

Résumé de l'épisode précédent

- La logique permet de mécaniser le raisonnement
- La logique informatique étudie l'automatisation des démonstrations
- Il faut distinguer la syntaxe de la sémantique
- Il existe plusieurs logiques avec des pouvoirs d'expression différents
- La programmation logique est un paradigme de programmation
- Prolog est un langage de programmation logique

Les termes complexes en Prolog

Cours « Découverte de l'intelligence artificielle »

Nicolas Delestre

Plan

1 Vocabulaire

2 Les termes complexes

- Les opérateurs
- Les expressions arithmétiques

3 Les listes

Précisions concernant la syntaxe et le vocabulaire 1 / 4

Termes

- Éléments de base du langage
- Deux types de termes : les termes simples et termes complexes

Terme simple : atome, constante ou variable

- Atome :
 - séquence de lettres (majuscule ou minuscule), de chiffres ou du tiret bas, commençant par une minuscule
 - séquence de caractères entourée de guillemets simples
 - séquence de caractères spéciaux (+, -, *, /, <, >, =, :, ., @, ~)
- Constante :
 - chaîne de caractères entourée de guillemets doubles
 - nombre : entier ou flottant par défaut représenté en décimale. Pour les entiers, possibilité d'utiliser le `_` comme séparateur : `1_000_000`

Précisions concernant la syntaxe et le vocabulaire 2 / 4

Terme simple : atome, constante ou variable (suite)

- Variable : séquence de lettres (majuscule ou minuscule), de chiffres ou du tiret bas, commençant par une majuscule ou un tiret bas (le tiret bas seul est appelé variable anonyme)

Terme complexe

- Association d'un foncteur (un atome) et d'arguments (n'importe quel type de terme) entre parenthèses, séparés des virgules
- L'arité d'un terme complexe est son nombre d'arguments

Atome logique

- Terme complexe particulier
- Met en relation logique les termes
- Composant des clauses

Clause

- Élément de la base de connaissance
- Affirmation inconditionnelle (ou fait) : un seul atome logique qui se termine par un point
« . »
- Affirmation conditionnelle (ou règle) : succession d'atomes logiques tel que le premier (le but) et le deuxième sont séparés par $:-$ et le deuxième et les autres (les prémisses) par $,$ qui se termine par $.$
La virgule lie les atomes par un « et » logique

- Il est à noter que $:-$ et la virgule ($,$) sont des termes complexes d'arité 2 et le point ($.$) est un terme simple

Précisions concernant la syntaxe et le vocabulaire 4 / 4

Prédicats

- Ensemble de clauses dont leurs premiers atomes logiques (leurs buts) ont le même foncteur et la même arité
- Les clauses d'un même prédicat sont liées par un « ou » logique

Programme Prolog

- Ensemble de prédicats
- L'exécution d'un programme revient à poser une question

Les termes complexes 1 / 3

- Les termes complexes (qui ne sont pas des atomes logiques) servent à structurer hiérarchiquement les données

Attention

- Ils ne calculent rien, ce ne sont pas des fonctions

Exemple (inspiré de [Bel94])

- Comment exprimer :
 - Annie possède une ford escort
 - Jérôme possède une renault twingo
 - Jérôme possède une console de jeu Sony playstation 5
 - Jérôme possède une console de jeu Microsoft Xbox One
 - Hélène possède une renault mégane
 - Hélène possède une console de jeu switch

Les termes complexes 2 / 3

Avec uniquement des prédicats

```

possede(annie,voiture_annie).
possede(jerome,voiture_jerome).
possede(jerome,console_jerome_1).
possede(jerome,console_jerome_2).
possede(helene,voiture_helene).
possede(helene,console_helene).

```

```

marque(voiture_annie,ford).
marque(voiture_jerome,renault).
marque(console_jerome_1,sony).
marque(console_jerome_2,microsoft).
marque(voiture_helene,renault).
marque(console_helene,nintendo).

