá en cada nodo una lista de eventos como se han descrito en los problemas previos. La consulta debe determinar si es que el árbol es bonito o no.

El árbol, para que se considere bonito, debe cumplir la siguiente propiedad: por cada nodo que no sea hoja, debe verificar si fue predominante el pasado o el futuro. Si en alguno de los dos hijos el pasado predominó, entonces en el nodo actual debe predominar el pasado. Si en ambos hijos predominó el futuro, entonces en el nodo actual debe predominar el futuro.

Por ejemplo, si la consulta recibe el siguiente árbol:

```
[[1, 4, 23, 10, 2, 5],
    [[1, 2, 3],
        [[1, 2],
            [], []],
        [[2, 3],
            [], []]],
    [[1, 2],
        [], []]]
```

Para simplificar la explicación, se tendrá en cuenta lo siguiente:

- nodo 1: [1, 4, 23, 10, 2, 5]
- nodo 2: [1, 2, 3]
- nodo 3: [1, 2]
- nodo 4: [1, 2]
- nodo 5: [2, 3]

El nodo 1 tiene como hijos al nodo 2 y nodo 3. Se aplica la consulta descrita en PantLema sobre el nodo 2 y el nodo 3, en el caso del nodo 2 se determina que predominó el pasado y en el caso del nodo 3 se determina que predominó el futuro. Entonces, como en alguno de los hijos del nodo 1 predominó el pasado, se debe verificar que en el nodo 1 también predominó el pasado. Se aplica la consulta del PantLema, se determina que en el nodo 1 también predomina el pasado. Por lo que, por ahora la propiedad descrita previamente se cumple.

El nodo 2 tiene como hijos al nodo 4 y nodo 5. Se aplica la consulta descrita en PantLema sobre el nodo 4 y el nodo 5, en el caso del nodo 4 se determina que predominó el futuro y en el caso del nodo 5 se determina que predominó el futuro. Entonces, como en ambos hijos predominó el futuro se debe verificar que en el nodo 2 predomina el futuro. Se aplica la consulta del PantLema, se determina que en el nodo 2 predomina el pasado. Por lo que, la propiedad descrita previamente NO se cumple. Por lo que el árbol no es bonito.

Cabe destacar que como los nodos 3, 4 y 5 son nodos hoja, no se debe revisar la propiedad.

```
?- arbolbonito(
[[1, 4, 23, 10, 2, 5],
    [[1, 2, 3],
        [[1, 2],
            [], []],
        [[2, 3],
            [], []]],
    [[1, 2],
        [], []]], ESBONITO)
ESBONITO = false
```

ESBONITO debe valer true si es que el árbol es bonito, sino debe valer false

7. Reglas del Juego

7.1. Archivos a entregar

- todos las consultas pedidas deben ir dentro de un mismo archivo en el mismo orden que salen en el enunciado de la tarea. Cabe destacar que en la sección tareas existirá un código base en el cual se podrán guiar.

7.2. Otras Reglas

- Cuidado con el orden y la identación de su tarea, puede llevar descuentos.
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- El código debe ser correctamente comentado, si no habrá descuento. Me refiero a descripción de las reglas y hechos.

- **La entrega debe realizarse en tar.gz y debe llevar el nombre: Tarea5LP_RolIntegrante.tar.gz**
- La entrega debe incluir un README.txt con el nombre del estudiante e instrucciones para utilizar su programa.
- **La tarea debe ser entregada el 4 de Julio a las 23:59, tareas atrasadas por más de 12 horas no serán aceptadas.**

8. Calificación

- Gablema (25 puntos)
- Pantema (25 puntos)
- PanGablema (25 puntos)
- GabPantema (25 puntos)
- Reglas de entrega (MAX -30 puntos)
- Falta de comentarios (-5 puntos c/u)
- Falta de README (-20 puntos)