Prolog

Uso de estructuras: programas ejemplo

David Gelpi Fleta

[correo]

Laboratorio de Introducción a los Sistemas Informáticos Inteligentes

Escuela Superior de Ingeniería Informática - Universidad Vigo

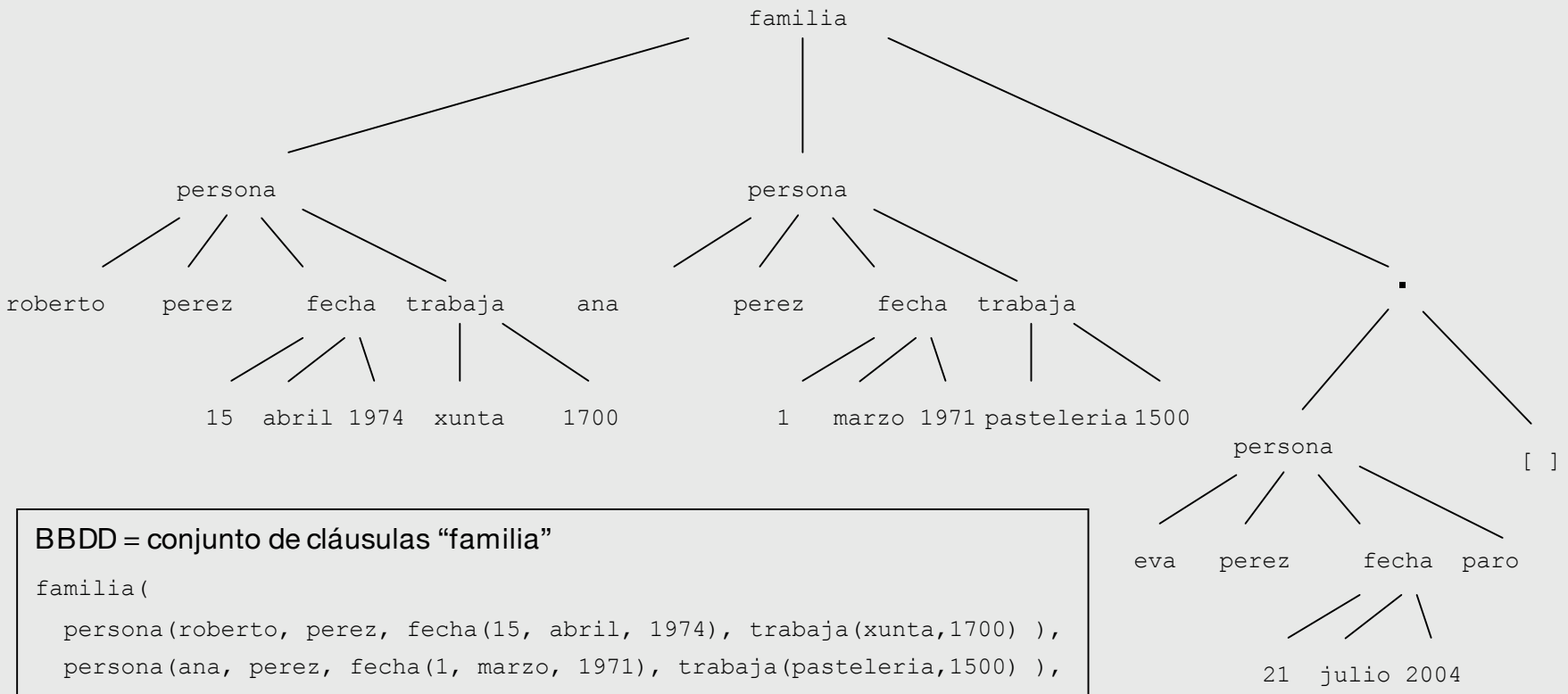
Nota:

Exposición basada en el capítulo 4 del libro Prolog: Programming for Artificial Intelligence de Ivan Bratko - Ed. Pearson - contenido en la bibliografía recomendada de la asignatura.

Contenido

- Bases de datos como ejemplo de información estructurada.
- Abstracción de datos.
- Selectores
- Problema del viajante de comercio.

Una base de datos en Prolog



BBDD = conjunto de cláusulas “familia”

```
familia(
  persona(roberto, perez, fecha(15, abril, 1974), trabaja(xunta,1700) ),
  persona(ana, perez, fecha(1, marzo, 1971), trabaja(pasteleria,1500) ),
          [ persona(eva, perez, fecha(21, julio, 2005), paro) ]).
```

familia(estructura(…, estructura(…), …), estructura (…), lista[ ]).

Prolog como lenguaje de consultas de BBDD

- Prolog permite referirnos a un objeto sin especificar sus componentes individuales: indicamos sólo la estructura.

