

Guía de Trabajos Prácticos XII – Programación Lógica IV

1. Cree un predicado en prolog que en una sola declaración del mismo y sin utilizar el operador “;”, valide que para una lista con igualdades aritméticas de la forma “exp1 = exp2” que todas las igualdades se cumplan.
2. Dado un predicado f/1 que se pueda manipular dinámicamente, y que siempre tiene como argumento un número entero, crear un predicado agregarf/1 que agregue una nueva declaración a f/1 con el argumento pasado como parámetro, siempre y cuando el argumento sea un número entero y el mismo no haya sido agregado con anterioridad.
3. Cree un programa en prolog que mediante el predicado forall/2 verifique si una lista contiene elemento todos del mismo tipo. Los tipos pueden ser: números, átomos, otros. El tipo otro es cualquier tipo que no sea número o átomo.
4. Genere un programa en prolog que resuelva el siguiente problema: teniendo una matriz de 3x3, se deben colocar todos y cada uno de los números de 1 al 9 de tal forma que la suma de cada una de las filas, cada una de las columnas y cada una de las diagonales principales dé el mismo número.

AYUDA: puede utilizar el predicado usado para encontrar las permutaciones de una lista.

Importante: NO USAR PREDICADOS PREDEFINIDOS.

5. Dado una lista de números enteros, crear un programa en prolog que compruebe si agregando operaciones aritméticas (+, -, *) y uno y solo un signo de igualdad (=) entre los números de la lista, se puede obtener una igualdad en la misma.

Ejemplo:

?- igualdad([2, 3, 4, 2], X).

X = [2, *, 3, =, 4, +, 2]

X = [2, *, 3, -, 4, =, 2]

NOTA 1: Validar que la lista contenga al menos dos elementos y solo contenga números enteros.

NOTA 2: Para simplificar el desarrollo, no se debe tener en cuenta la precedencia de los operadores, siempre se opera de izquierda a derecha.

Predicados aconsejados:

lista_valida/1: valida que el parámetro recibido sea una lista válida

lista_aritmetica/2: recibe como primer argumento una lista de enteros y retorna en el segundo la lista con los operadores aritméticos intercalados

operar/3: recibe en el primer argumento el resultado parcial, en el segundo una lista con el resto de la operación, y retorna en el tercer argumento el resultado final.

igualdad/2: recibe una lista de números enteros y retorna una lista con los números, los operadores y la igualdad.

6. Implemente en prolog los predicados que se describen a continuación teniendo en cuenta las siguientes consideraciones:

- No se deben usar predicados predefinidos
- Se estará trabajando con conjuntos
- Los mismos se representarán como listas de elementos
- Los conjuntos no pueden tener elementos repetidos, condición que hay que validar en los conjuntos de entradas y asegurar en el de salida

a) union(C1, C2, U): El predicado union/3 ejecuta la unión entre los conjuntos C1 y C2 y devuelve el resultado en el conjunto U.

union([1, 2, 3], [3, 4, 5], U).
U = [1, 2, 3, 4, 5].

b) interseccion(C1, C2, U): El predicado interseccion/3 ejecuta la intersección entre los conjuntos C1 y C2, y devuelve el resultado en el conjunto U.

interseccion([1, 2, 3], [3, 4, 5], U).
U = [3].

c) diferencia(C1, C2, U): El predicado diferencia/3 ejecuta la operación diferencia de conjuntos entre los conjuntos C1 y C2, y devuelve el resultado en el conjunto U.

diferencia([1, 2, 3], [3, 4, 5], U).
U = [1, 2].

IMPORTANTE: No utilizar predicados predefinidos y verificar que los conjuntos de entrada sean válidos, es decir que no contengan elementos repetidos, en caso de ser inválidos alguno de los dos, el predicado debe fallar.

7. Crear un programa en prolog que dada un conjunto de palabras (definidas como hechos mediante el predicado "p/1") y un patrón de búsqueda, retorne una lista con todas las palabras que coincidan con el patrón de búsqueda.

Se puede tomar como ejemplo las siguientes declaraciones:

p(árbol).
p(animal).
p(base).
p(caramelo).
p(caracol).
p(curso).
p(dedo).
p(dedal).
p(diferencia).
p(diferencial).
p(diferente).
p(elefante).
p(electricidad).
p(esfera).
p(fenómeno).
p(fiera).
p(guante).
p(hormiga).
p(hormigón).
p(irlanda).

El patrón de búsqueda será cualquier átomo que contenga letras en minúsculas y los símbolos '*' (representa una secuencia de cero o más letras no definidas), y '?' (representa una letra no definida).

La búsqueda no debe diferenciar entre vocales con y sin acento y solo se manejarán letras en minúscula tanto en la definición como en el patrón de búsqueda.

Ejemplos:

?- buscar(*, L).

L = [árbol, animal, base, caramelo, caracol, curso, dedo, dedal, diferencia, diferencial, diferente, elefante, electricidad, esfera, fenómeno, fieras, guante, hormiga, hormigón, irlanda].

?- buscar(arbol, L).

L = [árbol].

?- buscar(arbol*, L).

L = [árbol].

?- buscar(arbol?, L).

L = [].

?- buscar(ele*, L).

L = [elefante, electricidad].

?- buscar(ele*ad, L).

L = [electricidad].

?- buscar(*l, L).

L = [árbol, animal, caracol, dedal, diferencial].

ATENCIÓN: Se puede usar cualquier predicado predefinido. En particular para pasar un átomo a lista de átomos de los caracteres que lo componen se puede usar el predicado "atom_chars/2", por ejemplo:

?- atom_chars(hola, L).

L = [h, o, l, a].

ATENCIÓN: En caso de ser necesario, utilizar el apóstrofe para delimitar y explicitar los átomos.