Introducción a la Programación Lógica: PROLOG

UNIVERSIDAD DE VALLADOLID

Índice:

Introducción (I)

- Basado en la lógica de Primer Orden.
- Resolver problemas formulados como una serie de objetos y relaciones entre ellos: base de conocimiento
- Ejemplo: *Juan tiene un libro*
  - Objetos: "Juan" y "libro"
  - Relación: "tiene" (predicado)
  - Importancia del orden

```
tiene(juan, libro).
```

Introducción (II)

- Hecho: propiedad de un objeto
  - Ejemplo: *"el diamante es valioso"*

```
valioso(diamante).
```

- Reglas: forma de describir relaciones
  - Ejemplo: *"Dos personas son hermanas, si ambas son hembras y tienen los mismos padres"*
- Programación en PROLOG:
  - Declarar hechos sobre objetos y sus relaciones
  - Definir reglas sobre objetos y sus relaciones
  - Hacer preguntas sobre objetos y relaciones:
    - Trabajar contra una base de (datos) conocimiento

Hechos (I)

- Ejemplo:
  - Objetos: *Juan, Pepe, vino y cerveza*
  - Relación: A *<objeto>* le gusta *<objeto>*
  - Hechos: A *Juan* le gusta el *vino*
    A *Pepe* le gusta la *cerveza*
  - PROLOG:

```
a_le_gusta (juan, vino).
a_le_gusta (pepe, cerveza).
```

Hechos (II)

- Los nombre de objetos y relaciones comenzarán por minúsculas
- Primero la relación y después los objetos a modo de argumentos
- Se permite "_" para separar caracteres
- Al final debe aparecer un punto
- Coherencia con el orden de los objetos

  A Juan le gusta el vino ≠ Al vino le gusta Juan

Hechos (III)

- Admite predicados (relaciones) monádicos o poliádicos:

`valioso(diamante).`	*El diamante es valioso.*
`hembra(ana).`	*Ana es una hembra.*
`tiene(juan, oro).`	*Juan tiene oro.*
`padre(juan, maria).`	*Juan es el padre de María.*
`dar(juan, libro, maria).`	*Juan da un libro a María*

Preguntas (I)

- Formalmente igual que un hecho precedido del símbolo "?"
- En primera instancia, busca en la base de conocimiento, si el hecho inquirido está almacenado.
- Ejemplo:

Base de Conocimiento	Preguntas	Resp.
`gusta(jose, maria).`	`?- gusta(maria, jose).`	No
`gusta(maria, libro).`	`?- gusta(maria, libro).`	Yes
`gusta(juan, coche).`	`?- gusta(juan, pescado).`	No
`gusta(jose, pescado).`	`?- gusta(jose, pescado).`	Yes

Preguntas (II)

- Introducción de la base de conocimiento en SWI-PROLOG:
  - Doble click en el fichero con la extensión"pl"
- La respuesta "no" significa que "no" hay coincidencia con hechos almacenados en la base.
- Para averiguar qué le gusta a Maria, hay que recurrir a las variables.
  - ?- gusta(X, maria).

Variables (I)

- Formalmente comienzan por mayúscula
  - Instanciada: con un valor asignado
  - No instanciada: sin valor a priori
- Ejemplo:
  - ¿qué le gusta a Juan?

    ?- gusta(juan, X). *X está no instanciada*

    X = coche ; *X está instanciada con valor "coche"*
  - ¿qué le gusta a José?

    ?- gusta(jose, X). *X adquiere el primer valor encontrado*

    X = maria ; *pulsar sucesivamente ";" para pasar al siguiente*

    X = pescado ;

Variables (II)

- PROLOG localiza el primer hecho coincidente en el predicado y con los mismos argumentos literales. El sentido de búsqueda es de arriba abajo.
- En la variable almacenará el objeto, cuya posición coincida con ésta.
- Marca el hecho en la base y muestra la variable, ahora sí, instanciada: X=maria
- Si pulsa [RETURN], se detiene la búsqueda dándose por satisfechos con la respuesta.
- Si se teclea ";", X pasa a no instanciada y se repetiría el proceso pero empezando en la marca establecida en la última instancia.
- A esto se le denomina "resatisfacer" la pregunta.

Conjunciones (I)

- Ejemplo:

Base de Conocimiento
gusta(maria, comida).
gusta(maria, vino).
gusta(juan, vino).
gusta(juan, maria).

- Pregunta: ¿Juan y María se gustan?
  - ¿A Maria le gusta Juan? Y ¿A Juan le gusta Maria?

    ?- gusta(juan, maria), gusta(maria, juan).

Conjunciones (II)

- PROLOG intenta satisfacer el primer objetivo: ¿A María le gusta Juan?
- Si fuera así, establece una marca en la base y probará a satisfacer el segundo objetivo: ¿A Juan le gusta María?
- Si también resultase satisfactorio pone otra marca en la base, por si hubiera que "resatisfacer" a continuación.
- Nota: cada objetivo establece sus marcas.

Conjunciones (III)

- ¿Hay algo que les guste a Juan y a María también?

