PROGRAMACIÓN DECLARATIVA

LABORATORIO 3 (Más sobre listas, aritmética y computación simbólica)

Para medir lo aprendido durante la Práctica 3, los alumnos deberán enviar los programas y soluciones a las preguntas que a continuación se indican. El envío se realizará en un archivo ZIP de nombre:

gN_Apellidos_Lab3.ZIP

donde N es el número del equipo de trabajo. El archivo ZIP contendrá los ficheros:
Apellidos_Lab3.pl con el código Prolog de todos los programas solicitados.
Apellidos_Lab3.txt con los datos personales de la persona que hace el envío y el enunciado y solución de aquéllas preguntas que (no pudiéndose responder vía programa) se formulen en cada uno de los ejercicios del laboratorio.

En todos los casos, "**Apellidos**" son los apellidos del alumno que hace el envío.

Ejercicio 9.

Defina un predicado partir(L, L1, L2) que divida la lista L en dos partes L1 y L2, tales que los elementos de L1 son menores o iguales que un cierto elemento N perteneciente a L y los de L2 son mayores que ese elemento N. El elemento N seleccionado no se incluye en las listas partidas L1 y L2.

Ejercicio 10. (Ordenación rápida—quicksort—)

El algoritmo de ordenación rápida aplica la estrategia de "divide y vencerás" a la tarea de ordenar una lista. La idea es particionar una lista con respecto a uno de sus elementos (en principio elegido al azar), llamado el pivote, de forma que los elementos menores o iguales que el pivote queden agrupados a su izquierda, en una de las listas, y los elementos mayores que el pivote queden agrupados a su derecha, en la otra lista. Observe que, tras la partición, lo único seguro es que el pivote está en el lugar que le corresponderá en la ordenación final. Entonces, el algoritmo se centra en la ordenación de las porciones de la lista (que no están necesariamente ordenadas), lo que nos remite al problema original. Utilizando el predicado partir(L, L1, L2) del Ejercicio 10, dé una implementación recursiva del algoritmo de ordenación rápida, mediante la definición de un predicado: quicksort(Lista, ListaOrdenada).

Ejercicio 11. (Mezcla ordenada —merge_sort—)

Implemente en Prolog el algoritmo de mezcla ordenada para la ordenación de una lista de elementos, mediante la definición de un predicado: merge_sort(Lista, ListaOrdenada). Informalmente, este algoritmo puede formularse como sigue: Dada una lista, divídase en dos mitades, ordene cada una de las mitades y, después, "mezcle" apropiadamente las dos listas ordenadas obtenidas en el paso anterior.

Ejercicio 12.

El polinomio $Cnx^n + : : : + C2x^2 + C1x + C0$, donde Cn; ... ;C1;C0 son coeficientes enteros, puede representarse en Prolog mediante el siguiente término:

cn * x ** n + ... + c2 * x ** 2 + c1 * x + c0.

El operador "**" es un operador binario infijo. Cuando se evalúa la expresión "X ** n", computa la potencia enésima de X. Observe que el operador "**" liga más que el operador binario infijo "*", que a su vez liga más que el operador binario infijo "+" (por lo tanto no se requiere el uso de paréntesis). Observe también que, en la representación anterior, la variable x se trata como una constante. Defina un predicado eval(P, V, R) que devuelva el resultado R de evaluar un polinomio P para un cierto valor V de la variable x. A modo de ejemplo, el objetivo ?- eval(5 * x ** 2 + 1, 4, R). debe tener éxito con respuesta R = 81.

Ejercicio 13.

Utilizando la representación de los polinomios propuesta en el ejercicio anterior, defina un predicado d(P, D) que compute la derivada D de un polinomio P con respecto a x. Se requiere que el polinomio D, que se obtiene como resultado, se presente en un formato simplificado. Esto es, si lanzamos el objetivo:

?- d(2 * x ** 2 + 3, D).

debe devolver la respuesta "D = 4 * x " y no "D = 2* 2 * x ** 1 + 0".

Última modificación: jueves, 13 de marzo de 2014, 19:23:23