REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE FERHAT ABBES –SETIF-

MEMOIRE

Présente à la faculté des sciences
Département d'Informatique
Pour l'obtention du diplôme de

MAGISTER

Option : Matériel et Logiciel

Par
Mr. BELAZOUG MOUHOUB

Thème :

Interrogation d'une Base de Données à partir d'une Base de connaissances dans un Environnement de Développement de connaissances.

Soutenu le : 28/09/2011

Devant le jury

Président	: TOUAHRIA MOHAMED	**M.C.A**	**U.F.A. Sétif**
Rapporteur	: MOSTEFAI MOHAMMED	**Prof.**	**U.F.A. Sétif**
Examinateur	: KHABABA ABD ALLAH	**M.C.A**	**U.F.A. Sétif**

Dédicaces

À la mémoire
De ma chère Mère.

À mon chère père.
Et à mes sœurs et frères.

À mon épouse,
A mon enfant Mohamed.

Remerciements

Je voudrais de profiter de cet espace pour exprimer toutes mes gratitudes et mes reconnaissances à mon encadreur et enseignant Mr MOSTEFAI mohamed qui n'a pas cessé de m'encourager et de me guider ver la bonne voie tout au long de la réalisation de ce travail.

Je tiens également à remercier M. TOUAHRIA MOHAMED, Maître de Conférence à l'université de Sétif, pour son aide, et qui m'a fait l'honneur de présider le jury,

Ainsi que M. KHABABA ABD ALLAH, Maître de Conférence à l'Université de Sétif, pour avoir accepté de faire partie de ce jury.

Enfin, je salue l'ensemble des professeurs et étudiants du département informatique.

BELAZOUG MOUHOUB

SOMMAIRE

Chapitre 3 Techniques des Arbres de Décisions.

Chapitre 4 Présentation et Modélisation.

Chapitre 5 Implémentation et Validation.

Liste des Figures

Introduction

Les premiers systèmes experts sont apparus dans le milieu des années 70 par (SE) dont le but est de modéliser le comportement d'un expert humain dans un domaine bien circonscrit, on distingue les connaissances spécifiques aux domaines, qui sont énoncées de manière déclarative dans une base de connaissances, et des procédures d'utilisation de ces connaissances, qui constituent le moteur d'inférence.

Les problèmes qui font de l'acquisition des connaissances un *"goulot d'étranglement"* dans la construction d'un système expert sont donc multiples.

Dans certains domaines, les experts sont rares et de ce fait, il n'est pas toujours possible de s'assurer leur présence pendant de longues périodes à la fois pour des raisons géographique et économique. Du point de vue même de l'acquisition, il est évident que l'on peut être à la fois un excellent expert et un fort mauvais pédagogue, savoir utiliser des connaissances mais être incapable de les formuler.

Le travail proposé dans le cadre de la présente thèse de magister se situe dans le contexte d'une approche objet, dans les style SHIRKA, il s'inscrit dans le cadre des modèles centrés- objets, le but principale est de montrer l'intérêt et les apport d'un Langage déclaratif pour la représentation de connaissances (LRC) pour résoudre des problèmes de représentation, et en deuxième Temps créer un aiguillage durant le processus d'exécution par la mise en plage d'un couplage : Base de connaissances/Base de données.

Une fouille de données visant à extraire une connaissance d'une base de donnée via d'un processus d'extraction de connaissances a partir de données (E.C.D), en vue de constituer une base de connaissances initiale et en minimisant le recours à l'utilisateur, et d'autres part utilisé ces mécanisme d'interrogation des bases de données pour la classification des faits représentés dans notre langage (L.R.C).

Le chpaitre1 est un état de l'art sur les systèmes experts, en décrivant les systèmes experts généraux de deuxième génération (Shell), l'historique des systèmes experts, et les différentes méthodologies des connaissances.

Au chpaitre2 on présente l'ingénierie des connaissances : on décrit les principes de génie cognitif, les différents types de représentation des connaissances, aussi les phases d'acquisition de connaissances et les mécanismes d'un moteur d'inférence.

Au chpaitre3 est consacré à l'apprentissage automatique des arbres de décisions, on montre les techniques de construction des arbres de décisions, les algorithmes, les différentes techniques d'élagage, et a la fin on présente l'implémentation de notre module

d’apprentissage ***TreeClassifier*** qui utilise techniques des arbres de décisions (l’algorithme C4.5 de QUINLLAN).

Dans le chpaitre4 on présente, une modélisation de notre système Expert ***ExperTreeRule*** on utilisant UML, on a pris comme exemple la base de connaissance aide au voyageur.

Le chpaitre5 qui est le dernier chapitre correspond à l’implémentation et validation du système Expert Global de la base de connaissance (classes, objets et règles de productions), qui regroupe les deux modules, l’apprentissage Automatique & le Système Expert .

Chapitre 1

Etat de L'art

1. Introduction

L'Intelligence Artificielle (IA) est l'art de fabriquer des systèmes intelligents, systèmes capables de résoudre des problèmes dans des circonstances variées, en s'inspirant du raisonnement humain plutôt qu'en appliquant une démarche algorithmique.

Un domaine majeur de l'IA est la conception de systèmes déclaratifs, communément appelés Systèmes à Bases de Connaissances (SBC). Ces systèmes sont caractérisés par une séparation entre les connaissances nécessaires pour résoudre un problème et les mécanismes abstraits exploitant ces connaissances. Cette séparation permet de décrire les connaissances indépendamment de leur utilisation ultérieure. D'une part, ceci facilite la modification et l'ajout de nouvelles connaissances à la base. D'autre part, on peut fournir des justifications et des explications du comportement du système.

L'un des premiers défis dans le domaine l'intelligence artificielle (IA) a été de concevoir des systèmes à base de connaissances (SBC) pour simuler le raisonnement d'experts dans leurs domaines d'expertise. Cette tentative a conduit à de nombreux systèmes, notamment un ensemble de méthodologies visant à aider dans la modélisation des connaissances.

L'intelligence artificielle se situe au carrefour de l'informatique et ce qu'on appelle souvent les « sciences cognitives », qui visent à comprendre comment la connaissance naît, s'utilise, évolue et se transmet.

2.Domaines d'application des systèmes à base de connaissances

Les thèmes abordés par l'IA sont multiples. Ils concernent aussi bien le logiciel que le matériel. Un projet complet (par exemple le projet japonais dit de 5^{e} génération) mêle souvent plusieurs types de préoccupations (programmation, interfaçage, architecture des ordinateurs, conception des circuits, etc.). Voici les catégories principales:

- **Traitement du langage naturel**:

  La traduction automatique demande plus que la fabrication de dictionnaires et l'utilisation de règles de syntaxe. Traduire un texte nécessite une compréhension du texte ou, pour le moins, une certaine représentation qui fait intervenir le contexte, une connaissance experte de la matière et un certain sens commun.

  Actuellement le traitement du langage (traduction automatique, génération automatique de textes, analyse du langage naturel, ...) introduit de multiples outils:

✓ Utilisation de SCRIPTS qui sont des séquences standard de déroulement d'un dialogue.
✓ Représentation par dépendance conceptuelle.
✓ Réseaux neuronaux.

- **Recherche dans des bases de données:**

  Il s'agit de dépasser la recherche par mots clés pour effectuer des recherches à partir de demandes effectuées en langage naturel. L'ordinateur assiste l'utilisateur dans sa démarche.

- **Assistance**:

  C'est l'application commerciale principale des systèmes à base de connaissances. Il s'agit du domaine des systèmes experts qui sont caractérisés à la fois par une fonction (assistance qui peut remplacer celle d'un expert) et par un certain type de systèmes informatiques. Le fonctionnement d'un système expert est prototype des systèmes à base de connaissances.

- **Démonstration de théorèmes mathématiques**

  C'est un sujet classique qui a permis de développer les principales méthodes d'IA. Plusieurs langages de programmation utilisés en IA sont basés sur les techniques utilisées dans ce domaine (PROLOG, PLANNER).

- **Planning**:
  Ce sujet est également important dans le développement de l'IA. Un but est proposé, et l'ordinateur doit trouver l'ensemble des étapes qui permettent de l'atteindre. La robotique de même que la gestion de production est largement tributaire des recherches menées dans ce domaine.

- **Programmation automatique :**
  Il s'agit de la réalisation de programmes à partir de la description des fonctions à remplir. Sorte de « super compilateurs », ces systèmes 'achoppent à des problèmes ardus d'optimisation (reconnaître que plusieurs procédures sont équivalentes). Actuellement plusieurs systèmes d'assistance à la programmation existent (environnements de génie logiciel). Ils sont liés souvent à une méthodologie générale de résolution de Problème (SMX-Cogitor).

- **Résolution de problèmes:**
  Il s'agit principalement de problèmes à base de combinatoire (parcours, jeux,...). Le travail consiste à modérer au maximum l''explosion combinatoire ». C'est dans ce domaine que l'on trouve un des premiers programmes, le plus « fameux » de l'IA, le General Problem Solver (GPS) de Newell et Simon [New 72]. Le GPS résout onze types de problèmes: problèmes d'échec, intégration symbolique, puzzle,... Il n'est pas aussi efficace que les programmes spécialisés mais avait été conçu pour être « une série de leçons donnant une meilleure vue de la nature de l'activité de résolution de problèmes ainsi que des mécanismes à mettre en jeu en vue de leur accomplissement ».
  La structure du GPS est prototype des systèmes basés sur des recherches combinatoires. On donne des DONNEES, un BUT et des opérateurs. Le système applique les opérateurs aux données afin de réduire la différence entre données et but. ALICE (Laurière, 1978) est un système essentiel de résolution de problèmes. Une certaine heuristique est introduite pour tenter de réduire la combinatoire. Actuellement une perspective des plus prometteuses est la résolution de problèmes assistée par ordinateur.

- **Perception:**
  Ici, il s'agit d'identifier des objets complexes. Ce qui ne peut se faire que par un jeu d'hypothèses et de tests. Regarder n'est pas voir, écouter n'est pas entendre. Les systèmes qui regardent et écoutent doivent faire des hypothèses sur ce qui est vu ou entendu et procéder à des vérifications. Cela suppose également un grand nombre de connaissances à disposition du système. La technique des réseaux neuronaux s'impose de plus en plus dans ce domaine.

- **Enseignement assisté par ordinateur :**
  Des techniques d'intelligence artificielle peuvent être introduites dans des tutoriaux. Les tutoriaux intelligents sont constitués de plusieurs systèmes experts travaillant en collaboration.

3.Les disciplines de l'intelligence artificielle

L'intelligence artificielle n'a pas d'objet de recherche académique bien défini à part l'intérêt porté au "mental" et aux représentations de connaissances. Elle s'est divisée en de nombreuses sous-disciplines focalisant sur des problèmes bien distincts (tel que la vision, la résolution de problèmes, la compréhension du langage, l'apprentissage,…). Il n'existe pas de paradigme unifié de recherche et certaines branches de l'IA sont devenues des terrains d'échanges multidisciplinaires où se côtoient philosophes, psychologues, informaticiens et autres qui s'intéressent aux diverses problématiques de l'intelligence. Ainsi les branches fondamentales de

l'IA font partie des sciences cognitives. Forcément, il existe une multitude de façons de concevoir recherche fondamentale et recherche appliquée. Nous en distinguons cinq :

3.1 L'intelligence artificielle "pure". Cette école que l'on peut localiser par exemple dans certains groupes de recherche au MIT vise à découvrir ce qu'est l'intelligence au plan très général. La modélisation de l'esprit humain n'est pas au premier plan. Certains s'intéressent plutôt aux mécanismes qui constituent l'intelligence chez de simples agents autonomes 288 7. Epistémologie et méthodes de la modélisation IA [Brooks], d'autres cherchent à constituer des programmes capables de résoudre certaines tâches difficiles et spécialisées comme gagner aux échecs.

