

Systemes Experts

Cours de Prof. Abdellah AZMANI

Département Génie Informatique

Faculté des Sciences et Techniques de Tanger

Sommaire

I.	Introduction : Portées et domaines d'application.....	3
II.	Principe de fonctionnement d'un système expert	4
a.	Base de connaissances	5
b.	Base de faits	5
c.	Base de règles	6
d.	Le moteur d'inférence	7
e.	Chaînage avant :.....	8
f.	Chaînage arrière :.....	9
g.	Exercice :	10
III.	TRAITEMENT DE L'IMPRECIS ET DE L'INCERTAIN	11
IV.	TRAITEMENT DE LANGAGE NATUREL	15
a.	CONCEPTS ELEMENTAIRES DU LANGAGE	16
b.	Analyseur de bruits de fond	17
c.	Analyse par graphes d'états	17
h.	Analyse par formes grammaticales.....	19
V.	Modélisation Objet d'un Système Expert.....	22
a.	Spécification fonctionnelles	22
b.	Packages.....	23
c.	Cas d'utilisation	23
	Diagramme de UseCase de gestion du moteur d inférence.....	24
	Diagramme de UseCase de gestion des coefficients	24
d.	Diagrammes de scénarios	25
	Diagramme de scénario d'ajout d'une règle	25
	Diagramme de scénario de modification d'une règle	25
	Diagramme de scénario de recherche d'une règle	26
	Diagramme de scénario de suppression d'une règle	26
e.	Diagramme d'activité	27
	Diagramme d'activité de la gestion de la base de connaissance	27
	Diagramme d'activité de la gestion des coefficients	28
	Diagramme d'activité du moteur d inférence	28
f.	Diagrammes d'objet.....	29
	Diagramme d'objet :.....	29
g.	Diagrammes de séquence.....	29
	Diagramme de séquence d'ajout d'une règle	29
	Diagramme de séquence de modification d'une règle	30
	Diagramme de séquence de recherche d'une règle	30
	Diagramme de séquence de suppression d'une règle	31
h.	Modèle dynamique :	31
i.	Diagramme de classe	32
j.	Exercice d'application (Devoir)	32
VI.	Algorithmes.....	33
a.	Le chaînage avant	33
b.	Le chaînage arrière.....	34
c.	Le chaînage mixte	35
VII.	Exemples d'applications.....	36
a.	Application pour un diagnostique médical	36
b.	Application sur a base d'arbre de décision	38
c.	Exercice	39
VIII.	Projet de réalisation de Systèmes Experts.....	40

I. Introduction : Portées et domaines d'application

Impossibilité actuelle à programmer des systèmes informatiques capable de mimer un comportement intelligent général (comme nous en avons tous un !)

=> Création de programmes ne visant qu'un domaine limité et facile à mettre sous forme symbolique.

Ces programmes qui concentrent en eux les connaissances d'une discipline restreinte de telle façon qu'elles soient utilisables, sont appelés des systèmes experts : ils sont destinés à remplacer des experts humains ou à les aider.

Les systèmes experts peuvent être utilisés pour résoudre une gamme large des problèmes dont les algorithmes classiques ne peuvent pas servir, et cela dans plusieurs domaines

Portées	Domaines
▶ Diagnostique	▶ Médecine
▶ Ordonnancement et planification des tâches	▶ Finance
▶ Problèmes de classification	▶ Assurance
▶ Prise de décision	▶ Education et enseignement
▶ ...	▶ Traitement d'images
	▶ Agriculture
	▶ Traitement de la parole
	▶ ...

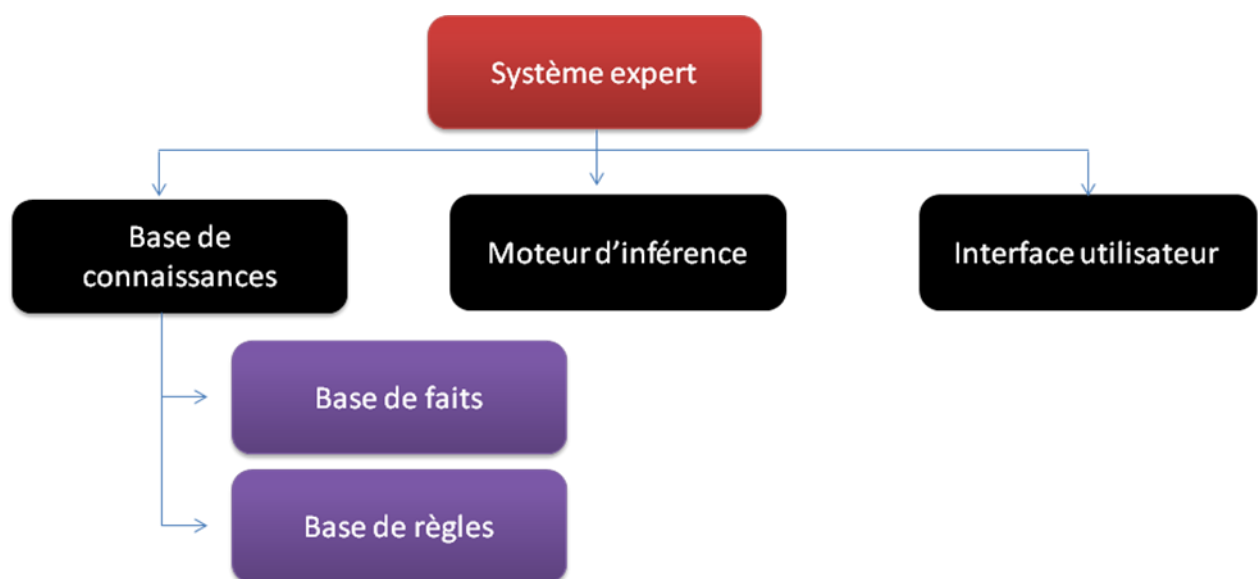
Les premiers et plus célèbres sont :

- **DENDRAL :**
chimie moléculaire, STANFORD.
- **MYCIN :**
diagnostic des infections bactériennes et indications thérapeutiques, STANFORD.
- **PROSPECTOR :**
aide à la recherche en exploitation minière.
- **FAULTFINDER :**
diagnostic de panne d'équipement.
- **PEACE :**
modélisation de circuits électroniques.

- **MECHO :**
résolution de problèmes en mécanique.
- **ORBI :**
évaluation de ressources pour la planification de l'environnement.
- **KBOI**
expertise en jardinage.
- **ORIENT EXPERT :**
orientation et élaboration de projet professionnel.

Il existe aujourd'hui des milliers de systèmes experts utilisés et en développement.

II. Principe de fonctionnement d'un système expert



Elaborer un ensemble de connaissances. Celles-ci correspondent à des règles de raisonnement qui ont été tirées d'un certain domaine de connaissances :

Diagnostic médical, analyse de circuit, exploration minière, etc.

a. Base de connaissances

- La base de connaissances contient toute l'information dont un expert humain aurait besoin pour s'acquitter de son travail, ceci dans un domaine donné. C'est la seule composante du système qui contienne les connaissances propres au domaine qu'il est censé recouvrir. Cette base de connaissances est elle-même divisée en deux composantes. La première partie contient des faits spécifiques du domaine, les connaissances factuelles. On parle alors de **base de faits**.
- La **base de règles**, contient les règles qui vont permettre au système de raisonner à partir des faits. Ces connaissances déductives sont représentées par des règles appelées **règles de production**.

b. Base de faits

- La **base de faits** est la mémoire de travail du système expert. Elle est variable au cours de l'exécution et vidée lorsque l'exécution se termine. Au début de la session, elle contient ce que l'on sait du cas examiné avant toute intervention du moteur d'inférences. Puis elle est complétée par les faits déduits par le moteur ou demandés à l'utilisateur. Par exemple, dans le domaine médical, la base de faits pourra contenir une liste de symptômes en début de session et un diagnostic lorsque celle-ci se terminera.
- **Le type de faits :**
Les faits peuvent prendre des formes dont les valeurs possibles sont :
 - Booléen : vrai, faux, exemple actif
 - Symbolique : exemple profession
 - Réel : exemple : rémunération

Un système expert qui n'utilise que des faits booléens est dit d'ordre 0. Un système expert qui utilise des faits symboliques ou réels, sans utiliser de variables, est d'ordre 0+ et un système utilisant toute la puissance de la logique du premier ordre est d'ordre 1.

La base des faits contient les différents faits appartenant au domaine d'expertise concerné, c'est l'ensemble des vérités connues.

- **Exemples :**
 - F1 : animal a des plumes
 - F2 : animal a un long cou
 - F3 : animal a de longues pattes

c. Base de règles

- Elle rassemble la connaissance et le savoir-faire de l'expert. Elle n'évolue donc pas au cours d'une session de travail.
- Une règle est de la forme :
 - **Si <conjonction de conditions> alors <conclusion>**
 - C'est à dire une suite de comparaison d'attribut pour la conjonction de condition (appelée aussi prémisse) et où les conclusions sont de la forme : <Fait> = <valeur>.
 - *Exemple :*
 - **SI <voiture_couleur = rouge ET voiture_marque = Ferrari> ALORS <conducteur = heureux>**
- On va prendre les attributs en prémisse, et on va voir si la prémisse est vérifiée. Par exemple, on va comparer voiture_couleur à rouge, et si c'est le cas, c'est-à-dire si la voiture dont on dispose est effectivement rouge, alors on passe à la prémisse suivante. Si la marque de la voiture dont on dispose est effectivement Ferrari, alors la règle se déclenche.
- Le déclenchement de la règle conduit à l'exécution des affectations de valeurs aux attributs présents en conclusion. Cela veut dire que l'on va pouvoir en tirer les conclusions. On va affecter les valeurs en conclusion à leurs attributs respectifs.
- Par contre, si l'une des prémisses n'avait pas été vérifiée alors la règle sera abandonnée.

Exemple d'une règle de la base animal :

Si	(est mammifère)
et	(est carnivore)
et	(a des rayures noires)
et	(est de couleur fauve)
alors	(est tigre).

La mise au point de ces règles est une partie très délicate et très longue dans la constitution des systèmes experts.

1. Les experts humains qu'on interroge pour élaborer ces bases de connaissances, en général, n'ont pas formalisé sous forme de règles (comme celles indiquées) tout ce qu'ils savent de leur domaine.
2. Certaines règles sont utilisées inconsciemment et il est donc nécessaire de réfléchir en profondeur à tout ce qui constitue le savoir de l'expert.

En médecine, il est souvent nécessaire d'énoncer les règles sous des formes non catégoriques, par exemple :

**Si le site de culture est le sang
Et si l'organisme est gram négatif
Et si l'organisme est esf de forme bâtonnet
Et si le patient est un hôte à risque
Alors
il est vraisemblable à 0,6 que l'organisme est le pseudomonas aeruginosa.**
(Extrait de MYCIN).

d. Le moteur d'inférence

Pour illustrer le principe de fonctionnement d'un moteur d'inférence, nous proposons la Base de connaissance avec des faits abstraits suivante :

R1 :	si B et D et E	alors	F
R2 :	si G et D	alors	A
R3 :	si C et F	alors	A
R4 :	si B	alors	X
R5 :	si D	alors	E
R6 :	si X et A	alors	H
R7 :	si C	alors	D
R8 :	si X et C	alors	A
R9 :	si X et B	alors	D

En plus de l'ensemble de connaissances, on donne une base de faits connus initialement :

B, C !

Ce qui correspond par exemple à des résultats d'analyses pour un patient donné.

Ensuite on pose une question :

Le but à atteindre est le fait H

=> On se demande si *H* résulte des faits par application des règles données dans la base de connaissance.

C'est là qu'intervient ce qu'on appelle le moteur d'inférences du système expert, c'est-à-dire le mécanisme (algorithme) qui va tenter de résoudre le problème posé.

Il y a deux types de fonctionnement pour un moteur d'inférences : **le chaînage avant** et le **chaînage arrière**.

- Pour exploiter cette connaissance, un moteur d'inférences est nécessaire pour relier la description d'un problème aux capacités d'analyse d'une situation donnée. De

façon générale et grâce à la structuration de la base de connaissances, le moteur d'inférences sera capable de répondre à des questions, de raisonner et de tirer les conséquences impliquées par la connaissance incluse dans le système.

