

Le langage Prolog

Un langage de programmation *logique*

Programmation logique et Prolog

- ▶ Historique
- ▶ Données, relations et faits
- ▶ Prédicats et formules
- ▶ Règles
- ▶ Clauses de Horn
- ▶ Démonstration en Prolog
- ▶ Stratégie de Prolog

Historique

- ▶ 1972 : création de Prolog par A. Colmerauer et P. Roussel à Marseille
 - Pour le traitement des langues naturelles
- ▶ 1977: premier compilateur par D.H. Warren à Edimbourg
- ▶ 1980: Borland Turbo Prolog
- ▶ 1995: ISO Prolog
- ▶ Prolog est le langage de l'IA classique, des systèmes experts

Utilisateur

Les chats tuent les souris.

Minou est un chat qui n'aime pas les souris qui mangent du fromage.

Miquette est une souris qui mangent du fromage.

Max n'est pas une souris.

Que fait Minou?

Machine

Le Minou ne aime pas les souris qui mangent le fromage.

Le Minou tue les souris.

Utilisateur

Qui est un chat?

Machine

Le Minou.

Utilisateur

Qu'est-ce que mange Miquette?

Machine

Le fromage.

Utilisateur

Qui n'aime pas les souris qui mangent du fromage?

Machine

Le Minou.

Utilisateur

Qu'est-ce que mange Minou.

Machine

Ce que mange les chats qui ne aiment pas les souris qui mangent le fromage.

Programmer en Prolog

- ▶ Prolog est un langage descriptif (faits et relations) et prescriptif (inférence).
 - Il permet de représenter et de manipuler des connaissances
- ▶ Résoudre des problèmes impliquant
 - Un domaine: ensemble d'objets;
 - Des connaissances: relations entre objets.

Programmer en Prolog

- ▶ Spécifier des faits
 - Énoncés incontestablement vrais à propos du domaine d'étude
- ▶ Définir des règles
 - Permettant d'établir de nouveaux faits
- ▶ Poser des questions
 - Répondues par l'interpréteur Prolog

Prolog repose sur la logique du premier ordre

Logique du premier ordre

- ▶ Ensemble de symboles (variables)
- ▶ Ensemble de relations
- ▶ Des connecteurs logiques
- ▶ Des quantificateurs ‘pour tout’ et ‘il existe au moins un’

Programmer en Prolog

- ▶ L'utilisateur spécifie le quoi pas le comment
 - Langage très expressif
 - Développement rapide (?)
 - Demande une bonne maîtrise de la logique des prédictats
 - Exige une bonne compréhension des mécanismes internes du langage

Les données

- ▶ Constantes ou Atomes
 - * Symbole: chaîne de caractères (minuscule)
 - * Nombres : entiers ou flottants
- ▶ Variables : une lettre ou un nom (majuscule)
 - * Exprimer une propriété concernant une catégorie d'objets
 - * Interroger Prolog à l'aide d'une question

Les relations

- ▶ Propriété qui lie un certain nombre d'objets
 - * *la possession lie le propriétaire et l'objet possédé*
- * Utilité des relations :
 - * lien entre objets
 - * propriété d'un objet
- ▶ Plusieurs possibilités pour établir la même relation
 - * *Jean est le pere de Paul* peut s'exprimer de deux façons.
 - * Attention aux ambiguïtés

Les faits

- ▶ Affirmation de l'existence d'une relation entre certains objets
 - * *Tous les hommes sont mortels*
 - * *Socrate est un homme*
- ▶ Un fait est une formule vraie à priori
- ▶ Cela constitue une partie des données d'un problème

Prédicats et formules(1)

- ▶ En Prolog, une relation possède :
 - * un nom
 - * un nombre d'arguments
 - * *pere(jean, paul)*
- ▶ En logique, relation = **prédicat**
 - * *pere*
- ▶ Application du prédicat à ses arguments = **formule**

Arité d'un prédicat

- ▶ Nombre d'arguments du prédicat = arité
 - * unaire → propriété de l'argument
 - * *homme(socrate)*.
 - * arité zéro → signification logique très restreinte
 - * $p()$ proposition vraie
 - * binaire:
 - * *pere(jean, paul).* *pere(paul, martin).*
 - * *mere(marie, paul).* *mere(marie, luc).*
- homme/1 pere/2

Formule Prolog

- ▶ Enoncent la dépendance d'une relation entre objets par rapport à d'autres relations
- ▶ Concernent des catégories d'objets
 - * fait : homme(socrate).
 - * règle : *si X est un homme alors X est mortel* qui s'écrit : mortel(X) :- homme(X).
 - Le « si » s'écrit « :- » en Prolog
- ▶ Il peut y avoir plusieurs conditions derrière le « :- », séparées par des virgules

Question Prolog

- ▶ Question: est-ce que socrate est mortel?
 - La question s'exprime avec « ?- » en Prolog
- ▶ En Prolog la question est un but.
- ▶ Un fait est une règle sans queue...
- ▶ Une question est une règle sans tête...
- ▶ *Le canard est un oiseau. Les oiseaux volent.
Est-ce que le canard vole?*

Exemple

```
aime(jean,sandra).  
aime(jean,paulette).  
aime(paulette,sam).  
aime(bruno,sandra).  
boit(paulette,vin).  
boit(bruno,jus).  
boit(sam,biere).  
fume(bruno).  
fume(sandra).  
fume(sam).
```

...exemple

danse(jean,paulette):- boit(paulette,vin).

danse(bruno,sandra):- aime(bruno,sandra), boit(sandra,eau).

danse(sam,paulette):- aime(paulette,sam),boit(sam,biere),
 \+fume(paulette).

