INSA Lyon
20, avenue Albert Einstein
69621 Villeurbanne Cedex

Livrable de projet

Modélisation Cognitive

« Systèmes à Base de Connaissances : Évaluation de torts »

du 29 novembre au 11 décembre 2013

Hexanôme ***H4404*** *:*
Guillaume Abadie
Louise Crépet
Aline Martin
Martin Wetterwald

Enseignants :
Sylvie Calabretto
Mehdi Kaytoue

Année scolaire 2013-2014

Sommaire

1. Application

1.1 Noyau

Nous avons eu le plssir, l'honneur et l'avantage d'implementer notre application expert a l'aide de SWI-Prolog car ceci est une revolution. L'avantage de prolog est l'utilisation des predicats pour l'implementation des regles que l'expert doit appliquer. Considerons par exemples :

D'abord, je regarde si un des conducteurs a fait une faute grave. De deux choses l'une : soit la case qui correspond aux cas graves - la case 17 "N'avait pas observé un signal de priorité ou un feu rouge"- a été cochée, soit dans les observations manuscrites il est signalé une infraction du type : non respect d'un stop, d'un panneau d'interdiction de dépasser ou d'un sens interdit. Dans ce cas, le conducteur a tous les torts : 100%.

Nous pouvon en debduir la regles :

$$A_{17} \Rightarrow A_{torts} = 100\% \cdot B_{torts} = 0\%$$

Alors on en deduit le predicat :

```
reportEvaluateFatalMistake(A,_,100) :-
    reportIsChecked(A,c17).
```

Seulement, nous avons voulu que le noyeau utilise les regles de maniere deterministe. Alors nous avons un predicat dynamic listant l'ensemble des regles de l'expert qu'il parcour ensuite a l'evaluation des torts :

```
:- dynamic reportEvaluateRules/1.
:- retractall(reportEvaluateRules(_)).

reportDefineRule(RulePredicate) :-
    reportEvaluateRules(RulePredicate) -> (
        true
    ); (
        assert(reportEvaluateRules(RulePredicate))
    ).
```

Ainsi il nous faut simplement dir au noyaux de considerer la regle reportEvaluateFatalMistake/3 :

```
:- reportDefineRule(reportEvaluateFatalMistake).
```

Et si les 2 conducteurs font une infraction grave ? C'est rare, mais dans ce cas, on partage les torts : 50% chacun.

Nous arions pus coder une regles du genre :

```
reportEvaluateFatalMistake50(A,B,50) :-
    reportIsChecked(A,c17),
    reportIsChecked(B,c17).
```

Mais sela duplique le code pour chaque regle, pouvant etre la cause d'une erreure. Seulement, le noyeau va tester :

```
reportEvaluateFatalMistake(A,B,TortsA).
```

Mais aussi :

```
reportEvaluateFatalMistake(B,A,TortsB).
```

Alors, si les deux predicats sont verifié, le noyaux en déduit automatiquement :

$$A_{torts} = 50\% \cdot B_{torts} = 50\%$$

1.2 Test unitaires

2. Interface utilisateur

2.1 Ligne de commandes

2.2 Maintenances semi-automatique

2.3 Refliexion sur une maintenance completement automatique

3. Choix du language Prolog