- Le moteur d'inférence va enchaîner les règles c'est-à-dire qu'il va effectuer un chaînage. Il en existe trois catégories, basées sur la manière dont les problèmes sont résolus ou types de raisonnement :
 - le moteur d'inférence, dit à « chaînage avant », qui part des faits et règles de la base de connaissance, et tente de s'approcher des faits recherchés par le problème ;
 - le moteur d'inférence, dit à « chaînage arrière », qui part des faits recherchés par le problème, et tente par l'intermédiaire des règles, de « remonter » à des faits connus;
 - le moteur d'inférence, dit à « chaînage mixte », qui utilise une combinaison des deux approches chaînage avant et chaînage arrière, précédemment citées.

e. Chaînage avant :

On part de la base de faits connus initialement et on déclenche des règles dont les prémisses sont entièrement contenues dans cette base de faits. On obtient ainsi une nouvelle base de faits et on poursuit jusqu'à ce que ou bien, on tombe sur le but recherché H ou bien, on sature la base de connaissance : plus aucune règle ne puisse s'appliquer.

Problématique :

Lorsque plusieurs règles peuvent s'appliquer, le moteur d'inférence choisit l'une d'elles,

⇒ **application des règles de choix : Les métarègles.**

Lorsqu'on a affaire à de vastes ensembles de règles, la croissance des faits déduits est souvent exponentielle et donc, la définition et la mise en place de ces métarègles est fondamentale.

Dans notre exemple :

Base de faits initiale : **B, C**

Applications de règles :

- **R4** donne **X** : **B, C, X**
- **R8** donne **A** : **A, B, C, X**
- **R6** donne **H** : **A, B, C, X, H succès**

Notre **métarègle** pour choisir parmi plusieurs règles : **prendre celle qui le plus de faits présents dans la base des faits connus.**

f. Chaînage arrière :

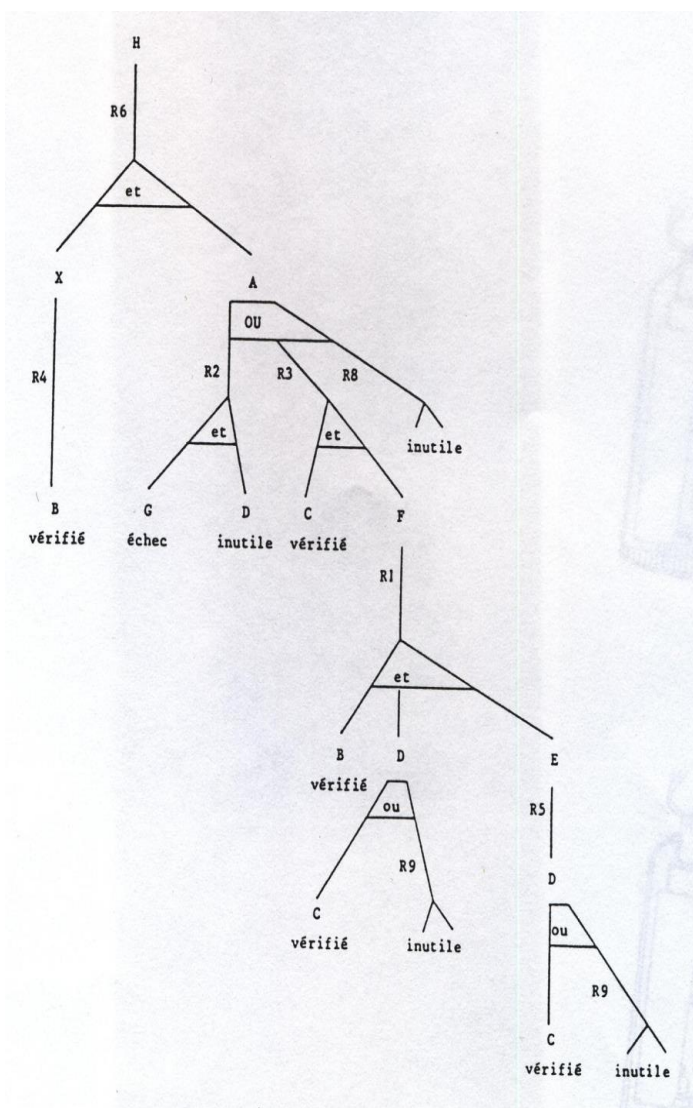
On regarde les règles qui ont le but fixé dans leurs conséquences.

On considère chacune de ces règles,

- si l'une d'elles a toutes ses prémisses dans la base de faits, on a un succès,
- sinon on considère les prémisses comme de nouveaux buts et on recommence.

On est ainsi amené à explorer un arbre qui, dans les cas simples, est fini (comme pour notre exemple).

Exemple : But H



Seule la règle R6 possède H comme conséquence, donc : nouveaux buts X, A
La règle R4 possède X comme conséquence, donc : nouveaux buts A, B

B est un fait déjà connu, donc : nouveaux but A

3 règles possèdent A comme conséquence, ce qui détermine 3 branches dans l'arbre d'exploration.

Branche 1 application de R2 : nouveaux buts G, D

G n'est jamais conséquence donc échec

Branche 2 application de R3 : nouveaux buts C, F

C est un fait connu, donc : nouveaux but F

R1 donne nouveaux buts B, D, E

B est un fait connu, donc : nouveaux buts D, E

R7 donne nouveaux buts C, E

C est un fait connu, donc : nouveau but E

R5 donne : nouveau but D

R7 donne : nouveau but C

C est fait donc succès

=> il est inutile d'essayer la branche 3.

Remarques :

- lorsqu'on utilise un procédé de chaînage arrière, on dit qu'on **cherche à effacer les buts**.
- Les méthodes de chaînage avant et de chaînage arrière ont chacune leurs intérêt propres, **selon la structure des problèmes l'une peut être plus intéressante que l'autre**.
- Beaucoup de **règles** avec les **mêmes prémisses pénalisent le chaînage** avant et inversement beaucoup de **règles** avec les **mêmes conclusions pénalisent le chaînage arrière**.
- Le principe du chaînage avant repose sur ce qu'on appelle en logique **le modus ponens** :
si $A \Rightarrow B$ et A , alors B
- Le principe du chaînage arrière repose sur ce qu'on appelle en logique **le modus tollens** :

si $A \Rightarrow B$ et non B , alors non A

On considère que la négation du but est ajoutée aux faits et on recherche par application successives du modus tollens la négation d'un fait, ce qui donnera une contradiction et donc la "démonstration" du but.

Indiquons encore qu'en général un système expert est conçu de manière :

1. à pouvoir être **mis à jour en permanence** : modification de la base de connaissances.
2. à pouvoir **indiquer en clair quel raisonnement il a utilisé** : Explication des résultats fournis.

g. Exercice :

Nous voulons créer un système qui assiste l'utilisateur à décider si la ville qu'il s'apprête à visiter, mérite d'être visitée (ville méritant le voyage). Le système doit lui poser un certain nombre de questions pour pouvoir décider.

Soit la base de règle suivante :

R1 : **Si** ville historique **alors** ville méritant le voyage

R2 : **Si** ville artistique **alors** ville méritant le voyage

R3 : **Si** nombreuses animations **alors** ville méritant le voyage

R4 : **Si** ville agréable **et** tradition gastronomique **alors** ville méritant le voyage

R5 : **Si** belle ville **et** nombreux monuments **alors** ville artistique

R6 : **Si** ville ancienne **et** nombreux monuments **alors** ville historique

R7 : **Si** nombreux concerts **et** nombreux théâtres **alors** nombreuses animations

R8 : **Si** activités sportives **et** traditions folkloriques **alors** nombreuses animations

R9 : **Si** espaces verts **et** climat agréable **alors** ville agréable

R10 : **Si** espaces verts **et** nombreux monuments **alors** belle ville

R11 : **Si** nombreux restaurants **et** bons restaurants **alors** tradition gastronomique

Construire l'arbre ET-OU dont le but est de vérifier si une ville mérite le voyage.

III. TRAITEMENT DE L'IMPRECIS ET DE L'INCERTAIN

Le savoir humain, que l'on cherche à représenter dans un système expert, est souvent entaché d'incertitude et d'imprécision.

- **L'incertitude porte sur la vérité ou la fausseté d'une affirmation.**
- **L'imprécision porte sur une affirmation exprimée à l'aide de quantificateurs vagues.**

À la question :

"Quel pourcentage de pièces défectueuses pensez-vous avoir ce mois-ci ?",

Réponse 1 : 2,371289 %	Réponse 2 : entre 0% et 80%
La réponse est très précise, mais on n'est pas certain qu'elle soit vraie (incertitude).	La réponse est certainement vraie, mais elle est très imprécise (imprécision).

Il est courant pour un expert de traiter ce type d'informations parce qu'il les obtient des capteurs physiques ayant une précision limitée ou d'observations humaines souvent très subjectives. La façon de combiner des coefficients varie d'un système expert à l'autre.

L'expert travaille, parfois, avec des informations dégradées :

observation impossible, appareil de mesures en panne...

Une autre source d'incertitude lors du raisonnement d'un expert est son propre savoir-faire, issu d'expériences et donc empirique (vague, floue).

Il y a donc trois sources d'incertitude :

- **observations imprécises,**
- **observations absentes,**
- **connaissances incertaines.**

C'est pour ces raisons que certains systèmes experts traitent l'incertitude.

La plupart utilisent un traitement numérique et associent à chacune des règles et à chacun des faits des coefficients de vraisemblance (CV).

Exemple :

dans le système MYCIN, les coefficients de vraisemblance sont compris entre -1 et 1.

-1 => faux de façon certaine

+1 => vrai façon certaine,
0 => absence d'information.

L'expert affecte à chaque règle un coefficient de confiance (CCR) qui exprime la confiance qu'il accorde à la règle.

Le coefficient associé au fait conclusion d'une règle se calcule de la façon suivante :

- si aucun coefficient n'avait encore été affecté au fait conclusion, alors on utilise la formule :

CV conclusion = CCR* minimum des CV des prémisses.

- si le fait conclusion avait déjà un coefficient de vraisemblance initial CV1 et qu'il reçoive le coefficient CV2, alors son coefficient final CVf sera calculé de la façon suivante :

CVf = CV 1 + CV2 - CV 1 * CV2 si CV1 et CV2 > 0.

CVf = CV 1 + CV2 + CV 1 * CV2 si CV 1 et CV < 0.

CVf = (CV1 + CV2) / (1 - min (CV 1, CV2)) sinon.

Ces formules permettent de garantir un coefficient entre -1 et +1 et de renforcer ou affaiblir un coefficient de vraisemblance selon les nouveaux résultats acquis.

Exemple : Soit la règle **R1 (CCR = 0.7)** dont les prémisses ont les coefficients de vraisemblance suivants :

R1 : **Si** la voiture a du mal à démarrer (**0.6**) et l'air est humide (**0.5**)
 Alors il faut changer les bougies

Remarquons tout d'abord qu'une règle aussi simple peut poser des problèmes :

- A partir de quel taux d'humidité peut-on affirmer que l'air est humide ?
- A partir de combien d'essais infructueux doit-on dire qu'une voiture démarre mal ?
- Quelle confiance à accorder à une telle règle ?

Si on ne sait rien du fait "il faut changer les bougies", l'application de la règle va lui affecter le coefficient de vraisemblance :

CV = 0.7*0.5 = 0.35

Si on savait déjà que "il faut changer les bougies" (0.30), le coefficient de vraisemblance de ce fait va devenir :

CVf= 0.3 +0.35 - 0.3* 0.35 = 0.54

Le déclenchement de la règle a donc renforcé la suspicion d'un problème de bougies.
Il est parfois difficile de savoir à partir de quel seuil de vraisemblance il faut considérer un fait.

Dans MYCIN, tout fait dont le coefficient de vraisemblance est inférieur en valeur absolue à 0.2 est considéré comme peu crédible et est éliminé de la base des faits.

Les inconvénients

Si l'idée de traiter l'incertain et l'imprécis est séduisante a priori, elle présente néanmoins de nombreux inconvénients :

- La présence de coefficients de vraisemblance implique de sérieuses modifications à apporter à un système fonctionnant en logique pure.
- L'explication du raisonnement est plus complexe car, pour un fait donné, plusieurs chemins ont pu être empruntés pour confirmer ou infirmer ce fait.
- Il est difficile et parfois impossible à un expert de chiffrer la confiance qu'il accorde à une connaissance.

Donner une confiance de 0.5 ou 0.6 à une règle, peut lui sembler complètement artificiel.

Il est de plus dans l'impossibilité de connaître à l'avance les répercussions de la modification d'un coefficient sur le système.

- A partir de quel coefficient de vraisemblance doit-on considérer qu'un résultat est acquis?
- Enfin, il existe souvent des cas limites pouvant avoir des conséquences désastreuses.

