Systèmes Experts

Annou Rayane
Mohamed Moudir
Madjid Selmane
Mayas Ould Kaci

April 12, 2024

UNIVERSITÉ
CAEN
NORMANDIE

Contents

1 Architecture d'un Système Expert **3**

2 Bases de Connaissance **4**
2.1 Bases de Règles 4
2.1.1 Règles avec Prémisses 4
2.1.2 Règles sans Prémisses 4

3 Bases de Faits **5**

4 Grammaire des Règles **7**

5 Fonctionnement des Systèmes Experts **8**
5.1 CHAINAGE-AVANT 9
5.2 CHAINAGE-ARRIÉRE 10

6 Tests **12**

7 Interface Graphique et Fonctionnalités **15**
7.1 Interface Graphique 15
7.2 Fonctionnalités 15

8 Compilation et Exécution des Tests **16**

Introduction

Les systèmes experts sont des programmes informatiques conçus pour imiter le raisonnement humain dans des domaines spécifiques. Ils utilisent une base de connaissances pour résoudre des problèmes complexes en appliquant des règles logiques.

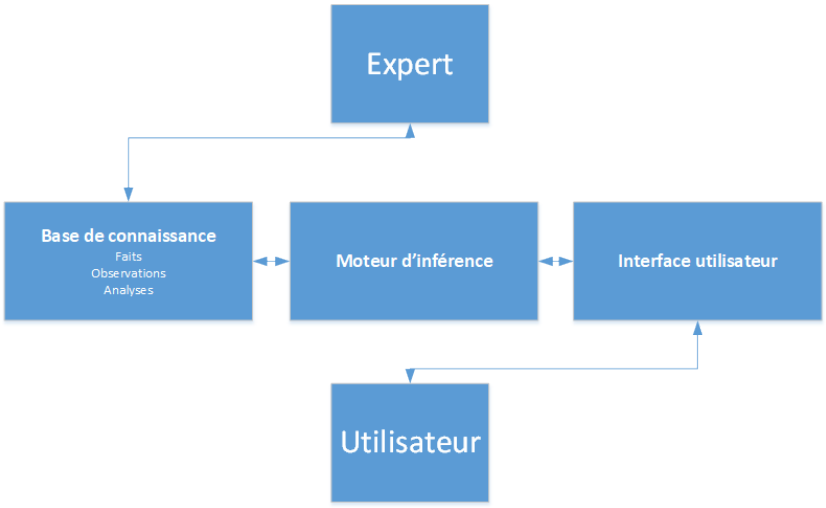

Figure 1: Systèmes experts

1 Architecture d'un Système Expert

Un système expert est généralement composé des éléments suivants :

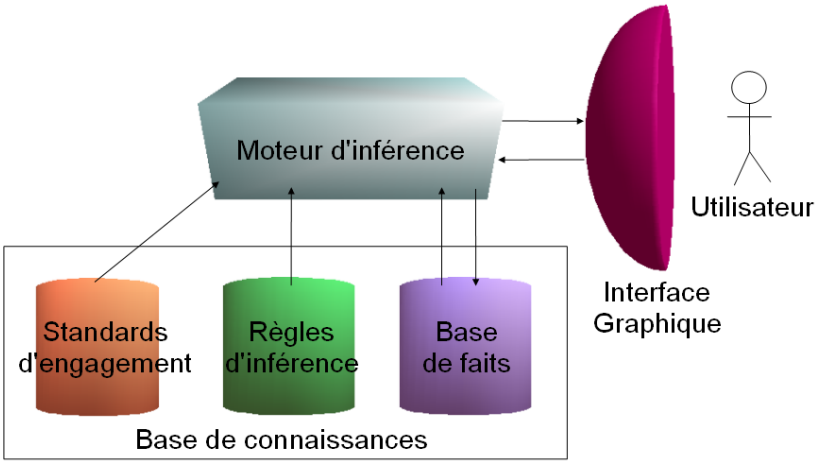

Figure 2: Architecture d'un Système Expert

- **Base de Connaissances** : Contient les faits et les règles sur le domaine spécifique.
- **Moteur d'Inférence** : Responsable de l'inférence logique en utilisant les règles de la base de connaissances.