3.2 L'intelligence artificielle "dure". Une tradition que l'on retrouve par exemple autour de Newell et Simon. Il s'agit de modéliser l'esprit humain en construisant un programme qui représente, à un certain niveau d'abstraction, les mécanismes humains. C'est ici que se retrouvent les fameuses thèses controversées comme "cognition = computation" et donc "esprit = programme" que nous allons discuter en détail plus bas.

3.3 L'intelligence artificielle "molle" (weak AI). Le but est similaire à celui de l'IA "dure", il s'agit de modéliser l'esprit humain. Toutefois, le programme n'a qu'un rôle de modèle heuristique. L'ordinateur et son programme ne sont pas mis en relation directe avec le cerveau et l'esprit. Il existe des variantes inductives et déductives de cette approche. La première cherche à construire l'objet "compréhension de l'intelligence" en construisant des programmes capables de résoudre certaines tâches très précises et ensuite de généraliser et tester une solution empiriquement trouvée. La deuxième est plus poppérienne : programmes et modèles sont destinés à tester certaines hypothèses.

3.4 L'intelligence artificielle "pratique". Il s'agit surtout (mais pas uniquement) des systèmes expert, sans doute la branche de l'I.A. la plus connue du grand public. Un système expert doit automatiser certaines tâches intellectuelles humaines ou encore assister un décideur humain ("decision support" ou "advisory systems"). Les tuteurs intelligents ("intelligent tutoring systems") sont une autre branche pratique comme certaines branches de la robotique. Dans toutes ces disciplines, "pratique" ne signifie pas absence totale de réflexion épistémologique, beaucoup d'impulsions théoriques ont leur origine dans des travaux pratiques. Curieusement les limites de l'IA se sont manifestées dans les systèmes experts plus que dans d'autres secteurs.

3.5 L'intelligence artificielle "outil de modélisation". On fait référence à des chercheurs dans d'autres disciplines faisant usage de certains langages de représentation et d'inférence pour décrire certains phénomènes, pour constituer des modèles ou pour formaliser des théories.

Nécessairement, les frontières entre ces orientations fondamentales de l'IA ne sont pas hermétiques. Toute tentative d'automatiser la résolution d'une tâche intelligente doit amener à se demander ce qu'est l'intelligence ou le "mental" et si la résolution artificielle se fait vraiment de façon intelligente. Réciproquement, même l'intelligence artificielle la plus spéculative doit se poser la question de l'opérationnalisation de ses concepts.

4.Intelligence artificielle et résolution de problèmes

4.1. Le General Problem Solver

Le "General Problem Solver" (GPS) de Newell et Simon est le système de résolution de problème le plus important dans l'histoire de l'intelligence artificielle (IA). Grâce à son mécanisme central - l'analyse moyens-fins ("means-ends analysis") - et grâce à son modèle de la mémoire centrale, le courant de recherche initié à la Carnegie-Mellon University a eu une grande influence sur l'évolution de l'IA et des sciences cognitives. Le "General Problem Solver" (GPS) a été le premier modèle complet du traitement humain de l'information. Quelque peu démodé aujourd'hui, il continue néanmoins à travers son successeur (le modèle SOAR) d'influencer d'une façon significative les modèles du traitement symbolique de l'information.

4.2.SOAR

Dans le livre "Universal Subgoaling and Chunking" de Laird, Rosenbloom et Newell (1986), l'architecture Soar est introduite. Venant de la tradition de Carnegie-Mellon et donc de GPS, il s'agit de nouveau d'un système de résolution de problème universel centré sur la recherche heuristique des espaces de problèmes. (Laird et al. 86: 286). Partant d'un état initial, le système tente d'appliquer un certain nombre d'opérateurs pour arriver à l'état désiré.

Lorsque le programme rencontre une difficulté, il crée un sous-but et le traite comme problème indépendant à résoudre. Il définit donc des hiérarchies de buts ayant des espaces de problèmes indépendents. Contrairement à GPS, le système choisit des méthodes de résolution différentes selon la nature du problème. Laird et al. Proposent dix sept méthodes de recherches heuristiques parmi lesquelles figure GPS. SOAR est bâti sur deux concepts importants: (1) la méthode faible universelle (angl. "universal weak method") qui intègre toutes les méthodes universelles connues et (2) le "sub-goaling" universel qui structure la résolution d'un problème difficile en tâches gérables. En plus, Soar possède une capacité d'apprentissage. Il peut se souvenir des séquences de résolution de problème ayant eu du succès et les réutiliser dans les nouvelles situations.

5.Les générations des Systèmes experts

5.1. La première génération

Les SE de cette génération étaient des applications ad-hoc et non réutilisable.

Chaque système possède son propre formalisme de représentation de connaissance, par conséquent chaque application a un moteur d'inférence rarement utilisable par d'autres applications.

L'ajout d'une nouvelle connaissance peut très bien avoir pour conséquence que le système ne soit plus capable de résoudre des problèmes qu'il savait résoudre auparavant.

- Quelques systèmes experts classiques :
  - ✓ **DENDRAL**, 1969, chimie, recherche la formule développée d'un corps organique à partir de la formule brute et du spectrogramme de masse du corps considère.
  - ✓ **CRYSALIS**, 1979, chimie, recherche la structure de protéines a partir de résultats d'analyse cristallographique.
  - ✓ **MOLGEN**, 1977, biologie, engendre un plan de manipulations génétiques en vue de construire une entité biologique donnée.
  - ✓ **PROSPECTOR**, 1978, géologie, aide le géologue à évaluer l'intérêt d'un site (1600 règles).
  - ✓ **ADM**, médecine, aide au diagnostique médical, qui donne des propositions diagnostiques devant un tableau sémiologique.

    Il permet de vérifier des hypothèses diagnostiques et de préciser une pathologie iatrogène devant un tableau associant signes cliniques et para cliniques.
  - ✓ **AM**, 1977, Mathématique, proposition de conjecture, de concepts intéressants (500 règles).
  - ✓ **MYCIN**, 1974, médecine, système d'aide au diagnostique et au traitement de maladies bactériennes du sang.

5.2. La deuxième génération :

L'introduction par Newel du ''Knowledge Level'' est a l'origine d'une évolution intéressante en matière de conception d'un système a base de connaissances : toute démarche va maintenant viser à séparer la spécification des connaissances nécessaires au traitement d'un problème de leur implémentation opérationnelle.

L'indépendance entre la base de connaissance et le moteur d'inférence est un élément essentiel des systèmes de cette génération. Cette indépendance nous permet d'évoluer les connaissances d'un système sans avoir a agir sur le mécanisme de raisonnement. Cette deuxième génération est caractérisée aussi par l'apparition des systèmes généraux (générateurs de systèmes experts).

Un générateur de système expert est un outil de développement de SE particulier, les systèmes généraux ne sont pas plus spécialises et peuvent contenir plusieurs bases de connaissances, on les appelle les systèmes vides ou les coquilles (**SHELL**), donc il est nécessaire d'avoir un formalisme de représentation des connaissances pour que le système puisse comprendre l'expertise et un mécanisme de raisonnement exploitant cette expertise.

6.Exemples de systèmes experts généraux

6.1.SNARK (Symbolique Normalised And Représentation of Knowledge)

Il a été conçu par J.L.Laurier en 1981 pour la mise en point de plusieurs systèmes experts dans des domaines de connaissances variées. SNARK (Symbolic Normalised And Représentation of Knowledge): a été conçu par J.L.Lauriereenl98L. Snark a été utilise pour la mise au point de plusieurs systèmes experts, dans des domaines de connaissances variées ; Il se compose d''un langage déclaratif de représentation, et d'un moteur d'inférence qui fonctionne en chaînage avant. Il est Caractérisé par les Les connaissances forme de règles : SI <Expg> ALORS <Expd> dans la logique du premier ordre, l'ensemble des faits est composée d'objets repérés par l'ensemble des propriétés qui s'y rattachent, Les relations utilisées, aussi bien dans la base de règles que dans la base de faits, sont exclusivement binaires.

Les règles peuvent être regroupées en paquets statiques, qui seront examinées successivement par le moteur, le raisonnement formalisé dans les règles peut être monotone ou non monotone ; il est irréversible.

SNARK permet d'utiliser des coefficients de vraisemblance dans l'écriture des règles et des faits, pour exprimer l'incertitude inhérente aux règles de connaissances exprimées par l'expert.

6.2. ART

Est apparu sur le marché en 1984, lancé par la jeune firme Californienne Inférence Corporation. C'est un outil très puissant qui combine des méthodologies d'IA sophistiquées, comme le maintien de vérité et une grande rapidité de fonctionnement, il est caractérisé par Une représentation de connaissances basée sur les schémas, et il est capable de travailler sur des propositions auxquelles sont attachées des valeurs de vérités déterminant si la proposition est vraie, fausse ou inconnue, ainsi que sur des propositions instanciables ou patterns. Son moteur d'inférence fonctionne en chaînage avant ou arrière, il permet de représenter l'incertitude par le biais de valeurs de confiance attachées aux faits eux mêmes.

ART est un outil de développement de systèmes experts puissant et bien intégré.

Les principaux atouts sont la complexité des modelés d'IA utilisés, ainsi que la vitesse apportée par son compilateur de connaissances.

6.3.MORSE

Développé par l'équipe de l'IA et système inférenciel de l'université d'ORSAY.

Il se caractérise par une base de faits formée du couple (Attribut, Valeur), les valeurs associées aux faits sont : numériques, booléens, et chaînes de caractères, les attributs sont mono values c'est à dire qu'il ne possèdent qu'une seule valeur au cours d'une session, le langage d'expression de connaissances est les règles de production, la stratégie de contrôle est le chaînage avant, Le système est ouvert, la partie conclusion possède des procédures gérant l'extérieur.

6.4. KEE (Knowledge Engineering Environnement) :

Il est apparu sur le marché en 1983. L'un des meilleurs atouts de KEE est son interface utilisateur sophistiquée et conviviale, elle inclut un éditeur graphique d'objet, un utilitaire graphique d'exploitation de raisonnement, et plusieurs fenêtres qui montrent en permanence ce que KEE est en train de faire et ce que l'utilisateur désire voir.

7.Méthodologie de développement des systèmes experts :

7.1. KOD

Méthode de recueil de la connaissance pour encadrer la spécification de systèmes experts, développé par l'anthropologue Claude Vogel (ethnologue de formation « converti » à l'IA et à la linguistique).

Le projet KOD marque la volonté d'unifier les méthodes de développement en informatique et en IA et de définir ainsi un génie cognitif pour les SBC comme il existe un génie logiciel en informatique classique.

7.2. KADS (Knowledge Acquisition and Desing Support 1985)

Jusqu'à sa dernière version CommonKADS (1992), est une méthodologie dont l'objectif est d'aider dans la modélisation des connaissances expertes, dans le but de réaliser des systèmes basés sur la connaissance.

7.2.1 Cycle de vie (KADS)

KADS représente à l'heure actuelle la réponse la plus avancée en matière de méthodologie de développement de SBC. Synthèse et amélioration de nombreuses techniques, elle permet de traiter tout le processus d'acquisition des connaissances, du recueil au développement d'un système complet. C'est une méthode dirigée par les modèles (par opposition aux methods dirigées par l'implémentation). Les modèles de KADS étant des représentations intermediaries partielles et orientées de l'expertise avant leur implantation dans un système ([De Greef 85]).

KADS propose un cycle de vie basé sur les méthodes de développement de logiciels.

Une analyse complète des données précède la conception et l'implantation du SE, ce qui diffère donc du prototypage rapide ou des approches incrémentales. Les résultats de la phase d'analyse (contraintes externes, Modèle Conceptuel des objets du domaine et des méthodes de résolution) servent d'entrée à la phase de conception où ils sont transformés en spécification de l'architecture fonctionnelle du SBC.

