Université Abderrahmane Mira de Bejaia
Faculté des Sciences Exactes
Département d'informatique
Année universitaire : 2022/2023

1ère Année Master SIA (S1) Module : **Intelligence artificielle**

TP4 : Systèmes à Base de Connaissances

Objectif du TP

Il s'agit d'étudier les mécanismes d'un moteur d'inférence en chaînage avant et arrière.

Activité 1

On donne du code partiel, à l'étudiant de corriger et d'apporter les améliorations nécessaires.
Base de connaissances pour le diagnostic automobile

- 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).

En chainage arrière, on démarre d'un but, et on cherche à le démontrer, le raisonnement s'arrête une fois le but démontré ou saturation de toutes les règles, la vérification du but n'est pas possible dans tous les cas.

Voilà comment fonctionne un moteur d'inférence avec chainage arrière, l'étudiant doit vérifier la syntaxe prolog.

La démarche est la suivante :

◆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

En Prolog (à améliorer)

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 ).
```

En chainage avant, on démarre d'un ensemble de règle, ou parfois d'un but, on s'arrête jusqu'à vérification du but (s'il a été défini ou saturation).
Voilà comment fonctionne un moteur d'inférence avec chainage avant, l'étudiant doit vérifier la syntaxe prolog.

La démarche est la suivante :

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
```

En Prolog (à améliorer)

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.
```

Activité 2
On donne une base de règles de test (à titre d'exemple) :
Les tests seront faits sur des requêtes sur la base de connaissances fournie.

- r1 : **si** mange_viande **alors** carnivore
- r2 : **si** dents_pointues **et** griffes **et** yeux_avant **alors** carnivore
- r3 : **si** mange_herbe **alors** non carnivore
- r4 : **si** mammifere **et** sabots **alors** ongule
- r5 : **si** mammifere **et** rumine **alors** ongule
- r6 : **si** mammifere **et** carnivore **et** brun **et** taches **alors** guepard
- r7 : **si** mammifere **et** carnivore **et** brun **et** raies **alors** tigre
- r8 : **si** ongule **et** long_cou **et** longues_pattes **et** taches **alors** girafe
- r9 : **si** ongule **et** raies **alors** zebre
- r10 : **si** oiseau **et** long_cou **et** longues_pattes **et** noir_et_blanc **et** non vol **alors** autruche
- r11 : **si** oiseau **et** nage **et** noir_et_blanc **et** non vole **alors** pingouin
- r12 : **si** oiseau **et** vole **alors** albatros
- r13 : **si** poils **alors** mammifere
- r14 : **si** lait **alors** mammifere
- r15 : **si** plumes **alors** oiseau
- r16 : **si** vole **et** pond_oeufs **alors** oiseau

1. Moteur en chaînage avant
On veut réaliser un moteur d'inférence en chaînage avant et qui ne constitue pas d'ensemble de conflit (dès qu'une règle est déclenchable, elle est déclenchée).

- Définir la base de règles grâce à un prédicat regle :

regle(ri) :- si(Liste de prémisses), alors(Liste de conclusions).

- Définir un prédicat permettant à l'utilisateur d'initialiser la base de faits. On utilisera le prédicat assert pour ajouter vrai(Fait) pour les faits positifs et faux(Fait) pour les faits négatifs.
- Définir un prédicat saturer qui sature la base de règles et produit une trace de son fonctionnement

On vous donne l'algorithme à implémenter en prolog :

```
Changement <-- Vrai
Tant que Changement est Vrai
    Changement <-- Faux
    Boucle sur les règles : soit R une règle de BaseRègles
        Si      R n'est pas marquée et si les prémisses de R appartiennent à
BaseFaits
        Alors   ajouter les conclusions de R à BaseFaits
                changement <-- Vrai
                marquer R
        FinSi
    FinBoucle
FinTantQue
```

Un exemple d'exécution souhaitée est :
?- faits([plumes,non(vole),nage,noir_et_blanc,mange_herbe]).
Yes
?- saturer.
r3
non(vole) non(carnivore) plumes nage noir_et_blanc mange_herbe
r15
non(vole) non(carnivore) plumes nage noir_et_blanc mange_herbe oiseau
r11
non(vole) non(carnivore) plumes nage noir_et_blanc mange_herbe oiseau
pingouin
Yes

2. Moteur en chaînage arrière

• Écrire un moteur d'inférence fonctionnant en chaînage arrière. On représentera la base de règles et la base de faits comme dans la première partie.

Exemple d'exécution :
?- satisfait(zebre).
poils dans la base de faits
mammifere satisfait grace a r13
sabots dans la base de faits
ongule satisfait grace a r4
raies dans la base de faits
zebre satisfait grace a r9
Yes