Primera práctica (Programación lógica pura)

Fecha de entrega: viernes 19 de abril de 2024
SUPERFICIES CARGADAS

Antes de empezar la práctica es **muy importante** empezar por leer las **Instrucciones generales para la realización y entrega las prácticas**, así como la **Preguntas frecuentes sobre Ciao, Deliverit, LPdoc, etc.**, disponibles en Moodle.

Instrucciones específicas

En esta práctica ponemos en práctica algunos de los conceptos estudiados en el bloque temático de programación lógica pura. Por ello, además de las instrucciones generales, como instrucciones específicas para esta práctica se deben usar sólo cláusulas (reglas y hechos), términos (constantes, variables, estructuras), y la unificación (términos en los argumentos, o llamadas a `=/2`), y la declaración inicial de módulo y el hecho `author_data/4` de las instrucciones generales para prácticas. No está permitido utilizar los predicados predefinidos de ISO Prolog (p.ej., aritmética, predicados metalógicos, predicados de inspección de estructuras, corte, negación por fallo, etc.). Tampoco está permitido utilizar las librerías del sistema (p.ej., las de predicados sobre listas, árboles, etc.). Si fuese necesario utilizar alguno de estos predicados, debéis programarlo vosotros mismos. Por ejemplo, si se necesitase definir la pertenencia a una lista mediante el predicado `member/3`, debéis programar una versión propia del predicado `member/3` a la que deberéis dar un nombre diferente (p.ej., `my_member/3`) para evitar el uso por defecto del predicado predefinido `member/3`. Se deberá tener especial cuidado con esto, dado que hay predicados predefinidos como `length/2` (que calcula la longitud de una lista) que utilizan aritmética de ISO Prolog en lugar de aritmética de Peano (que es la que se puede y debe utilizar en este ejercicio). Para aseguraros de que no usáis ninguno de estos predicados debéis usar el paquete `pure` en la declaración de módulo:

```
:- module(_,_,[pure,assertions,regtypes]).
```

---

1. ENUNCIADO DE LA PRÁCTICA (8.5 puntos)

En esta práctica representaremos superficies que están cargadas estáticamente, con diferentes valores de carga en cada punto de dichas superficies. Para simplificar el problema, discretizaremos varios aspectos. En primer lugar consideraremos que las cargas sólo pueden tomar los valores definidos por el siguiente predicado:

```
charge( 0 ).
charge( + ).
charge( ++ ).
charge( +++ ).
charge( ++++ ).
charge( +++++ ).
charge( ++++++ ).
charge( +++++++ ).
```

donde el valor de las cargas está representado por el número de símbolos '+'. Por ejemplo, la constante '++++' representa una carga de cuatro unidades de carga.

Por otra parte, discretizaremos también las dos dimensiones de la superficie, de manera que la consideramos formada por una malla de células, donde cada célula puede tomar un valor de carga de los definidos arriba. Un ejemplo de una superficie cargada puede ser el siguiente:

```
  +++     +++++++      0        +     ++++    0
+++++++     +++        0        +     ++++    0
  +++         0      +++++++    +       0    ++++
  +++     +++++++      0      ++++    +       0
+++++++       0        +++      +     ++++    0
  +++     +++++++      +++      +       0     +
+++++++     +++        0        +     ++++    0
```

Para representar estas superficies utilizaremos una lista de listas. Las listas interiores representarán de izquierda a derecha las células en la dimensión horizontal de la superficie y la lista exterior agrupará las diferentes líneas de células. Los valores almacenados en cada elemento de las listas son las cargas. Con esta representación, la anterior superficie se representa con la siguiente estructura:

```
[ [ +++ , +++++++ , 0 , + , ++++ , 0 ],
  [ +++++++ , +++ , 0 , + , ++++ , 0 ],
  [ +++ , 0 , +++++++ , + , 0 , ++++ ],
  [ +++ , +++++++ , 0 , ++++ , + , 0 ],
  [ +++++++ , 0 , +++ , + , ++++ , 0 ],
  [ +++ , +++++++ , +++ , + , 0 ,  + ],
  [ +++++++ , +++ , 0 , + , ++++ , 0 ] ]
```

Las superficies pueden tener un número arbitrario células en ambas dimensiones.

Las tareas a realizar son las siguientes. Las listas en las respuestas deben mantener el orden en el que aparecen las células en las listas de la estructura que describe la superficie.

1. Definir el tipo:

   `basic_surface(S):` **S** *es una superficie que contiene cargas. Debe tener al menos una línea y cada línea al menos una célula.*

2. Definir el tipo:

   `surface(S):` **S** *es una superficie como la anterior pero además todas las líneas deben tener el mismo número de células.*

3. Definir el predicado **`h_line/3`** tal que:

`h_line(S,N,C): C` *es la línea horizontal* `N`*-ésima de la superficie* `S` *(la lista con todas las células de esa línea).* `N` *es un número natural en representación de Peano. La primera línea es la 1 (es decir, la* `s(0)`*).*

4. Definir el predicado `v_line/3` tal que:

   `v_line(S,N,C): C` *es la lista formada por las células* `N`*-ésimas de todas las líneas horizontales de la superficie* `S`*. Es decir, si* `S` *está instanciada a la superficie ejemplo de arriba, la consulta* `v_line(S,s(s(s(0))),C)` *respondería* `C = [ 0, 0, +++++++, 0, +++, +++, 0]`.

5. Definir el predicado `v_lines/2` tal que:

   `v_lines(S,C): C` *es la lista de las lineas* verticales *de células de la superficie* `S`.

6. Definir el predicado `total_charge/2` tal que:

   `total_charge(S,T): T` *es la suma de todas las cargas de la superficie* `S`.

7. Definir el predicado `average/2` tal que:

   `average(S,A): A` *es la media de todas las cargas de la superficie. Es decir, la suma de todas las cargas (predicado anterior), dividido por el número total de celdas. El resultado se debe dar redondeado por truncado, es decir, devolviendo el natural anterior.*

---

2. MANUAL (1.5 puntos)

Ver las instrucciones generales para las prácticas. Los predicados deben ir documentados insertando en el código aserciones y comentarios del lenguaje Ciao Prolog de manera que se genere un manual automáticamente a partir de dicho código, usando la herramienta `lpdoc`. Esta herramienta está incluida en la instalación de Ciao Prolog y se puede también correr en el playground o desde Emacs o VSC.

---

3. PUNTOS ADICIONALES (para subir nota):

Ejercicio complementario en *codificación de casos de prueba*: Se valorará el uso de aserciones `test` que definan casos de prueba para comprobar el funcionamiento de los predicados. Estos casos de test se deben incluir en el mismo fichero con el código.

Hemos dejado en Moodle instrucciones específicas sobre cómo hacer todo ello y un ejemplo de código que tiene ya comentarios y tests, para practicar corriendo `lpdoc` sobre él y ejecutando los tests.