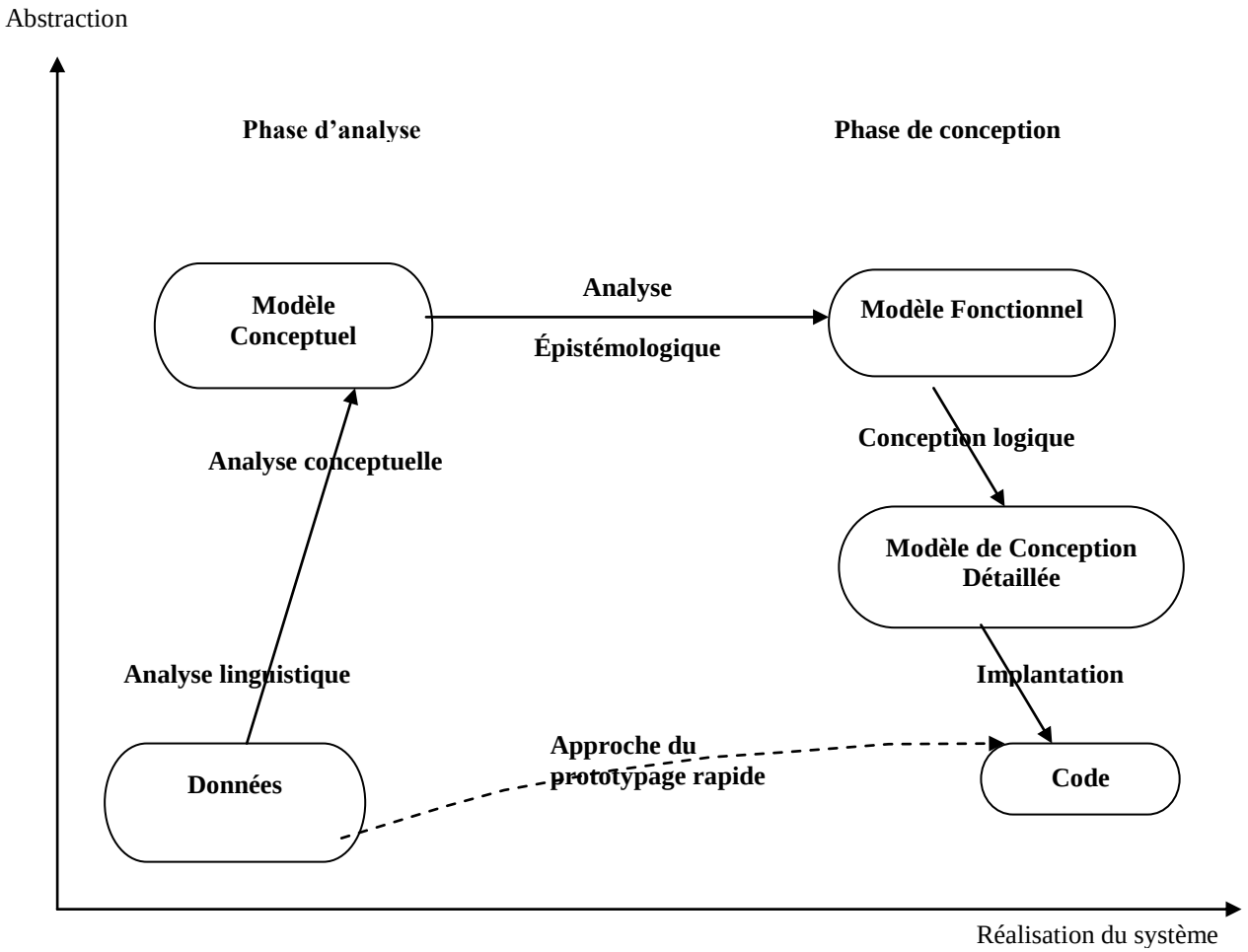


Figure 01: Modèles et espaces de développement de KADS (Extrait de [Schreiber 88])

7.2.2 Les modèles-étapes de KADS

Les différents modèles proposés par KADS permettent non seulement la "capture" et la représentation en machine d'un savoir-faire, mais aussi la compréhension du problème réel, de l'organisation et de ses besoins.

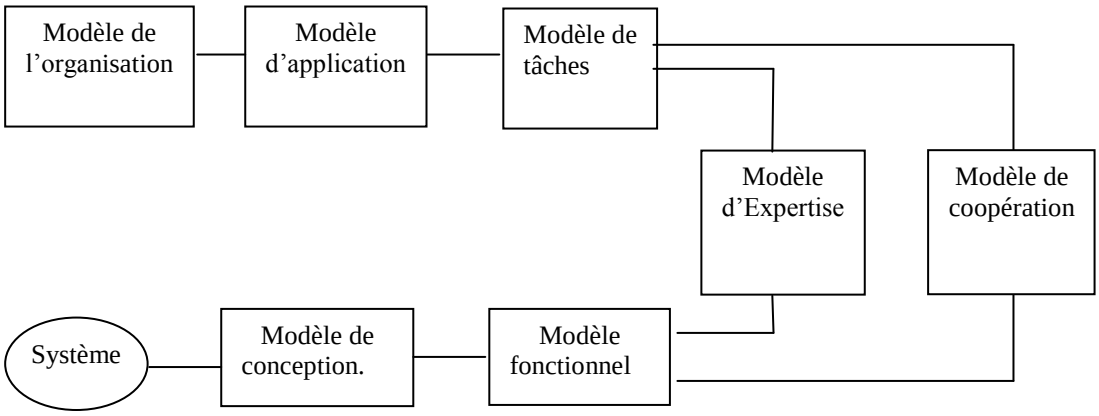


Figure 02: Modèles_Etapes de Kads

- ***Le modèle de l'organisation*** décrit l'environnement du SBC (avant et après son introduction pour apprécier son impact) ainsi que les interactions (au niveau humain ou social) avec les utilisateurs. Doivent donc être spécifiés (entre autres) : l'objectif du manager, le rôle du ou des experts, les types d'utilisateurs, leurs besoins et utilisations du SBC. Ceci est important pour permettre de bien cibler l'utilité et la fonction de l'outil, d'assurer son adéquation aux besoins des utilisateurs, de s'assurer de sa faisabilité.

- ***Le modèle d'application*** définit quel est le problème à résoudre, l'utilité d'un SBC, sa fonction et ses contraintes techniques (langage, système d'exploitation, performance, etc).

- ***Le modèle de tâches*** spécifie comment la fonction est décomposée par l'expert en tâches et sous-tâches, indique leurs entrées-sorties et les agents internes ou externes (l'utilisateur) qui les exécutera.

- ***Le modèle de coopération*** recense les tâches de transferts (e.g. «obtenir-de», «fournir-à» (l'utilisateur)) entre le système et son environnement (ex: l'utilisateur) et précise ainsi les rôles respectifs (qui fait quoi).

- ***Le Modèle d'Expertise*** utilise le modèle de tâches et décrit le comportement et le type de connaissances nécessaires au système pour la résolution du problème, indépendamment de toutes contraintes d'implantation. Il s'agit donc d'un modèle de la connaissance de l'expert, subjectif car il s'agit d'une vision particulière du monde réel, mais ce n'est pas un modèle cognitif de l'expert car il est orienté par ce que peut et doit faire le futur SBC. Ce modèle joue donc un peu le rôle de spécifications fonctionnelles dans les développements classiques: c'est "un cadre sémantique partagé par des utilisateurs d'un programme et ses développeurs qui leur permet de communiquer" [Auss 89] ou encore "le modèle qu'a en tête le concepteur quand il va implanter le SBC" [Brunet 91].

  - ***Le Modèle Conceptuel*** est la réunion des deux derniers modèles et décrit donc de façon abstraite les objets, comportements et opérations nécessaires au futur SBC. Par extension, il désigne toutes les activités menant à ce modèle (analyse linguistique puis conceptuelle).
    Certains générateurs de S.E. interprète partiellement le Modèle Conceptuel pour certains environnements ou type de tâches : Model-K [Karbach 91].

- ***Une phase de conception*** plus "classique" et plus courte que celle de l'analyse (ex: 1 à 2 mois pour un prototype ayant demandé 6 mois d'analyse [Brunet 91]) comprenant :
  - le modèle fonctionnel qui correspond à l'architecture fonctionnelle du futur SBC et s'obtient par une analyse épistémologique du Modèle Conceptuel; là peuvent intervenir les termes du formalisme de représentation des connaissances du langage d'implantation;
  - le modèle de conception détaillée, après une analyse logique effectuée sur le modèle fonctionnel.

7.3 CommonKADS

Créer en 1992, est la dernière version de la méthodologie KADS. Cette méthodologie propose deux phases de développement, la première étant la phase de spécification, dans laquelle se trouve cinq modèles conceptuels, et la deuxième, la phase de conception (modèle de conception). Les cinq modèles conceptuels sont les suivantes :

- Le modèle d'organisation : permet de représenter le processus, la structure ainsi que les ressources de l'organisation susceptible d'accueillir le système à base de connaissances.

- Identifier les tâches globales de l'organisation qui seront plus efficaces, automatisées, plutôt que réalisées avec l'intervention d'un agent (homme/programme) est l'un des objectifs du modèle des tâches.
- Le modèle d'agent permet de modéliser les capacités et les contraintes de l'agent (hommes/machines/programme) pour mener à bien la tâche à accomplir.
- Le modèle de communication permet de modéliser le système de communication dans lequel le système à base de connaissances doit évoluer.
- Le modèle conceptuel d'expertise, il permet de représenter le processus de résolution de problèmes d'un agent (hommes/ machines/programme).

7.4 La Méthodologie MKSM

MKSM (Methodology for Knowiedge System Management) est une méthode récente mise au point pour le CEA (Commissariat à l'Energie Atomique) en 1993 par E. Brunet et J.L. Ermine. Il s'agit de traiter les problèmes de capitalisation dés connaissances, précisément dans une optique de gestion des connaissances d'une organisation . Cette méthode a été utilisée par des entreprises telles que La Poste, Rhône-Poulenc, Cofmoga ou encore, Thomson. La société française KADE-TECK diffuse cette méthodologie sous une forme commerciale.

La m éthodologie continue d'être améliorée à l'Université d e T royes, s ous 1 a direction de J.L. Ermine, et porte désormais le nom de MASK (Mèthod for Analysing and Structuring Knowiedge).

MKSM propose à la fois des modèles permettant l'analyse et la modélisation des connaissances d'une organisation, et un cycle de développement de projet [Erm 96].

7.4.1 Les Modèles MKSM

Les modèles de connaissances proposés par la méthode MKSM se basent sur des fondements théoriques importants [Erm 96], découlant de deux hypothèses principales :

- L'hypothèse *systémique :* le patrimoine de connaissances d'une organisation forme un système complexe à part entière. Ce système de connaissances est, au sens de la théorie du système général, « un objet actif et stable et évoluant dans un environnement, et par rapport à quelque(s) finalité(s) ». Ce système peut donc être étudié selon trois points de vue : sa structure, sa fonction et son évolution ;
- L'hypothèse *sémiotique :* toute connaissance peut être observée selon trois niveaux indissociables : le niveau syntaxique (données, représentation de l'information), le niveau sémantique (signification de l'information) et le niveau pragmatique (rapport au contexte, à l'environnement dans lequel cette informations prend un sens). Cette vision peut être résumée par la formule : « un système de connaissances est vu comme de l'information qui prend une signification dans un contexte donné ».

7.4.2 7.4.2 Le Cycle de Vie d'un projet MKSM

La méthodologie MKSM se concentre sur l'analyse et la modélisation des systèmes de connaissances, laissant de côté les problèmes de réalisation et d'implantation. La conduite de projet MKSM comporte trois phases : le *cadrage,* le *cycle de modélisation* et le *schéma d'orientation.*

L'objet de la phase de cadrage est de déterminer le domaine d'application (assurance qualité, documentation, organisation, etc.), le niveau d'approche (stratégique, opérationnel...) ou le sujet d'étude du projet [Tix 96]. Cette phase permet ensuite de définir les modèles qui seront utiles au projet, afin de réduire au mieux l'effort de modélisation.

