TP5 – Système à base de règles en Prolog

Nathalie Guin, Alexis Lebis, Marie Lefevre, Baptiste Monterrat

ORGANISATION PRATIQUE

Objectif du TP : Illustrer le cours sur les systèmes à base de connaissances.

Livrables du TP : Ce TP prend la forme d'un projet. Il sera noté. Vous devez rendre un Compte Rendu correspondant au plan proposé ci-dessous (fichier nommé BIA2016_TP_Projet_Prolog_XX.pdf où XX indique votre nom de famille), accompagné du code source Prolog commenté. Le tout sera à déposer sur Tomuss sous la forme d'un fichier ZIP.

Date limite de rendu : Le vendredi 19/12/2016 minuit. Le dépôt restera ouvert jusqu'au lundi 02/01/2017 21h. Ensuite, tout retard sera pénalisé.

Contenu du CR :

- L'auteur : Nom, Prénom, Numéro d'étudiant ;
- Pour chaque partie :
  - Ce qui a été programmé par rapport au sujet (ce qui tourne et ce qui ne tourne pas) ;
  - Les questions que vous vous êtes posées, les choix que vous avez faits ;
  - Les jeux de tests effectués.

Organisation du travail : Le travail se fait de manière **individuelle**. Vous devez préparer le projet avant la séance du 13/12/2016 afin de pouvoir poser des questions à votre encadrant de TP et/ou lui montrer ce que vous avez fait.

Evaluation : Vous ferez vos tests sur le premier jeu de tests fourni puis vous définirez la base de connaissances pour le problème proposé. Votre code du moteur d'inférences doit donc être générique.

SYSTEMES A BASE DE REGLES

On considère des problèmes que l'on peut résoudre à l'aide d'un système à base de règles. Nous allons dans un premier temps définir un moteur d'inférences à chaînage avant, à chaînage arrière puis à chaînage mixte, que nous appliquerons sur un problème jouet. Vous définirez ensuite la base de règles relative à un problème de tourisme. Attention, votre moteur doit être générique et fonctionner sur le problème jouet, le problème de tourisme mais également sur un autre jeu de test (une autre base de règles) que nous utiliserons pour tester votre TP.

PARTIE 1 : MOTEUR EN CHAÎNAGE AVANT

Notre problème jouet est décrit comme suit. Nous disposons de 6 règles :

- R1 : A et B → C
- R2 : C et non(D) → F
- R3 : F et B → E
- R4 : F et A → non(G)
- R5 : non(G) et F → B
- R6 : A et H → L

Nous connaissons trois faits : A, C et D, et nous voulons démonter le fait E.

Nous allons réaliser un moteur d'inférences en chaînage avant qui fonctionne comme sur l'exemple suivant :

```
?- raz.                    /* on vide la base de faits */
true.

?- faits([a,c,non(d)]).    /* on charge les nouveaux faits */
true.

?- saturer.   /* on lance le moteur en chainage avant */
r2 : non(d) a c f
r4 : non(d) non(g) a c f
r5 : non(d) non(g) a c f b
r1 : non(d) non(g) a c f b
r3 : non(d) non(g) a c f b e
true.
```

Pour réaliser ce moteur d'inférences en chaînage avant, il faut :

Question 1 (1 pts) : Définir la base de règles sous forme de listes composées de listes de prémisses et de listes de conclusions, grâce à un prédicat `regle/1` :

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

Question 2 (1 pts) : 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.

Question 3 (1 pts) : Définir un prédicat permettant à l'utilisateur de vider la base de faits. On utilisera le prédicat `retractall` pour supprimer les faits.

Question 4 (5 pts) : Définir un prédicat `saturer` qui sature la base de règles et produit une trace de son fonctionnement. L'algorithme utilisé pour ce moteur sera le suivant :

```
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
```

Rappel 1 : La seule boucle qui existe en Prolog est la boucle mue par l'échec.

