PLC – Cours 4

Thi-Bich-Hanh Dao

Université d'Orléans

M1 Informatique

Plan

I. Coupure

Empêcher le retour en arrière

- En exploration de l'arbre de dérivation, le retour en arrière est systématique.
- Il peut causer de l'inefficacité : exploration de branches dont on sait à l'avance qu'elles ne mènent pas à une solution.
- Prolog fourni un mécanisme pour empêcher le retour en arrière : élagage dynamique de l'arbre de dérivation.

I. Coupure

Exemple 1 : le "if then else"

Le prédicat `max/3` :

```
max(X,Y,X) :- X>=Y.
max(X,Y,Y) :- X<Y.
```

- L'arbre de dérivation des buts `max(1,2,R)` et `max(2,1,R)`
- Le programme C qui implante le même algorithme serait

```
int max(int X,int Y) {
    if (X>=Y) return X;
    if (X<Y) return Y;
}
```

Comment programmer un "if then else" ?

I. Coupure

Exemple 2

- Relation entre X et Y
  - si $X < 3$ alors $Y = 0$
  - si $3 \leq X < 6$ alors $Y = 2$
  - si $6 \leq X$ alors $Y = 4$
- En Prolog

```
f(X,0) :- X<3.
f(X,2) :- X>=3, X<6.
f(X,4) :- X>=6.
```

- Les cas sont mutuellement exclusifs

I. Coupure

Exemple 2

But

```
?- f(1,Y).
```

- En choisissant la première règle, `Y` est instancié par 0.
- Après avoir donné la réponse `Y=0`, Prolog fait des retours en arrière successifs pour visiter tous les autres branches de l'arbre de dérivation.
- Cas mutuellement exclusifs ⇒ parcours inutile des autres branches.

I. Coupure

Le prédicat ! (coupure, cut)

- Dans les exemples `max/3` et `f/2`, il faudrait exprimer le fait que, lorsque un choix de valeurs des variables satisfait une certaine condition, alors on ne veut pas essayer d'autres choix.
- Dans le cas de `max`, la condition est `X>=Y`.
- Dans le cas de `f`, la condition est `X<3` pour la première règle et `X>=3,X<6` pour la seconde.
- Pour cela on place une coupure – prédicat prédéfini `!`, après la condition en question.
- Du point de vue de la sémantique déclarative, `!` équivalent à *true*.

I. Coupure

Programmes avec coupure

- Le prédicat `max/3` :

```
max(X,Y,X) :- X>=Y, !.
max(X,Y,Y) :- X<Y.
```

- Exemple 2 :

```
f(X,0) :- X<3, !.
f(X,2) :- X>=3, X<6, !.
f(X,4) :- X>=6.
```

I. Coupure

Coupure verte

- Quand on modifie un programme en utilisant la coupure, il faut que la sémantique opérationnelle reste inchangée (il s'agit d'une optimisation).
- Dans ces exemples, les coupures introduites ne changent pas la sémantique déclarative : des coupures vertes (green cut).
- L'addition et la suppression de coupures vertes dans un programme n'ont pas d'effet sur la signification du programme.

I. Coupure

Coupure rouge

- On peut reformuler la relation `max(X,Y,Z)` : si $X >= Y$ alors $Z = X$ sinon $Z = Y$
- En Prolog :

```
max(X,Y,X) :- X>=Y, !.
max(X,Y,Y).
```

- Programme plus efficace mais la sémantique déclarative a été modifiée : coupure rouge
- Si on enlève les coupures : le programme change la signification

```
max(X,Y,X) :- X>=Y.
max(X,Y,Y).
```


I. Coupure

Propriétés de la coupure (1/3)

- Une coupure élague toutes les clauses qui la suivent. Un but p unifié avec une clause c contenant une coupure qui réussirait, ne pourrait produire des solutions qui utilisent des clauses ayant une occurrence après la clause c.
- Exemple : Le prédicat `max/3` :

```
max(X,Y,X) :- X>=Y, !.
max(X,Y,Y) :- X<Y.
```

Dans l'évaluation du but `max(2,1,Z)`, puisque 2>1 réussit, la coupure fait que la seconde clause ne sera pas considérée.

I. Coupure

Propriétés de la coupure (2/3)

- Une coupure élague toutes les solutions alternatives pour la conjonction des buts qui apparaissent à sa gauche dans la clause, c-à-d un but conjonctif suivi d'une coupure produira au plus une solution.
- Exemple :

```
p(X) :- q(X), !.
q(a).
q(b).
```

Pour le but `p(X).`, Prolog réponde `X=a`. Sans la coupure les réponses seraient `X=a` et `X=b`.

I. Coupure

Propriétés de la coupure (3/3)

- En revanche, la coupure n'affecte nullement les buts à sa droite dans la clause. Ils peuvent produire plus d'une solution. Toutefois, dès que cette conjonction échoue, la recherche se poursuit à partir de la dernière alternative qui précède le choix de la clause contenant la coupure.
- Exemple : programme 17-cut.pl

I. Coupure

Exemple

Relation membre d'une liste

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

- Coupure verte ou rouge ?
- Solutions du but `membre(X,[1,2,3])` ?

Plan

II. Clauses et buts disjonctifs

- La clause

  `tete :- p1 ; p2 ; ... ; pn .`

  est équivalente à la séquence de clauses :

  `tete :- p1. tete :- p2. ... tete :- pn.`
- Puisque l'ordre des clauses est important, l'ordre des clauses d'une disjonction l'est aussi.

Plan

III. Négation

- Prolog permet la négation avec le prédicat `not/1` (ou `\+` en Gnu-Prolog)
- `not(A)` exprime le fait que A ne peut pas être prouvé : si un but A réussit alors `not(A)` échoue, si le but A échoue alors `not(A)` réussi.
- Ce n'est pas la négation logique ! `not(A)` est *défini par* :

```
not(A) :- call(A), !, fail.
not(A).
```

- Attention : nier des prédicats qui contiennent des variables peuvent donner des résultats inattendus.

III. Négation

Il faut que l'appel d'un but avec négation soit fait après l'instanciation des variables

```
marie(francois).
etudiant(rene).
etudiant_celibataire(X) :- not(marie(X)),  etudiant(X).

?- etudiant_celibataire(X).
no
?- etudiant_celibataire(rene).
yes
```


III. Négation

Version correcte

```
marie(francois).
etudiant(rene).
etudiant_celibataire(X) :- etudiant(X), not(marie(X)) .

?- etudiant_celibataire(X).
X = rene ;
```

Plan

1. Coupure
2. Disjonction
3. Négation
4. If then else
5. Quelques programmes de tri

IV. Le if then else

Une macro prédéfinie pour le "if then else", utilisant la coupure :

```
(P -> Q; R) :- P, !, Q.
(P -> Q; R) :- R.
```

Exemple d'utilisation : programme 23-ifthenelse.pl

```
maxi(A,B,C) :-
        A > B -> C=A ; C=B.
```


Plan

1. Coupure
2. Disjonction
3. Négation
4. If then else
5. Quelques programmes de tri

V. Quelques algorithmes de tri

- Tri lent : générer les permutations et tester si une permutation est triée.
- Tri par arbre binaire de recherche.
- Tri quicksort