Prenons pour exemple l'application des calculs précédents sur les deux règles suivantes :

- **R1** (0.7) Si je me rase avec un rasoir électrique, alors je ne fais pas une action dangereuse.
- **R2** (0.5) Si je prends un bain, alors je ne fais pas une action dangereuse.

Si je prends mon bain tout en me rasant avec un rasoir électrique.

⇒ les deux prémisses des règles sont vraies et donc on a 1 pour coefficient de vraisemblance.

Le système va appliquer R1 et déduire "je ne fais pas une action dangereuse (0.7)" puis va appliquer R2 et déduire "je ne fais pas une action dangereuse (0.5)" !

⇒ Pour le système, afin d'être sûr de ne pas faire une action dangereuse, il faut prendre son bain tout en se rasant avec un appareil électrique !

Il faut donc être particulièrement prudent dans l'utilisation de telles méthodes et certains n'hésitent pas à dire qu'il est préférable d'en éviter l'usage.

IV. TRAITEMENT DE LANGAGE NATUREL

Le traitement du langage naturel = écrire des programmes qui interprètent les expressions écrites dans une langue pour les convertir en une autre.

Langage naturel = proche du langage humain normal

= langage avec un alphabet, une syntaxe et une grammaire.

Applications typiques :

- L'interprétation des phrases humaines en commandes informatiques comprises par une machine.
- La conversion du langage informatique en phrases naturelles, comprises par un humain.
- La traduction d'un langage humain en un autre.
- Le contrôle syntaxique ou grammatical des phrases :

Vérificateurs orthographiques avec analyse des phrases pour corriger en fonction du contexte.

Exemple : la phrase "**le petite maison dans la prairie**" serait corrigée en "**la petite maison dans la prairie**"

Ici le programme trouve deux valeurs féminines pour le nom et l'adjectif

=> correction de l'article.

Dans ces systèmes, l'analyse syntaxique est le cœur du programme

=> similitude avec le système d'exploitation d'un ordinateur.

Exemple d'utilisation d'une interface en langage naturel :

Quel est votre problème ?

J'ai de l'acné.

Avez-vous d'autres symptômes ?

Mes yeux sont légèrement congestionnés.

Rien d'autre ?

Non.

Vous avez probablement une carence en Vitamine C, E et A.

Essayez de compléter l'apport de ces éléments.

Des pommades aux vitamines E et A peuvent traiter localement votre acné.

Les Interfaces en Langage Naturel (ILN) ne prennent pas en compte les différents aspects de la reconnaissance ou de la synthèse Vocale. Leur but est de rendre convivial le dialogue entre l'Homme et la Machine.

a. CONCEPTS ELEMENTAIRES DU LANGAGE

Une expression d'une sémantique.

La syntaxe = règles des symboles.

Exemple : un article en français.

Le langage est composé de syntaxe et de langage déterminant l'ordre et l'écriture qui précède toujours le groupe nominal en français

La sémantique = les idées exprimant le fond d'un texte.

Programmer une syntaxe est chose facile. En revanche faire comprendre le sens et la portée d'une expression ou d'une phrase reste encore une tâche difficile pour un ordinateur.

Exemple : la phrase "le fermier a du veau"

=> le fermier possède un veau

Ou

=> le fermier est devant une assiette de veau.

Actuellement les analyseurs ne sont pas capables de résoudre des analyses sémantiques complexes. Un analyseur coupe une phrase en blocs de symboles et détermine la syntaxe.

Types d'analyseurs

Il existe trois différents types d'analyseurs en ILN

- Les analyseurs de bruits de fond ;
- Les analyseurs à graphes ;
- Les analyseurs à base de règles grammaticales.

b. Analyseur de bruits de fond

Analyse simpliste, son principe balayer une phrase pour identifier certains mots clés. Le reste de la phrase est simplement ignoré ; ce sont les bruits de fond.

Création d'analyseur de bruits de fond:

- L'analyseur utilise un vocabulaire réduit et bien défini.
- Conservation du vocabulaire dans des listes ou des bases dynamiques.
- Chaque mot entré par l'utilisateur est traduit en symbole.
- Ce dernier sera utilisé pour activer la commande demandée.

Exemple :

Pour déclencher une commande permettant de vider une base de données.

On entre une phrase contenant les mots "vider" ou "initialiser " avec les mots "base" ou "données", par exemple : "initialiser la base".

Il faut créer une routine qui prend chaque mot de la phrase entrée et cherche une correspondance avec un mot clé.

Inconvénients de l'analyseur de bruits de fond:

=>Utiliser pour des environnements restreints impliquant un vocabulaire limité et peu de commandes associées.

=>Principal inconvénient = impossibilité d'analyser une phrase ambiguë.

Exemple :

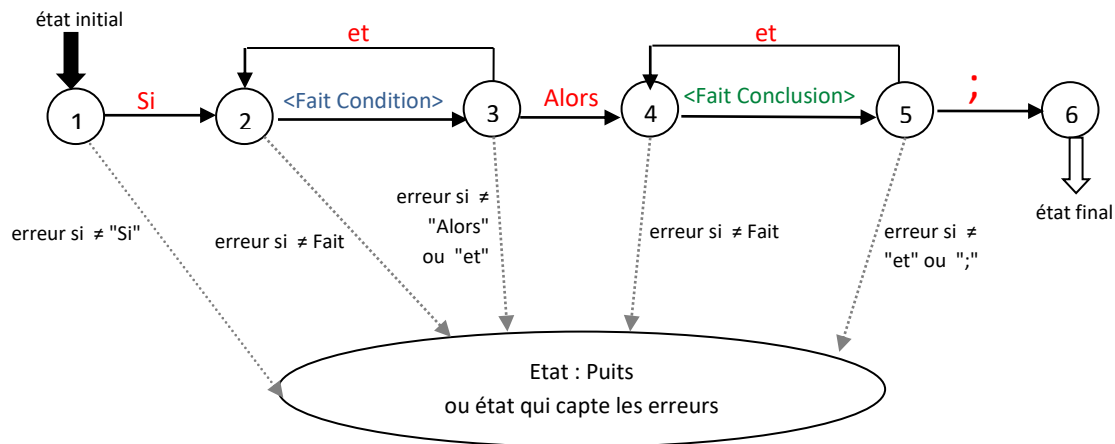
La phrase "Vider le bloc test de l'espace de données" pourra être interprétée comme une demande pour vider les données. Cela montre la différence entre la sémantique et la syntaxe et donc l'insuffisance des analyseurs simples.

c. Analyse par graphes d'états

Une machine à graphes voit la phrase comme une suite d'automates finis. Le traitement consiste à se déplacer dans l'espace défini, en passant d'un nœud à un autre. Le programme commence en reconnaissant un nœud ou un état donné. Il lit ensuite le premier mot qui lui permet de parcourir un chemin vers un autre nœud en déclenchant une action donnée. Le traitement s'arrête lorsque le nœud terminal ou état final est atteint.

Exemple d'analyse par graphes

Soit l'automate à états finis suivant :



Le rôle de cet automate est de reconnaître la bonne syntaxe d'une règle de production. Il possède 7 états dont un état initial et un état final.

Exemple d'une phrase

Si bouton appuyé = 2 **et** position ascenseur = 4

alors descendre 2 étages **et** ouvrir les portes.

Les mots clés sont : "**Si**", "**et**", "**alors**".

Le premier mot d'une phrase qui pourra être reconnue par cet automate est forcément un "**Si**". Le programme doit lire les mots suivants et former par concaténation une chaîne de caractères jusqu'à la rencontre d'un mot clé "**et**" ou "**alors**". La chaîne formée est vérifiée par rapport à un dictionnaire de faits possibles.

La rencontre du mot clé "**et**" quand doit absolument rencontrer un autre fait appartenant à la même partie "**condition**" ou "**conclusion**".

Le programme s'arrête quand on a plus de mots à lire ou quand on rencontre une erreur (état puits). On pourra envisager de terminer l'écriture d'une règle par un caractère spéciale, par exemple un ";".

Exercices :

- Ecrire un programme qui permet de reconnaître le conditionnel (if ... else...) ou l'itératif (while ...) d'un langage de votre choix.

- Construire le diagramme d'état d'une machine à laver dont le processus est le suivant:
 1. Verrouillage
 2. Remplissage de l'eau
 3. Actionnement du tambour
 4. Chauffage de l'eau
 5. Vidange
 6. Rinçage
 7. Essorage
 8. Déverrouillage

Inconvénient des automates :

Les analyses par automates d'états fonctionnent parfaitement lorsque le nombre de phrases types est limité.

Pour rendre un système efficace, il faut limiter le nombre de mots et de types de phrases acceptés, d'où la nécessité de faire une étude sérieuse du domaine et de sa limitation.

h. Analyse par formes grammaticales

L'analyseur par formes grammaticales est probablement le plus universel pour les traitements et interfaces en langages naturels. Puissant et complexe Prolog est prévu d'origine pour tenter d'utiliser la logique afin de formaliser les règles grammaticales.

Les règles de ce langage peuvent être des expressions écrites connues sous le nom de formes grammaticales. Ces formes peuvent être employées pour interpréter les Expressions de tous les langages, humain ou non.

Pour une telle étude, il faut structurer le langage selon ces règles.

Pour une étude simple, on utilisera un analyseur récursif descendant et libre du contexte (ne tient pas compte du sens de la phrase).

Création d'analyseur grammatical

Le traitement commence par couper les phrases en composants. Il commence par le groupe nominal et le groupe verbal. Poursuit l'analyse en cherchant les constituants de chaque groupe :

Noms, adjectifs, prépositions, verbes, etc.

L'opération se fait de haut en bas et commence par couper la phrase en deux éléments : groupe nominal, groupe verbal.

Chaque groupe est ensuite étudié par ses propres constituants. Certaines formes françaises peuvent être exprimées ainsi :

phrase = groupe nominal + groupe verbal.

groupe nominal = nom

groupe nominal = déterminant + nom

groupe nominal = déterminant + adjectif + nom
groupe nominal = préposition + groupe nominal

groupe verbal = verbe

groupe verbal = verbe + groupe nominal

groupe verbal = verbe + adverbe + groupe nominal

Soit la phrase suivante :

"Salim sort rapidement de la maison"

Groupe nominal = "Salim"

groupe verbal = "sort rapidement de la maison" = verbe + adverbe + groupe nominal;

Le dernier groupe nominal = "de la maison" = préposition + groupe nominal

Notons que la construction et l'analyse sont de type récursif.

Phrase = suite hiérarchique de structures

1° tâche = identification du groupe nominal et du groupe verbal.

2° tâche = analyse de chaque groupe dans l'ordre d'apparition tant que les composants ne sont pas identifiés comme éléments simples.

Exemple de système grammatical :

"Traitement de demandes d'horaires pour une compagnie d'aviation"

Analyse de phrases de type :

Quel avion va à Paris ?

Quel vol Royal Air Maroc va de Tanger à Bruxelles ?

Où va Fly Emirates vol 1324 ?"

On peut utiliser dans le programme trois bases de données :

- La première contient les informations sur les vols, les compagnies et les villes reliées
- La seconde base reçoit le dictionnaire des mots connus. Il sera présenté par un prédicat "mot".

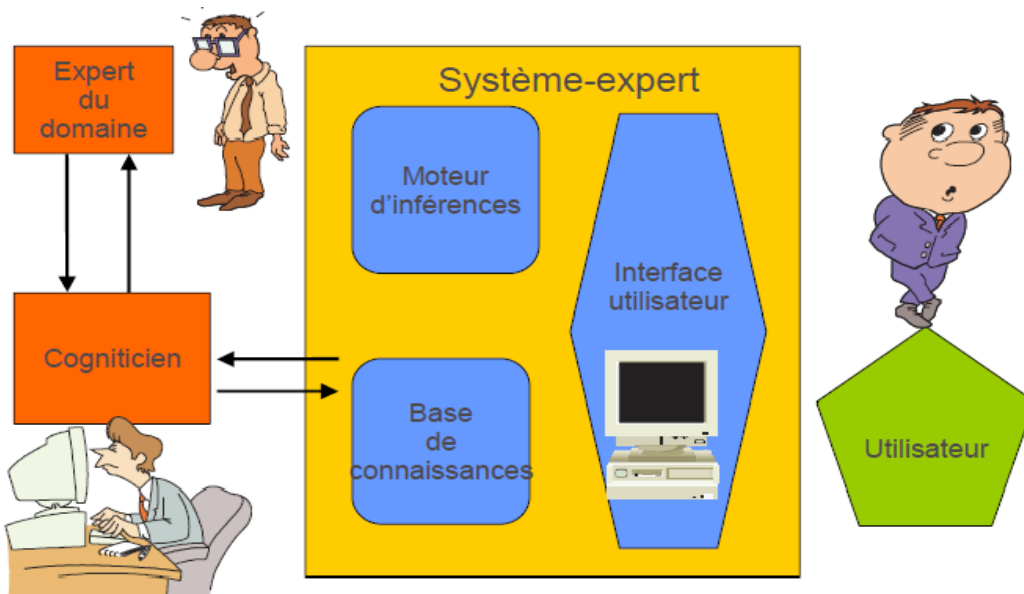
Ce prédicat pourra permettre de reconnaître le mot entré et la décision que l'analyseur doit prendre.