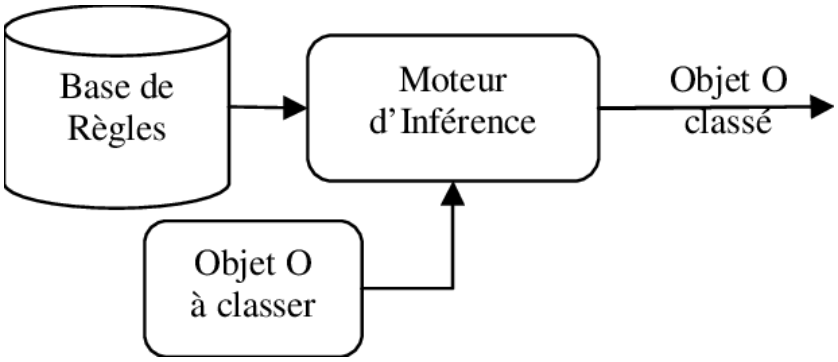

Figure 3: Moteur d'Inférence

- **Interface Utilisateur** : Permet à l'utilisateur de poser des questions au système expert et d'obtenir des réponses.

2 Bases de Connaissance

2.1 Bases de Règles

Les bases de règles constituent un élément central dans les systèmes experts. Elles sont utilisées pour formaliser la logique et le raisonnement qui sous-tendent le fonctionnement du système. Une base de règles est composée de différentes règles logiques qui décrivent les relations entre les faits et les conclusions.

2.1.1 Règles avec Prémisses

Les règles avec prémisses sont des règles logiques qui déclenchent une action seulement si certaines conditions, appelées prémisses, sont remplies. Voici un exemple de règle avec prémisses :

SI la température est élevée **ALORS** activer le système de refroidissement.

Dans cet exemple, la prémisses "la température est élevée" doit être vraie pour que l'action "activer le système de refroidissement" soit déclenchée.

2.1.2 Règles sans Prémisses

Les règles sans prémisses, ou règles de déduction directe, déclenchent une action indépendamment de toute condition. Elles sont souvent utilisées pour des conclusions qui peuvent être tirées directement à partir des faits disponibles. Voici un exemple de règle sans prémisses :

SI le diagnostic est la grippe **ALORS** prescrire des antipyrétiques.

Dans cet exemple, la règle déclenche l'action "prescrire des antipyrétiques" dès que le diagnostic de grippe est posé, sans condition préalable.

Les bases de règles peuvent contenir une combinaison de règles avec et sans prémisses, en fonction de la complexité du domaine d'application et des besoins du système expert.

En résumé, les bases de règles fournissent une structure logique pour la prise de décision et le raisonnement automatisé dans les systèmes experts, en utilisant à la fois des règles avec et sans prémisses pour déclencher des actions en réponse à des situations spécifiques.

En résumé, les bases de règles jouent un rôle crucial dans les systèmes experts en fournissant une structure logique pour la prise de décision et le raisonnement automatisé.

Dans l'exemple donné :

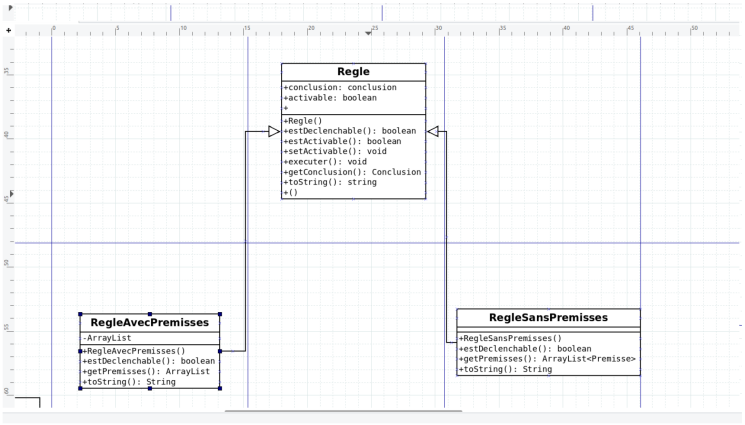

Figure 4: CDiagramme de class Regle

3 Bases de Faits

Les bases de faits sont des ensembles de données utilisées dans les systèmes experts pour représenter des informations sur le domaine spécifique. Elles peuvent être de différents types, notamment des faits booléens, des faits entiers et des faits symboliques.

