Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Conceptos de Inteligencia Artificial
&
Sistemas Inteligentes Artificiales

Trabajo Práctico: **Aritmética y Estructuras de Datos en Prolog**

Ejercicios

Observación: Para los ejercicios en los que se requiere definir predicados en Prolog para la resolución de problemas, se recomienda explicar la estrategia adoptada para su implementación (casos considerados, etc.).

1. *a*) El predicado predefinido `is/2` y los operadores relacionales permiten evaluar expresiones aritméticas apelando a las capacidades del procesador sobre el cual se esté ejecutando el intérprete de Prolog. Sin embargo, su utilización debe respetar ciertas restricciones. Indique claramente cuáles son dichas restricciones y sobre qué operandos se aplican.

   b) Determine y justifique la respuesta a las siguientes consultas en Prolog:

   i) `?- 7 is 4+3.`
   ii) `?- 10-3 is 10-3.`
   iii) `?- X is 5*6.`
   iv) `?- 5*6 is X.`
   v) `?- X is 'hola'.`
   vi) `?- X = 2 + 2, 4 is X.`
   vii) `?- 10-3 =:= 10-3.`
   viii) `?- X = 3+2, 5 < X.`
   ix) `?- X = 2*3, Y >=X, Y = 7-2.`
   x) `?- 10-3 =\= 7.0.`

2. Toda función n-aria puede ser representada mediante una relación n+1-aria al incorporar el resultado esperado como un argumento adicional en la relación. Teniendo en cuenta esto, defina predicados en Prolog para computar las siguientes funciones:

   a) Función suma.

   b) Función producto.

   c) Función factorial.

   d) Función de Fibonacci.

   Observación: adoptar la representación numérica tradicional para poder utilizar el predicado `is/2`.

3. Defina en Prolog el predicado `sn(N)` para verificar si `N` es un número codificado en la notación $s^n(0)$. ¿Es posible utilizar este predicado en forma reversible para generar todos los números en notación $s^n(0)$? ¿Cómo?

4. Defina un predicado en Prolog para sumar dos números representados en notación $s^n(0)$. Explique cuáles son los usos adicionales que se le puede dar a este predicado valiéndose de la reversibilidad de predicados en Prolog.

5. Considere el siguiente programa PROLOG:

```
p1(X):- X=1+2.      p3(X,Y,Z):- X=Y+Z, 7=X.     p5(X,Y):- p1(Y), X>Y.
p2(X):- X=3, Y=4.   p4(X):- X is 5*Y, Y=2.      p6(X):- p1(X), Y is X+1, p2(Y).
```

Indique las respuestas que daría el intérprete de PROLOG a las siguientes consultas, justificando adecuadamente en cada caso.

(i) `?- p1(3).` (ii) `?- p2(Y).` (iii) `?- p3(X,3,4).` (iv) `?- p4(10).` (v) `?- p5(4,X).` (vi) `?- p6(X).`

6. *a*) Defina en PROLOG el predicado separar(L,H,T) que recibe una lista L y retorna: en H la *cabeza* de L y en H la *cola* de L. Ejemplo:

?- separar([a,b,c],H,T).
H = a
T = [b,c]

b) En base al predicado definido en el inciso anterior, indique la respuesta a las siguientes consultas en PROLOG:

i) ?- separar([X,b,c],a,Y).
ii) ?- separar([a|L],X, [b,c]).
iii) ?- separar([a,[b,c]],X,Y).
iv) ?- separar([a,b,c],a,[b,c]).

7. Defina en PROLOG el predicado primerosTres(L,A,B,C) que recibe una lista L y retorna en A, B y C, respectivamente, los tres primeros elementos de L. Ejemplo:

?- primerosTres([a,b,c,d,e],A,B,C).
A = a
B = b
C = c

8. Brinde una sustitución unificadora para los siguientes pares de listas en PROLOG. En caso de no existir tal sustitución, indique claramente el motivo.

a) L_1 = [A, B, C] y L_2 = [1, 2, 3].
b) L_1 = [X | [ ]] y L_2 = [[hola]].
c) L_1 = [z | [ ]] y L_2 = [[ ] | Z].
d) L_1 = [1, X] y L_2 = [Y, [Z , T]].
e) L_1 = [1, X] y L_2 = [Y | [Z , T]].
f) L_1 = [1, X] y L_2 = [Y | [[Z , T]]].
g) L_1 = [a, b | C] y L_2 = [A | [B , c]].
h) L_1 = [a, b | C] y L_2 = [A | [B]].

