Documentació de la pràctica de PROLOG

Gener del 2012

Juan Gabriel Florit Gomila

Bernardo Miquel Riera

ÍNDEX DE CONTINGUT

1 Enunciat del problema

Lady Q, lady R i lady S el dia de la boda lluïen vestits ornats amb una flor dels famosos sastres Abreu, Bultó i Coromines: una duia una tulipa, l'altra un clavell i la tercera una rosa, però no en aquest ordre precisament. Els materials dels vestits eren la seda, l'organdí i la franel·la.

Determinau quin vestit, quina flor, quin sastre i de quin material és el vestit que correspon a cada núvia, si s'ha pogut observar que:

- Lady S no duia la rosa.
- Lady S duia el vestit d'Abreu.
- El vestit d'Abreu no era de franel·la.
- La rosa estava al vestit de Bultó.
- El vestit de Bultó és de seda.
- Lady R no duia la tulipa.
- La núvia que duia el vestit d'organdí no duia la tulipa.

2 Solució en PROLOG

La solució del problema amb PROLOG és similar a la de problemes semblants fets a classe. Consisteix en anar permutant les llistes segons ho necessitam. És a dir, feim permutacions d'una llista i en acabar, aplicam condicions per tal de descartar el major nombre de permutacions de les llistes. Així aconseguim retallar l'arbre de cridades recursives que es va formant amb PROLOG de tal manera que el cost computacional resulta menor que si permutéssim totes les llistes a l'inici. Un cop hem acabat PROLOG es troba amb l'operador CUT (“!”). D'aquesta manera evitam que ens torni la mateixa solució però permutada. Després el programa imprimeix per pantalla el resultat.

A continuació es pot veure el codi del programa (funcions auxiliars incloses):

```
misteri_misterios:-
      permutacio([lady_q,lady_r,lady_s],Dames),
      permutacio([abreu,bulto,coromines],Vestits),

      /*Lady S duia el vestit d'Abreu*/
      posicio(lady_s,Dames,P1),
      posicio(abreu,Vestits,P1),

      permutacio([tulipa,clavell,rosa],Flors),

      /*Lady S no duia la rosa*/
      posicio(rosa,Flors,P2),
      P1\=P2,

      /*La rosa estava al vestit de Bultó*/
      posicio(bulto,Vestits,P2),

      permutacio([organdi,seda,franella],Materials),
```

```
	/*El vestit de Bultó és de seda*/
	posicio(seda,Materials,P2),
	/*El vestit d'Abreu no és de franel·la*/
	posicio(franella,Materials,P3),
	P1\=P3,

	/*Lady R no du la tulipa*/
	posicio(lady_r,Dames,P4),
	posicio(tulipa,Flors,P5),
	P4\=P5,

	/*La dama del vestit fet d'organdí no porta la tulipa*/
	posicio(organdi,Materials,P6),
	posicio(_,Dames,P6),
	P5\=P6,
	!,/*evitam la permutació de la solució*/

	escriure([Dames,Vestits,Flors,Materials]).


/*FUNCIONS AUXILIARS: */

/*Escrivim el resultat del problema*/
escriure([[]|_]).
escriure(L1):-
	car(L1,LAtoms),
	LAtoms=[Dama,Vestit,Flor,Material],
	write(Dama),write(' va dur el vestit de '),
	write(Vestit),write(' amb la '),
	write(Flor),write(' i el material era '),
	write(Material),write('.'),nl,
	cdr(L1,L2),
	escriure(L2).

/*Extreu el primer element de cada llista*/
car([],[]).
car([[X|L1]|L2],[X|L3]):-car(L2,L3).

/*Torna les llistes sense el seu primer element*/
cdr([],[]).
cdr([[X|L1]|L2],[L1|L3]):-cdr(L2,L3).

/*Inserta un element dins la llista donada*/
insertar(E,L,[E|L]).
insertar(E,[X|Y],[X|Z]):-insertar(E,Y,Z).

/*Permuta una llista donada per paràmetre*/
permutacio([],[]).
permutacio([X|Y],Z):-permutacio(Y,L), insertar(X,L,Z).

/*Diu la posició on es troba un element*/
posicio(X,[X|L],1):-!.
posicio(X,[Y|L],P):-posicio(X,L,P1), P is P1+1.
```

Aquest programa torna com a solució el següent, que a més s'ha comprovat manualment que és l'únic resultat correcte:

```
lady_r va dur el vestit de bulto amb la rosa i el material era seda.
lady_s va dur el vestit de abreu amb la clavell i el material era organdi.
lady_q va dur el vestit de coromines amb la tulipa i el material era franella.
```

3 Solució en Java

La solució en Java ha resultat més costosa de realitzar que la de PROLOG. A més, el cost computacional és major degut a que l'algoritme de resolució no està del tot optimitzat (per exemple, es fan totes les permutacions al principi) per a no complicar precisament l'algoritme. S'ha intentat fer un bon disseny orientat objectes i tenim com a resultat distintes classes que interactuen entre si:

- *PracticaPROLOG.* Des d'on arranca el programa.
- *LlistesInformacio.* Classe que aglutina les llistes amb els 4 elements d'informació del problema així com mètodes per accedir-hi.
- MotorInferencia. Classe encarregada de resoldre el problema. L'algoritme segueix la mateixa filosofia que el realitzat amb PROLOG:
  1) Permutam les 4 llistes d'informació del problema (el mètode que permuta és recursiu).
  2) Després, mitjançant bucles anidats realitzam una comprovació per cada combinació possible de les 4 llistes permutades fins que es troba una solució. La comprovació també utilitza l'estratègia aplicada a PROLOG d'usar la posició dels elements dins les llistes per tal de veure si es satisfan les condicions. Quan es troba la solució es força la sortida dels bucles per a reduir el cost computacional i evitar obtenir la mateixa solució permutada.
  3) Finalment, mostram el resultat a l'usuari.

* Per a més informació anar al Javadoc del projecte de Netbeans adjunt.

** El codi Java no s'ha adjuntat al present document ja que és massa extens.

4 Conclusions: PROLOG vs Llenguatge procedimental

Com hem pogut comprovar al llarg del curs i de la pràctica, PROLOG és un llenguatge declaratiu dissenyat pel processament d'informació simbòlica, on s'indiquen un conjunt de fets coneguts i regles que descriuen les relacions entre objectes del problema. El motor d'inferència de PROLOG és l'encarregat de donar una solució emprant recursivitat (backtracking) i la unificació de patrons. Tots aquests motius fan de PROLOG un llenguatge idoni per a ser aplicat a la resolució de problemes lògics (per exemple els de IA). I és precisament en aquest àmbit on PROLOG pot ser aplicat amb comoditat. Per a altres àmbits deixa de ser indicat en detriment dels llenguatges imperatius/procedurals, on no sols es diu que es fa, sinó també com es fa.

Així doncs, l'enunciat que s'ha implementat en aquesta pràctica, al ser un problema de lògica, ha resultat força menys costosa la implementació amb PROLOG que amb el llenguatge procedural (en el nostre cas Java). Deim força menys costosa ja que hem hagut de lidiar amb una serie de problemes, tals com la falta d'experiència amb els llenguatges declaratius (i en concret de PROLOG) i la conseqüent corba d'aprenentatge, que ja tenim superada en quant a llenguatges procedurals com Java. Tot i així, ha quedat palesa la superioritat de PROLOG per a problemes de lògica.