PROGRAMACIÓN DECLARATIVA

LABORATORIO 6 (Estructuras de control y Entrada/Salida)

Para medir lo aprendido durante la Práctica 6, 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_Lab6.ZIP

donde N es el número del equipo de trabajo. El archivo ZIP contendrá los ficheros:
Apellidos_Lab6.pl con el código Prolog de todos los programas solicitados.
Apellidos_Lab6.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.

ENUNCIADO

Ejercicio 22.
Estudiar el comportamiento del programa

```
p(X) - q,r(X). t.
p(X) - s(X),t. a.
q - a,!,b. b :- fail.
q - c,d. c.
r(uno). d.
s(dos).
```

a) Realizar una traza y generar el árbol correspondiente al objetivo "?- p(X).''. b) Eliminar el corte del programa y repetir la experiencia. ¿Qué ramas fueron podadas?. c) ¿ El corte introducido en el programa es rojo o verde ?.

Observaciones.
a) Se dice que un corte introducido en un programa es verde, cuando no altera su significado declarativo. Esto es, el programa produce las mismas respuestas con y sin el corte. Por el contrario, si la introducción del corte altera el significado declarativo de un programa se dice que el corte es rojo.
b) Dado que la introducción del corte puede alterar el significado declarativo de un programa debe usarse con cuidado. La regla es, evitar el corte siempre que sea posible.

Ejercicio 23.
Definir un predicado add(X, L, L1) que añada un elemento X, a la lista L para dar L1, sin incurrir en repeticiones de elementos. Esto puede hacerse mediante la siguiente regla

Si X es miembro de la lista L entonces L1 = L,
sino L1 es igual a la lisa [X | L];

para cuya implementación es necesario el empleo del corte. Comprobar que omitiendo el corte o alguna construcción derivada del corte (e. g. el predicado not) sería posible la adición de elementos repetidos. Por ejemplo, llame a la nueva versión sin corte "add2" y compruebe que

?- add2(a, [a, b, c], L).
L = [a, b, c];
L = [a,a, b, c]

Así pues, en la solución de este ejercicio es vital el empleo del corte para obtener el significado esperado para nuestro predicado y no como un mero instrumento para aumentar la eficiencia.

Ejercicio 24.
Queremos definir un predicado diferente(X, Y) que sea verdadero cuando X e Y no unifican. Esto puede decirse en PROLOG teniendo en cuenta que

Si X e Y unifican entonces diferente(X, Y) falla,
sino diferente(X, Y) tiene éxito.

Definir este predicado empleando a) el corte, fail y true (llame a esta versión

"diferente_a"); b) empleando únicamente not (llame a esta versión "diferente_b").

Ejercicio 25.
El predicado predefinido repeat no es imprescindible en la programación con el lenguaje PROLOG. a) Para confirmar el anterior aserto, escribe un pequeño programa llamado cubo que solicite un número por teclado y calcule su cubo. El programa debe operar como a continuación se indica

?- cubo.
Siguiente número 5.
El cubo de 5 es 125
Siguiente número 3.
El cubo de 5 es 27
Siguiente número stop.
yes

b) Dar una versión empleando repeat (llame a esta versión "cubo_2").

Ejercicio 26.
Definir un predicado “cont(Fichero, Carac, Palab, Lineas)” que tome como argumento el nombre de un fichero de texto y cuente los caracteres, las palabras y las lineas contenidas en ese fichero. Como efecto lateral, el predicado cont/4 deberá imprimir el número de caracteres, palabras y líneas, procurando dar un formato agradable a la salida por pantalla.

Última modificación: jueves, 13 de marzo de 2014, 19:27:27