



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

<b>1</b>	<b>Application</b>	<b>1</b>
1.1	Noyau . . . . .	1
1.2	Test unitaires . . . . .	2
<b>2</b>	<b>Interface utilisateur</b>	<b>3</b>
2.1	Ligne de commandes . . . . .	3
2.2	Maintenances semi-automatique . . . . .	3
2.3	Reflexion sur une maintenance complètement automatique . . . . .	3
<b>3</b>	<b>Choix du langage Prolog</b>	<b>4</b>

# 1. Application

## 1.1 Noyau

Nous avons eu le plaisir, l'honneur et l'avantage d'implémenter notre application expert à l'aide de SWI-Prolog car ceci est une révolution. L'avantage de prolog est l'utilisation des prédicats pour l'implémentation des règles que l'expert doit appliquer. Considérons 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 pouvons en déduire les règles :

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

Alors on en déduit le prédicat :

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

Seulement, nous avons voulu que le noyau utilise les règles de manière déterministe. Alors nous avons un prédicat dynamique listant l'ensemble des règles de l'expert qu'il parcourt ensuite à l'évaluation des torts :

```
:- dynamic reportEvaluateRules/1.  
:- retractall(reportEvaluateRules(_)).  
  
reportDefineRule(RulePredicate) :-  
    reportEvaluateRules(RulePredicate) -> (  
        true  
    ); (  
        assert(reportEvaluateRules(RulePredicate))  
    ).
```

Ainsi il nous faut simplement dire aux noyaux de considérer la règle 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 erreur. Seulement, le noyau 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

A chaque implémentation d'une règle, un test unitaire est codé : il construit un rapport avec certaines cases cochées, et on donne la règle qui doit s'exécuter.

En cas de collision, la procédure qui exécute tous les tests s'arrête, et donne la règle qui a effectivement été utilisée : soit la nouvelle règle est redondante et est supprimée, soit l'une des règles doit être précisée. Généralement, on lui ajoute le prédicat "not(autreRegle)".

Par exemple, la règle 132 qui donne les torts quand l'accident a eu lieu alors que les deux véhicules roulaient en sens inverse entrain en collision avec la règle 131 qui donne les torts si l'accident se produit parce que l'un des véhicules a franchi l'axe médian, alors que les deux roulaient en sens inverse. La règle 132, qui avait cette forme :

```
reportRule132(A,B,50) :-
    reportReversedWays(A,B).
```

Est devenue :

```
reportReversedWays(A,B),
    not(reportRule131(A,B,_)).
```

## 2. Interface utilisateur

2.1 Ligne de commandes

2.2 Maintenances semi-automatique

2.3 Reflexion sur une maintenance complètement automatique

### 3. Choix du langage Prolog