La phase de cadrage permet également d'identifier et mettre en place les acteurs du projet : le comité de pilotage qui suit et oriente, le comité de projet qui réalise, et le comité technique qui conseille.

Le cycle de modélisation consiste à modéliser la connaissance au travers des différents types de modèles qui ont été présentés. La modélisation s'effectue par l'interview d'experts ou l'analyse de documentation (avec la présence de personnes du domaine). Cette phase s'achève par la constitution d'un *livre de connaissances* regroupant l'ensemble des modèles produits. Ce livre fournit un point de départ indispensable à tout projet opérationnel de traitement des connaissances.

Le schéma d'orientation doit permettre d'articuler la modélisation MKSM avec la partie opérationnelle du projet [Tix 96]. Le schéma d'orientation possède trois niveaux distincts :

- un niveau *stratégique* permettant de définir des objectifs de la gestion du patrimoine de connaissances considéré pour chacun d es quatre orientations possibles; 1es Ressources Humaines, le Management, la Production et la Recherche et Développement ;
- un niveau *tactique* établissant les actions à mener (en spécifiant notamment leurs objectifs spécifiques, les sources de connaissances, les résultats attendus, leurs conséquences, leur environnement) afin d'atteindre les objectifs ;
- un niveau *d'étude de risques* des différentes actions à mener classant les risques selon six axes : les enjeux externes, les enjeux internes, la facilité d'accès aux connaissances, la criticité des connaissances, l'intérêt des résultats possibles et l'intégration à l'environnement.

Le schéma d'orientation ne se construit pas à un moment précis du cycle de vie d'un projet, mais au moment où l'on estime avoir une vision suffisamment précise du système de connaissances.

7.5 La Méthodologie REX

REX (acronyme de Retour d'EXpérience) est une méthodologie dédiée à la capitalisation de l'expérience acquise durant la réalisation des activités d'une organisation [Mal 94]. A l'origine, cette méthodologie a été mise au point par le CEA dans le but de sauvegarder les connaissances acquises lors de là conception de réacteurs nucléaires. Elle a ensuite été utilisée dans d'autres domaines tels que l'aéronautique ou la conception de générateurs électriques.

Son application se fonde sur trois étapes :

- l'analyse des besoins et identification des sources de connaissances de l'organisation ; durant cette étape, il s'agit de spécifier et de dimensionner le système de gestion des connaissances qui sera mis en place; elle doit permettre d'identifier les spécialistes du domaine et d'estimer les *éléments de connaissances* susceptibles d'être produits ;
- la construction d'éléments de connaissances et la modélisation du domaine par un travail d'analyse documentaire et de recueil d'expérience (via des entretiens) ;
- la mise en place puis l'exploitation du système de gestion de connaissances créé.

7.5.1 Les Eléments de Connaissances

Le principe de base de la méthodologie REX consiste à composer des éléments de connaissances (EC) relatifs à une activité et qui, une fois constitués, pourront être valorisés par les utilisateurs [Tix 96]. L'ensemble de ces éléments d'expérience sont stockées dans ce qui est appelée une mémoire d'expérience. Les EC sont construits principalement à l'issue d'interviews de spécialistes de l'organisation, de l'analyse des documents et de l'interrogation des bases de données existantes. Un EC est typiquement constitué des éléments suivants : une *entête*, une *description* ou *corps*, une *liste de références.*

7.5.2 La mémoire d'expérience

Les éléments d'expérience constitués sont organisés de façon à être facilement réutilisables. Ils forment le cœur de la mémoire d'expérience. Celle-ci comporte en outre un réseau terminologique un modèle descriptif permettant de présenter le vocabulaire et les points de vue utilisés dans l'entreprise.

Le modèle descriptif permet de représenter les différents points de vue identifiés dans l'entreprise (par exemple le point de vue géographique). Chaque point de vue est représenté par un réseau sémantique constitué d'objets (concepts) liés entre eux par des liens de types : ensemble/élément, général/spécifique, proximité, évolution [Tix 96].

Le réseau terminologique quant à lui décrit le vocabulaire du domaine considéré. Il est également représenté par un réseau sémantique constitué d'objets reliés entre eux par des relations *sorte-de* et *concerne.*

Son objectif principal est de permettre des requêtes proches de la langue naturelle. Les EC sont rattachés à un ensemble d'objets définis par des points de vue. La représentation textuelle des EC peut être indexée par des termes identifiés par le réseau terminologique.

7.5.3 Le système de gestion des connaissances

La méthodologie REX propose un outil informatique permettant à la fois de mémoriser et d'accéder aux éléments d'expérience. L'interface permet de consulter la mémoire d'expérience au moyen de requêtes en langage naturel.

Le système retourne ensuite un ensemble d'objets (définis dans le réseau terminologique) et les EC qui leurs sont rattachées. REX est donc une méthodologie permettant l'analyse, la capitalisation, et l'opérationalisation des connaissances relatives à l'expérience.

7.6 Comparaison des méthodologies

Dans cette section on a présenté quelques méthodologies, mais d'autres également seraient intéressantes : citons notamment la méthodologie. MOKA (Methodology and tools Oriented to Knowledge-based engineering Application) qui offre un atelier complet (démarche, modèles de représentation et outils) [Cal 99] dédié au développement de Systèmes Experts et la méthodologie MCSC (Méthode de Conception de Systèmes d'information Coopératifs) spécialisée [Boug 01] dans les problématiques organisationnelles liées au travail coopératif.

Au travers de cette première synthèse, des critères permettant de caractériser les méthodologies de capitalisation de connaissances apparaissent, notamment les suivants :

- le type des connaissances ou des problèmes qu'elles proposent de traiter (connaissances experts, organisationnelles, etc.) ;
- leur niveau de formalisation des connaissances : selon que l'objectif d'exploitation des connaissances modélisées soit une réelle opérationalisation ou bien une simple consultation ;
- leur niveau d'outillage : certaines méthodologies proposent des environnements complets, comprenant des outils facilitant les phases de mise en oeuvre de la méthodologie depuis la capitalisation [Tix 96] jusqu'à l'opérationalisation des connaissances; d'autres se contentent de définir une démarche et un formalisme de modélisation ;
- leur complexité de mise en œuvre : le coût du travail à effectuer dans un projet de gestion des connaissances dépend directement de la méthodologie adaptée et de l'outillage disponible.

Le tableau suivant explicite la valeur de ces différents critères pour MKSM, REX et CommonKADS.

Méthodologie	Connaissances, problèmes traités	Niveau de formalisation	Niveau d'outillage	Complexité de mise en œuvre
MKSM	Objectif de création d'une mémoire d'entreprise : système organisationnel, activité, savoir et savoir-faire des experts.	Moyen, les connaissances modélisées (sous forme de livres de connaissances), mais les modèles utilisés ne permettent pas une opérationnalisation immédiate.	Faible, seuls des outils « classiques de bureautique sont utilisés pour la capitalisation.	Moyenne, nécessite la collaboration de nombreux acteurs de l'entreprise, tâche de modélisation longue mais souple (formalisation peu importante).
REX	Objectif de gestion du retour d'expérience (savoir et savoir-faire acquis au cours de projets).	Faible, seulement une ontologie du domaine et une simple structuration des éléments d'expérience sous forme de fiches.	Important, un outil permet la capitalisation et la construction des éléments d'expériences.	Moyenne, l'effort de capitalisation est limité car peu de formalisation, une solution opérationnelle est intégrée.
CommonKADS	Objectif de création de SBC : surtout connaissances expertes mais aussi sur l'organisation et sur le futur SBC.	Fort, les modèles KADS permettent de traiter les problèmes de représentation et de mise en œuvre des connaissances : un SBC peut découler directement de la modélisation.	Important», différents outils dédiés à la mise en œuvre de la méthodologie existent : Open-KADS, KADS-Tools.	Importante, complexité des modèles de représentation (formalisation importante), difficultés de mise en place d'une solution opérationnelle.

7.7 Ontologie

Il est difficile de définir ce qu'est une ontologie d'une façon définitive. Le mot est en effet employé dans des contextes très différents touchant la philosophie, la linguistique ou l'intelligence artificielle. Guarino [guarino95a], [guarino97a] part de sept interprétations possibles pour chercher à clarifier ce qu'est une ontologie. Outre le sens philosophique originel, une ontologie désigne en effet une modélisation conceptuelle, ou une représentation de cette modélisation. Dans les deux cas,

on parle d'ontologie formelle pour désigner aussi bien la rigueur de la modélisation que la structure de sa représentation. Cela nous amène à la notion de convention ontologique, qui est une modélisation conceptuelle du sens du formalisme de représentation. Enfin, on distingue différents types d'ontologies selon le domaine modélisé.

Les ontologies jouent un rôle central dans les systèmes à bases de connaissances. Les différentes méthodologies présentées ci-dessus possèdent toutes une partie dédiée à la représentation des connaissances ontologiques.

7.7.1 Définitions

Le terme ontologie est issu du domaine de la philosophie où il signifie « explication systématique de l'existence ».
Dans le cadre de l'ingénierie des connaissances, différentes définitions ont été établies.
Une définition générale et souvent reprise a été formulée par Gruber [Gru 93], une ontologie est une spécification explicite d'une conceptualisation, une conceptualisation étant définie comme étant une structure sémantique intensionnelle qui capture les règles implicites contraignant la structure d'un morceau de réalité.

D'autres auteurs apportent des définitions fondées sur l'approche qu'ils ont adoptée pour construire leurs ontologies. Il peut s'agir d'une approche linguistique pour Swartout [Swa 97], une ontologie est un ensemble de termes structuré de façon hiérarchique, conçu afin de décrire un domaine et qui peut servir de charpente à une base de connaissances.
Il peut également s'agir d'une ***approche formelle*** : Guarino et Giaretta [guarino95] ont défini l'ontologie comme étant une théorie logique qui rend compte partiellement mais explicitement d'une conceptualisation, adopter des principes de représentation permet également de s'appuyer sur une organisation précise des connaissances.

Une ontologie formelle est constituée d'une collection de noms pour les types de concepts et de relations. Ils sont organisés dans un ordre partiel par la relation type/sous-type [swa97], par opposition une ontologie informelle constituée d'un catalogue de types qui sont soit non définis, soit définis par des assertions en langage naturel. Guarino parle alors de *théorie ontologique* pour désigner une théorie logique cherchant à exprimer des connaissances ontologiques [guarino95].

Nous retiendrons que les ontologies donnent le cadre dans lequel exprimer les connaissances. Elles définissent des choix et des contraintes qu'il faut respecter pour que des affirmations aient une signification dans un domaine ou une application [Bac 01]. En effet, une ontologie définit un ensemble organisé (souvent sous forme de taxonomie) de concepts utilisables pour formuler des connaissances. Un concept représente un constituant de la pensée, né de l'esprit (un principe, une notion abstraite) pour lequel est explicitée une sémantique évaluable et communicable [Dieng 0 1].

Une ontologie renvoie donc aux connaissances statiques d'un domaine contrairement aux méthodes de résolution de problèmes qui reflètent des connaissances dynamiques (ou stratégiques) pour le raisonnement.

7.7.2 Le rôle des ontologies

Les ontologies rendent différents services, tels que la *définition d'un vocabulaire* permettant de formuler des savoirs, la spécification plus ou moins formelle du sens des termes et des relations entre les termes. Dans le cadre d'une mémoire organisatiomielle, les ontologies constituent en elles-mêmes une connaissance intéressante pour les utilisateurs, en établissant notamment des *référentiels terminologiques.* De plus, elles ont un rôle essentiel de mutualisation et de fédération des connaissances. En outre, elles permettent le multipoint de vue sur des connaissances, c'est à dire la représentation ou la traduction de connaissances avec des termes propres à un utilisateur ou à un service.

