2016-2017

Mathilde
Boltenhagen
11610748

TP5 - Système à base de règles en Prolog

Dans ce pdf, j'indique comment j'ai créé mes prédicats, disponibles dans le fichier BIA2016_TP_Projet_Prolog_Boltenhagen.pl.

PARTIE 1 : MOTEUR EN CHAINAGE AVANT

Question 1 :

Comme indiqué en TD, une solution alternative est de donner la règle sans contenu, comme un fait du type : **regle(nom,Liste_premisses,Liste_conclusions).**

Question 2 :

En suivant la forme de la fonction **faits(L)** de l'exemple, on met à vrai les éléments qui n'ont pas 'non' et inversement.

Question 3 :

En utilisant retractall, je vide la base des faits ainsi que les potentiels marqueurs utilisés dans la suite du TP de la manière suivante :

retractall(predicat(_))

Question 4 :

Dans mon prédicat **saturer2**, changement est défini par drapeau(changement) qui existe quand le changement de l'algorithme donné est faux et n'existe pas quand il est vrai.

Je crée un prédicat **valide(L)** retourne true si les éléments de L sont dans la base de faits. De même je crée **conclue(L)** qui met dans la base de faits.

Je crée un prédicat **affiche(X)** qui affiche non(X) ou X selon si X est vrai ou faux dans la base des faits.

PARTIE 2 : MOTEUR EN CHAINAGE ARRIÈRE

Question 5 :

Mon prédicat **satisfait(X) n**ote que X est dans la base de faits si c'est le cas et cherche à prouver X en fonction des règles où il est défini en conclusion sinon.

Ainsi le prédicat **satisfait([X|L])** permet de parcourir les variables en les parcourant en chainage arrière.

PARTIE 3 : MOTEUR EN CHAINAGE MIXTE

Question 6 :

Selon la définition de terminal et observable donné, j'ai défini **figure_droite(R,F)** et **figure_gauche(R,F)** qui indique si F est dans le membre de droite et respectivement de gauche de la règle R.

Avec ces prédicats, **auxobservable(F)** est une fonction auxiliaire appliquant figure_gauche à toutes les règles.

Ainsi **observable(F)** et **terminal(F)** vérifient respectivement si F est uniquement à droite ou uniquement à gauche dans toutes les règles.

Question 7 :

L'idée de la question 7 est de réprendre les deux chainages et de les combiner. Ainsi je crée **go()** qui commence par saturer la base des faits en chainage avant puis à l'aide d'une réponse de l'utilisateur, je conclue (ou non) de nouveaux faits par le chainage arrière. C'est le prédicat nommé **demander()** qui s'occupe de cette deuxième partie.

Comme l'indique l'énoncer, nous souhaitons demander à l'utilisateur son aide sur une variable uniquement si elle est la seule inconnue dans les pressmisses d'une règle. J'ai donc créé un prédicat **presque_declanchable(R,E)** qui vérifie que E est le seul non valide de R. La vérification se fait par l'intermédiaire du prédicat **seul_valide(L,E)** qui vérifie pour tous les premisses de R.

Le prédicat **etudie_reponse(R,E)** se contente de rentrer le fait suite à la réponse de l'utilisateur.

PARTIE 4 : NOUVELLE BASE DE CONNAISSANCES

Question 8 :

voir le fichier

Question 9 :

En lancant la base de faits demandé par la commande :

faits([parcsVerdoyants,avenuesLarges,monuments,restaurants3Toques,villeAncienne]).

J'obtiens en chainage avant :

saturer.
r7:parcsVerdoyants avenuesLarges monuments restaurants3Toques villeAncienne tresBonsRestaurants
r10:parcsVerdoyants avenuesLarges monuments restaurants3Toques villeAncienne tresBonsRestaurants vegetationAbondante
r4:parcsVerdoyants avenuesLarges monuments restaurants3Toques villeAncienne tresBonsRestaurants vegetationAbondante belleVille
r1:parcsVerdoyants avenuesLarges monuments restaurants3Toques villeAncienne tresBonsRestaurants vegetationAbondante belleVille villeMeritantVoyage

Question 10 :

En chainage arrière avec villeMeritantVoyage, on a :

raz.
faits([parcsVerdoyants,avenuesLarges,monuments,restaurants3Toques,villeAncienne]).
satisfait(villeMeritantVoyage).
monuments dans la base de faits
parcsVerdoyants dans la base de faits
avenuesLarges dans la base de faits
vegetationAbondante satisfait grace a r10
belleVille satisfait grace a r4
restaurants3Toques dans la base de faits
tresBonsRestaurants satisfait grace a r7
villeMeritantVoyage satisfait grace a r1

Question 11 :

Le but de la base des faits et de dire quelles villes valent le détour selon des critères que l'on définit. Par exemple si avoir des très bons restaurants et être une belle ville sont des bons points alors Strasbourg est une ville à visiter.

Ainsi j'ai rajouté trois règles définies par :

R11 : Si belle ville et très bons restaurants alors Strasbourg
R12 : Si végétation abondante, monuments et très bons restaurants alors Lyon
R13 : Si ville historique alors Dôle
R14 : Si mauvais temps alors Brest

En supposant que le mauvais temps ne soit pas en faveur d'une ville à visiter.

En rajoutant des prédicats indiquants que Strasbourg, Brest, Dôle et Lyon étaient des villes, je peux afficher la liste des grandes villes françaises selon l'exercice grâce à mon prédicat :

```
grandesVillesFrancaises(X):-ville(X),vrai(X).
```

En répondant "i." pour ville historique et musées et "n." pour mauvais temps, on obtient :

```
grandesVillesFrancaises(X).
X=Lyon;
X=strasbourg;
```

Valider ville historique ou musées impliquerait Dôle et valider mauvais temps impliquerait Brest.