- **Faits_booleens** : Cette base de faits contient des informations sous forme de variables booléennes, qui sont vraies ou fausses. Par exemple, les variables *avoirFaitPrison*, *habiteChateau*, *intelligent*, etc., représentent des caractéristiques ou des états d'individus dans le domaine spécifique.

- **Faits_entiers** : Cette base de faits contient des informations numériques entières. Par exemple, les variables *fortuneEnfant*, *fortune*, *fortuneParents*, etc., représentent des valeurs numériques telles que la fortune d'un individu ou de ses parents, le poids, la taille, etc.

- **Faits_symboliques** : Cette base de faits contient des informations sous forme de symboles ou de chaînes de caractères. Par exemple, la variable *profession* représente la profession ou le métier d'un individu.

- Les faits booléens incluent des informations sur les caractéristiques des individus, comme le fait d'être chercheur, thesard, malhonnête, etc.

- Les faits entiers incluent des informations numériques telles que la fortune, le poids, la taille, etc.

- Les faits symboliques incluent des informations sur la profession.

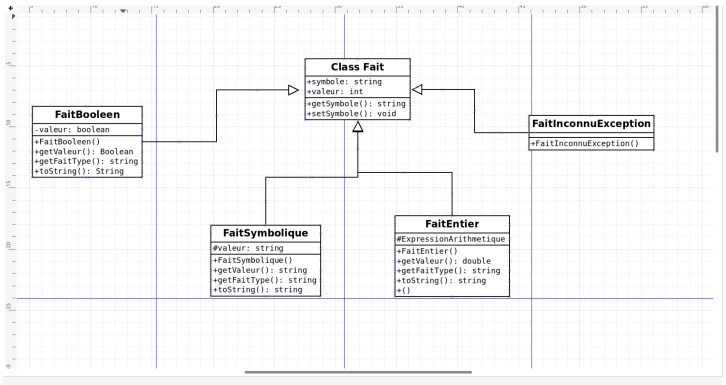

Figure 5: Diagramme de class fait

4 Grammaire des Règles

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.

Une règle typique peut être structurée comme suit :

SI *conditions* ALORS *conclusion*

Où :

- *conditions* représentent les prémisses ou les faits nécessaires pour que la règle s'applique.
- *conclusion* est le résultat ou la déduction qui est faite si les conditions sont satisfaites.

Par exemple, une règle dans un système expert médical pourrait être formulée comme suit :

SI le patient a de la fièvre ET des maux de tête ALORS le diagnostic est la grippe.

Cette règle indique que si un patient présente à la fois de la fièvre et des maux de tête, alors le diagnostic de la grippe est posé.

Spécifications pour la Base de Connaissance

Quelques notes à savoir pour créer une base de connaissance syntaxiquement correcte :

- 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.