Différentes ontologies d'organisations ont été développées, avec l'objectif de pouvoir être utilisées et intégrées à de nouvelles ontologies. Citons par exemple l'ontologie du projet TOVE

(Toronto Virtual Entreprise) ou bien l'ontologie PME [Kas 00]. Ces ontologies sont généralement dédiées à un domaine particulier.

7.7.3 Différents niveaux d'ontologies :

On distingue différents niveaux d'ontologies selon le domaine modélisé et éventuellement les tâches pour lesquelles elles sont conçues [chand 99], [burgun01].
Les ontologies d'application ont un domaine de validité restreint et correspondent l'exécution d'une tâche.

Les ontologies de domaine ont un faisceau plus large, une bonne précision et ne sont pas propres à une tâche particulière. Les ontologies générales ne sont pas propres à un domaine. Leur précision est moyenne. Les ontologies supérieures (upper level ontologies) représentent des concepts généraux.

7.8 Discussion

La gestion des connaissances peut être abordée de divers points de vue. Il est possible de faire une étude précise des besoins, au travers d'une analyse du système de connaissance ou bien au contraire de mettre directement en place des solutions génériques qui seront adaptées aux besoins par la suite.

Elle peut faire l'objet d'une étape de capitalisation des connaissances, c'est à dire tout d'abord d'une identification des connaissances à gérer, puis une explicitation et une modélisation (plus ou moins formelle) et enfin la constitution d'une mémoire collective (en structurant les connaissances modélisées).

En termes de solutions opérationnelles de gestion des connaissances, on peut différencier les solutions orientées information, des solutions orientées connaissances [Cho 99].

8.Apprentissage Symbolique

L'apprentissage est un terme général qui décrit le processus selon lequel l'être humain ou la machine peut accroître sa connaissance. Apprendre c'est donc raisonner sur son expérience : découvrir des analogies et des similarités, généraliser ou particulariser une expérience, tirer parti de ses échecs et erreurs passés pour des raisonnements ultérieurs (avant l'exploitation et tout au long du cycle de vie).

Les premiers essais en A.S.A. (d'Apprentissage Symbolique Automatique) dans les années 50 : ils consistaient en une tentative de modélisation incrémentale, où la connaissance est quasiment nulle au départ et s'accroît progressivement; le plus célèbre de ces modèles est le perceptron de Rosenblatt. Après lui, Samuel a créé un programme de jeu de dames, qui a posé le problème de la distinction entre ce qui est vraiment appris par le système et ce qui est communiqué au système par le programmeur plus ou moins implicitement; la différence s'est souvent avérée difficile à mettre en évidence, et l'A.S.A. rencontrait là sa première problématique.

Dans les années soixante, une autre approche, proposée par Buchanan et Winston, conduisait l'A.S.A. à l'acquisition de concepts et de connaissances structurées : le système DENDRAL, qui engendre des règles d'explication à partir de données de spectrographie de masse en chimie organique, en est la plus spectaculaire réussite.

En dehors de l'Intelligence Artificielle, il existe de nombreuses approches à l'acquisition automatique de connaissances : Systèmes Adaptatifs de l'Automatique, Inférence Grammaticale, Inférence Inductive fortement liée à l'informatique théorique, et les multiples méthodes numériques dont le connexionisme est la dernière incarnation.

L'apprentissage automatique propose la mise en œuvre des techniques inductives, déductives, inventives, abductives ou par analogie [Kod 96] permettant de doter les systèmes à bases de connaissances de capacités d'apprentissage.

L'apprentissage par induction reçoit un ensemble d'exemples d'apprentissage et doit produire des règles de classification, permettant de classer les nouveaux exemples. Le processus d'apprentissage cherche à créer une représentation plus générale des exemples, selon une *méthode de généralisation de connaissances*. Ce type de méthodes est aussi appelé apprentissage de concepts ou bien acquisition de concepts.

8.1. Objectifs de l'apprentissage automatique :

L'apprentissage automatique se définit par un double objectif: *un objectif scientifique* - comprendre et mécaniser les phénomènes d'évaluation dans le temps et d'adaptativité des raisonnements, et un objectif pratique - bâtir automatiquement des bases de connaissances à partir des exemples.

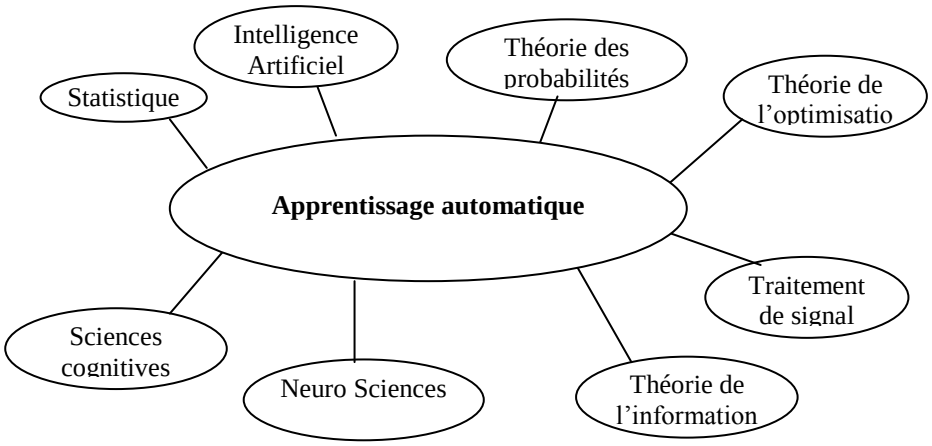


Figure 03 les Domaines d'Apprentissage automatique

L'apprentissage peut être défini par l'amélioration des compétences avec l'expérience [Hou 94]. En effet, l'apprentissage e st intimement 1ié à 1 a généralisation : apprendre c'est passer d'une succession de situations vécues à un savoir réutilisable dans des situations similaires [Kod 96]. Pour chacune des principales méthodes d'apprentissage existantes, plusieurs types de problèmes se posent :

- Problème du regroupement (qu'on appelle classification en analyse de données) : étant donnée une masse de connaissances, comment découvrir des traits communs entre elles de sorte que l'on puisse les regrouper en sous-groupes ayant une signification ?

- Problème de discrimination, qui est celui de l'apprentissage de procédures de classification : étant donné un ensemble d'exemples de concepts, comment trouver une méthode qui permette le plus efficacement de reconnaître à quel concept rattacher un percept ?

- Problème de la généralisation : comment à partir d'exemples concrets d'une situation ou d'une règle en déduire une formulation assez générale pour décrire cette situation ou cette règle, et comment expliquer la capacité de description de cette formulation.

8.2 Formes d'apprentissage

C'est depuis le « International Workshop in Machine Learning » de 1985, on distingue donc l'Apprentissage par Détection de Similarités (Similarity-Based Learning en anglais) de l'Apprentissage par Recherche d'Explications (Explanation-Based Learning en anglais).

La différence fondamentale entre les deux approches peut être exposée comme suit : en S.B.L., on apprend en détectant des similarités dans un ensemble d'exemples, et des dissimilarités entre les exemples et des contre-exemples; en revanche en E.B.L., on apprend à partir d'explications dérivées de l'analyse d'un exemple ou d'un contre-exemple du concept ou de la règle à apprendre.

En général, l'apprentissage se fait avec un système de résolution de problèmes [Kod 96]. Chaque fois que le système arrive à une solution, c'est bien sûr, soit un succès, soit un échec (on parle alors d'exemples négatifs). Un module analyse alors les raisons de ce succès ou de cet échec.

9.Conclusion

Les systèmes Expert généraux (shell) ne sont plus spécialisés et peuvent contenir plusieurs bases de connaissances.

L'indépendance entre la base de connaissance et le moteur d'inférence est un élément essentiel des systèmes généreux, cette indépendance nous permet d'évoluer les connaissances d'un système sans avoir à agir sur le mécanisme de raisonnement.

Chapitre 2

Ingénierie des connaissances

1. Introduction

L'ingénierie des connaissances constitue depuis de nombreuses années un domaine actif des recherches menées en intelligence artificielle autour de la conception et de la réalisation des systèmes à base de connaissances (SBC).

Elle consiste à concevoir des systèmes dont le fonctionnement permet d'opérationnaliser des connaissances portant sur le traitement ou la résolution d'un problème donné. La résolution de problèmes implique deux étapes essentielles : la modélisation du problème et d'une méthode de résolution dans un cadre théorique donné, l'opérationnalisation informatique du modèle obtenu.

Le génie cognitif fait référence à l'ensemble des méthodes utilisées pour les systèmes à bases de connaissances ("knowledge based systems").

De nombreux langages de description des connaissances ont été proposés pour décrire les modèles construits aux différentes étapes du cycle de développement d'un SBC. On distingue les langages de modélisation, des langages de représentation ou d'opérationnalisation des connaissances.

Les langages de modélisation offrent des primitives adéquates pour modéliser semi formellement et de façon structurée des connaissances. Ces notations, qui peuvent être graphiques ou textuelles, sont particulièrement pertinentes pour construire un modèle conceptuel d'une application au niveau connaissances. Leur principale caractéristique est d'être lisible et compréhensible par les différents intervenants humains impliqués dans la conception d'un SBC.

Les langages de représentation permettent le codage symbolique des connaissances, dans un formalisme caractérisant, par des règles ou des structures syntaxiques, les énoncés licites. L'objectif poursuivi est la réduction des ambiguïtés et des imprécisions inhérentes à la modélisation en privilégiant l'utilisation de langages formels.

2. Génie cognitif

2.1. Introduction

Le génie cognitif fait référence à l'ensemble des méthodes utilisées pour les systèmes à bases de connaissances ("knowledge based systems").

Il existe une différence importante entre l'IA du type "science cognitive" ou "pure" et 1'IA pratique ou appliquée. La deuxième doit fournir des systèmes capables de résoudre de façon optimale une certaine classe de tâches. La première modélise certains processus cognitifs ou le phénomène de 1'intelligence en soi. Le critère de succès dans ce cas est la question de savoir si le système est un bon modèle, voire une bonne théorie. Avec les IA plus fondamentales, on partage, le souci d'une certaine homomorphie structurelle ou fonctionnelle du modèle (il doit décrire quelque chose). Toutefois, selon notre opinion, on n'opère pas au même niveau de description (les processus cognitifs détailles) et on ne s'attache pas à la même fidélité structurelle du modèle. Avec 1'IA pratique, on partage le souci d'une certaine "completeness", de 1'inspectabilite, de la modularité et de la communicabilité du modèle au défaut d'une complexité plus adéquate mais ingérable. [Aus, 2000]

Pour appliquer la technique des systèmes experts, il n'est malheureusement pas suffisant de disposer de machines et d'environnements de développement performants. Le but final d'une application étant de reproduire le comportement d'un expert dans un domaine donné, il faut savoir observer, analyser et comprendre le comportement de cet expert avant de formaliser les connaissances acquises dans un langage accessible à la machine (ce processus d'acquisition et d'implémentation du savoir de 1'expert est appelé "transfert des connaissances").

Ce processus est très délicat, du fait de problèmes humains mais aussi du fait de la nature même de 1'expertise. Les sciences cognitives peuvent aider à comprendre la nature de ces difficultés et même faciliter dans une certaine mesure la réalisation de ce transfert des connaissances (on parle alors de "Génie cognitif").

2.2. Définitions

- Le génie cognitif (aussi appelé ingénierie de la connaissance, "knowledge engineering") fait référence à 1'ensemble des méthodes utilisées pour les systèmes a bases de connaissances. L'ingénierie des

connaissances correspond à l'étude de modèles symboliques formels plongés dans des systèmes d'usage : c'est l'ingénierie informatique et logique des modèles en fonction des usages qu'ils rendent possibles et des appropriations qu'ils permettent.