```

```

type_d_objet(voiture_annie,voiture).
type_d_objet(voiture_jerome,voiture).
type_d_objet(voiture_helene,voiture).
type_d_objet(console_jerome_1,console).
type_d_objet(console_jerome_2,console).
type_d_objet(console_helene,console).

```

```

modele(voiture_annie,escrot).
modele(voiture_jerome,twingo).
modele(console_jerome_1,playstation_5).
modele(console_jerome_2,xBox_One).
modele(voiture_helene,megane).
modele(console_helene,switch).

```

Qui possède une voiture de marque ford ?

```

?- possede(Personne,Objet),type_d_objet(Objet,voiture),marque(Objet,ford).
Personne = annie,
Objet = voiture_annie ;
false.

```

Les termes complexes 3 / 3

Avec les termes complexes (possede.pl)

```
possede(annie,voiture(ford,escort)).  
possede(jerome,console(sony,playstation_3)).  
possede(jerome,console(microsoft,xbox_one)).  
possede(jerome,voiture(renault,twingo)).  
possede(helene,console(nintendo,switch)).  
possede(helene,voiture(renault,megane)).
```

Exemple d'interaction

```
1 ?- [possede].  
true.  
  
2 ?- possede(X,voiture(_,_)).  
X = annie ;  
X = jerome ;  
X = helene.  
  
3 ?- possede(X,voiture(renault,_)).  
X = jerome ;  
X = helene.
```

Les opérateurs

Du sucre syntaxique

- Un opérateur permet de représenter de manière plus pratique des termes complexes ou des prédicats d'arité 1 ou 2, par exemple :

$op_g \text{ opérateur } op_d$ correspond à $opérateur(op_g, op_d)$

- Un opérateur est caractérisé par
 - une priorité : par exemple si l'opérateur $op1$ est prioritaire à l'opérateur $op2$, $op_g \ op1 \ op_m \ op2 \ op_d$ est interprété comme $op2(op1(op_g, op_m), op_d)$
 - une associativité (à gauche ou à droite) : par exemple si op est associatif à gauche, $op_g \ op \ op_m \ op \ op_d$ est interprété comme $op(op(op_g, op_m), op_d)$

- Il est à noter que $:-$ et $,$ sont des opérateurs d'arité deux
- Le prédicat *write_canonical/1* permet d'obtenir la représentation sous forme de termes toute expression Prolog

Les expressions arithmétiques

Des termes complexes

- La notation $2+3+4$ correspond au terme complexe $+(+(2,3),4)$
- Pour évaluer une expression arithmétique on utilise l'opérateur `is` qui correspond au prédicat $is/2$:
 X `is` $2+3$ correspondant au prédicat $is(X,2+3)$ qui est vrai lorsque X est associé à 5
- Les opérateurs `==`, `!=`, `<`, `>`, `=<`, `>=` évaluent les expressions arithmétiques (opérandes gauches et droites) avant de les comparer

Les listes

Définition

- Terme complexe ' $[|]$ ' / 2 tel que le premier arguments (un terme) est la tête de liste et le second la queue de liste (une liste).
- La liste vide est notée $[]$
- Constats :
 - Une liste permet de représenter une suite de termes
 - Sa définition récursive amène à écrire des prédicats récursifs

Facilité syntaxique, deux notations

- Les éléments de la liste sont entourés de crochets et séparés par des virgules : ' $[|]$ ' (1, ' $[|]$ ' (2, $[]$)) peut être notée $[1,2]$
- Une liste est entourée de crochet dont le premier élément est séparé du reste de la liste (une liste) par une barre : ' $[|]$ ' (1, ' $[|]$ ' (2, $[]$)) peut être notée $[1| [2| []]]$

Les différentes façons de désigner une liste

```
?- '[]'(1, '[]'(2, '[]'(3, []))) = [1,2,3].  
true.
```

```
?- [1|[2|[3|[]]]] = [1,2,3].  
true.
```

```
?- [1|[2|[3]]] = [1,2,3].  
true.
```

```
?- [1,2|[3]] = [1,2,3].  
true.
```

Quelques prédicats sur les listes 1 / 4

membre/2

```

/* membre(Element,Liste) est vrai si Element est un element de Liste*/
membre(Element,[Element|_]).
membre(Element,[_|Liste]) :- membre(Element,Liste).