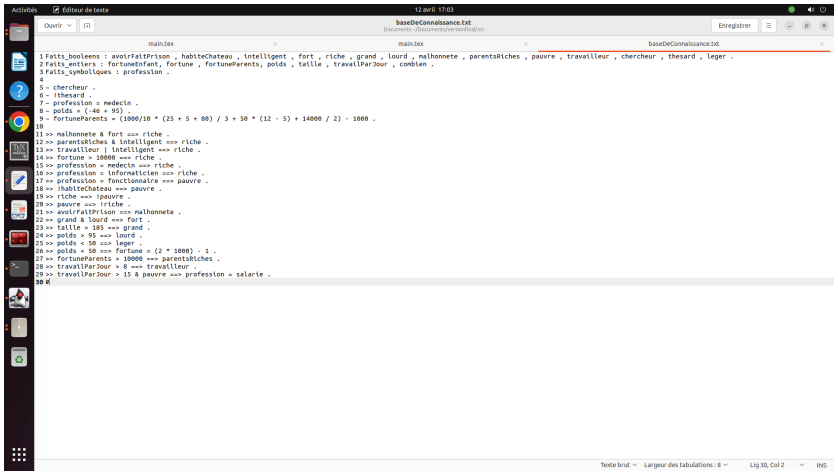

Figure 6: Base de Connaissance

- Il faut préciser dans les déclarations de faits le type au début de chaque déclaration (`Faits_booleens`, `Faits_entiers`, ou `Faits_symboliques`), suivi de deux points, puis des faits.
- À la fin d'un fichier, il est obligatoire de mettre le symbole `#`, qui signifie fin du fichier.

5 Fonctionnement des Systèmes Experts

Un système expert fonctionne en utilisant une base de connaissances contenant des faits et des règles sur un domaine spécifique. Lorsqu'il est interrogé ou confronté à un problème, le système applique des algorithmes d'inférence pour déduire des conclusions à partir de ces règles et de ces faits.

Il existe deux principales stratégies d'inférence utilisées par les systèmes experts : le chaînage avant et le chaînage arrière.

section Moteur d'inférence

Le moteur d'inférence est une composante centrale des systèmes experts. Il permet de tirer des conclusions logiques à partir des faits disponibles et des règles de connaissances prédéfinies. Le moteur d'inférence peut fonctionner selon différents modes, notamment le chaînage avant et le chaînage arrière, pour parvenir à une conclusion.

5.1 CHAINAGE-AVANT

Le chaînage avant est une technique de raisonnement dans laquelle le système explore les règles et déduit de nouveaux faits en se basant sur les faits connus. Cette méthode est guidée par les données et vise à activer toutes les règles activables afin de déduire des faits supplémentaires et mettre à jour la base de faits.

La procédure CHAINAGE-AVANT prend en entrée la base de faits (BF) et l'ensemble des règles (BR) du système. Elle itère à travers les règles activables pour déclencher celles qui sont prêtes, ajoutant ainsi de nouveaux faits à la base de faits jusqu'à ce qu'il n'y ait plus de règles activables.

Algorithme 1 : Procédure CHAINAGE-AVANT

Entrée : BF : Base de faits, BR : Ensemble des règles
Sortie : Base de faits mise à jour

```
1  pour chaque r dans BR faire
2  |  activable(r) ← VRAI;
3  fin
4  tant que VRAI faire
5  |  déclenchables ← ∅;
6  |  pour chaque r dans BR faire
7  |  |  si activable(r) = VRAI alors
8  |  |  |  pour chaque p dans Prémisse(r) faire
9  |  |  |  |  si !(p appartient à BF) or (p contradictoire avec BF)
   |  |  |  |   alors
10 |  |  |  |  |  continuer;
11 |  |  |  |  fin
12 |  |  |  fin
13 |  |  |  AJOUTER r à déclenchables;
14 |  |  fin
15 |  fin
16 |  si déclenchables = ∅ alors
17 |  |  ARRET;
18 |  fin
19 |  r ← CHOISIR(déclenchables);
20 |  activable(r) ← FAUX;
21 |  pour chaque c dans Conclusions(r) faire
22 |  |  AJOUTER c dans BF;
23 |  fin
24 fin
```

5.2 CHAINAGE-ARRIÉRE

Le chaînage arrière est une méthode de raisonnement utilisée dans les systèmes experts pour parvenir à une conclusion souhaitée en partant de celle-ci et en remontant la chaîne de raisonnement vers les faits initiaux. Contrairement aux approches de raisonnement direct telles que le chaînage avant, qui partent des faits disponibles pour arriver à une conclusion, le chaînage arrière commence par la conclusion recherchée et cherche à justifier celle-ci en trouvant les faits nécessaires pour la soutenir. Il explore les règles d'inférence en partant de la conclusion cible et en cherchant à satisfaire les prémisses des

règles pour arriver aux faits initiaux. Ce processus se poursuit de manière récursive jusqu'à ce que tous les prérequis nécessaires pour justifier la conclusion soient vérifiés, ou jusqu'à ce qu'il soit impossible de continuer en raison d'un manque de faits ou de règles appropriées.