• Le génie cognitif a pour but de déterminer les connaissances d'un expert dans un domaine précis et de les simuler. Le mot "connaissance" englobe à la fois des faits, des problèmes typiques et des heuristiques de résolution de problème. Au début des systèmes expert, le mot "connaissance"se réfèrait aux connaissances d'un expert acquises au terme d'une longue expérience. Ces connaissances sont souvent peu formalisées et inaccessibles au non-expert. En raison de cet intérêt porté aux connaissances spécialisées et floues, la méthodologie du génie cognitif repose souvent sur des recettes comme celles de [Buch 83] Pour construire un système expert.

2.3. Les stades du génie cognitif

[WAT 85] WATERMAN propose dans un projet de réalisation et de mise en exploitation d'un système expert: identification, conceptualisation, formalisation, implémentation et test.

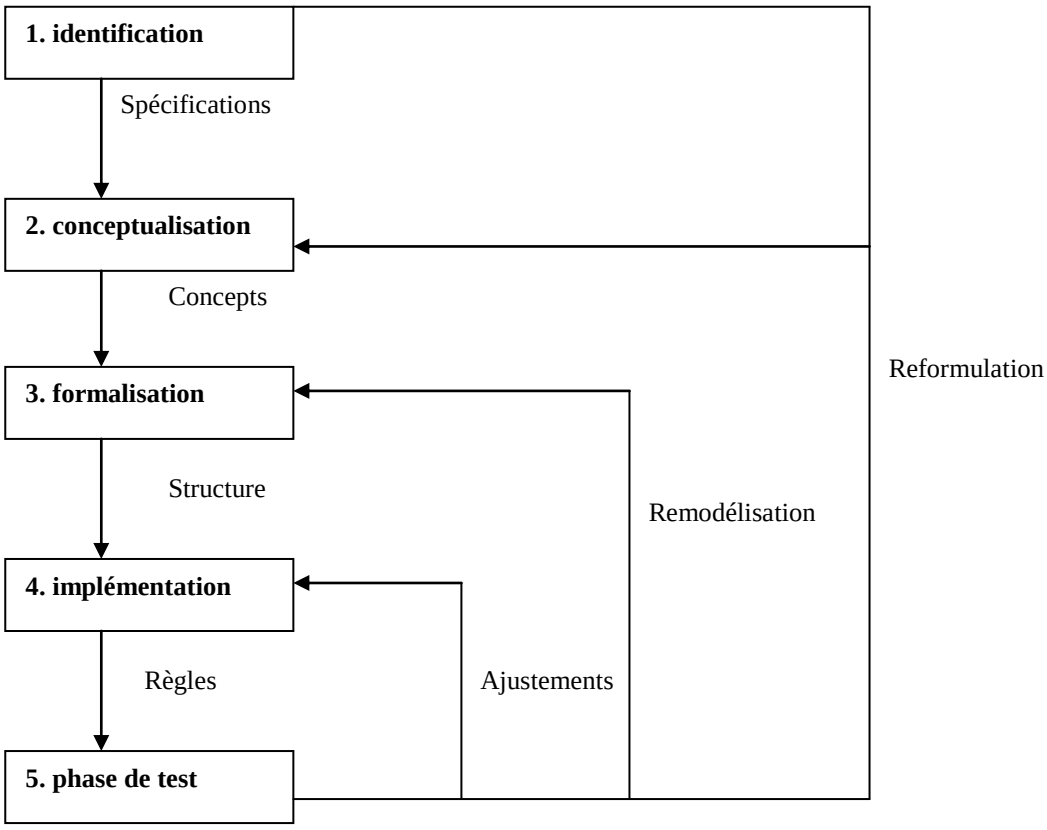


Figure 04: Les stades du génie cognitif selon Waterman

2.3.1. L'identification

il s'agit de définir les buts du projet, les tâches détaillées que le système devrait résoudre et d'identifier les contraintes techniques et pratiques (machines, logiciels, personnel, temps, etc. à disposition)

2.3.2. La conceptualisation et l'élicitation (obtention) des connaissances

il faut identifier et/ou constituer un vocabulaire plus précis pour décrire une activité de résolution de problème. Les concepts, relations, heuristiques, etc. identifiés lors de l'étape précédente doivent être rendus explicites.

2.3.3. La formalisation:

les concepts exprimés en langage naturel doivent être traduits dans un langage formel (dans un sens pas très strict du terme). Cela permet d'exprimer un problème ainsi que les heuristiques utilisées pour le résoudre avec des expressions quasi formelles. Ensuite, il faut rendre apparentes les interdépendances des données et des connaissances (flux d'information, identification de sous-problèmes, etc). En même temps, il faut choisir (ou construire) un logiciel approprié (un moteur d'inférence par exemple).

2.3.4. L'implémentation

Avec les connaissances formalisées, il faut construire un système informatique. Dans la plupart des cas, il s'agit d'écrire des règles informatiques. Parfois on construit un prototype rapide (un peu comme nos systèmes *-Lex) si cela n'a pas déjà été fait, afin de tester le bon choix des outils et la pertinence de l'analyse formelle.

2.3.5. La phase test

Chaque version du système doit être testée extensivement. Cela peut conduire à des multiples ajustements et affinages du système informatique, à une remodélisation complète du système qui renvoit à l'étape 3 ou encore à une reconceptualisation des tâches qui renvoie aux étapes 1 et 2. Le savoir-faire est dynamique et ne peut être saisi que dans une simulation dynamique. Aussi, ces itérations sont inévitables. Dans toutes les phases du projet, l'ingénieur cognitif (cognicien ou "knowledge engineer") doit travailler en coopération étroite avec l'expert.

Une implication importante de l'expert est surtout inévitable pour les phases 2 (élicitation du savoir) et 5 (test). A condition de pouvoir l'intéresser à un langage plus formel, il est aussi conseillé de l'associer aux phases plus techniques.

2.4. L'ingénieur cogniticien :

La vision "classique" de l'ingéniérie de la connaissance est très centrée sur le problème de bien "extraire" et de bien modéliser les connaissances de l'expert. Le rôle clé dans le dispositif appartient à l'ingénieur-cognicien. C'est lui qui "gère" les trois niveaux du "knowledge engineering" les plus importants :

- **Elicitation de la connaissance** : il s'agit d'extraire les connaissances de l'expert et d'autres sources disponibles.
- **Analyse de la connaissance :** il s'agit de structurer et d'organiser ces connaissances qui sont normalement hautement non-structurées et de formes diverses.

Au début du processus, il s'agit de voir d'abord quels sont les types de connaissance utilisés et dans un deuxième temps, il faut préparer l'implémentation, éventuellement par une formalisation non-informatique des connaissances.

- **Implémentation de la connaissance :** ensuite, il faut traduire ces connaissances

Structurées dans un format informatique.

2.5. L'expert

Il est considéré comme un spécialiste compétent dans un domaine bien cerné. Son habilité est due bien entendu, à sa formation, sa capacité d'analyse, mais aussi a sa compétence et à sa maîtrise des nombreux cas, qu'il a déjà rencontré au cours de son expérience professionnelle : il a acquis un savoir-faire grâce à son expérience diversifiée et de qualité.

Dans les compétences de 1'expert, on trouve ses capacités à :

• Poser et exprimer clairement des problèmes mal posés.

• Résoudre les problèmes nouveaux et délicats.

L'expert doit pouvoir intervenir directement sur les connaissances formalisées afin de les enrichir ou de les modifier jusqu'à qu'il soit satisfait.

2.6. Conclusion

Les sciences cognitives forment un champ pluridisciplinaire où les développements récents de 1'informatique et de 1'intelligence artificielle apportent leurs lumières sur des problèmes classiques de psychologie et de philosophie de 1'esprit. Les anciennes questions concernant le langage. Le raisonnement ou notre manière de voir le monde, voient une nouvelle forme.

3. Représentation des connaissances

3.1. Introduction

Résoudre un problème dans un domaine quelconque nécessite une grande quantité de connaissances lier à cette activité, Le but de la représentation de connaissances est de rendre compte, de modéliser un domaine particulier d'application de sorte que la représentation ou le modèle obtenu Soit manipulable par une machine.

Le modèle du traitement des connaissances provient non pas de l'informatique ou des mathématiques, mais des sciences humaines, en particulier de la psychologie cognitive.

3.2. types de Représentation des connaissances

La représentation des connaissances se situe au coeur de la problématique des sciences cognitives en général, et de l'intelligence artificielle en particulier.

Elle fait parfois appel à des théories développées antérieurement mais qu'elle utilise sous un angle particulier, parfois elle crée de nouveaux outils, conceptuels.

La représentation des connaissances est au cur des recherches en psychologie cognitive et en intelligence artificielle. Sa problématique peut s'exprimer ainsi [Ferber 89].

- Quelles sont les catégories conceptuelles a priori de la pensée et du raisonnement ?
- Quels sont, à partir de ces catégories fondamentales, les mécanismes pratiques qu'un système (individu ou machine) emploie de façon à pouvoir résoudre des problèmes, prendre des décisions et parvenir à ses fins ?

Ces deux interrogations se retrouvent dans tous les problèmes d'intelligence artificielle. Leur résolution nécessite de faire un certain nombre de choix.

1) Point de vue théorique : choix d'un formalisme permettant de représenter et de raisonner sur les connaissances.

2) Point de vue cognitif : choix des concepts nécessaires à la description et à la résolution du problème étudié.

3) Point de vue Implémentation : choix de moyens pratiques permettant la représentation efficace des données par un programme.

4) Point de vue méthodologique : choix des moyens permettant l'acquisition et la validation des connaissances.

Daprès la classification de J L Laurière [LAUR 82]. Les connaissances sont classées par rapport à leurs natures de représentation : procédurales ou déclaratives. Voir figure 5.

La représentation déclarative permet de spécifier un savoir indépendamment des contraintes et des méthodes d'utilisation. Elle permet de répondre à une question de type " quoi ".

La structure de contrôle est séparée des connaissances entrées sous formes de règles données en vrac. L'intérêt primordial est la modularité.

La représentation procédurale permet, de traiter des problèmes de type algorithmique c'est à dire complètement analysés et entièrement connus. Elle exprime un flot d'informations structurées et traduit "comment" transite la connaissance qui est une simulation du comportement réel.

Les représentations peuvent être regroupées dans les trois grandes classes suivantes :

Procédurales : incluant les automates finis et les programmes déclaratives : comprenant le calcul des prédicats et les règles de production structurées : réseaux sémantiques, frames, schémas, scripts, objets.

Certains modèles de représentation sont plus aptes à modéliser une connaissance fortement déductible, d'autres conviennent mieux à une connaissance descriptive et enfin d'autres encore à une connaissance structurée.

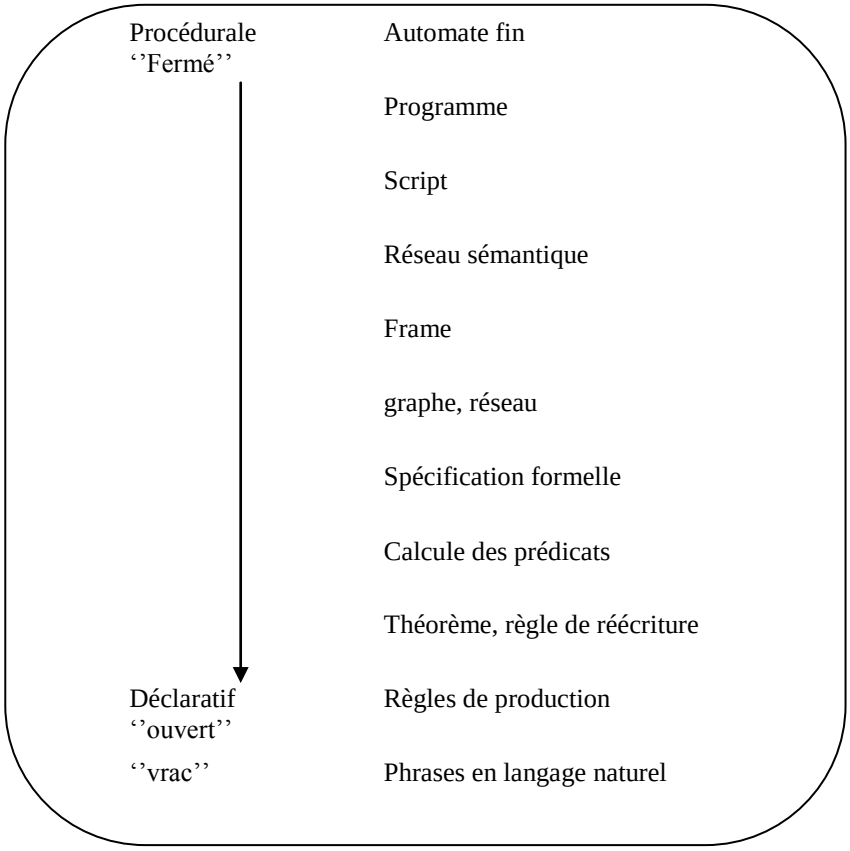


Figure 05 : Classification de J.L. Laurrière

3.3. Représentation procédurale

Cette représentation mélange des données limitées aux faits et contrôle. Ce mode de représentation de connaissance est peut utilisable car il est difficile à modifier et à entendre de l'évolution de la connaissance.