?-aime(jean,paulette)

yes

?-boit(bruno,biere)

no

?-danse(bruno,sandra)

no

?-danse(sam,paulette)

yes

...exemple

```
?- listing(aime)
aime(jean,sandra).
aime(jean,paulette).
aime(paulette,sam).
aime(bruno,sandra).
?- boit(Qui,biere)
Qui=sam
?-boit(jean,Quoi)
no
```

```
?-aime(jean,Qui)
Qui=sandra;
Qui=paulette;
No
?-danse(X,Y)
X=jean, Y=paulette;
X=sam, Y=paulette;
no
```

...exemple

```
malade(X):- fume(X), boit(X,Y), alcool(Y).  
alcool(biere).  
alcool(vin).
```

```
?- malade(X)  
X=sam
```

Un autre exemple

- * *la chèvre est un animal herbivore*
- * *le loup est un animal carnivore*
- * *les animaux carnivores sont cruels*
- * *un animal herbivore mange de l'herbe*
- * *un animal carnivore mange des animaux herbivores*

- ▶ Question: y a-t-il un animal cruel et que mange-t-il ?

Attention: la modélisation dépend des raisonnements que nous voulons mener!

Le Et et le Ou

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

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

Attention à la portée des variables!

On aurait pu écrire:

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

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

Clauses de Horn

- ▶ Ce sont les faits et les règles.
- ▶ Forme générale : $F :- F_1, F_2, \dots, F_n$.
 - * Se traduit par « F si F_1 et F_2 et...et F_n »
 - * F : formule atomique (doit être unique)
 - * F_i : littéraux (formules atomiques ou leur négation)
- ▶ F est la **tête** de la clause
- ▶ F_1, F_2, \dots, F_n est appelée la **queue** de la clause
- ▶ En Prolog : pour démontrer F , il faut démontrer F_1, F_2, \dots, F_n .
- ▶ Les clauses de Horn sont les seules formules permises en Prolog

Clauses de Horn

- ▶ Une règle est une clause dont la queue est non vide. La plupart des règles contiennent des variables.
- ▶ Définition d'une variable anonyme : « `_` »
 - * *a_un_salaire(X) :- employeur(Y,X).* peut s'écrire :
a_un_salaire(X) :- employeur(_,X).
- ▶ Déclaration d'un prédicat = ensemble de faits et de règles

Clauses de Horn

$b \leftarrow a_1, a_2, \dots, a_n$

- Si tous les termes a_i situés dans le corps de cette implication sont vrais, alors l'énoncé en tête d'implication, b , est vrai.
- Si le corps de l'implication est vide, il s'agit d'un fait (toujours vrai)

$b \leftarrow$

- Si la tête n'est pas spécifiée, il s'agit d'un but, c'est-à-dire une expression dont la véracité doit être vérifiée. L'usager peut aussi proposer un but au programme logique: il s'agit alors d'une requête. C'est le moyen par lequel l'usager interroge la base de données déductive

$\leftarrow a$

Clauses de Horn

- ▶ Les clauses Horn peuvent exprimer à peu près toute expression logique, même des algorithmes “mathématiques”.
- ▶ *L'hypothèse de fermeture du monde établie que toute affirmation dont la véracité ne peut être vérifiée doit être déclarée fausse.*

Programmes Prolog

- ▶ Programmes Prolog : succession de déclarations de prédictats
- ▶ Pas d'ordre à respecter
- ▶ Possibilité de plusieurs fichiers
 - * Exemple:
 - * *enfant(X,Y,Z)*
 - * *enfant(X,Y) :- enfant(X,Y,_); enfant(X,_,Y).*

Démonstration Prolog

- ▶ A partir d'un programme, on peut poser des questions
 - * Ex : *frere(paul, X)*. Pour trouver les frères de paul.
 - * C'est-à-dire déterminer les valeurs des variables (X) telles que la question soit déductible des faits et prédictats du programme.
- ▶ On parle de **Résolution de problème** ou de **démonstration de problème**.

