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

Lógica para Ciencias de la Computación

Trabajo Práctico "B"

Estructuras de Datos en Prolog

Primer Cuatrimestre de 2008

Observación importante: Las consultas relacionadas con los temas desarrollados en los trabajos prácticos serán únicamente respondidas durante las propias clases prácticas a lo largo del cursado.

Ejercicios

1. Proponer un predicado Prolog que verifique si su argumento representa un número natural codificados en notación $s^n(0)$. ¿Se podrá usar este mismo predicado para generar todos los números naturales? ¿De qué manera?

2. Definir un programa para sumar dos números naturales, representados en notación $s^n(0)$, a través de incrementos sucesivos.

3. En base al predicado para sumar definido en el inciso anterior, implementar un programa en Prolog para multiplicar dos números naturales mediante sumas sucesivas.

4. Una vez más, en base al predicado para multiplicar definido en el inciso anterior, implementar un programa en Prolog para calcular las potencias entero positivas de un número natural, apelando a multiplicaciones sucesivas.

5. Definir en Prolog los siguientes predicados recursivos para manejar listas:

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

   b) Insertar un elemento en una lista:

      1) Al comienzo de la lista.
      2) Al final de la lista.

   c) Concatenar dos listas.

   d) Buscar un elemento en una lista.

   e) Invertir una lista.

   f) Borrar un elemento en una lista:

      1) Borrando sólo una aparición del elemento.
      2) Borrando todas las apariciones de ese elemento.

   g) Cambiar un cierto elemento por otro:

      1) Cambiar sólo una aparición del elemento.
      2) Cambiar todas las apariciones de ese elemento.

h) Detectar si una lista resulta palíndroma.

i) Dada una lista de longitud n, generar otra lista de longitud $2n$ que sea palíndroma.

j) Desplazar una posición a la derecha a todos los elementos de una lista (*i.e.*, pasar de la lista $[x_1, x_2, \ldots, x_n]$ a la lista $[x_n, x_1, \ldots, x_{n-1}]$).

k) Desplazar una posición a la izquierda a todos los elementos de una lista (*i.e.*, pasar de la lista $[x_1, x_2, \ldots, x_n]$ a la lista $[x_2, \ldots, x_n, x_1]$).

6. Un conjunto puede ser modelado mediante una lista de elementos sin repeticiones. Adoptando esta representación, implementar las siguientes operaciones sobre conjuntos en lenguaje Prolog.

   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 un conjunto que contenga todos los elementos de esa lista.

   Observación: Adoptar una política adecuada para la validación de la entrada.

7. Considere un grafo $G = (V, A)$, donde V es el conjunto de vértices y A el conjunto de arcos (a_i, a_j), etiquetados por la distancia que separa al vértice a_i del vértice a_j. Adoptar una representación en Prolog apropiada para este tipo grafos. Luego, crear un programa Prolog que dados un grafo codificado de acuerdo a la representación adoptada y un par de vértices, calcule la distancia entre ellos. Implementar dos versiones: una que controle ciclos y otra que no.

8. Confeccionar un predicado Prolog que sea capaz de determinar el tipo de argumento que ha recibido (*i.e.*, si es una variable, un átomo, un entero o una lista).

9. Asumiendo que se dispone de un diccionario de pares (p_c, p_i), donde p_c es una palabra en castellano y p_i es su equivalente en inglés, implementar en Prolog un traductor de oraciones. Indicar claramente la convención adoptada para representar las oraciones.

10. Definir predicados recursivos para ordenar:

    a) Una lista de números.

    b) Una lista de palabras.

    Aclaración: representar las palabras mediante listas de letras.

11. Encontrar al menos dos soluciones alternativas para los predicados definidos en el ejercicio anterior.

12. Dada una serie de platos, clasificados en entradas, minutas o postres, y con su respectivo precio asociado, elaborar un predicado que encuentre un menú completo por un costo menor a n, para un n en particular.

