Conception et implémentation

IFT-17586 Intelligence artificielle I - E2003

Séance 8

- Conception de systèmes à base de connaissances en Prolog

Conception d'un SBC en Prolog

- Prolog : moteur de recherche en chaînage arrière
  - base de faits ⇒ faits Prolog
  - base de règles ⇒ règles de déduction Prolog
  - moteur d'inférence ⇒ moteur de recherche Prolog (questions)
- Construction d'une coquille de développement (méta-programmation)

Conception d'un SBC en Prolog

- Base de faits ⇒ un prédicat fait/1
  fait(F) où F est un fait du domaine
- Base de règles ⇒ ajout d'opérateurs
  si Condition alors Conclusion où :
  - Condition = Fait
  - Condition = Condition1 et Condition2
  - Condition = Condition1 ou Condition2
- Moteur d'inférence ⇒ 2 prédicats
  - ch_arrière/1 : prouver qu'un but est vrai
  - ch_avant/0 : déduire tous les faits possibles

Conception d'un SBC en Prolog

- Base de faits ⇒ un prédicat fait/1
  - Déclarer les prédicats dynamiques pour pouvoir faire des assertions, à l'aide du prédicat dynamic/1

```
%:- dynamic(NomPrédicat/Arité ).
:- dynamic( fait/1 ).
```

Conception d'un SBC en Prolog

- Base de règles ⇒ ajout d'opérateurs
  - Accepter les opérateurs « si, alors, et, ou » en les ajoutant avec le prédicat op/3

```
%:-op(Priorité,Type, Nom)
:-op(   800,    fx,     si ).
:-op(   700,    xfx,    alors ).
:-op(   300,    xfy,    ou ).
:-op(   200,    xfy,    et ).
```

Connaître tous les opérateurs disponibles :
?- current_op(P, T, N).

Conception d'un SBC en Prolog

- Base de connaissances pour le diagnostic automobile (p. 257)

```
%Base de faits
fait( essence(réservoir) ).
fait( essence(carburateur) ).
fait( tourne(moteur) ).
%Base de règles
si essence(moteur) et tourne(moteur) alors problème(bougies).
si not tourne(moteur) et not éclaire(phares) alors problème(batterie_ou_câbles).
si not tourne(moteur) et éclaire(phares) alors problème(démarreur).
si essence(réservoir) et essence(carburateur) alors essence(moteur).
```

Conception d'un SBC en Prolog

- Moteur d'inférence : chaînage arrière
- ?- ch_arriere(But).

But est vrai si But=Fait

ou si Règle = si Condition alors But et

Condition est vraie

- Condition= C1 et C2 : Condition est vraie si C1 est vraie et C2 est vraie
- Condition= C1 ou C2 : Condition est vraie si C1 est vraie ou C2 est vraie

Conception d'un SBC en Prolog

- Moteur d'inférence : chaînage arrière

```
ch_arrière( But ) :- est_vrai( But ).
est_vrai( Proposition ) :- fait( Proposition ).
est_vrai( Proposition ) :-
                si Condition alors Proposition,
                est_vrai( Condition ).
est_vrai( Cond1 et Cond2 ) :-
                est_vrai( Cond1 ), est_vrai( Cond2 ).
est_vrai( Cond1 ou Cond2 ) :-
                est_vrai( Cond1 ) ; est_vrai( Cond2 ).
```

Conception d'un SBC en Prolog

- Moteur d'inférence : chaînage avant

?- ch_avant.

NouveauFait (∉ Base de faits) est déduit
  si Règle=si Condition alors NouveauFait et
    Condition est un fait
      – Condition=Fait
      – Condition=C1 et C2 : Condition est un fait si C1 et C2 sont des faits
      – Condition=C1 ou C2 : Condition est un fait si C1 ou C2 sont des faits

Conception d'un SBC en Prolog

- Moteur d'inférence : chaînage avant

```
ch_avant :-
        nouveau_fait( Nouveau ),
        !,
        write( 'Nouveau fait : ' ), write( Nouveau ), nl,
        assert( fait( Nouveau ) ),
        ch_avant.
ch_avant :-
        write('Plus de nouveaux faits à déduire, '),
        write('la base de connaissances est saturée.'),
        nl.
```

Conception d'un SBC en Prolog

- Moteur d'inférence : chaînage avant

```
nouveau_fait( NouvFait ) :-
        si Condition alors NouvFait,
        not( fait(NouvFait) ),
        recherche_fait( Condition ).

recherche_fait( Condition ) :- fait( Condition ).
recherche_fait( Cond1 et Cond2 ) :-
        recherche_fait( Cond1 ), recherche_fait( Cond2 ).
recherche_fait( Cond1 ou Cond2 ) :-
        recherche_fait( Cond1 ) ; recherche_fait( Cond2 ).
```