Exercice :

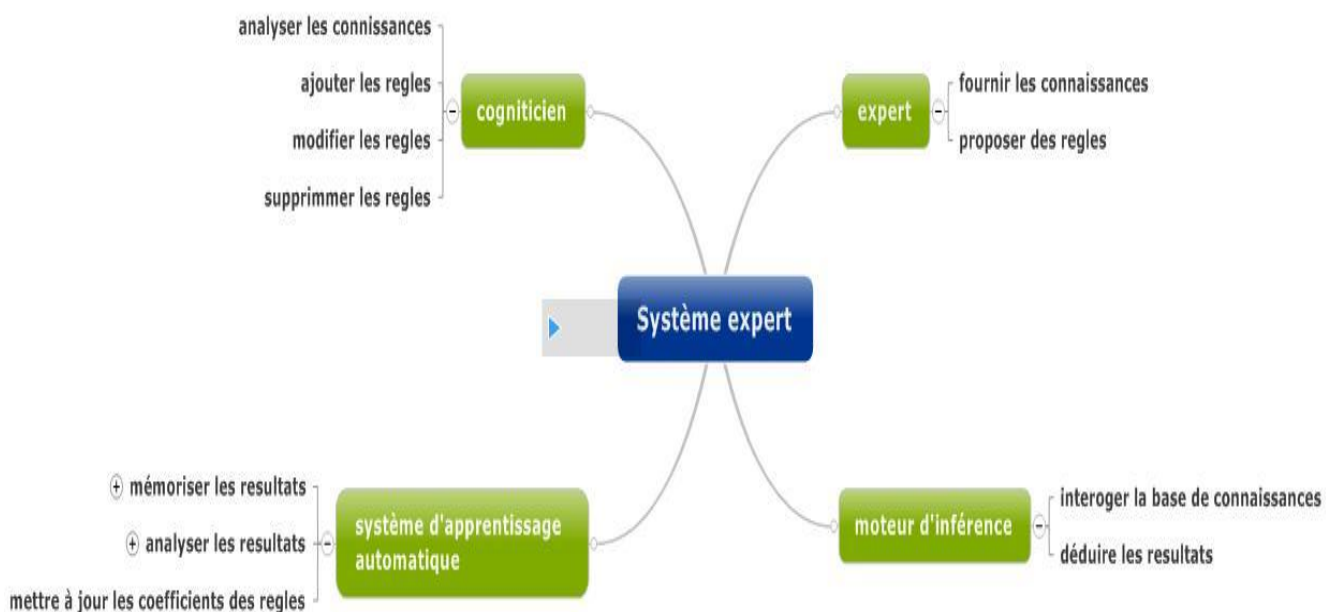
- Proposer une IHM intégrant un langage naturel pour un distributeur de boissons chaudes.

V. Modélisation Objet d'un Système Expert

- Spécification fonctionnelles
- Diagramme de Cas d'utilisation
- Diagrammes d'état
- Diagrammes de séquence
- Diagramme de classe

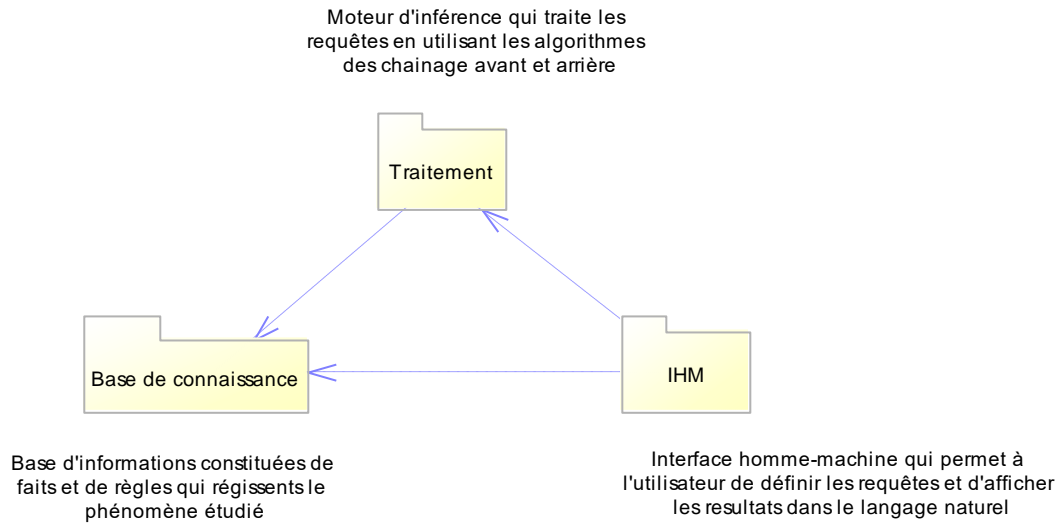


a. Spécification fonctionnelles



b. Packages

Le diagramme de package décrit le fonctionnement d'un Système expert et ses Sous-systèmes



c. Cas d'utilisation

Diagramme de UseCase de gestion de la base de connaissances

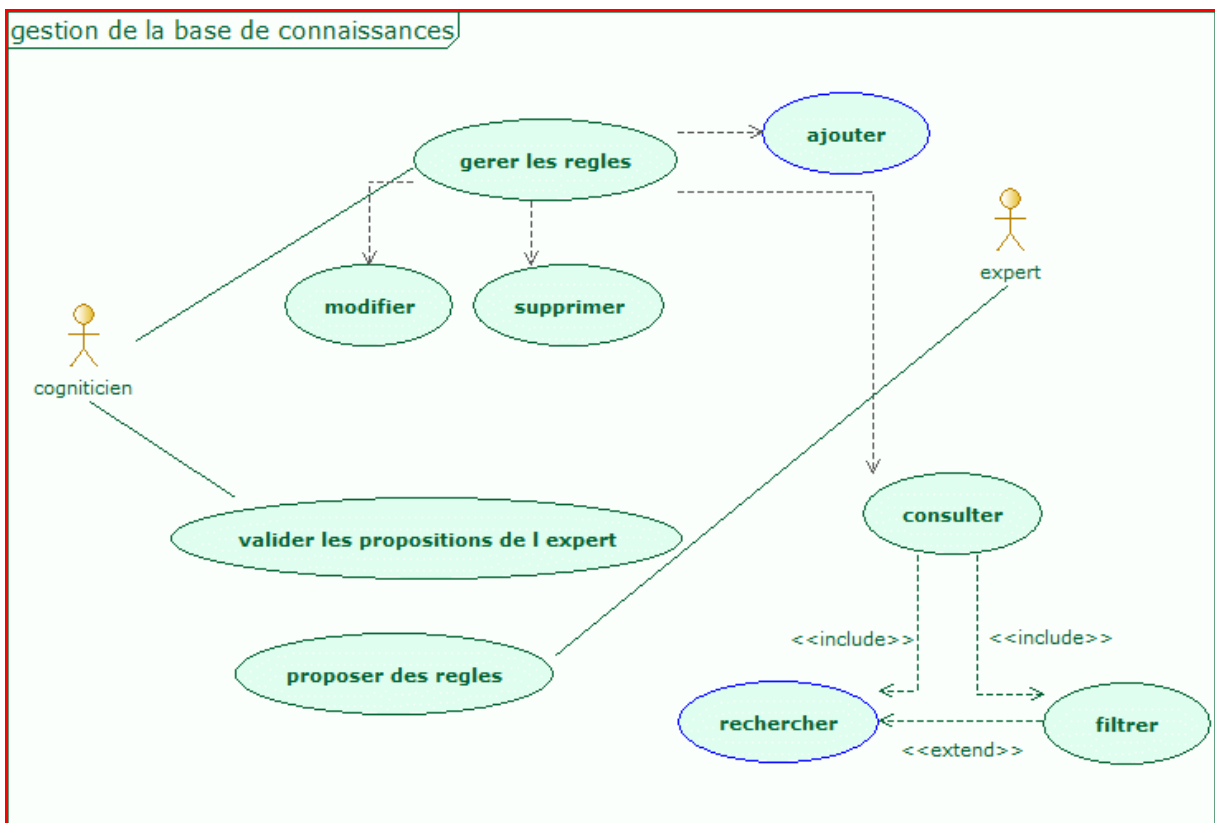
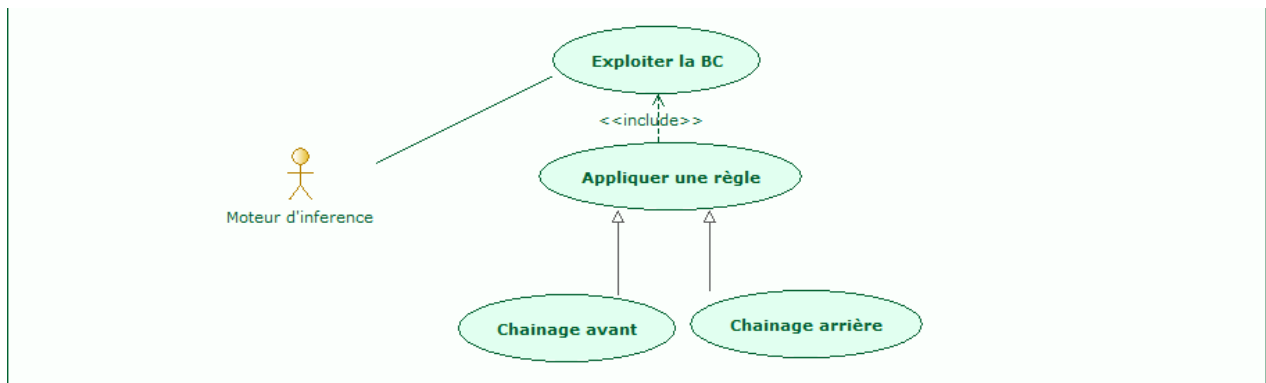


