Présentation de Prolog

J.-C. Boisson
Jean-Charles.Boisson@univ-reims.fr
(Auteur original Leonardo Brenner 2009-2010)

Licence 3 Informatique / Passerelle - Info0502 - Logique et programmation logique

2019-2020

Programmation Logique

Historique

- 1972 : création de Prolog par A. Colmerauer et P. Roussel à Luminy (proche de Marseille) ;
- 1980 : reconnaissance de Prolog comme langage de développement en Intelligence Artificielle ;
- Depuis plusieurs versions, dont une tournée vers la programmation par contraintes.

Programmation Logique

Prolog

Langage d'expression des connaissances fondé sur le langage des prédicats du premier ordre :

- Programmation déclarative ;
- L'utilisateur définit une base de connaissances ;
- L'interpréteur Prolog utilise cette base de connaissances pour répondre à des questions.

Constantes

- Nombres : 12, 3.5 ;
- Chaînes de caractères commençant par une minuscule ;
- Chaînes de caractères entre " " ;
- Liste vide [].

Variables

- Chaînes de caractères commençant par une majuscule ;
- Chaînes de caractères commençant par _ ;
- La variable "indéterminée" : _

Programmation Logique

Faits

Les faits sont des données élémentaires qu'on considère vraies.

p(...). avec p un prédicat.

Faits - Exemples

pere(jean, paul).

pere(albert, jean).

Règles

Les règles sont des relations qui permettent à partir de ces hypothèses d'établir de nouveaux faits par déduction.

p(...) :- q(...), ..., r(...).

Règles - Exemple

grandpere(X,Y) :- pere(X,Z), pere(Z,Y).

Questions

Les questions sont les conclusions qui Prolog essaie de démontrer en utilisant les faits et les règles.

s(...), ..., t(...).

Remarque

La virgule "," fait office de "et" entre relations.
Le point virgule " ;" représente le "ou".

Questions - Exemple

pere(jean,X), mere(annie,X).

Programmation Logique

Programme - pere.pl

```
pere(charlie, david).
pere(henri, charlie).
grandpere(X,Y) :- pere(X,Z), pere(Z,Y).
```

Programme - questions

```
grandpere(X,Y).
X=henri
Y=david
```

Exercice

On dispose des informations suivantes :

- La secrétaire déclare qu'elle a vu l'ingénieur dans le couloir qui donne sur la salle de conférences ;
- Le coup de feu a été tiré dans la salle de conférences, on l'a donc entendu de toutes les pièces voisines ;
- L'ingénieur affirme n'avoir rien entendu.

On souhaite démontrer que si la secrétaire dit vrai, alors l'ingénieur ment.

Exercice - solution

Faits relatifs à l'énigme :

tir(salleDeConference).

lieuxVoisins(salleDeConference,couloir).

(lieuxVoisins(couloir,salleDeConference).)

declarationSecretaire.

Exercice - solution

Faits relatifs à l'énigme :

tir(salleDeConference).

lieuxVoisins(salleDeConference,couloir).

(lieuxVoisins(couloir,salleDeConference).)

declarationSecretaire.

Les règles :

estPresent(ingenieur,couloir) :- declarationSecretaire.

entendreTir(Individu,Piece) :- tir(Piece), lieuxVoisins(Piece,PieceVoisine), estPresent(Individu,PieceVoisine).

Arithmétique

- Comparaisons : $>, <, >=, =<, =:=, = =$
- Affectation : *is*

  ?- X is 3+2.
  X=5
- Fonctions prédéfinies : $-, +, *, /,$' *mod*, *abs*, *min*, *max*, *sign*, *random*, *sqrt*, *sin*, *cos*, *tan*, *log*, *exp*, ...

Résolution ou démonstration

Pour démontrer les questions posées, Prolog utilise le Principe de l'unification. C'est-à-dire, Prolog essaie de remplacer les variables dans de règles ou les faits. Le résultat n'est pas forcément unique, mais représente l'unificateur le plus général.

Echec de l'unification

e(X,X) et e(2,3) ne peuvent être unifiés.

Résolution ou démonstration

Prédicat d'unification : "="
a(B,C) = a(2,3). donne pour résultat :
YES B=2, C=3

Point de choix

Plusieurs règles concernant une même question :

- essais consécutifs dans l'ordre de déclaration ;
- représentation sous forme d'arbre ;
- les noeuds de l'arbre sont appelés points de choix.

