

Système expert

MOUDIR Mohamed, SELMANE Madjid, ANNOU Rayane,
OULD KACI Mayas
Université de Caen

April 17, 2024

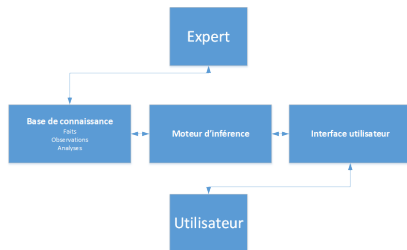
Introduction

Nous aborderons dans cette présentation un système expert 0+ :

- ▶ Qu'est-ce qu'un système expert 0+.
- ▶ Domaine d'utilisation.
- ▶ Architecture de notre projet.
- ▶ Guide d'utilisation.
- ▶ Test unitaire.

Qu'est-ce qu'un système expert 0+

- Un système expert est une application en intelligence artificielle symbolique qui utilise des faits connus et des règles d'inférence pour déduire de nouvelles informations ou faits.



Domaine d'utilisation

- ▶ Les systèmes experts peuvent être appliqués dans pratiquement tous les domaines où des connaissances spécialisées peuvent être formalisées et utilisées pour la prise de décision ou le raisonnement. Voici quelques exemples :
- ▶ Médecine : Pour l'aide au diagnostic médical en analysant les symptômes et en proposant des traitements appropriés.
- ▶ Finance : Pour l'analyse des investissements, la gestion des risques et la détection de fraudes.
- ▶ Systèmes d'information : Pour la gestion des connaissances, la recherche d'informations et la recommandation de contenu pertinent.

Architecture de notre projet

Notre projet se décompose en 3 parties :

- ▶ La Grammaire
- ▶ Moteur d'inférence
- ▶ Interface graphique

La grammaire

- ▶ La grammaire des règles dans un système expert définit la syntaxe et la structure des règles qui sont utilisées pour déduire des conclusions à partir des faits. Elle est essentielle pour assurer la cohérence et la compréhensibilité des règles dans la base de connaissances

La grammaire

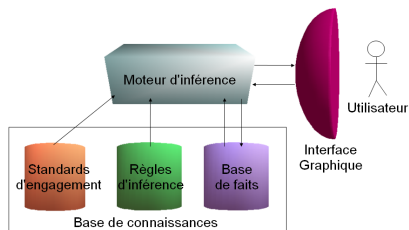
- ▶ Toutes les déclarations et les règles finissent par un point suivie d'un saut de ligne.
- ▶ Les faits déclarés sont séparés par des virgules.
- ▶ Il ne doit pas y avoir d'espaces ou de tirets dans un fait composé de plusieurs mots ; un fait est écrit en camel case.
- ▶ Il faut déclarer les faits booléens d'abord, ensuite les faits entiers, puis les faits symboliques pour assurer le bon fonctionnement du système.

Exemple de base de connaissance

```
main.tex      x      main.tex      x      baseDeConnaissance.
1 Faits_booleens : avoirFaitPrison , habiteChateau , intelligent , fort , riche , grand , lourd , malhonnete , parentsRiches , pauvre , travailleur , chercheur , thesard , l
2 Faits_entiers : fortuneEnfant , fortune , fortuneParents , poids , taille , travailParJour , combien .
3 Faits_symboliques : profession .
4
5 ~ chercheur .
6 ~ lthesard .
7 ~ profession = medecin .
8 ~ poids = (-46 + 95) .
9 ~ fortuneParents = (1000/10 * (25 + 5 + 80) / 3 + 50 * (12 - 5) + 14000 / 2) - 1000 .
10
11 >> malhonnete & fort ==> riche .
12 >> parentsRiches & intelligent ==> riche .
13 >> travailleur | intelligent ==> riche .
14 >> fortune > 10000 ==> riche .
15 >> profession = medecin ==> riche .
16 >> profession = informaticien ==> riche .
17 >> profession = fonctionnaire ==> pauvre .
18 >> !habiteChateau ==> pauvre .
19 >> riche ==> !pauvre .
20 >> pauvre ==> !riche .
21 >> avoirFaitPrison ==> malhonnete .
22 >> grand & lourd ==> fort .
23 >> taille > 185 ==> grand .
24 >> poids > 95 ==> lourd .
25 >> poids < 50 ==> leger .
26 >> poids < 50 ==> fortune = (2 * 1000) - 1 .
27 >> fortuneParents > 10000 ==> parentsRiches .
28 >> travailParJour > 8 ==> travailleur .
29 >> travailParJour > 15 & pauvre ==> profession = salarie .
30 #|
```


Moteur d'inférence

- ▶ Dans cette partie, nous avons implémenté :
- ▶ Une représentation des faits et des règles.
- ▶ Moteur d'inférence.