Diagramme de UseCase de gestion du moteur d'inférence



gestion du moteur d'inférence

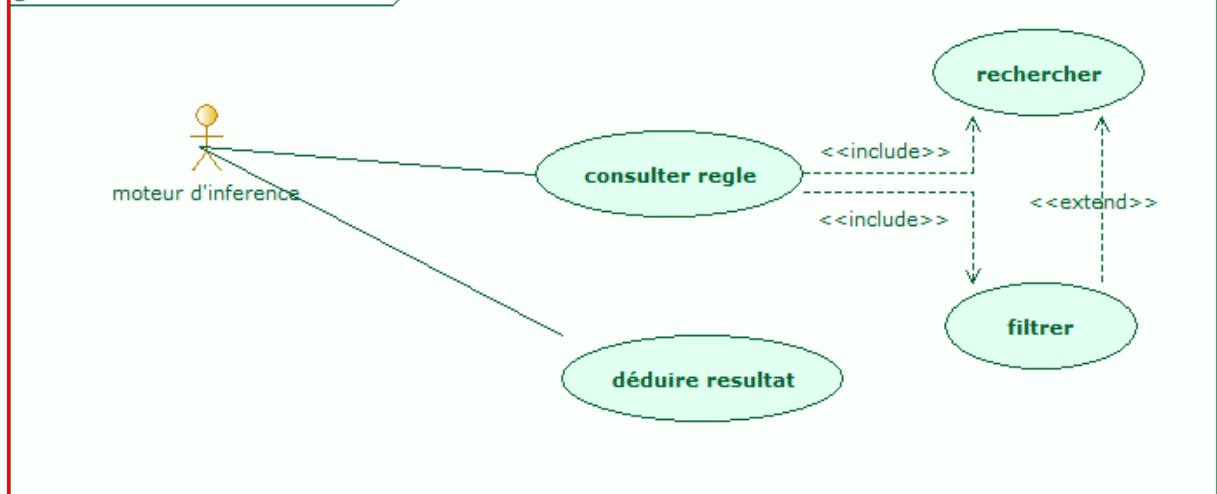
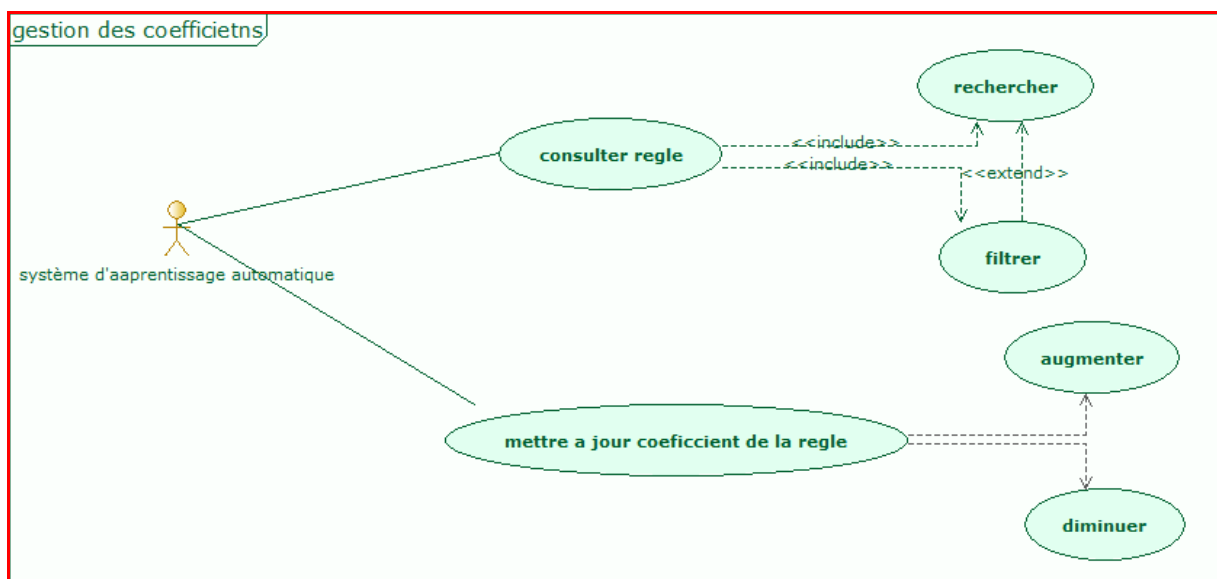


Diagramme de UseCase de gestion des coefficients



d. Diagrammes de scénarios

Diagramme de scénario d'ajout d'une règle

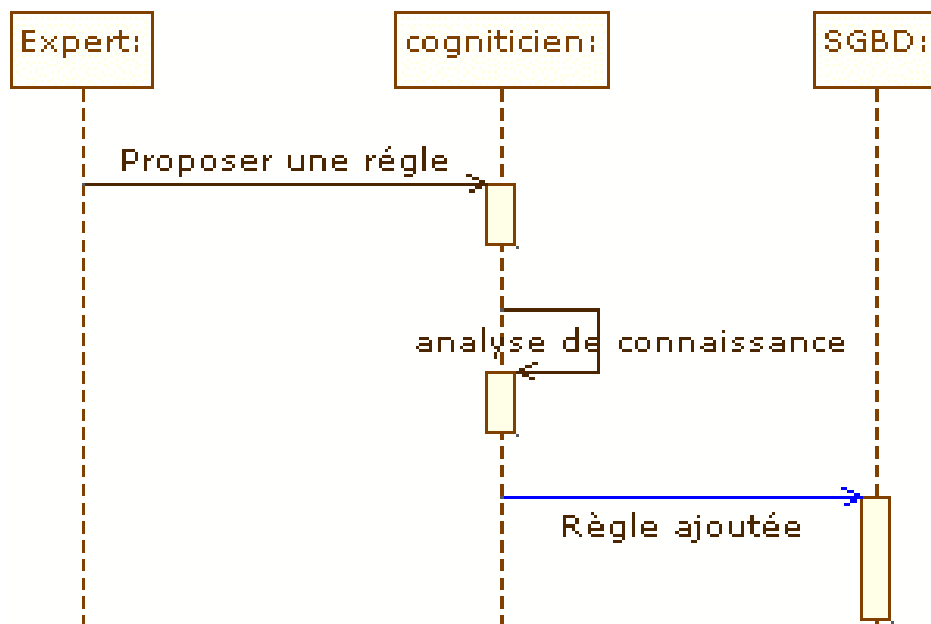


Diagramme de scénario de modification d'une règle

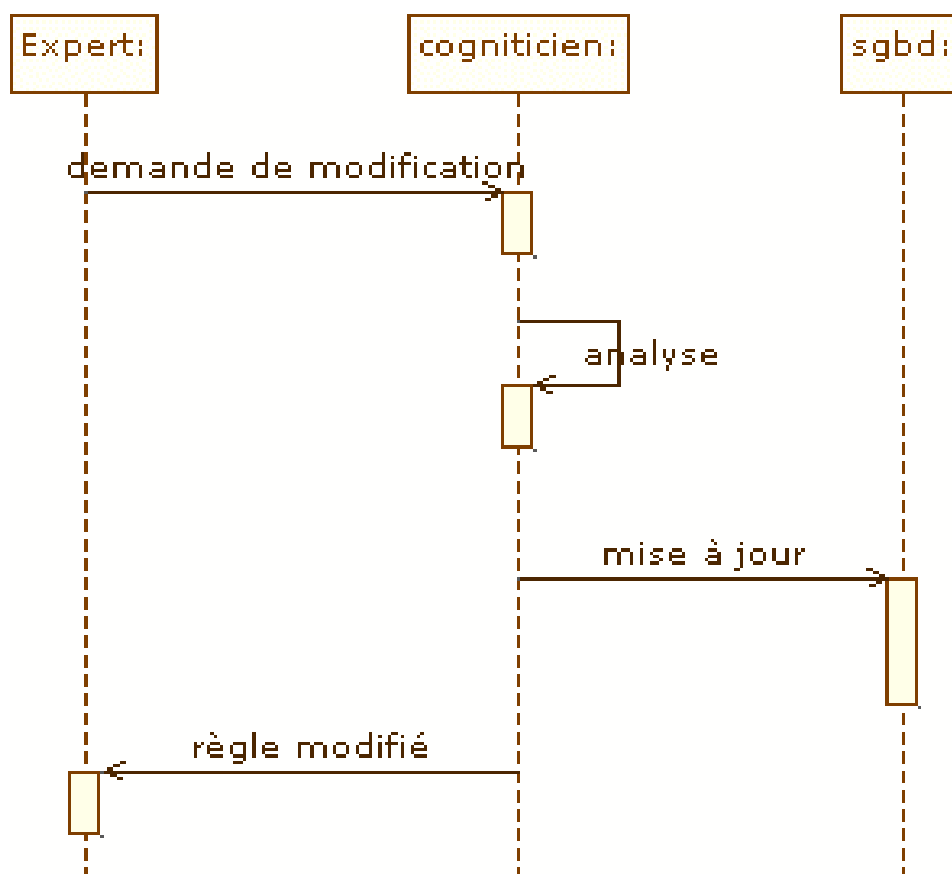


Diagramme de scénario de recherche d'une règle

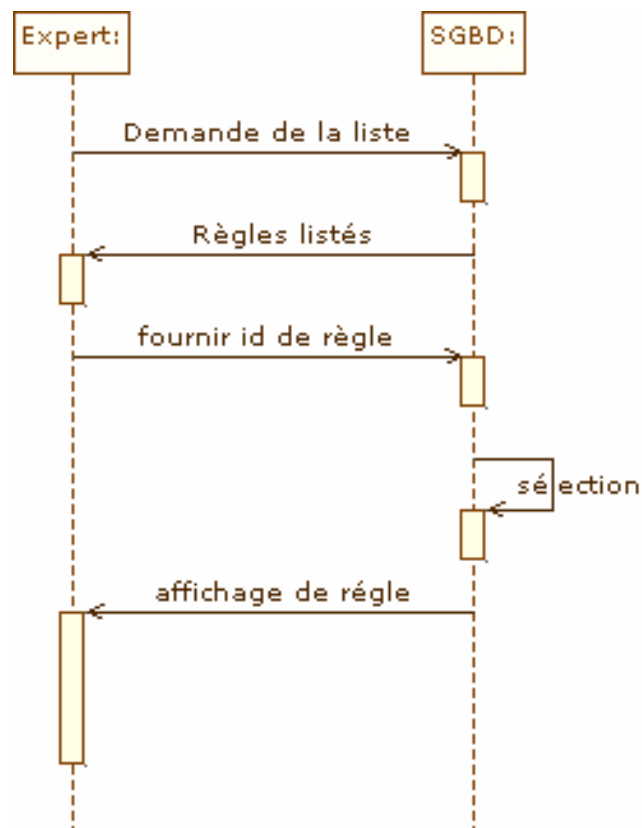
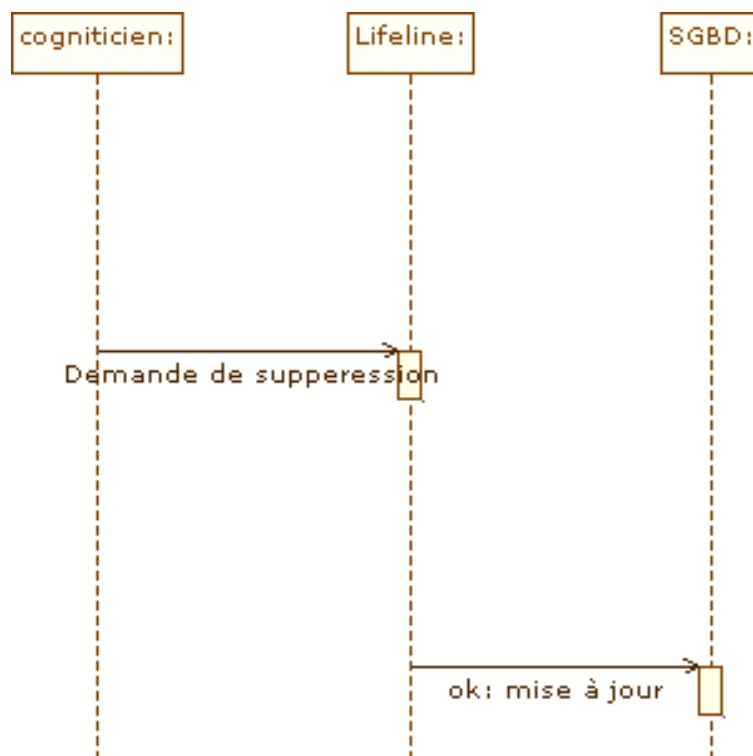


