

**Logique pour le prototypage rapide en informatique**

Aurélien Bénel  
*Systèmes d'information, management des connaissances et communication, Bureau T107*  
aurelien.benel@utt.fr

Outils mathématiques pour la modélisation, Université de technologie de Troyes, Automne 2007

---

---

---

---

---

---

---

---

**Plan**

2

- Programmation logique
- Aristote contre Prolog
- Prototypage rapide

---

---

---

---

---

---

---

---

**Programmation logique**

3

- Quels langage pour implémenter facilement des spécifications algébriques ?
  - ML (CAML),
  - Prolog,
  - LISP...

---

---

---

---

---

---

---

---

## Quand les logiciens font du Prolog sans le savoir...

4

### SPECIFICATIONS ALGÈBRIQUES

veutEtreAmiAvec(loulou, fifi)

$(\forall x) (\forall y) \text{ami}(x, y) \leftarrow$   
 $\text{veutEtreAmiAvec}(x, y)$   
 $\wedge \text{veutEtreAmiAvec}(y, x)$

$(\forall x) (\forall y) \text{amisCommuns}(x, y) =$   
 $\{ z \mid \text{ami}(z, x) \wedge \text{ami}(z, y) \}$

### PROLOG

veutEtreAmiAvec(loulou, fifi).

$\text{ami}(X, Y) :-$   
 $\text{veutEtreAmiAvec}(X, Y),$   
 $\text{veutEtreAmiAvec}(Y, X).$

$\text{amiCommunDe}(X, Y, Z) :-$   
 $\text{ami}(X, Y),$   
 $\text{ami}(X, Z).$

$\text{amisCommuns}(X, Y, E) :-$   
 $\text{setof}(Z, \text{amiCommunDe}(Z, X, Y), E).$

---

---

---

---

---

---

---

---

## Quels sont les cas qui posent problème ?

5

- Dans la logique classique (celle d'Aristote), on trouve :
  - Le principe d'identité,
  - Le principe de non-contradiction,
  - Le principe du tiers exclu,
  - Un principe "caché".
- Quels problèmes résolvent-ils ? Comment Prolog réagit-il en présence de ces problèmes ?

---

---

---

---

---

---

---

---

## 1. Principe de non-contradiction

6

- « Il est impossible d'affirmer et de nier [la même proposition] conformément à la vérité. »

Aristote, 1011b (*Métaphysique*)

---

---

---

---

---

---

---

---

## Prolog et la contradiction

7

% Histoire du marin (attribuée à Aristote)

```
il_prend_la_mer.  
il_est_sans_sa_femme:-  
    il_prend_la_mer.  
il_est_avec_son_bateau:-  
    il_prend_la_mer.  
il_est_heureux:-  
    il_est_sans_sa_femme, fail.  
il_est_heureux:-  
    il_est_avec_son_bateau.
```

```
PROLOG version XYZ  
?- il_est_heureux.  
Yes
```

En Prolog, si la réponse à une question est à la fois *oui* et *non*, c'est le *oui* qui l'emporte

---

---

---

---

---

---

---

---

## Prolog et la contradiction (suite)

8

% Histoire du marin (attribuée à Aristote)

```
il_prend_la_mer.  
il_est_sans_sa_femme:-  
    il_prend_la_mer.  
il_est_avec_son_bateau:-  
    il_prend_la_mer.  
il_est_heureux:-  
    il_est_sans_sa_femme, !, fail.  
il_est_heureux:-  
    il_est_avec_son_bateau.
```

```
PROLOG version XYZ  
?- il_est_heureux.  
No
```

... à moins que l'on précise à l'aide d'un *coupe-choix* que l'on s'en tient au premier résultat.

---

---

---

---

---

---

---

---

## 2. Principe du tiers exclu

9

- « Chaque chose, nécessairement, est ou n'est pas, sera ou ne sera pas, même si on ne peut pas dire laquelle des deux est nécessaire. »

Aristote, 19a (*De l'interprétation*)

---

---

---

---

---

---

---

---

## Prolog et le tiers exclu

10

```
% La bataille navale de demain  
% (Aristote vs Diodore Chronos)
```

En Prolog  
la réponse à une question  
est **non** par défaut.

```
PROLOG version XYZ  
?- il_y_aura_une_bataille_navale_demain.  
No
```

---

---

---

---

---

---

---

---

## 3. Principe d'identité

11

- « Se demander pourquoi une chose est elle-même, c'est enquêter dans le vide [...], le fait qu'une chose est elle-même est la seule réponse et la seule cause. »