  Ejemplos:

  - `?- familia( persona(_, porto,_,_),_,_)`

    devuelve todas las familias de apellido “porto”

  - `?- familia(_,_, [_,_,_ ])`

    devuelve todas las familias con tres hijos

  - `?- familia(_, persona(Nombre, Apellido,_,_), [_,_,_ ])`

    devuelve el nombre y el apellido de todas las mujeres casadas que posean tres hijos.

- Así, podemos definir varios procedimientos para trabajar con las estructuras contenidas en la BBDD “familia”:

```
marido(X):- familia(X, _, _).
esposa(X):- familia(_,X,_).
hijo(X):- familia(_,_,Hijos), member(X,Hijos).
existe(Persona):- marido(Persona) ;
                  esposa(Persona) ;
                  hijo(Persona).
**cumpleaños(persona(_,_,Fecha,_),Fecha).
```

Abstracción de datos

- Trataremos de organizar varias piezas de información en unidades naturales a ser posible de manera jerárquica.
- La información así estructurada debe poseer significado conceptualmente.
- Cada unidad de información debería ser fácilmente accesible en el programa.
- Los detalles de la implementación de la estructura serán invisibles para el usuario de la estructura.
- El programador piensa en objetos y relaciones entre ellos, no en cómo está representada la información.
- ¿Cómo se lleva a cabo esto en Prolog?:

  Ejemplo BBDD “familia”:

  cada familia es una colección de piezas de información

  estas piezas se dividen en unidades naturales como persona o familia

  pueden ser tratadas como objetos simples

Selectores

- Un selector es una relación que permite acceder a objetos sin conocer los detalles sobre cómo está representada la información.
- En nuestra BBDD "familia" permitirá acceder a componentes particulares de una familia sin conocer los detalles de la figura de la transparencia 4.

Selector:

selector(Objeto, Selección)

- *selector* es la relación selector
- *Objeto* es el objeto que contiene a
- *Selección* ,el componente seleccionado.
- El nombre del *selector* será el mismo que el del *Componente seleccionado*

Ejemplo:

```
esposa( familia(_,Esposa,_), Esposa).
hijos( familia(_,_,ListaHijos), ListaHijos).
```

Mejoras en la implementación de programas

Podemos escribir un programa mediante el conjunto de hechos que representan la información (bbdd) más los selectores que permitan acceder a la información.

- El uso de selectores en un programa permite abstraernos de cómo está representada la información estructurada.
- Para manipular la información es suficiente con conocer las relaciones definidas por los selectores.
- Los programas que emplean selectores resultan más fáciles de modificar:

  al variar la representación de la información (base de conocimiento) no es necesario modificar la manera de acceder a ella (selectores).
- Variar la representación de la información puede mejorar la eficiencia del programa.

Problema del viajante de comercio (TSP)

- Este programa se emplea para planificar viajes en avión.
- Contesta a cuestiones como:
  - ¿Qué días de la semana existe vuelo directo por la tarde entre dos determinadas ciudades?
  - ¿Cómo puedo viajar entre dos ciudades el martes?
  - Si quiero visitar tres ciudades en determinado orden, empezando el martes y retornando el viernes, ¿qué ruta debo seguir si sólo dispongo de un vuelo al día?
- Programa = BBDD + Selectores.
  - BBDD = conjunto de cláusulas "horario" que contienen la información sobre los vuelos.
  - Selectores: cómo consultar la información.
- Pasos en la implementación del problema:
  1. Definir la BBDD con los horarios de los vuelos.
  2. Establecer qué consultas (selectores) extraen la información de la BBDD.

TSP: BBDD

Definir la BBDD con los horarios de los vuelos: representación de la información.

Programa

BBDD
(representación)

+

Selectores
(acceso)

```
horario(Lugar1, Lugar2, ListaVuelos).
```

ListaVuelos = [Vuelo1, Vuelo2, …, Vuelon]
lista de items estructurados, separados por el operador "/"

```
LitaVuelos =
    [HoraSalida/ HoraLlegada/ NumVuelo/ ListaDias, ...]
```

HoraSalida
estructura de dos componentes separados por el operador ":"

```
HoraSalida = 9:40
```

ListaDias
lista de días o el átomo "todos"

```
ListaDias = [lu, ma, mi, ju, vi, sa ]
```

Cláusula horario:

```
horario( edimburgo, londres,
    [9:40 / 10:50 / ba4733 / todos,   % Vuelo1
    13:40 / 14:50 / ba4773 / todos,   % Vuelo2
    19:40 / 20:50 / ba4833 / [lu,ma,mi,ju,vi,sa]
     ]).
```

TSP: selectores

Definir los selectores del programa: cómo acceder a la información

- El principal problema es encontrar *rutas* entre dos ciudades dadas en un determinado día de las semana.

```
selector( Objeto, Selección)
```

```
ruta( Lugar1, Lugar2, Dia, Ruta)
```