13. Sea G la siguiente gramática:

$$G = \{S \to aSc,\ S \to T,\ T \to bTc,\ T \to bc\}$$

Confeccionar un programa Prolog que determine si una cadena w pertenece a $L(G)$, el lenguaje generado por G.

Observación: Emplear certeramente al predicado append/3.

14. Escribir un predicado Prolog que dada una lista L y un número n, rote la lista L a la derecha n posiciones. Posteriormente, escribir otro para rotar la misma lista n posiciones a la izquierda.

15. Escribir un predicado Prolog que dadas dos listas L_1 y L_2, determine si L_1 es prefijo de L_2.

16. Implementar en Prolog los siguientes algoritmos de ordenamiento, los cuales a partir de una lista de enteros retornan su imagen ordenada:

    a) Insert-sort.

    b) Select-sort.

    c) Quick-sort.

17. Sean consideradas como válidas las expresiones descriptas por la siguiente BNF:

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

Asumiendo que $+$, $/$, $*$ y $-$ denotan las operaciones sobre conjuntos de unión, unión exclusiva, intersección y diferencia respectivamente, definir en Prolog un predicado eval/2 que tome una expresión y retorne el resultado correspondiente a su evaluación. Por ejemplo:

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

18. Una pila de bloques puede ser modelada mediante un conjunto de hechos de la forma sobre(Bloque1,Bloque2), denotando que Bloque1 está apoyado sobre Bloque2. Definir el predicado encima(Bloque1,Bloque2), que verifique si Bloque1 está por encima y en la misma pila que Bloque2.

Pista: La relación encima/2 es la clausura transitiva de la relación sobre/2.

19. Confeccionar un predicado Prolog que relacione una lista de elementos con su imagen aplanada. Por ejemplo:

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

20. Adoptando una representación razonable para *árboles binarios de búsqueda*, implementar a través de predicados Prolog las siguientes relaciones:

a) crearArbolVacio(X): retorna en X un árbol vacío.

b) insertar(E, A, NA): inserta el elemento E en el árbol A, retornando el nuevo árbol en NA (asumir que si el elemento a ser insertado ya pertenece al árbol, entonces se retorna el mismo árbol).

c) eliminar(E, A, NA): elimina el elemento E del árbol A, retornando el nuevo árbol en NA. Se debe manejar apropiadamente la eliminación de elementos tanto en las hojas del árbol como en sus nodos internos.

d) altura(A, X): retorna en X la altura del árbol A.

e) balanceado(A): determina si el árbol A está balanceado o no.

21. Considerando como expresiones válidas a las definidas mediante la siguiente **BNF**,

$$E ::= E + E \mid E - E \mid E * E \mid E/E \mid (E) \mid N$$
$$N ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

definir mediante predicados Prolog las siguientes relaciones:

a) generaArbol(E, A): a partir de una expresión E, retorna en A el árbol binario asociado a esta expresión, adoptando nuevamente una representación razonable para los árboles binarios.

b) post(A, P): a partir de un árbol A, representando a una expresión, retorna en P su recorrido en postorden.

c) eval(P, V): a partir de un recorrido P en postorden, de un árbol que representa una expresión, evalúa esta expresión y retorna su resultado en V.

Ejercicios opcionales

1. Definir un predicado Prolog que reciba como argumento una lista de elementos y retorne el elemento que más veces se repite, junto con la cantidad de veces que lo hace (asumir que el elemento que más veces se repite es único). Por ejemplo:

   ?- masveces([a,b,c,d,e,a,b,c,d,a,b,c,a,b,a],E,V).

   E = a

   V = 5

   yes

2. Rehacer el ejercicio 13, en esta ocasión sin emplear el predicado append/3.

3. Investigar cómo funciona el algoritmo de ordenamiento conocido como *shell-sort* para luego implementar un predicado en Prolog que a partir de lista de enteros y retorne su imagen ordenada.

Referencias

[SS94] Sterling, L., and Shapiro, E. *The Art of Prolog*, 2nd ed. The MIT Press, 1994.