Diagramme de scénario de suppression d'une règle



e. Diagramme d'activité

Diagramme d'activité de la gestion de la base de connaissance

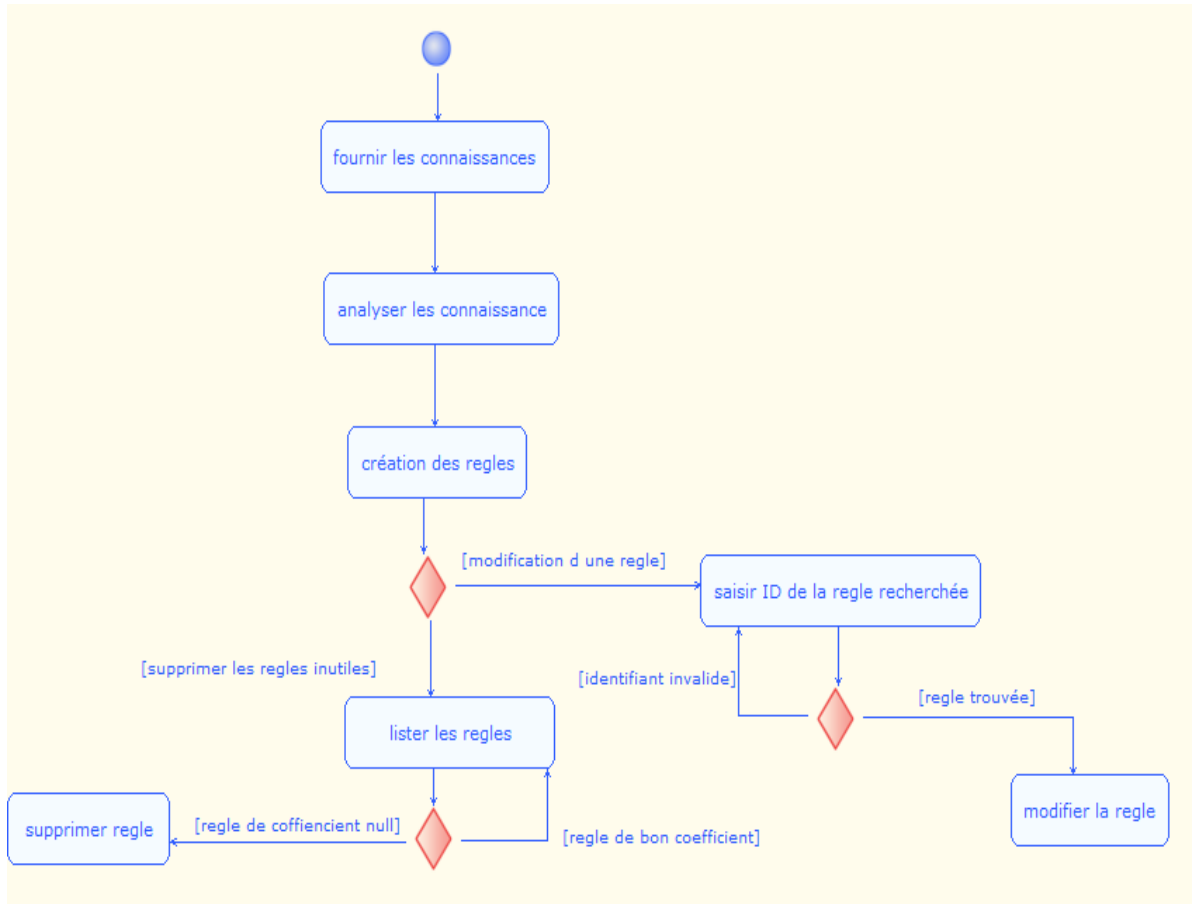


Diagramme d'activité de la gestion des coefficients

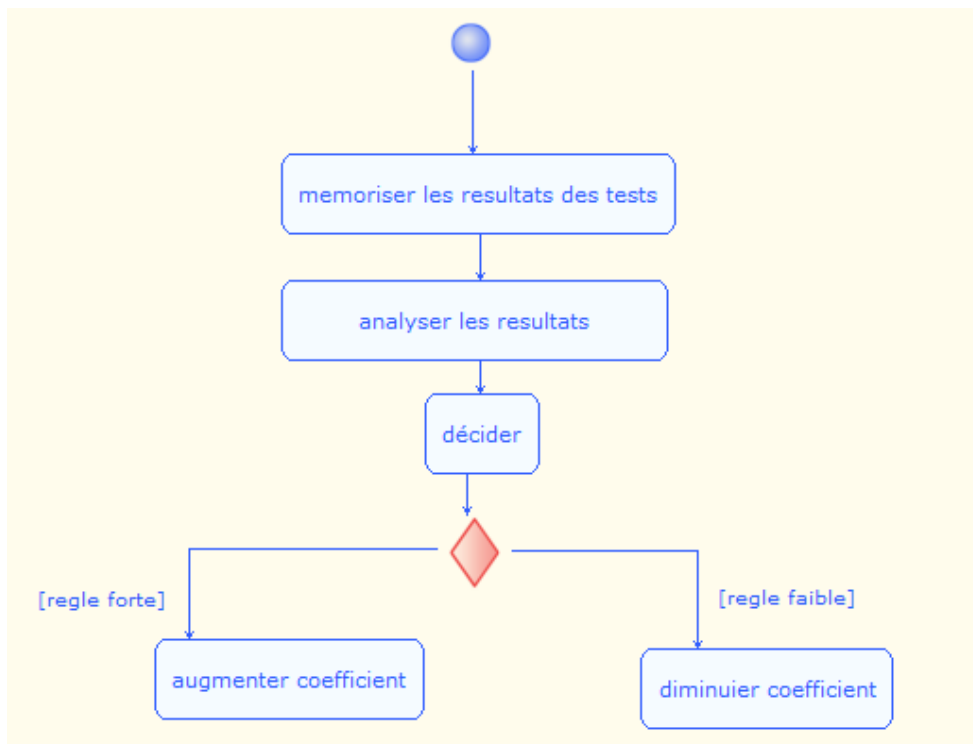
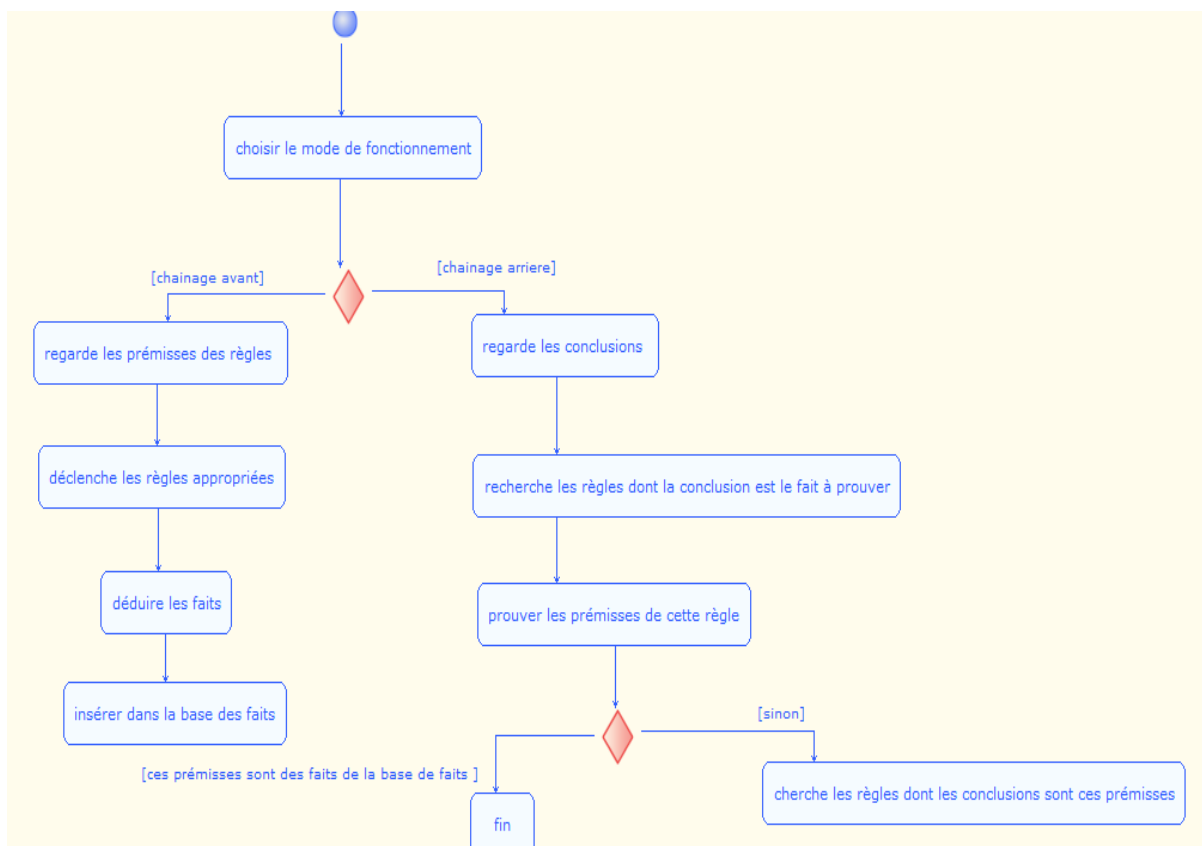
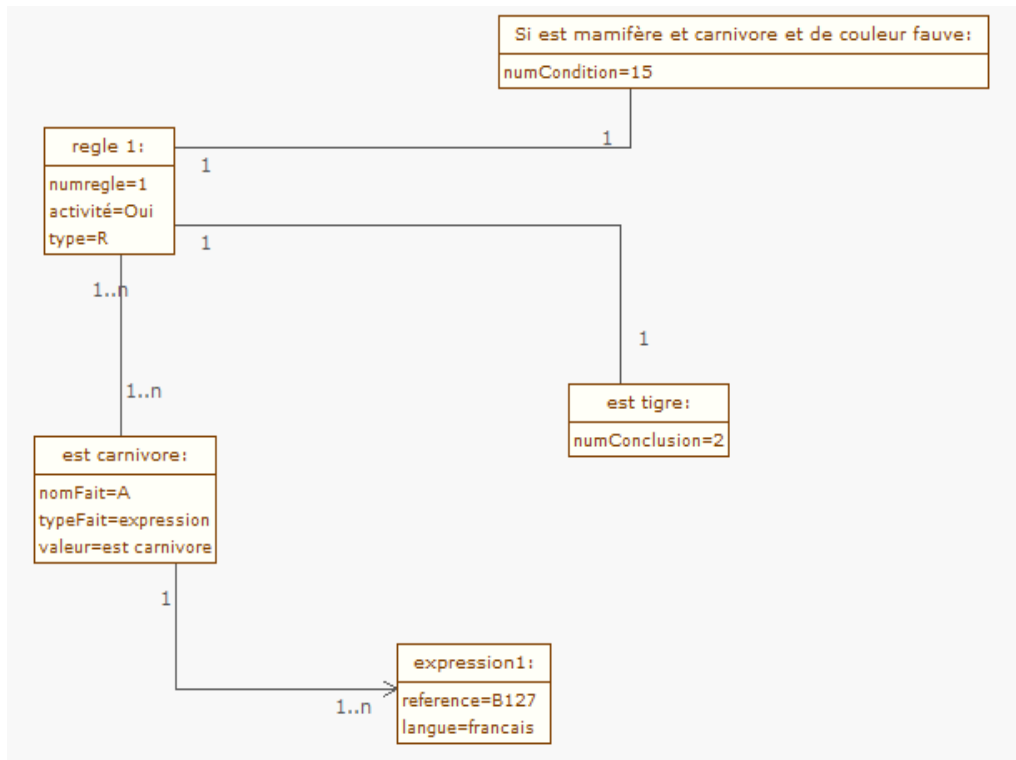


Diagramme d'activité du moteur d'inférence



f. Diagrammes d'objet

Diagramme d'objet :