```

Exemple de questions

```

?- membre(b,[a,b,c]).
true ;
false.

```

```

?- membre(X,[a,b,c]).
X = a ;
X = b ;
X = c ;
false.

```

```

?- membre(a,[a,b,a,c]).
true ;
true ;
false.

```

```

?- membre(a,L).
L = [a|_4450] ;
L = [_4448, a|_4456] ;
L = [_4448, _4454, a|_4462] ;
L = [_4448, _4454, _4460, a|_4468] ;
L = [_4448, _4454, _4460, _4466, a|_4474] ;

```

Quelques prédicats sur les listes 2 / 4

supprimer/3

```

/* supprimer(Element,Liste1,Liste2) est vrai lorsque Liste2 possede tous les elements de Liste1 sauf Element */
supprimer(_,[],[]).
supprimer(Element,[Element|Queue],QueueSansElement) :- supprimer(Element,Queue,QueueSansElement).
supprimer(Element,[ElementDifferent|Queue],[ElementDifferent|QueueSansElement]) :- Element\=ElementDifferent,
    supprimer(Element,Queue,QueueSansElement).

```

Exemple de questions

```

?- supprimer(1,[1,2,1,3],L).
L = [2, 3] ;
false.

```

```

?- supprimer(X,[1,2,1,3],[2,3]).
X = 1 ;
false.

```

```

?- supprimer(2,L,[1,3]).
ERROR: Out of local stack
Exception: (1,595,448) supprimer(2, _9572670, [1, 3]) ?

```


Quelques prédicats sur les listes 3 / 4

longueur/2

```
/* longueur(Liste,Longueur) est vrai lorsque Longueur est égal au nombre d'éléments de Liste */  
longueur([],0).  
longueur([_|Queue],Longueur) :- longueur(Queue,LongueurQueue),Longueur is LongueurQueue+1.
```

Exemple de questions

```
?- longueur([1,2,3],3).  
true.  
?- longueur([1,2,3],X).  
X = 3.
```

Quelques prédicats sur les listes 4 / 4

scinder_liste/2

```

/* scinder_liste(Liste,PremierPartie,DeuxiemePartie) est vrai lorsque Liste est scindée en deux liste de même
   longueur (à plus ou moins un près) */
scinder_liste([],[],[]).
scinder_liste(Liste,PremierPartie,DeuxiemePartie) :- append(PremierPartie,DeuxiemePartie,Liste), longueur(
    PremierPartie,LongueurPremierePartie), longueur(DeuxiemePartie,LongueurDeuxiemePartie),
    LongueurPremierePartie =:= LongueurDeuxiemePartie.
scinder_liste(Liste,PremierPartie,DeuxiemePartie) :- append(PremierPartie,DeuxiemePartie,Liste), longueur(
    PremierPartie,LongueurPremierePartie), longueur(DeuxiemePartie,LongueurDeuxiemePartie),
    LongueurPremierePartie+1 =:= LongueurDeuxiemePartie.

```

Exemple de questions

```

?- scinder_liste([1,2,3,4],L1,L2).
L1 = [1, 2],
L2 = [3, 4] ;
false.

```

```

?- scinder_liste([1,2,3],L1,L2).
L1 = [1],
L2 = [2, 3] ;
false.

```

Conclusions

Nous avons dans ce cours

- précisé le vocabulaire du Prolog
- 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

Références

- [Bel94] P. Bellot.
Objectif Prolog.
Masson, 1994.
- [Tri22] Markus Triska.
The power of prolog.
<https://www.metalevel.at/prolog>, 2022.