M1Info/BIA/2011-2012
TP5/6 PROLOG
(PROgrammation LOGique)

Illustration du cours « Système à Base de Connaissances » : Etude pour la réalisation d'un moteur d'inférence en PROLOG

Objectif des deux séances de TP

- **Avertissement** : Ce TP est noté. Vous devez rendre un CR correspondant au plan proposé et un code commenté avant le **5 décembre 2011** à votre encadrant (le tout sous format électronique : .PDF ou .DOC pour le CR et .TXT ou .PL pour les sources commentés).
- **Objectif des deux séances de TP**
  Il s'agit d'étudier les mécanismes d'un moteur d'inférence en chaînage avant simple, arrière et mixte. Les tests seront faits sur des requêtes sur la base de connaissances fournie.

On donne une base de règles de test (à titre d'exemple) :

- 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 d'ordre zéro, fonctionnant en chaînage avant, en régime irrévocable et monotone, et qui ne constitue pas d'ensemble de conflit (dès qu'une règle est déclenchable, elle est déclenchée).

1. Définir la base de règles grâce à un prédicat `regle/1` :

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

2. 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.
3. Définir un prédicat `saturer` qui sature la base de règles et produit une trace de son fonctionnement (on pourra s'inspirer du troisième moteur d'inférence présenté en cours).

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 d'ordre 0 fonctionnant en chaînage arrière. On représentera la base de règles et la base de faits comme dans la première partie.

Exemple d'exécution :

```
2 ?- 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
```

3. Chaînage mixte ou autre amélioration

Dans cette troisième partie du TP, nous vous proposons d'effectuer une amélioration du fonctionnement du moteur d'inférence. Vous pouvez choisir une amélioration qui vous intéresse et la réaliser. Si vous n'avez pas d'idée, nous vous proposons de réaliser un moteur fonctionnant en chaînage mixte selon la description ci-dessous.

On distingue deux catégories de faits :
- les faits « terminaux », qui ne figurent jamais en partie gauche d'une règle,
- les faits « observables », qui ne figurent jamais en partie droite d'une règle.

1. Définir le prédicat `terminal(F)` (respectivement `observable(F)`) vrai si le fait F est terminal (resp. observable).
2. Il est possible que la base de faits fournie par l'utilisateur ne permette pas en chaînage avant de conclure sur un fait terminal. Dans ce cas, on n'a pas répondu à l'utilisateur sur le végétal qu'il observe (dans notre exemple). On veut donc lui poser des questions afin de conclure sur un fait terminal.

Pour cela, on procède en deux temps :
- On cherche une règle « presque déclenchable », c'est-à-dire dont toutes les prémisses sont vraies sauf une portant sur un fait qui est inconnu et qui porte sur un fait observable.
- On essaie de prouver ce fait inconnu en chaînage arrière. Si le chaînage arrière aboutit à essayer de prouver un fait observable inconnu, on pose une question à l'utilisateur sur ce fait observable. L'utilisateur peut répondre « oui », « non », ou « je ne sais pas ». Dès qu'on a une réponse « oui » ou « non » à une de nos questions, on peut ajouter un fait à la base de faits et relancer le moteur. Si l'on n'arrive toujours pas à conclure sur un fait terminal, il faut réitérer le processus. *Attention à ne pas poser deux fois la même question à l'utilisateur.*

Exemple d'exécution :

```
   ?- faits([brun,dents_pointues,griffes]).
   Yes
   ?- go.
   Est-ce que mange_viande ? (o/n/i)
   |: i.
   Est-ce que yeux_avant ? (o/n/i)
   |: o.
   r2
   brun dents_pointues griffes yeux_avant carnivore
   Est-ce que mange_herbe ? (o/n/i)
   |: n.
   Est-ce que poils ? (o/n/i)
   |: o.
   r13
   non(mange_herbe) brun dents_pointues griffes yeux_avant carnivore poils
mammifere
   Est-ce que sabots ? (o/n/i)
   |: n.
   Est-ce que rumine ? (o/n/i)
   |: i.
   Est-ce que taches ? (o/n/i)
   |: o.
   r6
   non(mange_herbe) non(sabots) brun dents_pointues griffes yeux_avant
carnivore poils mammifere taches guepard
   Yes
```

Remerciements
Ce projet est issu du cours de Prolog de Nathalie Guin (Département Informatique - Université Lyon1).