g. Diagrammes de séquence

Diagramme de séquence d'ajout d'une règle

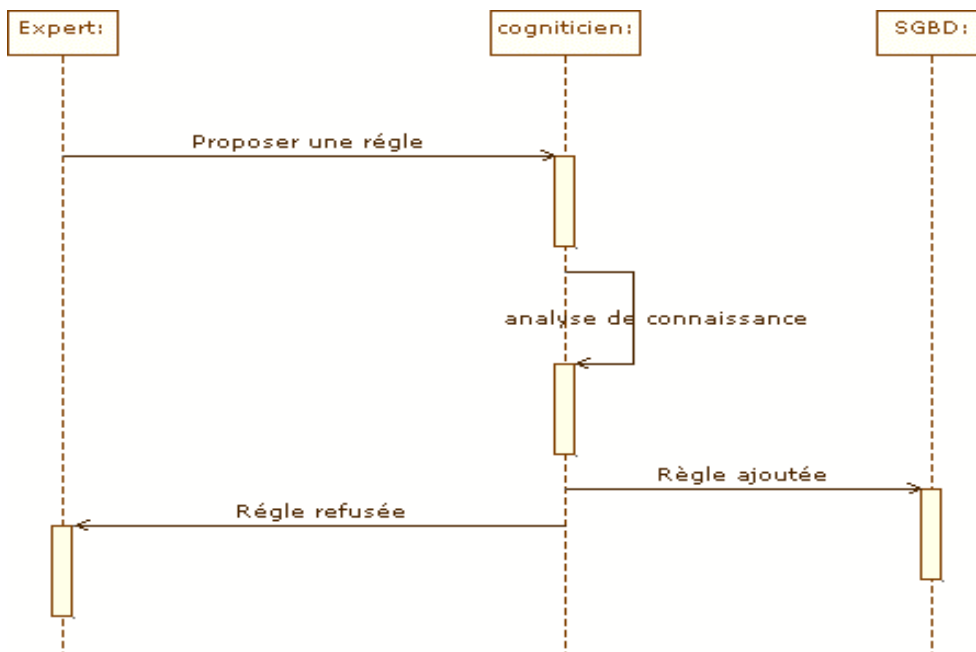


Diagramme de séquence de modification d'une règle

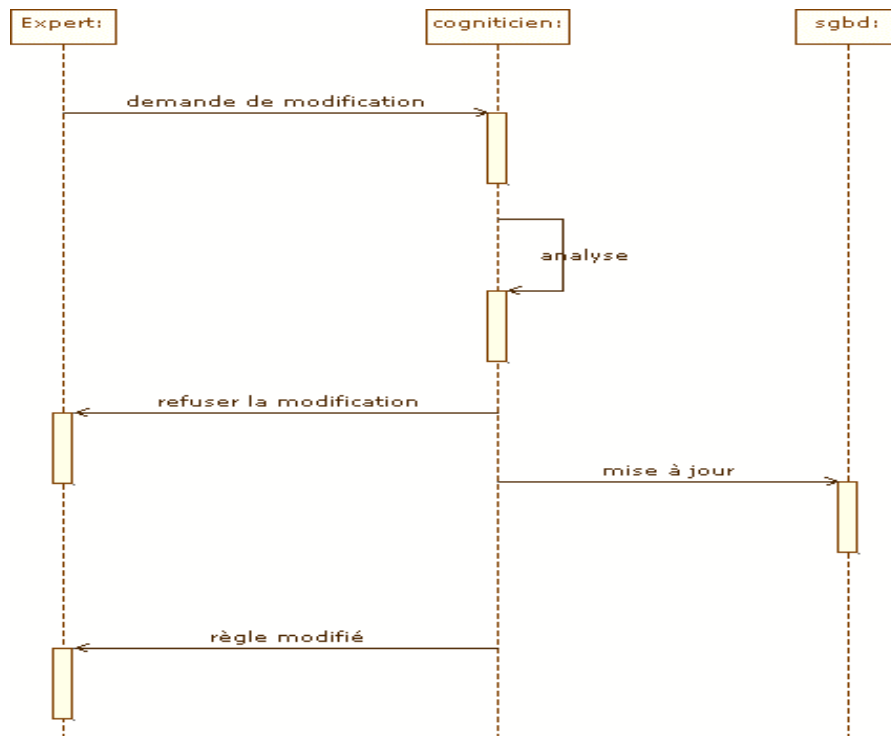


Diagramme de séquence de recherche d'une règle

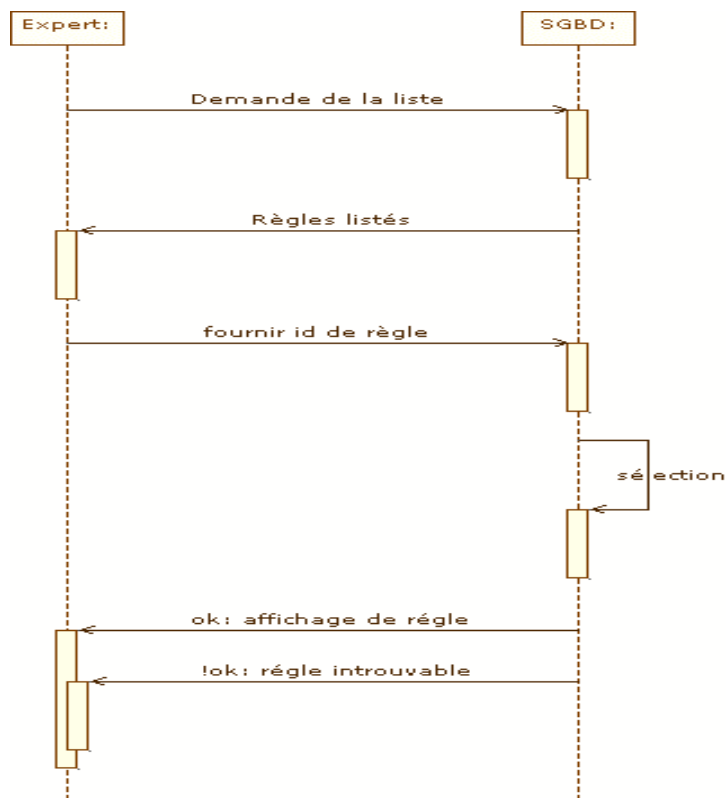
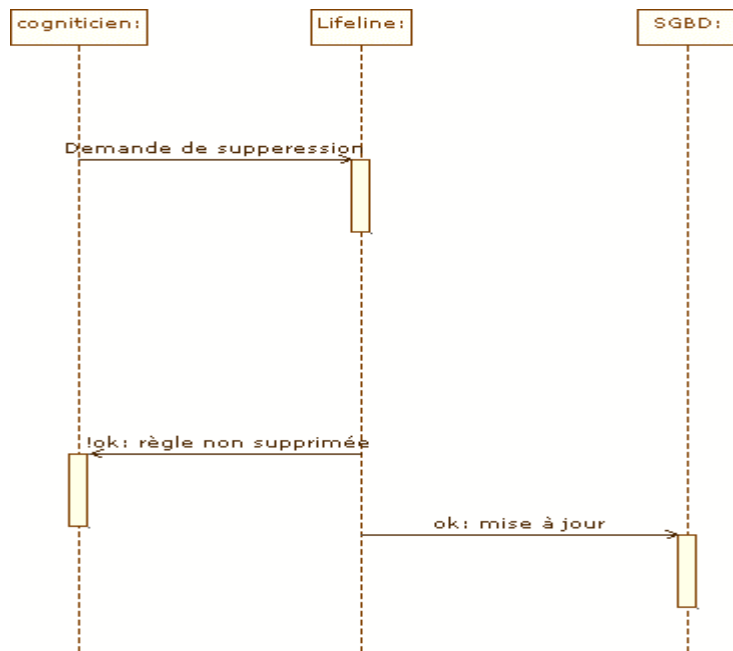
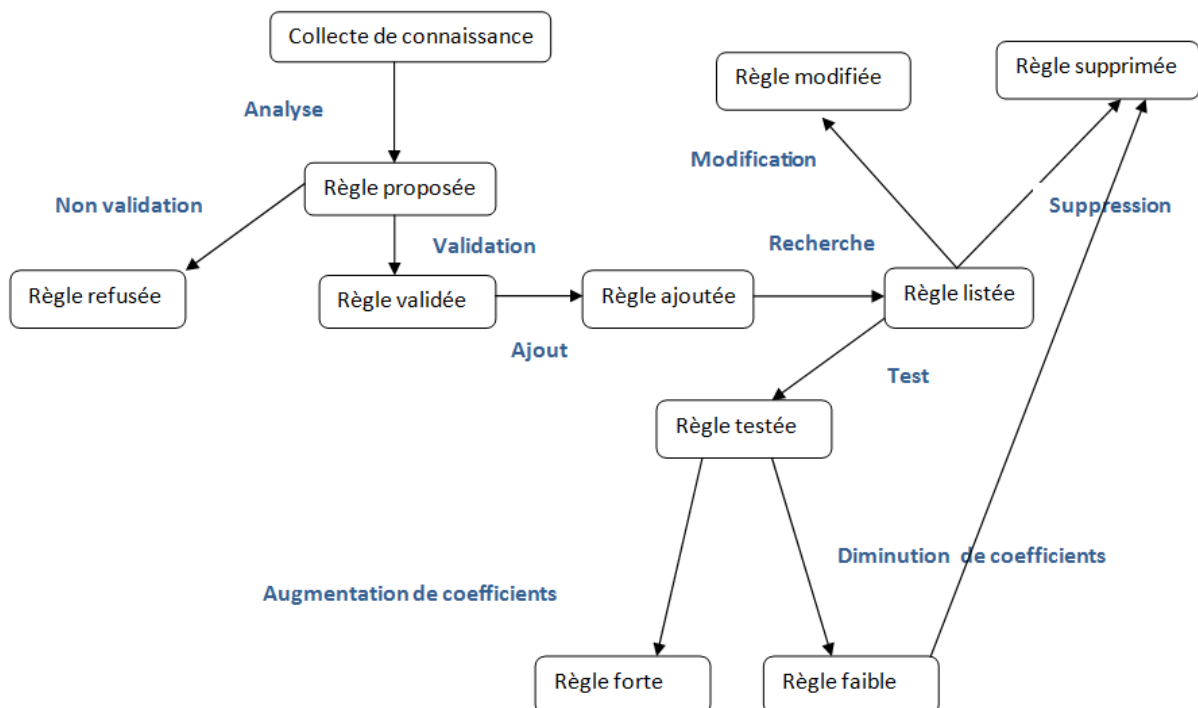


Diagramme de séquence de suppression d'une règle

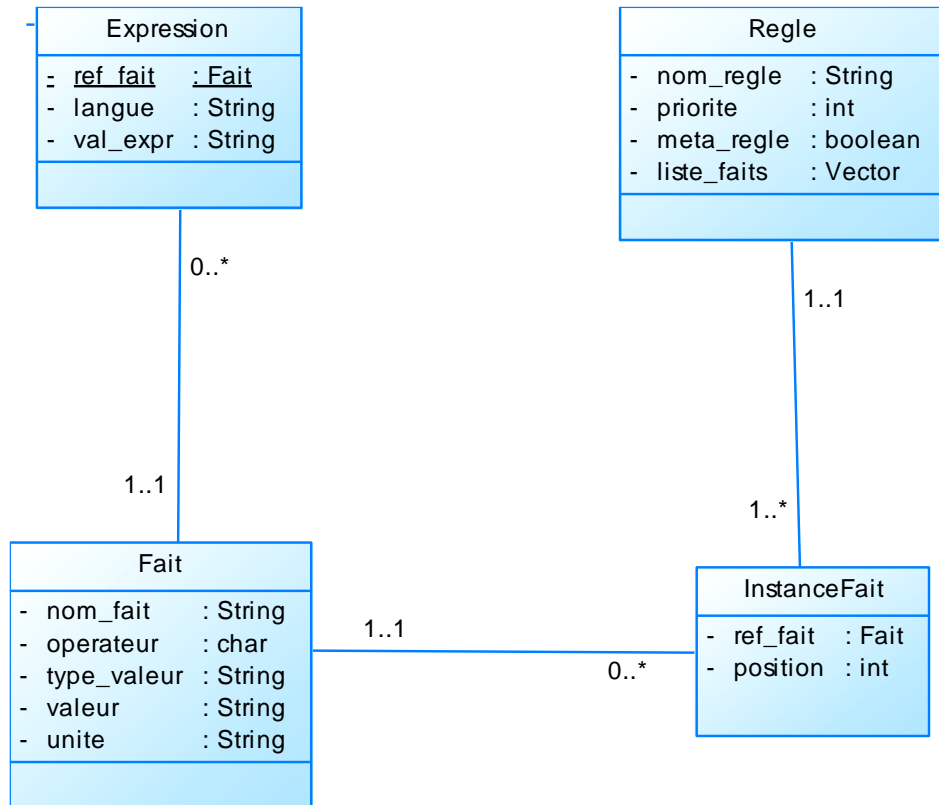


h. Modèle dynamique :



i. Diagramme de classe

Ce diagramme de classe contient finalement quatre classes qui permettent de représenter les données manipulées par un système expert.

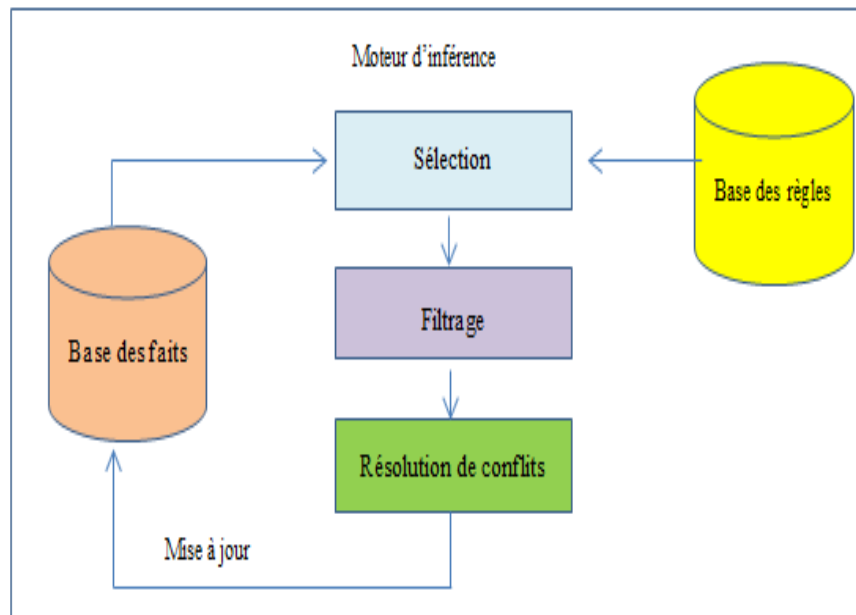


j. Exercice d'application (Devoir)

Revoir les diagrammes proposés afin de les améliorer et de les adapter par rapport au mini-projet que vous devez réaliser (voir la dernière page du présent document).

