Programación Lógica

Obligatorio 1 – 2012

Facultad de Ingeniería
Instituto de Computación
Grupo de Procesamiento de Lenguaje Natural

El objetivo de este obligatorio es entrenarse en la implementación de predicados en Prolog de nivel fácil a intermedio.

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, de otros años, de cualquier índole, o hacer público código a través de cualquier medio (news, correo, papeles sobre la mesa, etc.).

Forma de entrega

La entrega se realizará a través del espacio eva del curso, en una página destinada a tal fin que se habilitará cerca de la fecha de entrega. Se debe entregar un solo archivo '`grupo#.pl`', donde **#** es el número de grupo que realiza la entrega, conteniendo los ítems indicados en el apartado **Entregable**.

Fecha de entrega

Los trabajos deberán ser entregados siguiendo el procedimiento descrito anteriormente antes del domingo 13 de mayo de 2012 a las 23:30 horas, sin excepciones. No se aceptará ningún trabajo pasada la citada fecha.

Observaciones

La implementación debe realizarse de manera que pueda ser ejecutada en la plataforma SWI-Prolog, en sus versiones para Windows o Linux.

Predicados a implementar

El presente obligatorio tiene como objetivo ejercitarse en cuanto a la implementación de programas Prolog, mediante la construcción de una serie de predicados.

Los predicados se dividen en tres grupos de complejidad creciente. Salvo que se especifique lo contrario, la eficiencia en la ejecución de los mismos no se considerará un aspecto eliminatorio. Sin embargo, se valorará positivamente implementaciones que sean simples y claras, y a la vez eficientes. Este aspecto es importante ya que parte de la implementación podrá ser utilizada para el siguiente obligatorio.

1) Predicados simples sobre listas:

`nth(+Pos,?L,?E)`
E es el elemento en la posición Pos (contando a partir de 1) de la lista L.

`choose(+N,?L,?Sub)`
Sub es una subsecuencia de largo N de la lista L.

`remove_alter(?L,?R)`
R es la lista producto de borrar alternadamente la lista L, comenzando por el primero.

`insert_in_order(+X,+L,?R)`
R es el resultado de insertar ordenadamente el elemento X en la lista ordenada L

2) Predicados avanzados sobre listas:

`nth_with_merge(+Pos,+L1,+L2,?E)`
E es el elemento en la posición Pos de la lista producto de combinar ordenadamente las listas ordenadas L1 y L2.

`member_sorted(+X,+L)`
X es un elemento de la lista ordenada L, no se debe recorrer la lista innecesariamente.

`permutation(+X,?Y)`
La lista Y es una permutación de los elementos de la lista X.

`quicksort(+L,?S)`
S es la lista ordenada de L utilizando el algoritmo quicksort.

`insertion_sort(+L,?S)`
S es la lista ordenada de L utilizando el algoritmo de insertion sort: obtener cada elemento de la lista L e insertarlo ordenadamente en una nueva lista.

3) Predicados sobre matrices:

`matrix(+X,+Y,?M)`
M es una matriz de X filas e Y columnas.
La matriz se representa mediante una lista de X filas, donde cada fila es una lista de Y celdas.

`cell_at(+X,+Y,?M,?P)`
P es la celda en la fila X y columna Y de la matriz M, las filas y columnas comienzan en 1.

`select_column(?M,?C,?MRest)`
C es la primera columna de la matriz M, MRest es la matriz M sin su primera columna.

`latin_square(+N,?Resul)`
La matriz Resul representa un cuadrado latino de orden N.
Un cuadrado latino de orden N es una matriz compuesta enteramente por enteros entre 1 y N, y cuyas filas y columnas no contienen elementos repetidos.

Entregable

El archivo a entregar debe contener:

1. Implementación de todos los predicados solicitados.
2. Cada predicado debe contar con comentarios documentando su funcionamiento. Si se implementan predicados auxiliares, los mismos también deben estar documentados.