Facultad de Ciencias - UNAM
Lógica Computacional 2023-1
Práctica 5 - Ejercicios de listas

Favio E Miranda Perea
Javier Enríquez Mendoza
José Manuel Madrigal Ramírez

19/noviembre/2022
Fecha de entrega: 24/noviembre/2022 a las 23:59 hrs

Introducción

El objetivo de esta práctica es familiarizarnos más con el uso de listas en Prolog, Crea un archivo llamado `basedc2.pl` para escribir tu código.

Para recordar la sintaxis de las listas, realiza las siguientes consultas en la terminal de *SWI-Prolog* y escribe en un archivo `ReadMe.txt` una breve explicación de los resultados obtenidos.

- `[a,b,c,d] = [a|[a,b,c]].`
- `[a,b,c,d] = [Head|Tail].`
- `[a,b,c,d] = [a|X].`
- `[a,X] = [a|[a,b,c]].`
- `[a,b,c,d] = [a,b|[c,d]].`
- `[a,b,c,d] = [X,Y|_].`
- `[] = [[]].`
- `[X] = [[]].`

Procesamiento de listas

1. Definamos un predicado llamado `pura_a(X)` que es verdadero sólo si X es una lista que no contenga otro caracter más que *a's*.

```
?- pura_a([a,a,a,a]).
true.
?- pura_a([a,a,b,a,c]).
false.
?- pura_a([]).
true.
```

Pista: Para resolver este ejercicio primero tenemos que verificar que la cabeza de la lista sea una 'a'. Esto puede quedar especificado con un hecho (es decir, no hay necesidad de utilizar el símbolo :-). Después verificaremos el resto de la lista, esta vez con una regla recursiva.

2. Ahora definamos un predicado llamado `reemplaza_a_b_c(L_Entrada,L_Salida)` tal que `L_Salida` es una lista que se obtiene a partir de `L_Entrada` reemplazando las *a's* por *b's*, las *b's* por *c's* y las *c's* por *a's*.

```
?- reemplaza_a_b_c([a,b,c,1,3,a,b,c], L).
L = [b, c, a, 1, 3, b, c, a] ;
false.
?- reemplaza_a_b_c([], L).
L = [].
?- reemplaza_a_b_c([a,b,c], [b,c,a]).
true.
?- reemplaza_a_b_c([a,b,c], [b,c,b]).
false.
```

Pista: El predicado se verifica cuando ambas listas son vacías. Ese puede ser nuestro caso base. Después habrá que considerar cuatro casos más, un por cada letra y otro más en caso de que ambas listas tengan por cabeza una letra distinta a las que estamos reemplazando.

3. Definamos un predicado llamado `longitud(L, Longitud)` que resulta verdadero cuando `Longitud` es el número de elementos en `L`.

```
?- longitud([1,2,3],Longitud).
Longitud = 3.
?- longitud([1,2,3],3).
true.
?- longitud([],Longitud).
Longitud = 0.
?- longitud([1,2],4).
false.
```

Observa cómo los predicados anteriores difieren en sus resultados. `pura_a(X)`

simplemente evalúa la lista y regresa verdadero o falso dependiendo de su composición. Por otro lado, `reemplaza_a_b_c(L_i, L_o)` procesa cada elemento de la lista de entrada para generar una lista de salida del mismo tamaño. Finalmente el predicado `longitud(L,T)` regresa un resultado que no es una lista.

4. Definamos un predicado llamado `sumaUno(Li, Lo)` donde `Li` es una lista de enteros y `Lo` es la lista que resulta de sumarle uno a cada elemento de Li.

```
?- sumaUno([1,2,3],L).
L = [2,3,4].
```

5. Escribamos la definición del predicado `contiene_0(L)` Que se verifica si `L` es una lista que contiene un *0*.

```
?- contiene_0([1,2,0,1,s,f]).
true.
?- contiene_0([1,2,1,s,f]).
false.
```

6. Recordemos de álgebra lineal que una operación básica de vectores es el producto escalar. Definamos un predicado que lleve a cabo esta operación, así `multEscalar(X, L, R)` se verifica cuando `R` es el resultado realizar el producto escalar entre `X` y el vector `L`.

```
?- multEscalar(3, [1,2,3], R).
R = [3, 6, 9].
```

7. El producto punto era otra operación de vectores que consistía en multiplicar las entradas correspondientes de dos vectores y sumar los resultados. Definamos un predicado llamado `prodPunto(V1, V2, R)` que devuelva en la variable `R` el resultado de hacer el producto punto entre los vectores `V1` y `V2`.

```
?- prodPunto([1,2,3], [4,5,6], R).
R = 32.
```

8. Define un predicado `max(L, M)` que se verifique si `M` es el máximo elemento de una lista de enteros `L`.

```
?- max([1,2,3], R).
R = 3.
?- max([], R).
false.
```

9. Este es un ejercicio más interesante. Vamos a aplanar listas con un predicado llamado `aplana[]`, veamos algunos ejemplos:

```
?- aplana([[1],2,[[[[c]]]],[[[d]]]], X).
X = [1, 2, c, d] .
?- aplana([],X).
X = [].
```

Pistas:

- *Para aplanar una lista que no esté vacía tenemos que aplanar su cabeza y su cola. Podemos concatenar las listas resultantes utilizando el predicado* `append(L1,L2,Lr)` *que se encuentra predefinido en Prolog.*
- *Si la cabeza de la lista no es una lista, no tendría sentido aplanarla. ¿Qué podemos hacer para lidiar con este caso adecuadamente? Recordemos que una lista es, o bien la lista vacía* `[ ]` *, o un par de elementos separados por un pipe y encerrados entre corchetes* `[_|_]`.

10. Por último, define el predicado elementoN(N, L, R) que es verdadero cuando `R` es el *N-ésimo* elemento de la lista L.

```
?- elementoN(3,[1,2,3,4,5,6],R).
R = 3.
?- elementoN(0,[1,2,3,4,5,6], X).
false.
?- elementoN(4,[32,41,45,15,76], X).
X = 15.
```

Pista: Tenemos que recorrer recursivamente la lista mientras mantenemos guardada la posición a la que pertenece la cabeza de la lista actual en relación a la lista de entrada. Esto significa que necesitamos cuatro argumentos:

- *La posición del elemento que buscamos.*
- *La posición del elemento en el que estamos.*

- *La lista de entrada que se reducirá a cada paso de la recursión.*
- *El resultado*

De este modo la recursión se detendrá cuando el primer y el segundo argumentos sean el mismo. Por lo tanto, la definición del predicado orignal, con tres argumentos, debe llamar un predicado auxiliar de cuatro argumentos.

Entrega

La entrega se realizará por **parejas** y consistirá en un archivo comprimido que debe contener:

- El archivo `basedc2.pl` con el código necesario que da solución a los ejercicios.
- Un archivo readMe.txt que incluya la respuesta al primer ejercicio, sus nombres y una bitácora sobre el desarrollo de la práctica donde indiquen los problemas que se presentaron y cómo los resolvieron. Asimismo incluyan comentarios individuales al respecto.

Envíen su archivo comprimido a mi correo: *jose.manuel.madrigal.ramirez@gmail.com* con el siguiente formato:

Practica5-Apellido1Nombre1Apellido2Nombre2.zip

Cualquier duda que tengan no duden en enviarme un correo o comunicarse por Telegram. 😄