VI. Algorithmes

a. Le chaînage avant



Architecture technique d'un système expert basé sur un chaînage avant

- Pour déduire un fait particulier, on déclenche les règles dont les prémisses sont connues jusqu'à ce que le fait à déduire soit également connu ou qu'aucune règle ne puisse être déclenchée.
- Plus précisément l'utilisateur du système expert rentre des faits. A partir de ces faits rentrés, le système va essayer de déduire toutes les conclusions possibles, c'est-à-dire tous les attributs qui pourront être affectés. Il est clair que dans cet ensemble de conclusions possibles, il en est qui intéresseront l'utilisateur, il en est d'autres qui ne lui diront rien, il en est enfin qui manqueront.
- Les faits sont de la forme : attribut = valeur
- On va prendre chaque fait et on va examiner toutes les règles où ce fait apparaît en prémisses. Grâce au mécanisme explicité plus haut, on va voir pour chacune de ces règles si elle est déclenchée. Si elle l'est, on va affecter les attributs en conclusion des valeurs qui leur correspondent. On dira que les faits ont été propagés.
- Ces attributs affectés feront partie du résultat final de l'expertise ; et, en même temps, ils seront eux-mêmes propagés.
- On fait cela jusqu'à l'épuisement des faits, et on communique les résultats à l'utilisateur.

Algorithme du chaînage avant

ENTREE : BF, BR, F

DEBUT

TANT QUE F n'est pas dans BF

ET QU'il existe dans BR une règle applicable FAIRE

 choisir une règle applicable R (étape de résolution de conflits, utilisation d'heuristiques, de métarègles)

 BR = BR - R (désactivation de R)

 BF = BF union concl(R) (déclenchement de la règle R, sa conclusion est rajoutée à la base de faits)

FIN DU TANT QUE

SI F appartient à BF ALORS

 F est établi

SINON

 F n'est pas établi

b. Le chaînage arrière

- Le mécanisme de chaînage arrière consiste à partir du fait que l'on souhaite établir, à rechercher toutes les règles qui concluent sur ce fait, à établir la liste des faits qu'il suffit de prouver pour qu'elles puissent se déclencher puis à appliquer récursivement le même mécanisme aux faits contenus dans ces listes.
- Le chaînage arrière est un mécanisme d'induction, c'est-à-dire qu'on vérifie les hypothèses en remontant depuis l'objectif. On cherche ainsi à vérifier si un fait est possible : celui-ci étant choisi selon des critères extérieurs.
- L'algorithme dans son principe est assez simple : pour chaque attribut, on définit ses sources.
- Si l'attribut est initial c'est-à-dire s'il n'apparaît qu'en prémisse de règle, les sources sont réduites aux questions.
- Si l'attribut est final, les sources sont les règles dans lesquelles l'attribut est en conclusion.
- Si l'attribut est intermédiaire, apparaissant à la fois en prémisse et en conclusion, alors les sources peuvent être les règles, ou alors on peut se replier sur les questions au cas où elles n'auraient rien donné.
- Si, une fois la question posée, l'utilisateur répond, la valeur qu'il donne est affectée à l'attribut et c'est bon. Si les sources sont les règles, on prend chacune des règles où l'attribut apparaît en conclusion, et on essaye d'évaluer les attributs qui sont en prémisse. Si la règle se déclenche, l'attribut en conclusion est évalué.
- L'algorithme de chaînage arrière est nettement plus compliqué que le précédent et nous nous contenterons d'étudier un exemple.

- L'exécution de l'algorithme de chaînage arrière peut être décrit par un arbre dont les noeuds sont étiquetés soit par un fait, soit par un des deux mots et, ou. On parle d'arbre et-ou.
- Sur l'exemple précédent, on obtient l'arbre de la figure de la page suivante.

Algorithme de chaînage en arrière

```
DEBUT
  Phase de selection ; phase de filtrage
  SI ensemble de règles applicable vide
    ALORS questionner utilisateur
  SINON
    TANTQUE but non résolu ET il reste des règles sélectionnées
      FAIRE
        Résolution de conflits
        Ajouter les sous-buts
        Résoudre les sous-buts si possible
    FINTANTQUE
FIN
```

c. Le chaînage mixte

- L'algorithme de chaînage mixte combine, comme son nom l'indique, les algorithmes de chaînage avant et de chaînage arrière. On peut alors aussi bien raisonner à partir des faits que l'on connaît comme des prédicats ou comme objectifs.
- C'est aussi une extension du chaînage arrière qui supprime ce dernier car il pallie à un problème qu'il pose. En effet, le chaînage arrière pose le problème du retard à l'évaluation. Lorsque l'on évalue un but par chaînage arrière, l'évaluation peut conduire à des conclusions sur d'autres attributs, qui ne sont pas faites et pour lesquelles d'autres procédures de chaînage arrière sont inutilement invoquées.
- De là l'idée de combiner le chaînage arrière, inefficace à lui seul, à du chaînage avant. Ainsi, après chaque évaluation d'une prémisse de règle, une propagation en avant de cette évaluation est faite pour tirer l'ensemble des conclusions qu'il est possible d'en tirer.

Algorithme du chaînage mixte

ENTREE : F (à déduire)

DEBUT

TANT QUE F n'est pas déduit mais peut encore l'être FAIRE

Saturer la base de faits par chaînage AVANT

(c'est-à-dire, déduire tout ce qui peut être déduit)

Chercher quels sont les faits encore éventuellement déductibles

Déterminer une question pertinente à poser à l'utilisateur et

ajouter sa réponse à la base de faits

FIN DU TANT QUE

FIN

VII. Exemples d'applications

a. Application pour un diagnostic médical

Exemple de base de règles :

R1 : Si douleur abdominale et nausées **et** fièvre

Alors suspicion maladie de digestion

R2 : Si suspicion maladie de digestion **et** maux de tête **et** fièvre importante

et pas de jaunissement des yeux

Alors dysenterie

R3 : Si suspicion maladie de digestion **et** fatigue **et** coloration jaune de la peau

Alors jaunisse

R4 : Si suspicion maladie de digestion **et** fièvre importante **et** douleur irradiante à droite

Alors coliques hépatiques

R5 : Si suspicion maladie de digestion **et** faible fièvre

et douleur vive en bas à droite de l'abdomen

Alors appendicite

R6 : Si maux de tête

Alors suspicion maladie respiratoire

R7 : Si toux

Alors suspicion maladie respiratoire

R8 : Si suspicion maladie respiratoire **et** douleur à la racine du nez **et** fièvre

Alors sinusite

R9 : Si suspicion maladie respiratoire **et** irritation de la gorge et sensation de sécheresse

Alors inflammation de la gorge

R10 : Si inflammation de la gorge **et** fièvre et gorge rouge

Alors angine

R11 : Si inflammation de la gorge **et** toux rauque

Alors laryngite

R12 : Si suspicion maladie respiratoire **et** crachat mousseux après 1/4 h de toux

et respiration rapide

Alors oedème aigu du poumon

Exemple de dialogue entre le SE et l'utilisateur:

Entrez le nom de la base : médecin

Quoi de neuf : respiration rapide

Quoi de neuf : toux

Je déduis : suspicion de maladie respiratoire

J'ai utilisé la règle : 7

Quoi de neuf : crachats mousseux après 1/4 h de toux

Je déduis : oedème aigu du poumon

J'ai utilisé la règle : 12

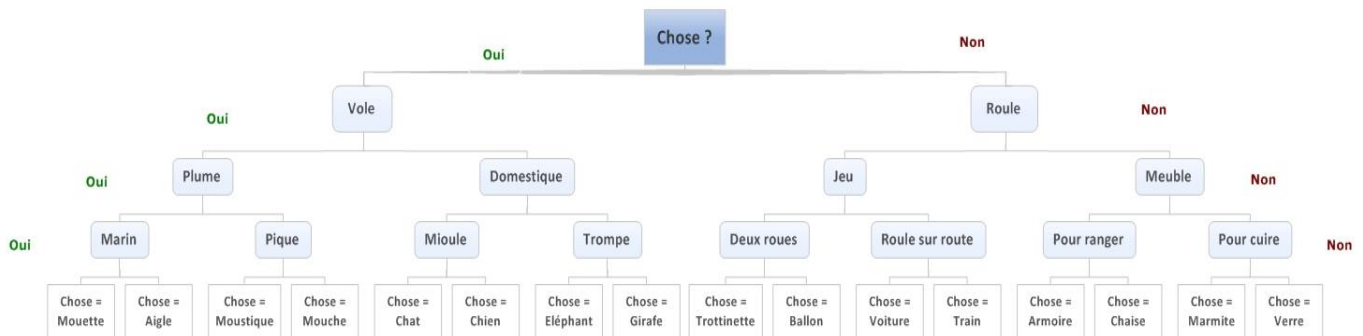
Critiques

Si le SE précédent a le grand mérite d'être rapidement réalisable, il a en contrepartie l'inconvénient d'être, bien sûr, très limité.

b. Application sur la base d'arbre de décision

Réaliser un Système Expert « Chose » qui devine si la Chose recherchée est un animal ou une chose non vivante.

L'arbre à décision suivante donne un exemple de raisonnement suivi. Quant aux exemples de trace, ils reflètent le dialogue naturel de l'IHM entre le Système Expert (représenté par la suite par : 🖨) et l'Utilisateur (représenté par la suite par : 🧠).



Principe d'un arbre de décision

Exemple de dialogue entre le SE et l'Utilisateur (ex. 1)

(l'UTILISATEUR lance le chaînage mixte en demandant au système de déterminer quelle est la chose ?)

🧠 - chose ?

🖨 - est-ce que vivant ?

🧠 - NON

🖨 - est-ce que roule ?

🧠 - NON

🖨 - est-ce que meuble ?

🧠 - NON


🖨 - chose="verre"

Exemple de dialogue entre le SE et l'Utilisateur (ex. 2)


(l'UTILISATEUR lance le chaînage mixte en demandant au système de déterminer quelle est la chose ?)


🧠 - chose ?


(l'UTILISATEUR lance le chaînage mixte en demandant au système de déterminer quelle est la chose ?)


 - est-ce que vivant ?


(première question posée par le SYSTEME EXPERT)


 - OUI


 - est-ce que vole ?


 - NON

 - est-ce que domestique ?

 - OUI

 - est-ce que miaule ?

 - OUI

 - chose="chat"

c. Exercice

1. Imaginer une session d'utilisation d'un système expert qui utilise la base de règles "ville méritant le voyage" vue dans la première partie (page 11).

2. Construire l'arbre de décision relative à la base de connaissance suivante :

- BF = {pommes, poires, abricots, farine, beurre, sucre, sel}

- Base de règles :

R1 : Si farine et beurre et œufs et sel alors pâte

R2 : Si pommes et sucre alors pommes sucrées

R3 : Si pommes sucrées et pâte alors tarte aux pommes

R4 : Si abricots et pâte alors tarte aux abricots

R5 : Si poires et pâte alors tarte aux poires

R6 : Si cerises et pâte alors tarte aux cerises

VIII. Projet de réalisation de Systèmes Experts

Des sujets vous seront proposés pour réaliser votre premier système expert. Dans ce qui suit, vous allez trouver des indications et des algorithmes qui vont vous permettre de faire votre réalisation. Les éléments abordés dans cette section seront traités en cours dans le cadre d'atelier pratiques. De même, votre réalisation doit être testée sur la base des deux exemples d'applications proposés dans cette section.

Indication à suivre pour la Réalisation de votre premier SE

Nous allons réaliser un petit système expert possédant les caractéristiques suivantes :

- architecture classique :
 - base de faits,
 - base de règles
 - et moteur d'inférence possédant uniquement un chaînage avant.
- Langage simple et clair pour exprimer un problème éditeur de règles
 - un langage pour une programmation objet : C++, Java, PHP, Ruby, Python, ...
 - un langage pour une programmation logique : Prolog ou Caml
- Choisissez une plateforme selon la manière d'utiliser ce premier système expert:
 - Plateforme FULL Web : .Net, J2EE, Ruby On Rails, Laravel, Django, ...
 - Application Desktop utilisable sur un serveur local

Étapes à suivre pour construire un tel système expert :

- Modéliser par une représentation UML votre projet (améliorer celle présentée précédemment).
- Définir une syntaxe et réaliser un éditeur de règles
- Développer et déployer votre application en tenant compte :
 - Développer une interface Homme-Machine
 - Gestion de la base des faits
 - Gestion de la base des règles
 - Programmation des modules ou programmes suivants :
Filtrage, Dédution, Chaînage avant et Chaînage arrière
- Faire des essais et apporter des améliorations