README
PRÁCTICA 05

Liprandi Cortes Rodrigo: 317275605
Tinoco Miguel Laura Itzel: 316020189

11.Diciembre.2020

1 Observaciones:

Explicaciones de las soluciones.

1. **Relación 'elemento'**

   Primero definimos el caso base, este lo definimos como cuando sólo se tiene un elemento en la lista y es el elemento que estamos buscando En el caso recursivo, checamos la cabeza de la lista, si es el elemento entonces devuelve True, si no, busca en el resto de la lista.

2. **Relación 'suma'**. Devuelve la suma de los elementos de una lista.

   Creamos el hecho suma el cual cuenta con dos argumentos, la lista que contiene los elementos a sumar, y S la variable que sera el resultado de la suma.

   Usamos recursion para resolver el problema, tenemos el caso base en que si la lista es vacía S es 0 suma([],0), el caso recursivo lo implementamos con una regla, si la lista tiene al menos un elemento volvemos a aplicar suma con la cola de la lista y una nueva variable S2 tal que S es igual a la suma de la cabeza de la lista con S2.

3. **Relación 'palíndromo'**

   Primero definimos la relación concatena, que lo que hace es añadir un elemento al final de una lista, el caso base es añadir un elemento a una lista vacía, el recursivo va hasta el final de la lista y lo añade.

   Luego definimos la relación reversa, que nos devuelve la reversa de una lista, el caso base es la lista con un sólo elemento, en el recursivo lo que hicimos fue quedarnos con la cabeza de la lista, concatenarla a la reversa del resto de la lista y devolverla.

   En palindromo sólo checamos que si la lista pasada es igual a su reversa.

4. **Relación 'ocurrir'**. Devuelve el número de ocurrencias de un elemento en una lista.

   Creamos el hecho ocurrir el cual cuenta con 3 argumentos, una lista, el elemento de interés, y una variable que guardara el número de ocurrencias.

   Usamos recursion para resolver el ejercicio, tenemos el caso base en donde la lista es vacía y N es 0, el caso recursivo lo definimos con dos reglas, en las que la lista tiene un elemento o mas, si la cabeza es igual al elemento de interés se volverá aplicar ocurrir con la cola de la lista, el mismo elemento y una nueva variable N1 y N sera la suma de N1 mas 1, si es diferente se volverá aplicar ocurrir con la cola de la lista, el mismo elemento y N (es decir no se modificara la variable).

5. **Relaciones de equivalencia**

   Definimos 3 gatos, hicimos relaxiones auxiliares para hacer amigos al gato 1 y al gato 2, al gato 2 y al gato 3, para hacer más grande al gato 1 que el gato 2, y más grande el gato 2 que el gato 5.

   En todas las relaciones es necesario que los parametros sean gatos.

   En ser(X,X), sólo checamos que X sea gato. Puesto que ya está definido que X=X.

   En amigos(X,Y) checamos si en nuestras relaciones auxiliares X es amigo de Y o si Y es amigo de X.

   En esmasgrande(X,Y) checamos en nuestras relaciones auxiliares si X es mas grande que Y, o si X es más grande que un gato Z y ese gato Z es más grande que Y.

6. **Relación 'in_order'**. Obtiene el recorrido in order de un árbol binario.

   Primero definimos la relación 'arbolB' para construir arboles binarios, tenemos el caso base de árbol vacio = null, y el caso recursivo, nodo(lado izquierdo, raíz, lado derecho) es un árbol binario si el lado izquierdo y derecho son arboles binarios.

   También definimos la relación concatena que une dos listas, cuenta con tres argumentos las dos listas a concatenar y la lista que sera el resultado de la concatenación, usamos la definición recursiva.

   Finalmente definimos la relación in_order la cual cuenta con dos argumentos, el árbol binario, y una variable que sera la lista con los elementos del árbol ordenados por el recorrido, usamos la definición recursiva del recorrido. Tenemos el caso base donde el árbol y la lista son vacíos, y el caso recursivo, en el cual el árbol tiene un elemento o mas, entonces se vuelve aplicar in_order con el lado derecho del árbol y una lista "Ini", también se aplica con el lado derecho del arbol y una lista "Ind" y finalmente se concatena "Ini" con la raíz del árbol e "Ind".

7. **Relación abuelo**

Aquí tomamos los hechos más básicos del texto, escencialmente quiénes estaban involucrados y quién era padre/madre de quien.

Definimos relaciones padre y madre para que nos devolvieran respectivamente el padre de la persona o la madre de la persona.

Después hicimos una relación ancestro, esta buscaba el padre de la persona, si no estaba definido, entonces buscaba la madre.

En abuelo(X,Y) lo que hicimos fue llamar al ancestro del ancestro de X y a ese lo definimos como Y.