- Les automates finis

Un automate est une représentation procédurale de la connaissance utilisée pour représenter des protocoles ou représenter des plannings d'actions. Une action se trouve dans un nœud et un arc représente la règle à entreprendre (à exécuter) ou prédicat pour passer d'un état à un autre .

- Les programmes

Un programme est une suite finis d'instructions ordonnées, exécutables par ordinateur. Les programmes sont souvent représentés par des procédures ou des morceaux de programmes à exécuter en cas de besoin par une série d'appels. Il existe deux sortes d'appels : appel directe, attachement procédural.

3.4. Représentation Logiques :

Un langage symbolique issu des logiques mathématiques permet de formuler des descriptions sous une forme proche du langage courant et représentable dans un langage de programmation. La représentation logique est à la base de systèmes d'I.A. comme les démonstrateurs de théorèmes.

3.4.1. Logique des propositions

Le calcul des propositions est une théorie assez simple, mais ayant une utilité fondamentale notamment dans la réalisation des systèmes d'I.A.

La logique d'ordre (0) ou calcul des propositions permet la manipulation d'assertions ainsi que la création de nouvelles assertions à partir des anciennes.
"Il fait beau" est une assertion

La logique propositionnelle est appelée calcul des propositions et étudie des énoncés qui sont soit vrais, soit faux. La manipulation et création d'assertions repose sur la règle (P & (P => Q)) => Q dite de "modus ponens".
Et sur la règle (!Q & (P => Q)) => !P dite de "modus tollens".

3.4.2. Logique des prédicats

Le calcul des propositions est trop pauvre pour représenter des activités du monde réel. La logique du premier ordre introduit les notions de variables et de quantificateurs.
Les modes d'inférences sont le modus ponens et la spécification universelle.
Celle-ci consiste à substituer une variable quantifiée par une constante. Pour des expressions est réservée l'appellation d'unification.

3.5. Règles de production

Ce modèle de représentation est très répandu. Il est proche de la formulation naturelle des raisonnements et est donc facile à utiliser. Les règles de production permettent de représenter des connaissances dynamiques.

SI *Prémisse(s)*

ALORS *Conséquence(s).*

Figure 06 : syntaxe d'une règle de production

Par le fait que la conséquence d'une règle peut être prémisse d'une autre, l'ensemble des règles de production est organisé en réseau. Ce type de représentation permet de résoudre des problèmes de nature déductive et inductive. Selon l'utilisation qu'on en fait, on distingue différentes classes de règles de production qui sont :

- les règles de réécriture,
- les règles d'inférence,
- les règles de cohérence pondérées : elles peuvent faire partie de la base de connaissance et assurent une certaine validité au raisonnement. La pondération permet au système de rejeter toutes les incohérences existantes,
- les règles d'équivalence.

❖ **Avantages :**

- Modularité : caractéristique du savoir humain, très bien représentée par les règles. En effet, les règles peuvent, indépendamment les unes des autres, être ajoutées ou supprimées,
- Facilité de manipulation et de compréhension,
- Les règles peuvent être données en vrac,
- La contradiction et la redondance sont aisément détectées par traitement

❖ **Inconvénients :**

- Lenteur des systèmes qui les utilisent
- Complexité quand il y a augmentation du nombre de règles
- Inefficace pour traiter les relations statiques entre les objets
- Il ne prend pas en compte tous les liens de causalité et d'autres qui peuvent exister dans la connaissance.
- Les notions de classification hiérarchique et de contexte ne peuvent pas être utilisées,
- il répond très mal au besoin de représenter des objets manipulés,
- les règles sont activées indépendamment les unes des autres, il n'y a pas d'inter-relations possibles.

REGLE N° 11

Si

Voyage1.Moyen=”Voiture”

Voyage1.Ville1=”BBA”

Voyage1.Vilel2=”Setif”

Alors

Voyage1.Prix=100 DA

Figure 07: Exemple de règles.

3.6. Réseaux sémantiques

Issus des modèles psychologiques de Quillian et Raphael, (1968), les réseaux sémantiques sont des graphes dont les noeuds représentent des objets, événements ou concepts et les arcs représentent des relations tout type entre les noeuds.

Introduite pour représenter le sens des mots, cette technique a été utilisée dans le système expert Prospector (Recherche géologique). Les propriétés et les attributs se transmettent de noeud en noeud en glissant le long du réseau comme des associations d'idées.

Ils permettent d'exprimer sous forme de liens et de gérer des relations entre objets hiérarchisés (héritage de propriétés, traitement d'exceptions). Ils permettent de représenter des connaissances statiques et dynamiques. La représentation graphique facilite la compréhension lorsqu'il s'agit pour l'utilisateur d'examiner le contenu de la base de connaissances.

❖ Avantages

- La représentation graphique facilite la lecture et la compréhension,
- la structure avec noeuds et liens est proche des symboles et pointeurs utilisés en informatique,
- Ce mode de représentation est très utilisé en I.A,
- Il permet une gestion correcte des exceptions.

❖ Inconvénients

- Impossibilité d'exprimer des connaissances de types procédurales ou quantifiés,
- Il n'existe pas de procédure manipulant rapidement des réseaux sémantiques.
- Difficulté pour fournir des explications sur leur raisonnement.

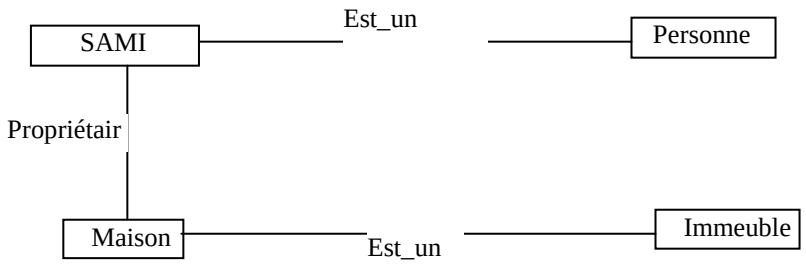


Figure 08: Représentation d’un simple réseau sémantique

3.7. Objets structurés

Le concept d'objet structuré est générique. Il recouvre en fait, les schémas : dans les travaux du psychologue Bartlett sur la mémoire, les frames : introduites par Minsky ont perdu leur richesse première dans les travaux effectuées par la suite, les scripts dû à Schank et Abelson qui décrivent l'encadrement d'évènements, les objets pris au sens de Smalltalk.

3.7.1. Frame

Les "frame" ou cadres ou encore schémas sont issu des travaux de Minsky [MINSKY 75]. Ils supposent que la connaissance d'un homme n'est pas en vrac mais structurée autour d'unités d'informations. Tous les scénarios de la vie courante peuvent se représenter sous la forme de frame.

Un frame est une structure de données incluant des informations à la fois déclaratives et procédurales.

Le format général d'un frame possède normalement trois niveaux: frame / slot / facet.

```
(frame (slot1 (facet 1 valeur 1)
" "
(facet n valeur n)
)
"
(slot q (facet 1 valeur 1)
" "
(facet n valeur n)
      )
)
```

Figure 09 : La notation "frame" (ou "slot and filler")

3.7.2. Script

C'est une structure qui permet de décrire un scénario stéréotypé et la suite des personnages mis en cause. Il peut faire appel à d'autres scripts, et il s'intéresse beaucoup à l'aspect dynamique de la connaissance.

Les "scripts" (une invention du groupe d'intelligence artificielle de Yale) ont comme objet les connaissances spécialisées, par opposition aux connaissances de nature plus générale comme les heuristiques. Dans le modèle de Yale, les "scripts"servent à interpréter des événements fréquemment rencontrés et à y participer. Schank et Abelson [SCH, ABEL 77] postulent qu'il existe des milliers de tels "scripts"dans la mémoire humaine.

Ces scripts nous permettent de faire des économies de raisonnement et ils guident l'interprétation d'événements rencontrés dans des récits ou des activités courantes. Un "script" contient des mécanismes spéciaux pour gérer les connaissances stéréotypiques que l'on possède sur des institutions et des conventions de comportement. Il crée des attentes sur les objets que l'on peut rencontrer dans une situation d'un certain type et il fait des prédictions sur la suite des événements qui peut arriver.

Il existe plusieurs types de script: le "script" le plus souvent modélisé dans des programmes de Yale est situationnel. L'exemple le plus connu est le "script" du "restaurant" qui a déjà inspiré Tolman [Tolman 51].

1. Entrer au Restaurant.
2. S'asseoir la table.
3. Appeler le garçon.
4. Lire le menu.
5. Commander un plat.
6. Manger.
7. Payer la note.
8. Sortir de Restaurant.

Figure 10. *Script Aller au Restaurant.*

3.8. Conclusions

L'étude de ces différents modes de représentation de connaissances nous permet de déduire que :

1. Une représentation est toujours une approximation.

2. Il n'existe pas une représentation qui soit uniformément meilleure qu'une autre.
3. Représenter implique nécessairement de sacrifier certaines propriétés de l'entité représentée.
4. Il n'existe pas de représentation idéale, le choix d'un mode dépend de la nature des connaissances, du degré de contrôle que l'on veut exercer sur le raisonnement et du type de problème à résoudre.
5. Il faut toujours rechercher un compromis entre clarté et puissance de représentation d'une part et efficacité des raisonnements d'autre part.

La tendance actuelle de l'IA de faire coexister dans le même système plusieurs modes de représentation de façon à mieux rendre compte de la diversité des connaissances mises en œuvre.

4.Acquisition des connaissances

4.1. Introduction

La conception d'un système expert exige un travail de transfert de connaissances entre sources d'expertises (experts humains ou documents) et un outils informatique de à disposer ensuite d'un système à base de connaissances (SBC), pouvant être consulté comme un expert.

Traditionnellement cette construction met en jeu un ou plusieurs expert, ainsi q'un ou plusieurs ingénieur de connaissances (les cogniticiens) chargés d'extraire la connaissance des experts pour la traduire dans le formalisme de représentation des connaissances offert par un générateur de systèmes experts cible.

La phase de transfert d'expertise est souvent considérée comme le goulot d'étranglement du développement d'un système expert .en effet. Elle peut durer des mois voire des années et. Compte-tenu de la complexité des connaissances de l'expert. Qui parvient difficilement à expliciter ses processus mentaux. La connaissance extraite risque souvent d'être inexacte. Incomplète. Voire inconsistante. Diverse recherche sont donc menées pour aider l'expert et le cogniticien dans cette phase de transfert d'expertise [Dieng 01] :

Certains chercheurs réfléchissent à des méthodes inspirées du génie logiciel (description de cycle de vie d'un SBC. méthode générales) ou de la psychologie (analyse de la nature de l'expertise technique de recueil des connaissances).

