Résumé de l'épisode précédent

Nous avons...

- précisé le vocabulaire du Prolog : terme (simple et complexe), atome logique, clause (fait, règle), prédicat, programme
- présenté en détail les termes complexes
- présenté les opérateurs qui permettent de plus facilement utiliser certains termes complexes ou certains prédicats d'arité un ou deux
- présenté les expressions arithmétiques qui sont des termes complexes par défaut non évaluées
- présenté les listes et quelques algorithmes

INSA ROUEN NORMANDIE

Fonctionnement du moteur Prolog

Cours « Découverte de l'intelligence artificielle »

Nicolas Delestre

INSA
ROUEN NORMANDIE

Plan

INSA ROUEN NORMANDIE

Définitions 1 / 2

Unification

« Procédé par lequel le Prolog essaie de rendre deux formules identiques en donnant des valeurs aux variables qu'elles contiennent » [Bel94]

`p(X,a)` et `p(b,a)` peuvent être identiques si on unifie `X` à `b`

Point de choix

Le fait d'avoir des clauses compatibles pour essayer de démontrer l'atome logique d'une question

```
p(a).
p(A) :- q(A).
```

Vouloir démontrer `p(X)` amène un point de choix avec deux choix d'unification

Définitions 2 / 2

Pas de démonstration

- Le fait de faire une unification au niveau d'un point de choix (rentrer dans une branche de la démonstration)
- Si la clause courante est une règle, un pas de démonstration entraîne la démonstration des prémisses de la règle

Backtracking

- Le fait de ne plus avoir à/pouvoir faire de pas de démontration
- Le moteur prolog remonte jusqu'au dernier point de choix et essaye d'appliquer le pas de démonstration suivant

INSA ROUEN NORMANDIE

Fonctionnement du moteur prolog 1 / 3

Pour une question donnée (conjonction d'atomes logiques)

Étape 1 : liste des clauses compatibles

- Le moteur prolog sélectionne les clauses dont les buts peuvent s'unifier avec le premier atome de la question (clauses compatibles) : c'est un point de choix
  - l'ordre de la liste des clauses compatibles est l'ordre du programme Prolog (cela peut donc avoir des conséquences sur les résultats)
  - si cette liste est vide, alors le moteur fait un backtracking avec la valeur *false*
  - sinon il passe à l'étape 2

INSA ROUEN NORMANDIE

Fonctionnement du moteur prolog 2 / 3

Étape 2 : pas de démonstration

- Pour chaque clause compatible, le moteur tente d'unifier les termes de la question avec les termes de la conclusion. S'il y a arrive, c'est un pas de démonstration
- Si la clause courante est une règle, Il essaye alors de démontrer une à une toutes les prémisses (qui en quelques sortes deviennent de nouvelles questions) :
  - Si toutes les prémisses sont vrai, prolog « retourne » la valeur *true* et les unifications ayant permis la démonstration (solution). Si la clause courante n'est pas le dernier élément de la liste, le moteur propose de poursuivre la recherche de solution.
- Si la clause courante est un fait, il « retourne » *true*. Si la clause courante n'est pas le dernier élément de la liste, le moteur propose de poursuivre la recherche de solution.

INSA ROUEN NORMANDIE

Fonctionnement du moteur prolog 3 / 3

Exemple

```
1 p(1,2).
2 p(7,8).
3 p(9,10).
4 p(X,Y) :- q(X,Y).
5 p(5,Y) :- Y=6.
6 p(X,6) :- X=5.
7 q(7,8).
```

```
?- p(7,8). % lig. 2 OK lig. 4 OK
true ;
true.
?- p(1,2). % lig. 1 OK lig. 4 KO
true ;
false.
```

```
?- p(3,4). % lig. 4 KO
false.
?- p(5,6). % lig. 4 KO lig. 5 OK
                       lig. 6 OK
true ;
true.
```