Résolution

La résolution est une règle d'inférence qui permet de déduire des faits à partir d'autres faits (processus de déduction). Ainsi, à partir de

$$\begin{aligned}m &\leftarrow a, b, c, d, e \\d &\leftarrow f, g, h\end{aligned}$$

il est possible d'écrire, en combinant ces deux axiomes

$$m, \cancel{d} \leftarrow a, b, c, \cancel{d}, e, f, g, h$$

$$m \leftarrow a, b, c, e, f, g, h$$

Unification

- ▶ Exemple :
 - * *frere(X,Y) :- homme(X), enfant(X,Z), enfant(Y,Z), X \= Y.* où \= représente le prédicat de différence.
 - * *frere(paul,Qui)* : tentative d'unification avec la tête de la clause *frere(X,Y)*
- ▶ Définition : procédé par lequel on essaie de rendre deux formules identiques en donnant des valeurs aux variables qu'elles contiennent.

Unification

- ▶ Résultat : c'est un **unificateur** (ou substitution), un ensemble d'affectations de variables.
 - * Exemple : $\{X=paul, Qui=Y\}$
- ▶ Le résultat n'est pas forcément unique, mais représente l'unificateur le plus général.
- ▶ L'unification peut réussir ou échouer.
 - * $e(X,X)$ et $e(2,3)$ ne peuvent être unifiés.

Unification

```
vole(X):-oiseau(X).  
oiseau(canard).
```

```
?- vole(canard)  
?- oiseau(canard)
```

Unification

▶ Prédicat d'unification : « = »

- * $a(B,C) = a(2,3)$. donne pour résultat :
 $YES \{B=2, C=3\}$
- * $a(X,Y,L) = a(Y,2,carole)$. donne pour résultat :
 $YES \{X=2, Y=2, L=carole\}$
- * $a(X,X,Y) = a(Y,u,v)$. donne pour résultat :
 NO

Etapes de démonstration

- ▶ Si l'unification échoue : situation d'échec sur la règle considérée pour démontrer la formule.
- ▶ Si l'unification réussit : substitution des variables présentes dans la queue de la clause par les valeurs correspondantes des variables de l'unificateur.

Etapes de démonstration

- ▶ Démonstration de cet ensemble de formules dans l'ordre de leur citation pour enrichir le système avec les valeurs obtenues des variables.
- ▶ A la fin, l'ensemble des couples valeur-variable des variables présentes dans la question initiale forme la *solution* affichée par Prolog.

Exemple

```
ancetre(X, Y) :- pere(X, Y).  
ancetre(X, Y) :- ancetre(X, Z), pere(Z, Y).  
pere(phiippeIII, phiippeIV).  
ancetre(hugues_capet, phiippeIII).
```

Exemple

```
?- pere(phiippeIII, phiippeIV) .
```

yes

```
?- pere(phiippeIV, phiippeIII) .
```

no

```
?- pere(X, phiippeIV) .
```

X= phiippeIII

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

X= phiippeIII,

Y= phiippeIV

Exemple

```
?- philippeIII = philippeIII  
yes
```

```
?- philippeIII = philippeIV  
no
```

```
?- X = philippeIII  
X= philippeIII
```

L'unification est un mecanisme interne de Prolog

```
?- philippeIII = X
```

```
X= philippeIII
```

```
?- pere(X,philippeIV) = pere(phiippeIII, philippeIV) .
```

```
X= phiippeIII
```

```
?- pere(X,Z) = pere(phiippeIII, Y) .
```

```
X= phiippeIII
```

```
Z= Y= _8625
```

```
?- ((phiippeIII,philippeIV), hugues_capet)=(X,Y) , X=(M,N) .
```

```
X= phiippeIII,philippeIV,
```

```
Y= hugues_capet,
```

```
M= phiippeIII,
```

```
N= philippeIV
```

```
?- X = pere(X,X) .
```

```
pere(pere(pere(pere(pere(pere(pere(
```

Exemple

```
?- pere(phiippeIII,phiippeIV)  
yes
```

```
?- pere(X,phiippeIV)  
X= phiippeIII
```

```
?- pere(phiippeIII,X)
```

Un Autre exemple

```
%des faits en Prolog  
homme(john).  
homme(peter).  
homme(marc).  
femme(mary).  
femme(louise).
```

```
%interrogation des faits  
?- homme(john).  
Yes
```

```
?- homme(mary).  
No
```

```
?- homme(simon).  
No
```

```
?- homme(M).  
M = john ;  
M = peter ;  
M = marc ;  
No
```

```
% predicat parent/2:  
parent(john,peter).  
parent(peter,marc).  
parent(peter,louise).  
parent(mary,marc).  
parent(mary,louise).
```

```
%les enfants de Peter?  
?- parent(peter,C).  
C = marc ;  
C = louise ;  
No
```

```
%les parents de Louise?  
?- parent(P,louise).  
P = peter ;  
P = mary ;  
No
```

```
% Est-ce que John a des enfants?  
?- parent(john,_).  
Yes
```

```
% Est-ce que Marc a des enfants?  
?- parent(marc,_).  
No
```

```
%definir une regle Prolog  
hasChildren(P) :- parent(P,_).
```

```
?- hasChildren(peter).  
Yes
```

```
?- hasChildren(marc).  
No
```