9. Defina en Prolog predicados para realizar lo siguiente:

   a) Dados dos elementos, crear una lista con esos dos elementos.

   b) Insertar un elemento en una lista:

      i) Al comienzo de la lista.

      ii) Al final de la lista.

   c) Concatenar dos listas.

   d) Determinar si un elemento aparece en una lista.

   e) Contar la cantidad de veces que un determinado elemento aparece en una lista.

   f) Invertir una lista.

   g) Eliminar un elemento de una lista:

      i) Eliminar sólo una aparición del elemento.

      ii) Eliminar todas las apariciones del elemento.

   h) Reemplazar un elemento de una lista por otro:

      i) Reemplazar sólo una aparición del elemento.

      ii) Reemplazar todas las apariciones del elemento.

   i) Ordenar los elementos de una lista de menor a mayor (Observación: puede asumir que los elementos de la lista son números).

   j) Desplazar una posición a la derecha a todos los elementos de una lista. Ejemplo: dada la lista $[x_1, x_2, \ldots, x_n]$ deberá retornarse la lista $[x_n, x_1, \ldots, x_{n-1}]$).

   k) Desplazar una posición a la izquierda a todos los elementos de una lista. Ejemplo: dada la lista $[x_1, x_2, \ldots, x_n]$ deberá retornarse la lista $[x_2, \ldots, x_n, x_1]$).

10. Un conjunto puede modelarse en Prolog mediante una lista de elementos sin repeticiones. Adoptando esta representación, defina en Prolog predicados que implementen las siguientes operaciones sobre conjuntos:

    a) Comprobar si una lista de elementos constituye un conjunto válido.

    b) Determinar si un elemento pertenece a un conjunto.

    c) Incorporar un elemento a un conjunto.

    d) Unir dos conjuntos.

    e) Intersectar dos conjuntos.

    f) Calcular la diferencia entre dos conjuntos.

    g) Dada una lista de elementos con repeticiones, construir el conjunto que contenga todos los elementos de la lista.

    Observación: En cada caso deberá adoptarse una política adecuada para la validación de los datos de entrada.

11. Considere la siguiente BNF que describe una serie de expresiones sobre conjuntos:

$$E ::= E + E \mid E * E \mid E - E \mid (E) \mid C$$
$$C ::= \text{“una lista representado a un conjunto”}$$

Asumiendo que $+$, $*$ y $-$ denotan las operaciones sobre conjuntos de *unión*, *intersección* y *diferencia* respectivamente, defina en PROLOG un predicado eval/2 que reciba una expresión y retorne el resultado correspondiente a su evaluación. Ejemplo:

```
?- eval([a,b,c,d,e] + ([h,i,j,k,l] * ([j,k,l,m,n,t,u] - [m,n])),A).
A = [a,b,c,d,e,j,k,l]
```

12. Considere un dominio de aplicación en el que se desea representar información sobre las conexiones entre ciudades de una región. Por ejemplo, considere la región del Noreste de Bahía Blanca. A continuación se muestra la abstracción del problema, donde la información es representada mediante cláusulas de PROLOG:

```
ruta(bahia,medanos).
ruta(bahia,punta).
ruta(bahia,dorrego).
ruta(bahia,tornquist).
ruta(punta,dorrego).
ruta(tornquist,pringles).
ruta(pringles,tresarroyos).
ruta(dorrego,tresarroyos).
```

   a) ¿Qué ocurre al incorporar como primera cláusula del programa anterior la siguiente regla?: ruta(C1,C2) :- ruta(C2,C1).

   b) ¿Qué sucede al reemplazar la regla definida en el inciso anterior por esta otra?: conexion(C1,C2) :- ruta(C2,C1).