Un exemple 1 / 3

Reprenons l'exemple sur la généalogie

```
1  /* homme(X) est vrai si X est un homme */
2  homme(patrick).
3  homme(gerard).
4  homme(louis).
5  homme(pierre).
6
7  /* femme(X) est vrai si X est une femme */
8  femme(therese).
9  femme(sandrine).
10 femme(muriel).
11 femme(germaine).
12 femme(yvette).
13
14 /* enfant_parents(Enfant,Parent1,Parent2) est vrai si Enfant est un enfant de Parent1 et Parent2 */
15 enfant_parents(gerard,germaine,louis).
16 enfant_parents(therese,yvette,pierre).
17 enfant_parents(patrick,gerard,therese).
18 enfant_parents(muriel,gerard,therese).
19 enfant_parents(sandrine,gerard,therese).
20 enfant_parents(astride,jean,therese).
```

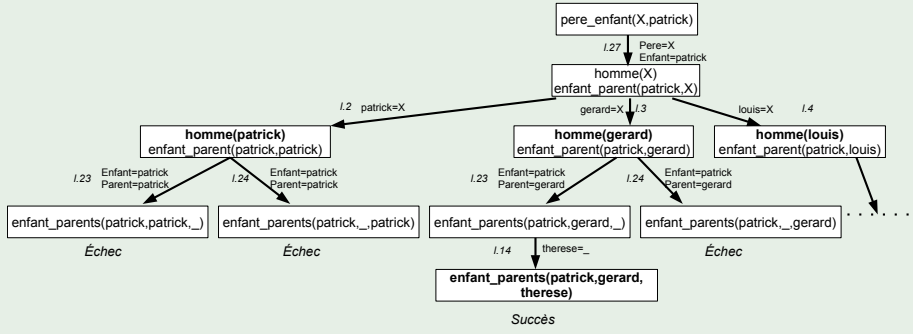
Un exemple 2 / 3

```
22 /* enfant_parent(Enfant,Parent) est vrai si Enfant est un enfant de Parent */
23 enfant_parent(Enfant,Parent) :- enfant_parents(Enfant,Parent,_).
24 enfant_parent(Enfant,Parent) :- enfant_parents(Enfant,_,Parent).
25
26 /* pere_enfant(Pere,Enfant) est vrai si Pere est pere de Enfant */
27 pere_enfant(Pere,Enfant) :- homme(Pere), enfant_parent(Enfant,Pere).
28
29 /* mere_enfant(Mere,Enfant) est vrai si Mere est mere de Enfant */
30 mere_enfant(Mere,Enfant) :- femme(Mere), enfant_parent(Enfant,Mere).
31
32 /* grandpere_enfant(GrandPere,Enfant) vrai si GrandPere est grand-pere de Enfant */
33 grandpere_enfant(GrandPere,Enfant) :- pere_enfant(GrandPere,Parent), pere_enfant(Parent,Enfant).
34 grandpere_enfant(GrandPere,Enfant) :- pere_enfant(GrandPere,Parent), mere_enfant(Parent,Enfant).
35
36 /* grandmere_enfant(GrandMere,Enfant) vrai si GrandMere est grand-mere de Enfant */
37 grandmere_enfant(GrandMere,Enfant) :- mere_enfant(GrandMere,Parent), pere_enfant(Parent,Enfant).
38 grandmere_enfant(GrandMere,Enfant) :- mere_enfant(GrandMere,Parent), mere_enfant(Parent,Enfant).
```

INSA ROUEN NORMANDIE

Un exemple 3 / 3

pere_enfant(X,patrick).



Le *cut*

Définition

Le prédicat *!/0* (le *cut*) sert à indiquer au Prolog que si la démonstration en cours est valide, alors toutes les unifications issues des points de choix précédents sont à supprimer (plus de backtracking)