Représentation des faits et des règles

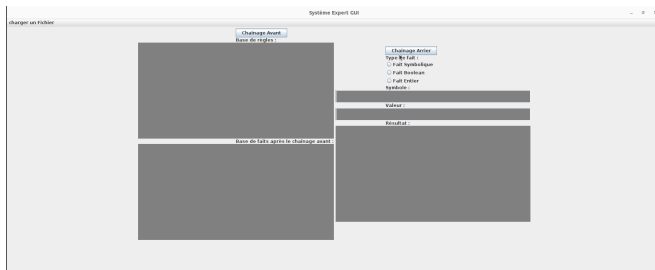
- ▶ Pour les faits, on a trois types selon leur valeur : fait entier, fait symbolique, fait booléen.
- ▶ Pour les règles, on a deux types :
- ▶ Règles sans prémisse : c'est une règle sans condition qui contient juste une conclusion, par exemple "profession = médecin".
- ▶ Règles avec prémisse : c'est une règle qui est composée de deux parties, d'une condition et d'une conclusion, par exemple "si poids > 50 alors léger".

Moteur d'inférence

- ▶ Le moteur d'inférence est le cœur d'un système expert, responsable de l'application des règles d'inférence pour déduire de nouveaux faits à partir des faits existants. Il existe deux principaux types de moteurs d'inférence : le chaînage avant et le chaînage arrière.
- ▶ Chaînage avant : à partir d'une base de connaissances existante, le moteur déduit de nouveaux faits pour les ajouter à la base des faits.
- ▶ Chaînage arrière : il part d'un objectif à atteindre en essayant de le prouver.

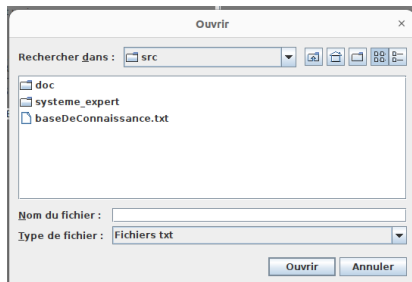
Fonctionnement du Projet

Voici un aperçu de l'interface graphique de notre système expert, conçue pour offrir une expérience utilisateur intuitive et efficace.



Inclusion du Fichier de Base de Connaissance

- ▶ L'utilisateur doit importer le fichier `baseDeConnaissances.txt` qui contient la base de connaissances initiale.
- ▶ Après l'importation, les règles de la base de connaissances seront affichées dans la section "Base de Règles".



Chaînage Avant

- ▶ L'utilisateur choisit le type de chaînage (avant/arrière) qu'il souhaite utiliser.
- ▶ Si l'utilisateur sélectionne le chaînage avant :
- ▶ Une nouvelle base de faits, contenant les faits déduits après l'exécution du chaînage avant, sera affichée dans la section dédiée.

Chainage Avant
Base de règles :

```
Regle sans premisses: chercheur est vrai
Regle sans premisses: thesard est faux
Regle sans premisses: profession = medecin
Regle sans premisses: poids est (46.0 + 95.0)
Regle sans premisses: fortuneParents est 11016.666666666666 - 1000.0
Si malhonnete est vrai fort est vrai ALORS riche est vrai
Si parentsRiches est vrai intelligent est vrai ALORS riche est vrai
Si travailleur est vrai intelligent est vrai ALORS riche est vrai
Si fortune >10000.0 ALORS riche est vrai
Si profession = medecin ALORS riche est vrai
Si profession = informaticien ALORS riche est vrai
Si profession = fonctionnaire ALORS pauvre est vrai
Si habiteChateau est faux ALORS pauvre est vrai
Si riche est vrai ALORS pauvre est faux
Si pauvre est vrai ALORS riche est faux
Si avoirFaitPrison est vrai ALORS malhonnete est vrai
Si grand est vrai lourd est vrai ALORS fort est vrai
Si taille >185.0 ALORS grand est vrai
Si poids >95.0 ALORS lourd est vrai
Si poids <50.0 ALORS leger est vrai
Si poids <50.0 ALORS fortune est 2000.0 - 1.0
Si fortuneParents >10000.0 ALORS parentsRiches est vrai
Si travailParJour >8.0 ALORS travailleur est vrai
Si travailParJour >15.0 pauvre est vrai ALORS profession = salaire
```