Arbre de recherche

On parle d'arbre de recherche d'une question :

- Racine de l'arbre : question
- Noeud : points de choix (formule à démontrer)
- Passage d'un noeud vers son fils en considérant l'une des règles et en effectuant une unification est un pas de démonstration ;
- Noeuds sont crées de gauche à droite dans l'ordre de déclaration des règles ;
- Noeuds d'échec : aucune règle ne permet de démontrer la première formule du noeud ;
- Noeuds de succès : ne contient plus aucune formule, tout a été démontré et les éléments de solution sont trouves en remontant vers la racine de l'arbre

Stratégie de Prolog

- Pour résoudre une question, Prolog construit l'arbre de recherche de la question ;
- Parcours en profondeur d'abord :
  - noeud de succès : c'est une solution, Prolog l'affiche et cherche d'autres solutions ;
  - noeud d'échec : remontée dans l'arbre jusqu'à un point de choix possédant des branches non explorées.

Appel récursifs

Il faut :

- Choisir sur quoi faire l'appel récursif ;
- Choisir comment passer du résultat de l'appel récursif au résultat que l'on cherche ;
- Choisir le(s) cas d'arrêt.

Factorielle

```
fact(1, 1).
fact(A, B) :- C is A-1, fact(C, D), B is A*D.
```

Attention

Il faut faire des cas exclusifs.

Factorielle

```
fact(1, 1).
fact(A, B) :- C is A-1, fact(C, D), B is A*D.
```

fact(5,X).

fact(5,X) = fact(4,Y) et X = 5*Y

fact(4,X) = fact(3,Y) et X = 4*Y

fact(3,X) = fact(2,Y) et X = 3*Y

fact(2,X) = fact(1,Y) et X = 2*Y

fact(1,X) = fact(1,1) et X = 1

Factorielle

```
fact(1, 1).
fact(A, B) :- C is A-1, fact(C, D), B is A*D.
```

fact(5,X).

fact(5,X) = fact(4,Y) et X = 5*Y **: X = 120**

fact(4,X) = fact(3,Y) et X = 4*Y **: X = 24**

fact(3,X) = fact(2,Y) et X = 3*Y **: X = 6**

fact(2,X) = fact(1,Y) et X = 2*Y **: X = 2**

fact(1,X) = fact(1,1) et X = 1

Si j'appuye sur la touche "ENTER", Prolog s'arrête ici.

fact(5,X).

fact(5,X) = fact(4,Y) et X = 5*Y **: X = 120**

fact(4,X) = fact(3,Y) et X = 4*Y **: X = 24**

fact(3,X) = fact(2,Y) et X = 3*Y **: X = 6**

fact(2,X) = fact(1,Y) et X = 2*Y **: X = 2**

fact(1,X) = fact(1,1) et X = 1

Si j'appuye sur la touche " ;", Prolog cherche un prochain cas valide.

fact(1,X) = fact(0,Y) et X = 1*Y

fact(0,X) = fact(-1,Y) et X = 0*Y

fact(-1,X) = fact(-2,Y) et X = -1*Y

...

ERROR : Out of local stack

Factorielle - exclusion

```
fact(1, 1).
fact(A, B) :- A>1, C is A-1, fact(C, D), B is A*D.
```

fact(5,X).

fact(5,X) = fact(4,Y) et X = 5*Y : **X = 120**

fact(4,X) = fact(3,Y) et X = 4*Y : **X = 24**

fact(3,X) = fact(2,Y) et X = 3*Y : **X = 6**

fact(2,X) = fact(1,Y) et X = 2*Y : **X = 2**

fact(1,X) = fact(1,1) et X = 1

Si j'appuye sur la touche " ;", pas d'autre unification possible.

Factorielle - comme une fonction

```
fact2(A, B) :- fact2(A, 1, B).


fact2(1, A, A).
fact2(A, B, C) :- A > 1, D is B*A, E is A-1, fact2(E, D, C).
```

Factorielle - comme une fonction

```
fact2(A, B) :- fact2(A, 1, B).
fact2(1, A, A).
fact2(A, B, C) :- A > 1, D is B*A, E is A-1, fact2(E, D, C).
```

fact2(5,X).

fact2(5,X) = fact2(5,1,X)

fact2(5,1,X) = fact2(4,5,X)

fact2(4,5,X) = fact2(3,20,X)

fact2(3,20,X) = fact2(2,60,X)

fact2(2,60,X) = fact2(1,120,X)

fact2(1,120,X) = fact2(1,120,120)

Factorielle - comme une fonction

```
fact2(A, B) :- fact2(A, 1, B).
fact2(1, A, A).
fact2(A, B, C) :- A > 1, D is B*A, E is A-1, fact2(E, D, C).
```

fact2(5,X).

fact2(5,X) = fact2(5,1,X) **: X = 120**

fact2(5,1,X) = fact2(4,5,X) **: X = 120**

fact2(4,5,X) = fact2(3,20,X) **: X = 120**

fact2(3,20,X) = fact2(2,60,X) **: X = 120**

fact2(2,60,X) = fact2(1,120,X) **: X = 120**

fact2(1,120,X) = fact2(1,120,120)