Programación Lógica

Laboratorio 1 – 2018

Facultad de Ingeniería
Instituto de Computación

El objetivo de este obligatorio es experimentar en aspectos básicos de la Programación Lógica mediante la implementación de predicados simples en Prolog.

Nota previa - IMPORTANTE

Se debe cumplir íntegramente el "Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios", disponible en:
`http://www.fing.edu.uy/inco/pm/uploads/Ense%f1anza/NoIndividualidad.pdf`

En particular está prohibido utilizar documentación de otros grupos o de otros años, de cualquier índole, o hacer público código a través de cualquier medio (EVA, correo, papeles sobre la mesa, etc.).

Predicados a implementar

El laboratorio consiste en la implementación de los predicados detallados a continuación. Solamente puede utilizarse Prolog puro, incluyendo el manejo de listas y la parte de aritmética. No puede utilizarse ninguna otra funcionalidad de SWI-Prolog, a menos que se indique específicamente. La implementación debe realizarse de manera que pueda ser ejecutada en la plataforma SWI-Prolog. Se valorarán positivamente implementaciones que sean simples y claras, y a la vez eficientes.

1) Predicados sobre listas

largo(+L,?N) ← *N es el largo de la lista L*.
Ej.: `largo([a,b,c],3).`

largoAc(+L,+Ac,?N) ← *N es* el largo de la lista L, el predicado es *tail-recursive*

ultimo(?L,?U) ← *el último elemento de la lista L unifica con U.*
Ej.: `ultimo([pedro,ana,luisa],luisa).`

penultimo(?L,?U) ← *el penúltimo elemento de la lista L unifica con U.*
Ej.: `penultimo([pedro,ana,luisa],ana).`

diferentes(+L) ← *todos los elementos de la lista L son distintos entre sí*
Ej.: `diferentes([a,b,c])`

diferentes (+L1,+L2) ← *L1 y L2 tiene el mismo largo y todos los elementos correspondientes son distintos*
`Ej.: diferentes([a,b,c],[1,2,3]); diferentes([1,2,3],[3,1,2]).`

simplificada(+L1,?L2) ← *L2 contiene los mismo elementos que L1 y en el mismo orden, pero con una sola instancia de elementos consecutivos repetidos en L1.*
Ej.: `simplificada([a,a,a,a,b,c,c,a,d,d,b,b],[a,b,c,a,d,b]).`

empaquetada(+L1,?L2) ← *L2 contiene listas con los elementos consecutivos repetidos (ocurren 1 o más veces consecutivas) de L1, en el mismo orden*
Ej.: `empaquetada([a,a,a,a,b,c,c,a,d,d,b,b],[[a,a,a,a],[b],[c,c],[a],[d,d],[b,b]]).`

comprimida(+L1,?L2) ← *L2 contiene pares formados por los elementos de L1, en el mismo orden y la cantidad de veces consecutivas que ocurren*
`comprimida([a,a,a,a,b,c,c,a,d,d,b,b],[p(a,4),p(b,1),p(c,2),p(a,1),p(d,2),p(b,2)]).`

Descomprimida(?L1,+L2) ← *L2 contiene pares formados por los elementos de L1, en el mismo orden y la cantidad de veces consecutivas que ocurren*
descomprimida([a,a,a,a,b,c,c,a,d,d,b,b],[p(a,4),p(b,1),p(c,2),p(a,1),p(d,2),p(b,2)]).

(Notar que comprimida y descomprimida representan la misma relación pero tienen distinto modo de instanciación de las variables)

2) Predicados sobre matrices

(Se representan las matrices como listas de filas, siendo cada fila una lista de elementos.)

matrizFija(?M,?N,+E,?A) ← *A es una matriz de M filas y N columnas. Cada celda debe tener el valor E. La matriz se representa mediante una lista de M filas, donde cada fila es una lista de N celdas.*
Ej.: `matriz(2,2,-10,[[-10,-10],[-10,-10]]).`

valor_celda(+I,+J,+A,?E) ← *E es el contenido de la celda (I,J) de la matriz A.*

nuevo_valor_celda(+I,+J,+A1,+E,?A2) ← *A2 es una matriz que contiene el valor E en la celda (I,J) y en el resto de las celdas contiene los mismos valores que A1.*

cartesiano(+V1,+V2,?M) ← *V1 y V2 son vectores de igual dimensión N, M es una matriz NxN de pares (V1i,V2j) como elemento fila i y columna j de la matriz, siendo V1i el i-ésimo elemento de V1 y V2j el j-ésimo elemento de V2*