Aristote, 1041a (*Métaphysique*)

---

---

---

---

---

---

---

---

## Prolog et l'identité

12

```
PROLOG version XYZ  
?- X=socrate.  
X = socrate  
yes  
?- socrate=X.  
X = socrate  
yes  
?- X=socrate, X=aristote.  
no
```

En Prolog :  
- l'unification (=) n'est pas une égalité,  
- l'unification n'est pas non plus une affectation.

```
PROLOG version XYZ  
?- 2=1+1.  
no  
?- 2 is 1+1.  
yes  
?- 1+1 is 2.  
no
```

En Prolog :  
- l'unification porte sur la structure,  
- l'unification à la valeur (*is*) ne fonctionne que dans un seul sens.

---

---

---

---

---

---

---

---

#### 4. Principe caché : la portée existentielle

13

- Pour Aristote, il y a 4 types de propositions :
  - Tous les *SUJETs* sont *PREDICATs*
  - Quelques *SUJETs* sont *PREDICATs*
  - Quelques *SUJETs* ne sont pas *PREDICATs*
  - Aucun *SUJET* n'est *PREDICAT*
- Or on ne peut parler que de choses qui existent, les propositions donnent donc une valeur d'existence aux sujets.

Exemple : prendre "homme" comme sujet et "mortel" comme prédicat.

---

---

---

---

---

---

---

---

#### Prolog et la portée existentielle (formelle)

14

% Tout est substance

substance(\_).

AMZI! PROLOG  
?- substance(X).  
X = H110;  
no

GNU PROLOG  
?- substance(X).  
yes

% Tous les  
% philosophes sont  
% des hommes

homme(X):-  
philosophe(X).

AMZI! PROLOG  
?- homme(X).  
no

GNU PROLOG  
?- homme(X).  
..existence\_error  
..philosophe..

---

---

---

---

---

---

---

---

#### Prolog et la portée existentielle (formelle), suite...

15

% Callias est un  
% homme  
homme(callias).

% Socrate est un  
% philosophe  
philosophe(socrate).

% Tous les  
% philosophes sont  
% des hommes  
homme(X):-  
philosophe(X).

PROLOG version XYZ  
?- not(philosophe(callias)).  
yes  
?- not(philosophe(X)).  
no  
?- homme(X), not(philosophe(X)).  
X = callias ;  
no

En Prolog,  
il faut "générer puis tester" :  
Chaque variable doit être  
unifiée à une instance dès sa  
première apparition.

---

---

---

---

---

---

---

---

## Pourquoi Prolog gère-t-il mal ce cas ?

16

- Ces cas problématiques (et les principes associés) sont décrits par Aristote :
  - Non dans ses livres de logique,
  - Mais dans ses livres de métaphysique.
- En quoi, *l'interprétation* de la logique (et de la vérité), a-t-elle changée au cours de l'histoire ?

---

---

---

---

---

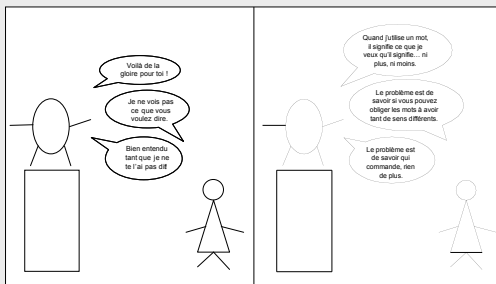
---

---

---

## Logique contemporaine

17



---

---

---

---

---

---

---

---

## Statut des programmes

18

- Intelligence artificielle au sens fort ?
  - cycle en V (très critiqué)
- Purement formels ?
  - jeux
- Modèles ?
  - cycle en spirale

---

---

---

---

---

---

---

---

## Application : Prototypage rapide

19

- But : Comparer des algorithmes de tri.
- Jeu de test : liste de nombres réels.
- Est une liste :
  - Une liste vide [],
  - Une liste [E|L] composée d'un élément E suivi d'une liste L.