Rappel 2 : Lorsque vous voulez utiliser un prédicat relatant un fait (par exemple `vrai/1`) dans un autre prédicat, mais que votre base de faits n'est pas encore remplie, vous devez le déclarer dynamiquement :

```
:- dynamic vrai/1, faux/1.
```

PARTIE 2 : MOTEUR EN CHAINAGE ARRIERE

Question 5 (5 pts) : Écrire un moteur d'inférences d'ordre 0 fonctionnant en chaînage arrière. On représentera la base de règles et la base de faits comme dans la première partie.

Pour rappel, le chainage arrière consiste à assigner un but au système et à unifier la partie conclusion des règles avec ce but. Les nouveaux buts à démontrer sont alors les prémisses de la règle sélectionnée. Si les prémisses ne sont pas dans la base de faits, ni démontrables *via* une autre règle, vous ferez échouer la démonstration.

Sur notre problème jouet, son exécution pourra donner par exemple :

```
?- raz.                    /* on vide la base de faits */
true.

?- faits([a,c,non(d)]).    /* on charge les nouveaux faits */
true.

?- satisfait(e) .          /* on cherche à démontrer E */
c dans la base de faits
non(d) dans la base de faits
f satisfait grace a r2
c dans la base de faits
non(d) dans la base de faits
f satisfait grace a r2
a dans la base de faits
non(g) satisfait grace a r4
c dans la base de faits
non(d) dans la base de faits
f satisfait grace a r2
b satisfait grace a r5
e satisfait grace a r3
true

?- satisfait(l).         /* on cherche à démontrer L */
a dans la base de faits
false.
```

PARTIE 3 : MOTEUR A CHAINAGE MIXTE

Nous vous proposons d'effectuer une amélioration du fonctionnement du moteur d'inférences en mettant en œuvre un 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.

Question 6 (2 pts) : Définir le prédicat `terminal(F)` (respectivement `observable(F`)) qui est vrai si le fait F est terminal (resp. observable).

Il est possible que la base de faits fournie par l'utilisateur ne permette pas, en inférant en chaînage avant, de conclure sur un fait terminal. Dans ce cas, on ne peut pas répondre à l'utilisateur (par exemple pour prouver que L est vrai dans notre problème jouet). 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.
- 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.

Question 7 (5 pts) : Mettre en œuvre ce chainage mixte et les interactions nécessaires avec l'utilisateur.

Sur notre problème jouet, en rajoutant la règle R7 : E et L → M, son exécution donnera :

```
?- raz.
true.

?- faits([a,c,non(d)]).
true.

?- go.
r2 : non(d) a c f
r4 : non(d) non(g) a c f
r5 : non(d) non(g) a c f b
r1 : non(d) non(g) a c f b
r3 : non(d) non(g) a c f b e

j essaie de prouver h

Est-ce que h ? (o/n/i)
|: o.
h car observé
r6 : non(d) non(g) a c f b e h l
r7 : non(d) non(g) a c f b e h l m

true .
```

PARTIE 3 : NOUVELLE BASE DE CONNAISSANCES

Question 8 (1 pts) : Tester votre moteur d'inférences amélioré sur la base de règles ci-dessous :

- R1 : Si belle ville et très bon restaurants alors ville méritant le voyage.
- R2 : Si ville historique alors ville méritant le voyage.
- R3 : Si autochtones accueillants et traditions folkloriques alors ville méritant le voyage.
- R4 : Si monuments et végétation abondante alors belle ville.
- R5 : Si tradition culinaire alors bons restaurants.
- R6 : Si restaurants 3 étoiles alors très bons restaurants.
- R7 : Si restaurants 3 toques alors très bons restaurants.
- R8 : Si musées et ville ancienne alors ville historique.
- R9 : Si Provence et bord de mer alors autochtones accueillants.
- R10 : Si parcs verdoyants et avenues larges alors végétation abondante.

Question 9 (1 pts) : Qu'obtenez-vous en chainage avant avec la base de faits suivante :

- Parcs verdoyants
- Avenues larges
- Monuments
- Restaurants 3 toques
- Ville ancienne

Question 10 (1 pts) : Qu'obtenez-vous avec la même base de faits et le but suivant :

- Mérite le voyage

Question 11 (1 pts) : Complétez la base de règles pour décrire les grandes villes françaises afin de trouver les villes qui valent le détour !