`Ej.: cartesiano([1,2],[a,b],[[(1,a),(1,b)],[(2,a),(2,b)]]).`

composicion(+M1,+M2,?M) ← *M es la composición elemento a elemento con el el operador '+' de las matrices de igual dimensión M1 y M2*

`Ej.: composicion([[1,2],[3,4]],[[a,b],[c,d]],[[1+a,2+b],[3+c,4+d]])`

3) Cuadrados latinos

Un cuadrado latino de orden N es una matriz de NxN elementos en la que cada casilla está ocupada por uno de N símbolos de tal modo que cada uno de ellos aparece exactamente una vez en cada columna y en cada fila.

La siguiente matriz es un cuadrado latino:

1	2	3
2	3	1
3	1	2

latino(+N,+E,?Lat) ← *Lat es un cuadrado latino de orden N sobre el conjunto de elementos E*

ej.: latino(2,[a,b],[[a,b],[b,a]]).

Se debe realizar pruebas para N creciente, comenzando en 2 y terminando en el máximo que su implementación pueda tolerar.

4) Cuadrados greco-latinos

Un cuadrado greco-latino, o cuadrado de Euler o cuadrado latino ortogonal de orden n es una matriz cuadrada n×n con elementos en dos conjuntos S y T (eventualmente iguales), ambos con n elementos. Cada celda contiene un par ordenado (s, t), siendo s un elemento de S y t de T, de forma que cada elemento de S y cada elemento de T aparezca exactamente una vez en cada fila y una vez en cada columna y que no haya dos celdas conteniendo el mismo par ordenado.

La siguiente figura es un ejemplo de cuadrado grecolatino de 3x3, con S={A,B,C} y T={1,2,3}

A 1	B 2	C 3
C 2	A 3	B 1
B 3	C 1	A 2

Se desea implementar el siguiente predicado:

cuadrado_GL(+N,+L1,+L2,?C) ← *C es un cuadrado grecolatino de orden N, siendo L1 y L2 los elementos de S y de T respectivamente*

ej.: cuadrado_GL(3,[a,b,c],[1,2,3],[[a+1,b+2,c+3],[c+2,a+3,b+1],[b+3,c+1,a+2]]

Se debe realizar pruebas para N creciente, comenzando en 2 y terminando en el máximo que su implementación pueda tolerar.

5) Pruebas de eficiencia

a. Matrices .Utilizando solamente los predicados pedidos para matrices, realice las siguientes pruebas para diferentes valores de M y N:

- Cree una matriz de tamaño MxN.
- En cada posición (I,J) de la matriz poner el valor (I + J).

b. Cuadrados latino y greco-latino. Realice pruebas para N = 2,3,4,5, 6, ...

Entrega

Los trabajos deberán ser entregados siguiendo el procedimiento descrito anteriormente antes del viernes 27/04/2018 a las 23:55, sin excepciones. No se aceptará ningún trabajo pasada la citada fecha.

La entrega se realizará a través del EVA del curso. Se debe entregar un archivo llamado 'lab1_#######.zip', donde ####### es el número de cédula del estudiante que realiza la entrega, conteniendo:

1. La Implementación de un módulo cuya interfaz exporte los predicados solicitados, en un archivo llamado 'lab1.pl'. Cada predicado debe contar con un breve comentario, usando la notación vista en el curso, documentando su funcionamiento. Si se implementan predicados auxiliares, los mismos también deben estar documentados.
2. Un informe, en un archivo llamado 'informe.pdf', analizando los problemas en eficiencia en los predicados sobre matrices y los casos sin solución y problemas de eficiencia que haya encontrado en los ejercicios de cuadrado latino y greco-latino. Se sugiere utilizar el predicado time/1 SWI-Prolog (notar que también existen profile/1 y statistics/2).