La methode CHAINAGE-ARRIÉRE prend en entrée la base de faits (BF) et l'ensemble des règles (BR) du système et la conclusion a demontrer.Elle utilise ces informations pour déterminer si le fait donné peut être prouvé dans la base de faits en utilisant les règles de la base de règles.

Algorithme 2 : Algorithme du CHAINAGE-ARRIERE

Entrée : BF : Base de faits, BR : Ensemble des règles, f : Conclusion à prouver (ou bien le fait)
Sortie : Vrai si la conclusion (ou fait) est prouvée, Faux sinon

```
1  si f appartient à BF alors
2  |  retourner VRAI
3  fin
4  pour chaque r dans BR faire
5  |  activable(r) ← VRAI
6  fin
7  conflit ← ∅
8  pour chaque r dans BR faire
9  |  si activable(r) = VRAI et f appartient à Conclusions(r) alors
10 |  |  pour chaque p dans Prémisses(r) faire
11 |  |  |  si p appartient à BF et p non contradictoire avec BF
   |  |  |   alors
12 |  |  |  |  AJOUTER r à conflit
13 |  |  |  fin
14 |  |  fin
15 |  fin
16 fin
17 si conflit = ∅ alors
18 |  retourner FAUX
19 fin
20 r ← CHOISIR(conflit)
21 activable(r) ← FAUX
22 Soient p_1, ..., p_n les Prémisses(r);
23 retourner CHAINAGE-ARRIERE(BF, BR, p_1) ET ... ET
     CHAINAGE-ARRIERE(BF, BR, p_n)
```

v

6 Tests

Dans le cadre de notre projet, nous avons accordé une attention particulière à la qualité du code en créant des tests unitaires pour les classes et les méthodes les plus cruciales. Notre objectif était de garantir le bon fonctionnement de ces éléments essentiels.

Les classes que nous avons choisi de tester comprennent des composants clés tels que le Moteur d'Inférence, PremisseBoleenne, Premisse Symbolique, entre autres. Chaque classe a fait l'objet de tests unitaires individuels, où nous avons évalué le comportement attendu de ses méthodes dans différentes conditions.

Pour simplifier l'exécution de ces tests et faciliter la validation globale des fonctionnalités principales, nous avons également développé une classe de test globale nommée TestGlobal.java. Cette classe regroupe tous les tests unitaires des classes importantes, ce qui permet une exécution rapide et efficace de l'ensemble des tests critiques en une seule fois.

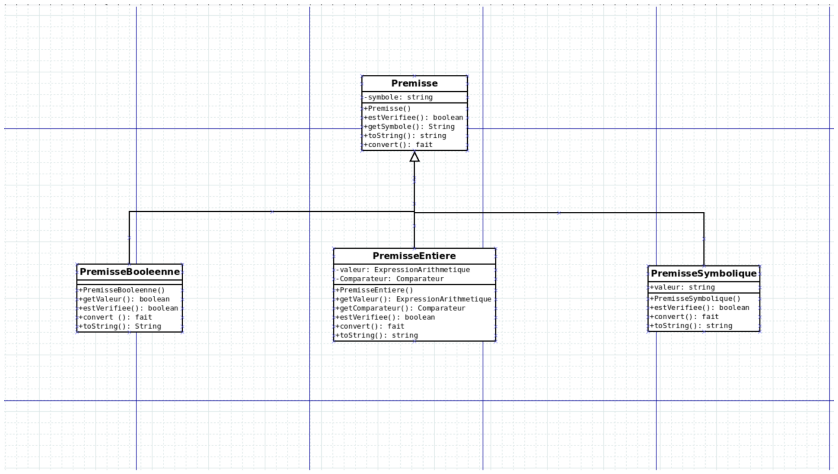

Figure 7: Captures d'écran des tests unitaires

7 Interface Graphique et Fonctionnalités

Dans cette section, nous décrivons l'interface graphique que nous avons développée pour notre système expert, ainsi que ses fonctionnalités principales.