  ?- gusta(maria, X), gusta(juan, X).
  X=vino;
- Pasos:

gusta(maria, comida).	*X=comida; Marca.*
?- gusta(juan, X=comida).	*No. Resatisfacer 1º obj.*
gusta(maria, vino).	*X=vino; Marca.*
?- gusta(juan, X=vino).	*Yes. Marca. Mostrar X.*

- Los objetivos se van satisfaciendo de izda. a dcha.
- Supone un proceso de instanciar y desinstanciar, satisfacer y resatisfacer objetivos: reevaluación (backtracking – próximos capítulos)

Reglas (I)

- Para declarar un hecho dependiente de otros.
- Ejemplo:

  X es un pájaro si:
  X es un animal, y
  X tiene plumas.
- Formalmente se compone de:

  <Cabeza> :- <cuerpo>
  <cuerpo> = hechos con conjunciones y variables
- Ejemplo:

  Juan es amigo de los amantes del fútbol
  Juan es amigo de X, si a X le gusta el fútbol
  gusta(juan, X) :- gusta (X, futbol)

Reglas (II)

- Ejemplo:

Base de Conocimiento
varon(alberto). varon(eduardo). hembra(alicia). hembra(victoria). padres(eduardo, victoria, alberto). padres(alicia, victoria, alberto). hermana_de(X,Y):- hembra(X), padres(X,M,P), padres(Y,M,P).

Preguntas
?- hermana_de(alberto, eduardo). No
?- hermana_de(alicia, victoria) No
?- hermana_de(alicia, eduardo). Yes
?– hermana_de(alicia, X). X=eduardo; X=alicia
?- hermana_de(X,eduardo). X=alicia

Reglas (III)

- Al igual que en las conjunciones:
  - Proceso de instanciar y desinstanciar, satisfacer y resatisfacer de izquierda a derecha.

Base de Conocimiento
varon(alberto). varon(eduardo). hembra(alicia). hembra(victoria). padres(eduardo, victoria, alberto). padres(alicia, victoria, alberto). hermana_de(X,Y):- hembra(X), padres(X,M,P), padres(Y,M,P).

?– hermana_de(alicia, X).
X=eduardo;
X=alicia

→ hermana_de(X=alicia,Y(=X)):-
?- hembra(alicia). YES
?- padres(alicia, M, P). M=victoria P=alberto
?- padres(Y, victoria, alberto). Y=eduardo
?- padres(Y, victoria, alberto). Y=alicia

?- hermana_de(X,eduardo).
X=alicia

→ hermana_de(X,Y=eduardo):-
?- hembra(X). X=alicia
?- padres(alicia, M, P). M=victoria P=alberto
?- padres(Y, victoria, alberto). Y=eduardo, M=victoria, P=alberto

?- hembra(X). X=victoria
?- padres(victoria, M, P). NO

Reglas (IV)

- Dar información a PROLOG sobre predicados:

  hechos o reglas (=cláusulas)

```
/* 1 */ ladron(juan).
/* 2 */ gusta(maria,comida).
/* 3 */ gusta(maria, vino).
/* 4 */ gusta(juan,X):- gusta(X,vino).
/* 5 */ puede_robar(X,Y) :-
            ladron(X),
            gusta(X,Y).

?- puede_robar(juan,X).
X=maria;
```

- Comentarios igual que en C
- Un mismo tipo de definición mediante tres cláusulas:
  /*2*/, /*3*/ y /*4*/
  Dos hechos y una regla.

Ejercicios (I)

- Modificar la regla *hermana_de* para que una misma persona no sea hermana de sí misma.

  Para ello, téngase presente que una relación que refleja que dos objetos son distintos recurre al operador /=

  juan \= pedro

- Pedro y Maria se casan y tienen tres hijos: Juan, Margarita y Paula. A su vez, cada uno de ellos se casa con Virginia, José y Manuel respectivamente. Como consecuencia, Juan y Virginia tienen dos hijos: Isabel y Fernando. Por su parte, José y Margarita solamente tienen a Felipe. Finalmente Paula y Manuel son padres de Javier, Eva y Alicia.

Ejercicios (II)

- Constrúyase el árbol genealógico a base de las relaciones:

  progenitores_de(X, Y, Z) /* X e Y son progenitores de Z*/
  matrimonio(X, Y) /* X e Y están casados */

- Obtener las reglas para definir las relaciones:

```
padre_de (X, Y) / madre_de(X, Y)                 /* X es padre/madre de Y */
hijo_de(X, Y) / hija_de(X, Y)                    /* X es hijo/a de Y */
hermano_de(X, Y) / hermana_de(X, Y)              /* X es hermano/a de Y */
tio_de(X, Y) / tia_de(X, Y)                      /* X es tío/a de Y */
sobrino_de(X, Y) / sobrina_de (X, Y)             /* X es sobrino/a de Y */
primo_de(X, Y) / prima_de(X, Y)                  /* X es primo/a de Y */
cunhado_de(X, Y) / cunhada_de(X, Y)              /* X es cuñado/a de Y */
abuelo_de(X, Y) / abuela_de(X, Y)                /*X es abuelo/a de Y*/
```