Prolog

Une introduction à la programmation déclarative

Eric Salvat

IMERIR
salvat@imerir.com

2010-2011

Introduction

Objectifs :

- Utilisation de la logique des prédicats en programmation
- Etude d'un nouveau paradigme de programmation : la programmation déclarative
- Un premier pas vers les systèmes experts

Bref Historique

PROLOG = PROgrammation en LOGique.

- Alain Colmerauer (univ. Aix-Marseille) 1972
- Robert Kowalsky (univ. of Edinburgh-UK)

Domaines d'application

- Bases de données relationnelles
- Logique mathématique
- Conception assistée par ordinateur
- Traitement automatique du langage naturel
- Résolution symbolique d'équations
- Systèmes experts
- Aide à la décision
- . . .

Principe de la programmation déclarative

- déclarer des sur les objets manipulés et leurs relations,
- définir des (généralités) sur les objets et leurs relations,
- puis, interroger le « système » sur l'existence de relations entre objets.

Mécanisme de déduction

- Pour répondre aux requêtes, l'interpréteur Prolog utilise un mécanisme de **déduction indépendant** du problème posé (par opposition à un algorithme spécifique)
- C'est
- Interpréteur Prolog =

Syntaxe

- Instruction → :
  - faits
  - règles
  - but (ou requête)
- Structure de données → :
  - constantes
  - variables
  - termes composés

Un premier exemple

- 2 prédicats binaires :
  : X est étudiant de la promotion Y
  : X est apprenti de la promotion Y
- Soient les faits (ou programme) suivant :

```
etudiant(jean,colmerauer).
etudiant(tom,colmerauer).
etudiant(julie,humpich).
apprenti(luc,colmerauer).
apprenti(marie,humpich).
```

Éléments de syntaxe

- prédicats et constantes commencent par une minuscule ;
- les commencent par une ;
- nom du prédicat suivi des objets entre parenthèses et séparés par des virgules ;
- chaque déclaration se termine par un ;
- Attention à l'arité et à l'ordre des arguments d'un prédicat.

Utilisation du Programme

- L'exécution d'un programme ne se fait que lors d'une interrogation.
- Il faut donc poser une **question** (ou requête, ou but) à Prolog.
- Une telle question prend la même forme qu'un fait,
- seul le contexte d'interprétation change.
- `?-` est le prompt de l'interpréteur Prolog

Exemple

```
?- apprenti(luc,colmerauer).
```

Exemple

```
?- etudiant(luc,colmerauer).
```


Requête et variable

- Lorsqu'une requête contient une variable, Prolog cherche de la variable pour les quelles la questions se déduit du programme.

Exemple

```
?- apprenti(luc,X).
```

Exemple

```
?- etudiant(X,colmerauer).
```

Requête avec plusieurs variables

Exemple

```
?- etudiant(X,Y).
```


Requête composée

- le prédicat `promo(P,A)` : indique que la promotion `P` est en année `A`
- On enrichit le programme avec les faits suivants :

```
promo(colmerauer,1).
promo(humpich,2).
```

Exemple

```
?- etudiant(Nom,Promo),promo(Promo,An).
```

Principe d'effacement d'un But : la SLD Résolution

- Pour répondre à une requête , Prolog procède par
- en cherchant parmi les prédicats du programme ceux qui aux prédicats de la requête
- dans l'ordre des prédicats de la requête (de gauche à droite)
- dans l'ordre d'entrée des prédicats dans le programme (de haut en bas)
- Ces deux ordres sont importants et influent sur le résultat d'une requête.

Remarque

Tous les faits portant sur un même prédicat doivent être saisis dans la base, les uns après les autres (en un seul bloc !). L'ensemble des faits portant sur un même prédicat forme la du prédicat.

Exemple

```
etudiant(jean,colmerauer).
etudiant(tom,colmerauer).
etudiant(julie,humpich).

apprenti(luc,colmerauer).
apprenti(marie,humpich).

promo(colmerauer,1).
promo(humpich,2).
```

```
?-apprenti(X,Y).
```

Un autre exemple : but composé

Pour des raisons de présentation, on renome les prédicats `etudiant` et `apprenti` respectivement en `etud` et `app`.