7.1 Interface Graphique

Notre interface graphique est conçue pour offrir une expérience utilisateur conviviale et intuitive. Elle se compose de cinq panneaux (JPanel) principaux, chacun offrant une fonctionnalité spécifique :

1. **Panneau de Base de Règles** : Ce panneau affiche la base de règles du système expert, permettant à l'utilisateur de visualiser les règles logiques qui déterminent le raisonnement du système.

2. **Panneau de Base de Faits** : Ce panneau affiche la base de faits après l'exécution du chaînage avant, montrant les faits déduits à partir des règles et des faits initiaux.

3. **Panneau de Symboles et Valeurs** : Ce panneau permet à l'utilisateur de saisir les symboles et les valeurs nécessaires pour interroger le système expert ou exécuter des chaînages avant et arrière.

4. **Panneau de Résultats** : Ce panneau affiche les résultats de l'exécution du chaînage avant ou arrière, permettant à l'utilisateur de voir les conclusions déduites par le système expert.

5. **Panneau de Contrôle** : Ce panneau contient deux boutons, l'un pour exécuter le chaînage avant et l'autre pour exécuter le chaînage arrière. Ces boutons permettent à l'utilisateur de contrôler le fonctionnement du système expert.

7.2 Fonctionnalités

Notre interface graphique offre les fonctionnalités suivantes :

- Visualisation de la base de règles et de la base de faits.
- Saisie des symboles et des valeurs pour interroger le système expert.
- Exécution du chaînage avant pour déduire de nouveaux faits à partir des règles.

- Exécution du chaînage arrière pour prouver une conclusion à partir des faits disponibles.
- Affichage des résultats de l'exécution du chaînage avant.

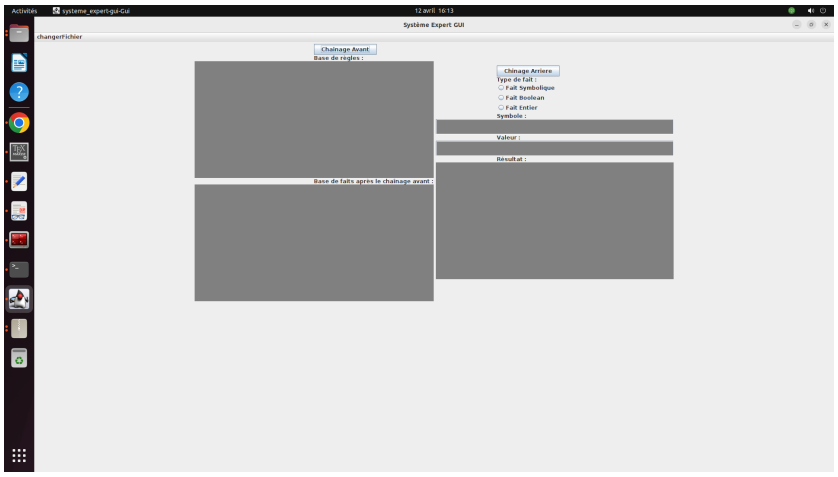

Figure 8: Interface graphique

8 Compilation et Exécution des Tests

Pour compiler et exécuter les tests unitaires dans votre projet, suivez les instructions ci-dessous :

Compilation des classes de test :

```
javac -cp .:../lib/junit-4.13.2.jar -d ../build/ systeme_expert/tests/
```

Exécution des tests unitaires :

```
java -cp .:../lib/junit-4.13.2.jar:../lib/hamcrest-core-1.3.jar:../bui
```

Cette commande exécute les tests de la classe TestGlobal. Si vous souhaitez exécuter les tests d'une autre classe, remplacez TestGlobal par le nom de la classe de test correspondante.