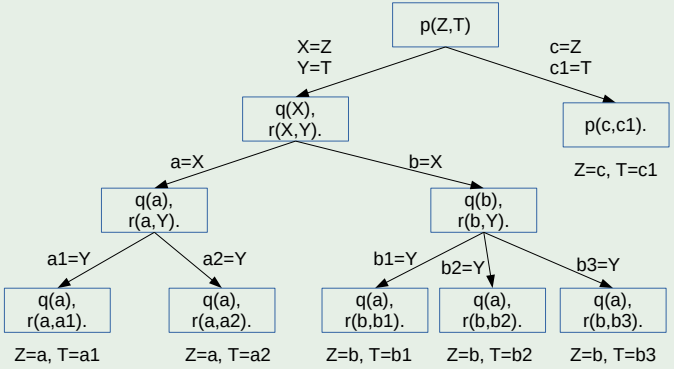
INSA ROUEN NORMANDIE

Compréhension par l'exemple 1 / 4

Un exemple sans cut

```
q(a). q(b).
r(a,a1). r(a,a2).
r(b,b1). r(b,b2). r(b,b3).

p(X,Y) :- q(X), r(X,Y).
p(c,c1).
```




```
?- p(Z,T).
Z = a,
T = a1 ;
Z = a,
T = a2 ;
Z = b,
T = b1 ;
Z = b,
T = b2 ;
Z = b,
T = b3 ;
Z = c,
T = c1.

?-
```

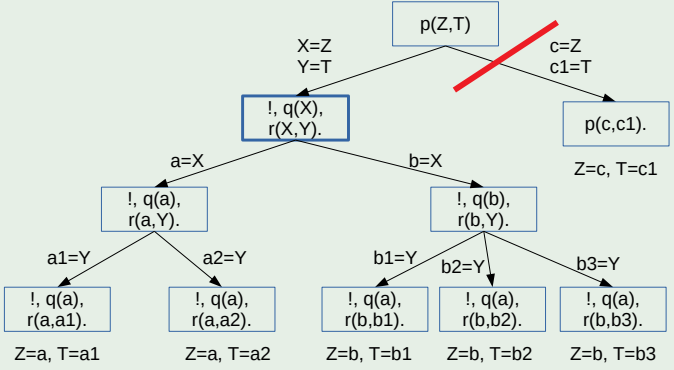
inspiré de « Introduction à PROLOG » Christine Solnon

Compréhension par l'exemple 2 / 4

Un exemple avec un cut au début

```
q(a). q(b).
r(a,a1). r(a,a2).
r(b,b1). r(b,b2). r(b,b3).
```

```
p(X,Y) :- !, q(X), r(X,Y).
p(c,c1).
```




```
?- p(Z,T).
Z = a,
T = a1 ;
Z = a,
T = a2 ;
Z = b,
T = b1 ;
Z = b,
T = b2 ;
Z = b,
T = b3.

?-
```

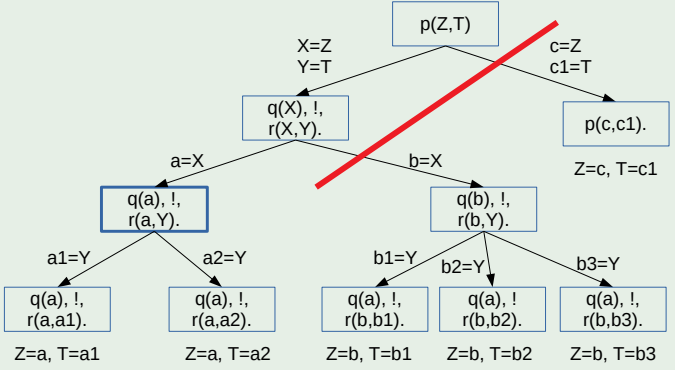
inspiré de « Introduction à PROLOG » Christine Solnon

Compréhension par l'exemple 3 / 4

Un exemple avec un cut au milieu

```
q(a). q(b).
r(a,a1). r(a,a2).
r(b,b1). r(b,b2). r(b,b3).
```

```
p(X,Y) :- q(X), !, r(X,Y).
p(c,c1).
```




```
?- p(Z,T).
Z = a,
T = a1 ;
Z = a,
T = a2.

?-
```