   c) Defina en PROLOG el predicado triDirecta(C) que determine si existen rutas que conecten a la ciudad C en forma directa con otras 3 ciudades.

   d) Defina en PROLOG el predicado camino(Origen,Destino,Camino) que retorne en Camino la lista con las ciudades que forman un camino entre la ciudad Origen y la ciudad Destino. OBSERVACIÓN: Deberá contemplarse la existencia de ciclos.

   e) Re-escriba la base de conocimiento brindada de forma tal de añadir a cada ruta información referente a la distancia entre las ciudades que conecta.

   f) Dada la representación del inciso anterior, defina en PROLOG el predicado distancia(Origen,Destino,Valor) que retorne en Valor la distancia que existe entre la ciudad Origen y la ciudad Destino (para algún camino).

13. Una lista L se dice *palíndroma* si el reverso de L es igual a L. Teniendo en cuenta esto, defina en PROLOG predicados para realizar lo siguiente:

   i) Determinar si una lista es *palíndroma*.

   ii) Dada una lista L de longitud n retornar una lista de longitud $2 * n$, obtenida a partir de L, que sea palíndroma.

14. Defina en PROLOG el predicado aplanar/2 que recibe una lista de elementos y retorne su *imagen aplanada*. Ejemplo:

```
?- aplanar([a,[b,c],[[d,e],f],[],[[[]],g,h]], X).
X = [a,b,c,d,e,f,g,h].
```

15. Defina en PROLOG un predicado que dada una lista L y un número N, desplace todos los elementos de la lista N posiciones a la derecha. Análogamente, defina un predicado para desplazar los elementos de la lista N posiciones a la izquierda.

16. Defina en PROLOG un predicado que reciba dos listas L_1 y L_2, y determine si L_1 es *prefijo* de L_2. (OBSERVACIÓN: explique la política adoptada para la identificación del *prefijo* de una lista).

17. Defina en PROLOG el predicado masVeces/3 que reciba como argumento una lista de elementos y retorne el elemento que más veces se repite, así como también la cantidad de veces que lo hace. (OBSERVACIÓN: puede asumir que el elemento que más veces se repite es único). Ejemplo:

    ?- masVeces([a,b,c,d,e,a,b,c,d,a,b,c,a,b,a],E,V).
    E = a
    V = 5

18. Implemente en PROLOG un predicado espejo/2 que reciba como argumento una lista L y retorne el *espejo* de L. El espejo de L corresponde a la lista con los elementos de L en orden inverso, siendo aplicada esta política también a las sublistas dentro de L. Por ejemplo:

    ?- espejo([a,b,c], Rta).
    Rta = [c,b,a]
    ?- espejo([a,b,[c,d],[e,[f,g]]], Rta).
    Rta = [[[g,f],e],[d,c],b,a]

19. Implemente en PROLOG un predicado eliminar_espejos/3 que reciba como argumento una lista E y una lista L y retorne la lista resultante de eliminar todas las apariciones del espejo de E <u>*dentro de*</u> L. (OBSERVACIÓN: Deberá contemplarse el caso en que el espejo de E sea un elemento de una sub-lista de L, a cualquier nivel). Por ejemplo:

    ?- eliminar_espejos([a,b,c], [d,h,1,[a,b,c],[4],[7,[c,b,a]], [[c,b,a]]], Rta)
    Rta = [d,h,1,[a,b,c],[4],[7],[]]
    ?- eliminar_espejos([a,b,c], [c,b,a], Rta).
    Rta = [c,b,a]

20. Implemente en PROLOG un predicado eliminar_vocales/2 que reciba como argumento una lista de letras L (posiblemente conteniendo otras listas anidadas) y retorne la imagen aplanada L_A de L, eliminando además todas las vocales de L.

    OBSERVACIÓN: Dada una lista L = [a,[b,c,e],f,[i,[h]]], su imagen aplanada (sin eliminar las vocales) es L_A = [a,b,c,e,f,i,h].

    Por ejemplo:

    ?- eliminar_vocales([a,b,c], Rta).
    Rta = [b,c]
    ?- eliminar_vocales([a,[b,c,e],f,[i,[h]]], Rta).
    Rta = [b,c,f,h]