Conception et implémentation

IFT-17586 Intelligence artificielle I - A2002

Séance 9

- La méta-programmation

© Capus et Tourigny, 2002

Méta-programmation

- Méta-programmes en Prolog :
  - écrire des méta-interpréteurs Prolog
  - implémenter d'autres paradigmes de programmation (orienté objet, par filtrage, etc.)
- Prédicats prédéfinis :
  - call/1
  - clause/2
  - op/3
  - etc.

© Capus et Tourigny, 2002

Méta-programmation

- Le plus simple méta-interpréteur Prolog :

  ?- call(But).

  Vrai si But est vrai

  Faux si But est faux

  Définition du prédicat « prouver/1 » :

  prouver(But) :- call(But).

  Remarque : le travail est laissé à call/1

Méta-programmation

- Autre méta-interpréteur Prolog :

  ?- clause(Tête, Corps).

  Tête = partie gauche (jamais une variable)

  Corps = true si pas de partie droite

  = (But, Autres_Buts) si existe

  – Avantage : permet de connaître les clauses appelées (tracer la résolution)
  – Inconvénient : ne traite pas tout (le prédicat !/0, par exemple)

Méta-programmation

- Méta-interpréteur :

```
prouver( true ) :- !.
prouver( ( But1, But2 ) ) :-
        !, call( But1 ), call( But2 ).
prouver( But ) :-
        clause( But, Corps ), prouver( Corps ).
```

Remarque : avec clause/2, les prédicats devront être déclarés dynamiquement

```
:- dynamic( Prédicat/Arité ).
```


Méta-programmation

- Un résolveur par résolution-réfutation :

$\boxed{\sim a \vee b}$ et $\boxed{\sim b \vee c}$ $\Rightarrow$ $\boxed{\sim a \vee c}$

S'il existe 2 clauses C1 et C2 telles que P est une sous expression disjonctive de C1 et ¬P est une sous expression disjonctive de C2

alors

- enlever P dans C1 (la nouvelle clause est CA)
- enlever ¬P dans C2 (la nouvelle clause est CB)
- ajouter dans la base de clauses la clause (CA $\vee$ CB)

Méta-programmation

- Ajouter la base de clauses :
  - Définir des opérateurs : négation et disjonction

    :- op(100, fy, ~), op(120, xfy, v).
  - Écrire les clauses de la base

    clause(~a v b).

    clause(~b v c).

    clause(a).

    clause(~c).

    Le prédicat clause/1 doit être déclaré dynamiquement.

Méta-programmation

- Définition d'un formalisme à l'aide d'un opérateur : Conditions ---> Actions
  - Déclaration à l'aide du prédicat op/3 :

    op(Priorité, Type, Nom)

    current_op(Priorité, Type, Nom)

    :- op(800, xfx, --->). % début de programme
  - Définition dans le code Prolog :

    demontrer :- Conditions ---> Actions,

    tester(Conditions),

    executer(Actions).

Méta-programmation

```
% Tester toutes les conditions :
tester( [ ] ).
tester( [Condition|Reste] ) :- call( Condition ),tester( Reste ).
% Exécuter toutes les actions jusqu'à stop :
executer( [stop] ) :- !.
executer( [ ] ) :- demontrer.
executer( [Action|Reste] ) :- call( Action ), executer( Reste ).
```


Méta-programmation

S'il existe 2 clauses C1 et C2 telles que
P est une sous-expression disjonctive de C1
et
¬P est une sous-expression disjonctive de C2
alors

- enlever P dans C1 (la nouvelle clause est CA)
- enlever ¬P dans C2 (la nouvelle clause est CB)
- ajouter dans la base de clauses la clause (CA ∨ CB)

⇒ Enlever P dans C sachant que P est une sous-expression disjonctive de C (on obtient CA)

Méta-programmation

```
% Cas général
[   clause( C1 ), effacer( P, C1, CA ),
    clause( C2 ), effacer( ~P, C2, CB ),
    not deja_fait( C1, C2, P ) ]
--->
[   assert( clause( CA v CB ) ),
    assert( deja_fait( C1, C2, P ) ) ].
```


Méta-programmation

```
% effacer( P, C, CA )
% Si on efface une sous expression disjonctive P
% de C alors on obtient CA
effacer(X, X v Y, Y).
effacer(X, Y v X, Y).
effacer(X, Y v Z, Y v ZA) :- effacer(X, Z, ZA).
effacer(X, Y v Z, YA v Z) :- effacer(X, Y, YA).
```

Méta-programmation

S'il existe 2 clauses C1 et C2 telles que
C1 = ¬C2

alors

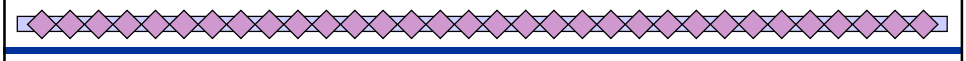
- Les 2 clauses s'annulent (résolvant = {})
- On obtient une contradiction

Méta-programmation

```
% Cas finaux
% si clause vide alors il y a une contradiction
% sinon il n'y a pas de contradiction
[ clause( X ), clause( ~X ) ] --->
        [ write(' Contradiction trouvée '), stop].

[ ] ---> [ write(' Pas de contradiction '), stop].
```

Méta-programmation



```
% Supprimer une clause vraie
% Exemple : ~a v b v a
[ clause(C), effacer(P, C,_), effacer(~P, C, _ ) ] --->
    [ retract( clause(C) )].

% Simplifier une clause
% Exemple : a v ~b v a
[ clause(C), effacer(P, C, CA), effacer(P, CA, _) ] --->
    [ retract( clause( C ) ), assert( clause( CA ) ) ].
```

© Capus et Tourigny, 2002