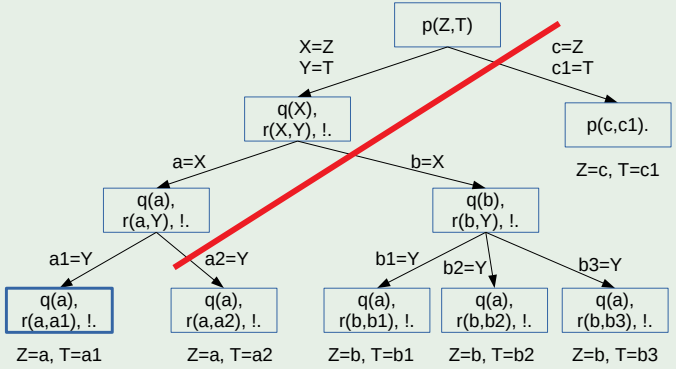
inspiré de « Introduction à PROLOG » Christine Solnon

Compréhension par l'exemple 4 / 4

Un exemple avec un cut à la fin

```
q(a). q(b).
r(a,a1). r(a,a2).
r(b,b1). r(b,b2). r(b,b3).
```

```
p(X,Y) :- q(X), r(X,Y), !.
p(c,c1).
```




```
?- p(Z,T).
Z = a,
T = a1.

?-
```

inspiré de « Introduction à PROLOG » Christine Solnon

Cut vert et rouge

Définition

- Un cut est dit vert lorsque son utilisation n'élimine pas de solution
- Dans le cas contraire il est dit rouge

Exemple de *cut* vert

```
insererOrdreCroissant(Element,[],[Element]) :- !.
insererOrdreCroissant(Element,[Tete|Queue],[Element,Tete|Queue]) :- Element=<Tete, !.
insererOrdreCroissant(Element,[Tete|Queue],[Tete|InsertionElementDansQueue]) :-
    Element>Tete,
    insererOrdreCroissant(Element,Queue,InsertionElementDansQueue).
```

Exemple de *cut* rouge

```
membre(Element,[Element|_]) :- !.
membre(Element,[_|Queue]) :- membre(Element,Queue).
```

Le métaprédicat *not/1*

Définitions

- Un métaprédicat est un prédicat qui accepte un prédicat comme paramètre
- Le métaprédicat *not/1* est vrai si la démonstration de son paramètre échoue, sinon il échoue

Attention

Il faut s'assurer que toutes les variables utilisées comme paramètres du prédicat sont bien unifiées

Exemple

```
p(a).
```

```
?- X=b,not(p(X)).
X = b.

?- not(p(X)), X=b.
false.
```

Le métaprédicat *maplist/3*

Définition

- Le métaprédicat *maplist/3* applique un prédicat à une liste d'arguments : `maplist(predicat, listeArguments, Resultat)` : chaque élément de `listeArguments` est unifié à l'avant dernière variable libre de `predicat`
- Si l'une de ces applications échouent, *maplist/3* échoue, sinon on obtient dans la liste `Resultat` toutes les unifications réussies de la dernière variable libre de `predicat`
- Il est à noter que ce métaprédicat reprend un outil du paradigme de la programmation fonctionnelle

Exemple

```
plus(X,Y,R) :- R is X+Y.
```

```
?- maplist(plus(2),[1,2,3],L).
L = [3, 4, 5].
```

Le métaprédicat *findall/3*

Définition

- Le métaprédicat *findall/3* permet d'obtenir une liste d'associations qui satisfont un prédicat : `findall(Variable, predicat, Resultat)` : la liste `Resultat` contient les termes unifiés à `Variable` qui satisfont `predicat`.
- Il est à noter que ce métaprédicat reprend un outil du paradigme de la programmation fonctionnelle

Exemple

```
?- findall(Personne, enfant(Personne, gerard), L).
L = [patrick, muriel, sandrine].
```

INSA ROUEN NORMANDIE

Conclusions

Nous avons dans ce cours

- défini les éléments de base du fonctionnement du moteur : unification, point de choix, pas de démonstration et backtracking
- décrit le fonctionnement du moteur prolog
- étudié le rôle du predicat *cut/0*
- vu quelques métaprédicats

INSA ROUEN NORMANDIE

Références

[Bel94] P. Bellot.
Objectif Prolog.
Masson, 1994.

[Tri22] Markus Triska.
The power of prolog.
https://www.metalevel.at/prolog, 2022.

INSA ROUEN NORMANDIE