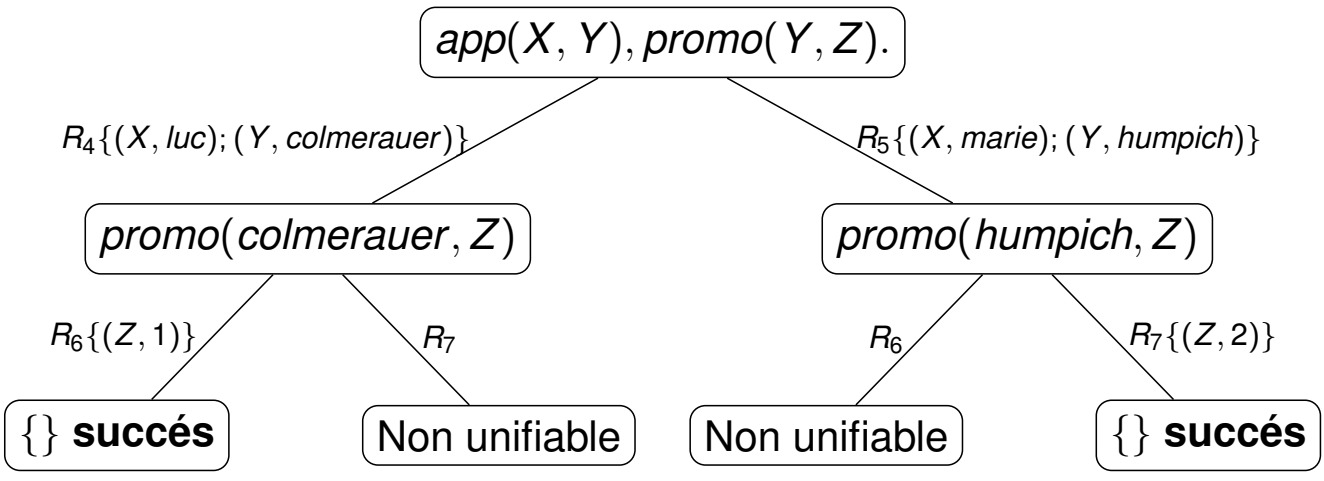
R_1 `etud(jean,colmerauer).`
R_2 `etud(tom,colmerauer).`
R_3 `etud(julie,humpich).`

R_4 **`app(luc,colmerauer).`**
R_5 **`app(marie,humpich).`**

R_6 **`promo(colmerauer,1).`**
R_7 **`promo(humpich,2).`**

`?-app(X,Y),promo(Y,Z).`

Schéma de l'effacement d'un but : l'arbre SLD



Les règles

- Les règles permettent d'exprimer des généralités.
- Par exemple on peut définir le prédicat binaire `app_meme_promo` qui indique que deux apprentis sont dans la même promotion par :

- Les règles permettent aussi de définir la notion d'élève dans une promotion :

SLD résolution avec règle

R_1 `etud(jean,colmerauer).`
R_2 `etud(tom,colmerauer).`
R_3 `etud(julie,humpich).`

R_4 `app(luc,colmerauer).`
R_5 `app(julien,colmerauer).`
R_6 `app(marie,humpich).`

R_7 `promo(colmerauer,1).`
R_8 `promo(humpich,2).`

R_9 `app_meme_promo(X,Y) :-app(X,P),app(Y,P),X\==Y.`

R_{10} `eleve(X,P) :-app(X,P).`
R_{11} `eleve(X,P) :-etud(X,P).`

`?-app_meme_promo(julien,A).`

Schéma de l'effacement d'un but : l'arbre SLD

La coupure (ou cut - !)

- Le cut (représenté par le symbôle " !") est un outil de contrôle de la résolution ;
- Il n'a pas de signification logique ;
- l'effacement du cut réussit toujours et agit par effet de bord sur la résolution en cours ;
- il interdit les retours arrière (backtrack) au delà du point où il est effacé ;
- Sur l'arbre SLD, cela revient à couper certaines branches ;
- Le cut ne peut pas être satisfait plus d'une fois et la règle dans laquelle il apparaît non plus.

Utilisation du cut dans une requête

- Si on ajoute un cut en fin de requête, on n'obtient que la première réponse.
- exemple : `apprenti(X,Y),promo(Y,Z),!.`

Cut et règles

- Le cut peut être utilisé dans la définition d'un prédicat (i.e. dans une règle).
- Soit dans un soucis d'optimisation :

- Soit pour définir une alternative (type "if-then-else")

Exemple

Les listes

- Une liste est représentée entre [], les éléments étant séparés par des virgules
- e.g.
- La liste vide :
- De façon générique, une liste est composée d'un élément E suivi du reste de la liste Q. Ce qui se note $[E|Q]$

Exemple

$$\begin{aligned} [1,2,3,4] &= \\ &= \\ &= \\ &= \end{aligned}$$

Listes et unification

Deux listes $[E_1|L_1]$ et $[E_2|L_2]$ sont

- égales si et seulement si :
  - et
- unifiables si et seulement si :
  - et

Exemple

L_1	L_2	`unification`
`[X,Y,Z]`	`[1,2,3]`	
`[X|Y]`	`[1,2,3]`	
`[X|Y]`	`[1]`	
`[[1,Y]|Z]`	`[[X,2],[3,4]]`	

Arithmétique

- Une expression artihmétique est un terme : elle est évaluée par Prolog
- On dispose des opérateurs classiques : + - * / mod ...
- L'affectation se fait à l'aide du prédicat `is` :
- Prolog dispose également de prédicats de comparaison : `> < =< >= =\= = :=`

Exemple

```
?-

?-

?-
```

Meta prédicats

- Ce sont des prédicats qui peuvent prendre en argument un predicat
- Ils permettent de manipuler un programme

: affiche la définition du prédicat `Pred`

: insère un fait ou une clause dans la base

: supprime un fait ou une clause de la base

: supprime tous les faits ou clause dont la tête correponds à `Head`

etc.