- Una *ruta* (selector, relación) es una secuencia de *vuelos* que satisfacen los siguientes criterios:
  - El origen de la ruta es `Lugar1`
  - El destino de la ruta es `Lugar2`
  - Todos los vuelos se encuentran en el mismo día de la semana, `Día`.
  - Todos los vuelos en `Ruta` están en la relación `horario`.
  - Existe suficiente tiempo para realizar el trasbordo entre dos vuelos si la ruta consta de varios vuelos.
- Objeto Ruta: lista de objetos estructurados separados por el operador "/"

```
Ruta = [De/ Hacia/ NumVuelo/ HoraSalida ]
```

TSP: selectores

- La relación ruta entre dos ciudades consta de dos reglas:
  - si existe vuelo directo:

```
ruta( Lugar1, Lugar2, Dia, [Lugar1/ Lugar2/ NumVuelo/ HoraSalida ]) :-
            vuelo(Lugar1, Lugar2, Dia, NumVuelo, HoraSalida, HoraLlegada).
```

  - si existe vuelo indirecto entre las ciudades:

```
ruta( Lugar1, Lugar2, Dia, [Lugar1/ Lugar3/ NumVuelo/ HoraSalida|
    RestoRuta ]) :-
    ruta(Lugar3, Lugar2, Dia, RestoRuta),
    vuelo(Lugar1, Lugar2, Dia, NumVuelo1, HoraSalida1, HoraLlegada1),
    salida(RestoRuta, HoraSalida2),
    trasbordo(HoraLlegada1, HoraSalida2).
```

- Ahora debemos definir las relaciones que componen el selector `ruta`.

TSP: resto de relaciones

```
% Selector ruta y relaciones

vuelo( Lugar1, Lugar2, Dia, NumVuelo, HoraSalida, HoraLlegada):-
    horario( Lugar1, Lugar2, ListaVuelos),
    member(HoraSalida/ HoraLlegada/ NumVuelo/ ListaDias,
           ListaVuelos),
    diaVuelo( Dia, ListaDias).

diaVuelo( Dia, ListaDias):-
   member( Dia, ListaDias).

diaVuelo( Dia, todos):-
    member(Dia,[lu,ma,mi,ju,vi,sa]).

horaSalida( [P1/ P2/ NumVuelo/ Salida | _ ], Salida).

trasbordo( Horas1:Minutos1, Horas2:Minutos2):-
   60 * (Horas2 - Horas1) + Minutos2 - Minutos1 >= 40.

member( X, [X | L ] ).

member( X, [Y | L ] ):-
   member( X, L).
```

TSP: BBDD

```
% Base de datos con los vuelos
timetable( edinburgh, london,
           [9:40 / 10:50 / ba4733 / alldays,
            13:40 / 14:50 / ba4773 / alldays,
            19:40 / 20:50 / ba4833 / [mo,tu,we,th,fr,su] ] ).
timetable( london, edinburgh,
            [9:40 / 10:50 / ba4732 / alldays,
            11:40 / 12:50 / ba4752 / alldays,
            18:40 / 19:50 / ba4822 / [mo,tu,we,th,fr] ] ).
timetable( london, ljubljana,
           [ 13:20 / 16:20 / jp212 / [mo,tu,we,fr,su],
            16:30 / 19:30 / ba473 / [mo,we,th,sa] ] ).
timetable( london, zurich,
           [  9:10 / 11:45 / ba614 / alldays,
            14:45 / 17:20 / sr805 / alldays ] ).
timetable( london, milan,
           [  8:30 / 11:20 / ba510 / alldays,
            11:00 / 13:50 / az459 / alldays ] ).
timetable( ljubljana, zurich,
           [ 11:30 / 12:40 / jp322 / [tu,th] ] ).
timetable( ljubljana, london,
           [ 11:10 / 12:20 / jp211 / [mo,tu,we,fr,su],
            20:30 / 21:30 / ba472 / [mo,we,th,sa] ] ).
timetable( milan, london,
           [  9:10 / 10:00 / az458 / alldays,
            12:20 / 13:10 / ba511 / alldays ] ).
timetable( milan, zurich,
           [  9:25 / 10:15 / sr621 / alldays,
            12:45 / 13:35 / sr623 / alldays ] ).
timetable( zurich, ljubljana,
           [ 13:30 / 14:40 / jp323 / [tu,th] ] ).
```

TSP: cuestiones

- ¿Qué días de la semana hay vuelo directo por la tarde de Ljubliana a Londres?:

```
?- vuelo( ljuljana, london, Dia, _, HoraSalida: _, _),
HoraSalida >= 18.
      Dia = lu;
      Dia = mi;
      ...
```

- ¿Cómo puedo viajar de Ljubljana a Edimburgo en martes?:

```
?- ruta(ljubljana, edimburgo, ma, Ruta).
Ruta= [ljubljana/ zurich/ jp322/ 11:30,
       zurich/ londres/ sr806/ 16:10,
       londres/ edimburgo/ ba4822/ 18:40 ]
```