Base de faits après le chaînage avant :

Base De Faits:
profession(medecin);
FaitEntier(symbole='fortuneParents', valeur=10016.666666666666);
FaitBooleen(symbole='parentsRiches', valeur=true);
FaitBooleen(symbole='lourd', valeur=true);
FaitEntier(symbole='poids', valeur=141.0);
FaitBooleen(symbole='chercheur', valeur=true);
FaitBooleen(symbole='pauvre', valeur=false);
FaitBooleen(symbole='thesard', valeur=false);
FaitBooleen(symbole='riche', valeur=true);

Chaînage Arrière

- ▶ Si l'utilisateur opte pour le chaînage arrière :
- ▶ Il devra spécifier le type de fait qu'il souhaite étudier, en précisant le symbole et sa valeur.
- ▶ Le système analysera alors si le fait spécifié peut être déduit à partir des règles et de la base de faits actuelle.
- ▶ Le résultat sera affiché pour indiquer si le fait peut être confirmé ou non.

Chainage Arrière

Type de fait :

☐ Fait Symbolique

☒ Fait Boolean

☐ Fait Entier

Symbole :

pauvre

Valeur :

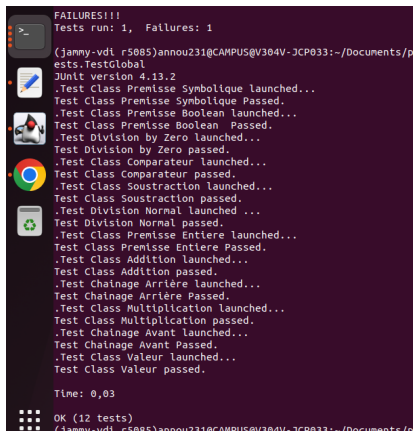
vrai

Résultat :

le fait FaitBoolean{symbole='pauvre', valeur=false} n'est pas prouver

Tests

- ▶ Pour assurer la fiabilité et la robustesse de notre système expert, nous avons développé des tests unitaires ciblant les méthodes et classes les plus essentielles de notre application.
- ▶ Nous avons utilisé JUnit 4 pour l'implémentation des tests.
- ▶ Chaque classe importante a son propre test unitaire.
- ▶ De plus, nous avons mis en place un test global nommé "GlobalTest", qui regroupe l'ensemble de ces tests unitaires.



```
FAILURES!!!
Tests run: 1, Failures: 1

(jammy-vdi r5085)annou231@CAMPUS/V304V-JCP033:~/Documents/p
ests.TestGlobal
JUnit version 4.13.2
.Test Class Premisse Symbolique Launched...
.Test Class Premisse Symbolique Passed.
.Test Class Premisse Boolean Launched...
.Test Class Premisse Boolean Passed.
.Test Class Premisse Boolean Passed.
.Test Division by Zero Launched...
.Test Division by Zero passed.
.Test Class Comparateur Launched...
.Test Class Comparateur passed.
.Test Class Soustraction Launched...
.Test Class Soustraction passed.
.Test Division Normal Launched ...
.Test Division Normal passed.
.Test Class Premisse Entiere Launched...
.Test Class Premisse Entiere Passed.
.Test Class Addition Launched...
.Test Class Addition passed.
.Test Chainage Arrière Launched...
.Test Chainage Arrière Passed.
.Test Class Multiplication Launched...
.Test Class Multiplication passed.
.Test Chainage Avant Launched...
.Test Chainage Avant Passed.
.Test Class Valeur Launched...
.Test Class Valeur passed.

Time: 0,03

OK (12 tests)
(jammy-vdi r5085)annou231@CAMPUS/V304V-JCP033:~/Documents/p
```