---

---

---

---

---

---

---

---

## Application : Prototypage rapide

20

Exemple : comparer des algorithmes de tri

- Ordonner deux réels

```
% minMax(Float1, Float2, Min, Max)

% pour Float1 < Float2
minMax(X, Y, X, Y):-
    X<Y, !.

% pour le cas général
% (sauf cas précédent)
minMax(X, Y, Y, X).

...
```

---

---

---

---

---

---

---

---

## Prototypage rapide : Tri à bulle

21

```
...
% bulle(ListeInitiale, ListeBullee)
...
```

```
% pour un singleton
```

```
bulle([X], [X]):-!.
```

```
% pour une liste non vide quelconque
% (sauf cas précédent)
```

```
bulle([X | L1], [Min, Max | L2]):-
    bulle(L1, [Y | L2]),
    minMax(X, Y, Min, Max).
...
```

```
...
% triBulle(ListeInitiale, ListeTrie)
...
```

```
% pour une liste vide
```

```
triBulle([], []).
```

```
% pour une liste non vide
```

```
triBulle(L1, [Min | L2]):-
    bulle(L1, [Min | L3]),
    triBulle(L3, L2).
...
```

---

---

---

---

---

---

---

---

## Prototypage rapide : Tri par fusion (1)

22

```
...
% fusion (ListeTrie1, ListeTrie2,
% ListeTrieFusion)

% lorsque l'une des listes est vide
fusion([], L, L):-!.
fusion(L, [], L).

% lorsque les listes sont non vides
fusion([X | L1], [Y | L2], [X | L3]):-
    X < Y, !,
    fusion(L1, [Y | L2], L3).
fusion(L1, [Y | L2], [Y | L3]):-
    fusion(L1, L2, L3).
```

---

---

---

---

---

---

---

---

## Prototypage rapide : Tri par fusion (2)

23

```
...
% couper(Liste, Position, Taille,
% Moitie1, Moitie2)
couper(L, Position, Taille, [], L):-
    Position > Taille/2, !.

couper([X|L], Position, Taille, [X|M1], M2):-
    Position2 is Position+1,
    couper(L, Position2, Taille, M1, M2).

% couper(Liste, Moitie1, Moitie2)
couper(Liste, M1, M2):-
    length(Liste, Taille),
    couper(Liste, 1, Taille, M1, M2).
...
```

---

---

---

---

---

---

---

---

## Prototypage rapide : Tri par fusion (3)

24

```
...
% triFusion(ListeInitiale, ListeTrie)
triFusion([], []):-!.

triFusion([X], [X]):-!.

triFusion(Initial, Trie):-
    couper(Initial, Initial1, Initial2),
    triFusion(Initial1, Trie1),
    triFusion(Initial2, Trie2),
    fusion(Trie1, Trie2, Trie).
```

---

---

---

---

---

---

---

---

## Prototypage rapide : Comparaison sur des jeux de test

25

```
...
test1([46,2,4,46,248,5, ...
test2([46,2,4,46,248,5, ...
```

GNU PROLOG

```
?- test1(L),length(L,X).
L=[46,2,4,48,248,5,56, ...
X=100
yes
```

```
?- test1(L1),triBulle(L1, L2).
L2=[-2673,-9,1,2,4,5,46,48,56...
(40 ms) yes
```

```
?- test1(L1),triFusion(L1, L2).
L2=[-2673,-9,1,2,4,5,46,48,56...
(10 ms) yes
```

GNU PROLOG

```
?- test2(L),length(L,X).
L=[46,2,4,48,248,5,56, ...
X=600
yes
```

```
?- test2(L1),triBulle(L1, L2).
L2=[-3568,-2673,-749,-245,-9, ...
(1230 ms) yes
```

```
?- test2(L1),triFusion(L1, L2).
L2=[-3568,-2673,-749,-245,-9, ...
(60 ms) yes
```

---

---

---

---

---

---

---

---

## Pour aller plus loin

26

- Amzi! Prolog, Interpréteur Prolog gratuit, Disponible sur Internet : <http://www.amzi.com/download/>
- Bellot P., *Objectif Prolog*, Paris : Masson, 1994.
- Blanché R., *La logique et son histoire*, Paris : Armand Colin, 2002. (Note : Edition originale publiée en 1970, revue et complétée en 1996).
- Colmerauer A., Roussel P., *La naissance de Prolog*, 1992. Disponible sur Internet : <http://www.lim.univ-mrs.fr/~colmer/texte.htm#early>
- GNU Prolog, Interpréteur/Compilateur Prolog en open-source, Disponible sur Internet : <http://www.gnu.org/software/gprolog/>
- Sowa J.F., *Knowledge Representation: Logical, Philosophical and Computational Foundations*, Pacific Grove: Brooks/Cole, 2000.

---

---

---

---

---

---

---

---