D'autres proposent des outils destinés à aider l'expert ou le cogniticien. Certains outils permettent l'extraction de la connaissance à partir d'interviews de l'expert ou de documents écrits. Dans certain cas. La présence d'un cogniticien n'est plus nécessaire.

L'expert pouvait manipuler directement l'outil pour y entrer son expertise. autres outils considèrent l'acquisition des connaissances comme une activité de modélisation et sont guider par des modèles, certains outils utilisent des techniques inductives pour apprendre de nouvelles connaissances à partir d'exemples.

4.2. Les classes de problèmes

L'analyse des différentes classes de problèmes traités par les SBC doit permettre de mieux orienter le processus d'acquisition. Un certain nombre de travaux aident à la compréhension des types de problèmes possibles et des modes de résolution qui leur sont associés.

D'après Clancey [Clan 85] les problèmes sont classés en problèmes d'analyse (interprétation d'un système) et problèmes de synthèse (construction d'un système). Si l'on caractérise par ses entrées, ses sorties et le système lui-même. Les problèmes d'analyse peuvent se répartir en trois classes de problèmes suivant ce qui inconnu : les problèmes d'identification ou de reconnaissance permettent d'identifier un système dont on connaît les entrées et les sorties : les problèmes de prédiction permettent de prédire les sorties de système, connaissant ses entrées et le système lui-même : les problèmes de contrôle permettent de trouver les entrées adéquates d'un système connu pour obtenir les entrées désires.

Pour la synthèse. On distingue les spécifications (contraintes qu doit satisfaire le système à construire). La conception et l'assemblage correspondant a la construction physique du système. la conception peut consister en une configuration pour structurer un système ou une planification pour construire un plan d'action.

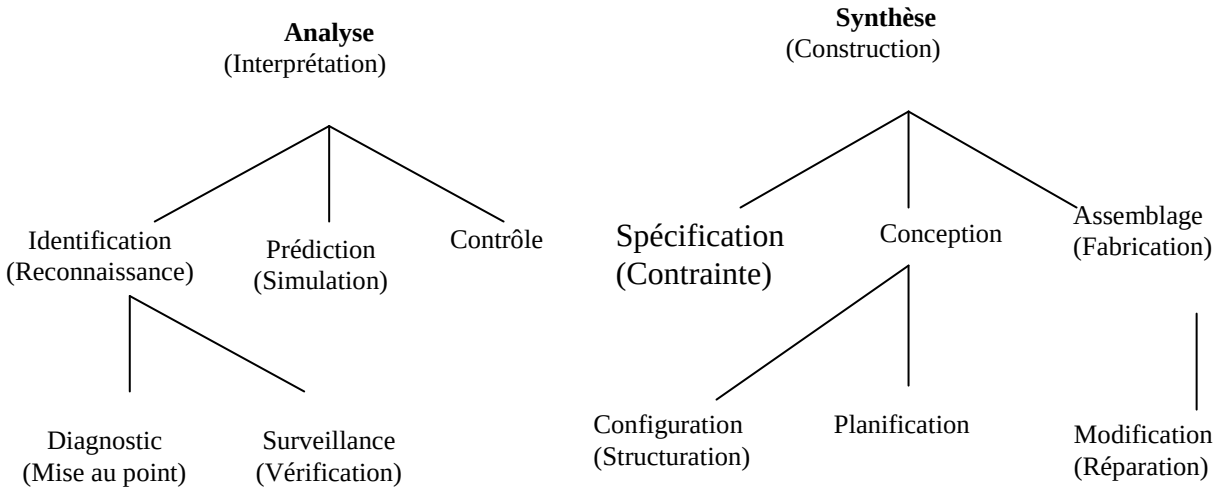


Figure 11 : Classification des problème proposé par clancey

Autres travaux, CHANDRASEKARAN & all reposent sur l'idée que tout SBC devrait être construit grâce a des briques de base, appelées taches génériques, chacune étant associée à un type élémentaire de résolution de problème, chaque tache générique est caractérisée par sa fonction, la représentation et l'organisation des connaissances et ses stratégies d'inférence.

Cela permet d'aborder l'acquisition des connaissances et la construction d'un SBC à un niveau d'abstraction plus élevé. Un certain nombre de taches génériques ont été identifiées.
A chacune d'elles est associée une stratégie d'acquisition de connaissances adaptée [Dieng 90].

4.3. Le cycle de vie d'un système à base de connaissances

Souvent, les méthodes d'acquisition des connaissances sont associées à une certain vision du cycle de vie du SBC, les premiers propositions décrive le cycle de vie d'un SBC étaient souvent basées sur le prototype rapide. Elles distinguaient les phases suivantes :

- Etape d'identification qui permet la mise en place des acteurs et des ressources. Le choix d'un problème approprié et d'objectifs précis.
- Développement en quelques mois d'une maquette : cette phase comprend la conceptualisation du problème (à partir d'entretiens cogniticien- expert. Explication des principaux concepts. relation et stratégies de résolution du problème). Puis la formalisation (représentation dans un formalisme) .l'implantation dans un outil choisi de façon adéquate et enfin des tests pour valider cette maquette.
- Développement d'un système complet (par extension ou modification totale de la base de connaissances de la maquette).
- Validation du système complet : cette phase consiste en une évaluation du système obtenu par rapport à des bibliothèques de cas tests ou par d'autres experts que ceux ayant participé à son développement.
- Intégration dans l'entreprise (avec une éventuelle connexion à d'autres logiciels) et maintenance.

Dans une approche différente. Appelée l'acquisition structurée des connaissances. L'implantation n'à lieu que plus tard. Une fois la connaissance décrite une représentation intermédiaire.
Le cycle de vie proposé se rattache à cette seconde approche :

- Définition du problème.
- Modélisation de la connaissances : cette phase permet de recueillir la connaissance et de développer des modèles des domaine indépendamment des décisions d'implantation. On obtient alors un ensemble de modèles su lesquels sera basée la conception du système : le recueil des

connaissances 1 (description textuelle de la connaissance du domaine) et l'analyse (grâce à laquelle on obtient un modèle objet et un modèle de résolution de problèmes).

- ❖ Conception du système.
- ❖ Implantation et tests.

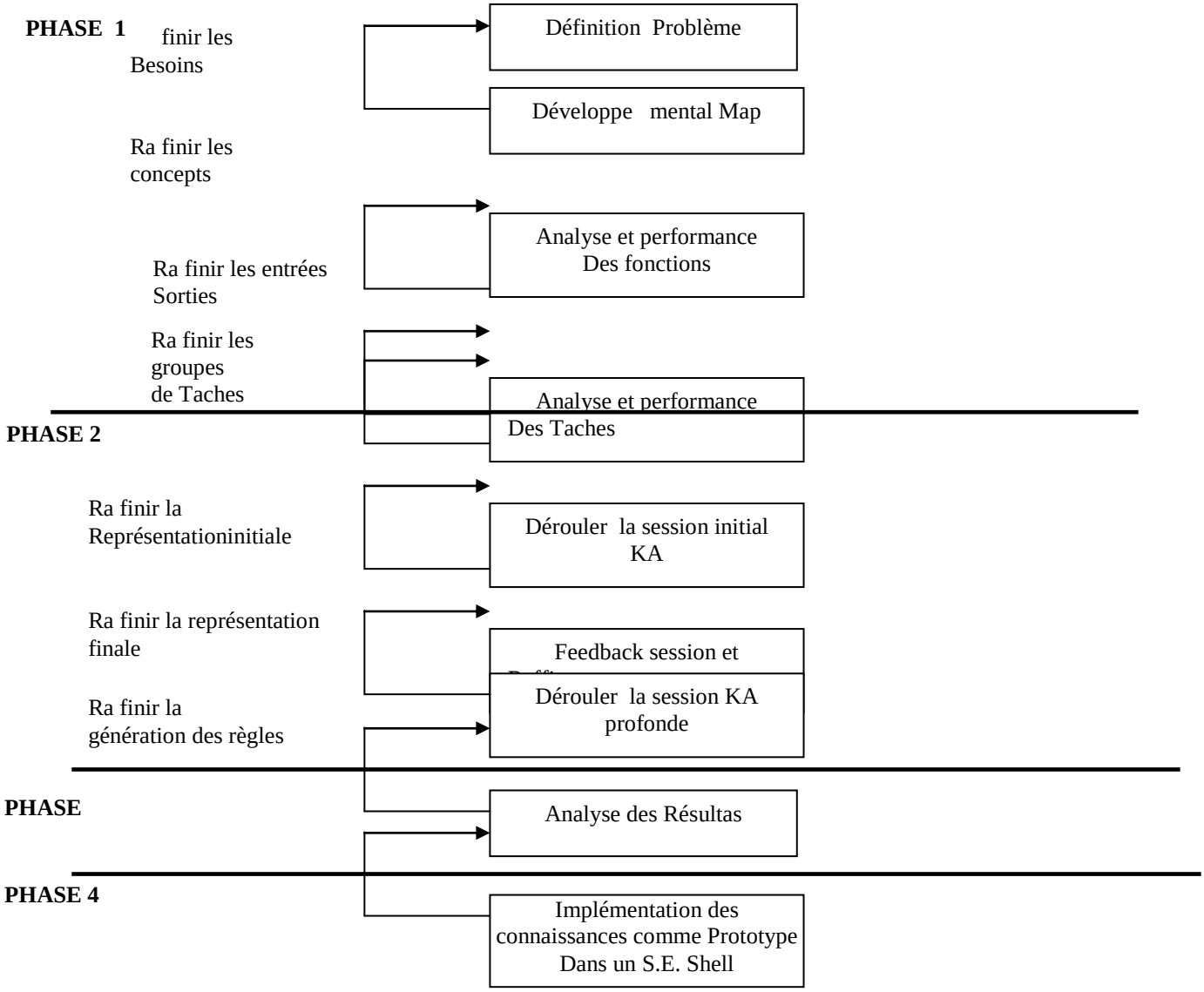


Figure 12 : les phases d'acquisition des connaissances

4.Moteur d'inférence

4.1. Introduction

Notre système expert (ExperRule) est un langage à base de règles d'ordres 0, 1, et 2 dont le moteur d'inférence associé fonctionne en chaînage avant. La principale difficulté pour mettre en oeuvre ce genre de programme réside dans l'algorithme de pattern matching, ou algorithme d'instanciation, qui doit trouver, au vu de la base de faits courante, les substitutions valables des variables d'une règle.

De nombreux et importants travaux de recherche ont été consacrés à trouver des méthodes générales pour résoudre des problèmes de satisfaction de contraintes, qui consistent à trouver les assignations correctes à des variables soumises à un ensemble de contraintes.

On peut citer à ce sujet le langage ALICE de Jean-Louis Laurière [LAUR 76], qui est un résolveur général de problèmes combinatoires, et les nombreux travaux sur les méthodes de propagation de

contraintes. Or il s'avère que, si l'on prend le contre-pied de ce qui est à la base de l'algorithme basé sur RETE [Forg 82].

Le système expert (ExperRule) a la possibilité d'avoir des prémisses entièrement variables (ordre 2), cette possibilité lui offre des avantages essentiels pour la représentation des connaissances.

4.2. Type de fonctionnement du Moteur inférence :

Ce critère permet également à l'utilisateur de conserver un minimum la maîtrise de l'inférence.un Système expert peut être d'ordre 0 ou d'ordre 1 (d'autres ordres 0+, 0++, 1+ ont été introduits pour rendre compte de dispositifs particuliers).

De nombreuses autres caractéristiques différencient les systèmes existants. Une de celles ci est la façon d'utiliser les règles: de la prémisse vers la conclusion (chaînage avant) ou de la conclusion vers la prémisse (chaînage arrière).

4.3. Instanciation d'une règle

Trouver les instanciations d'une règle, c'est résoudre le problème combinatoire qui consiste à trouver les "assignations de valeurs" (les substitutions) des variables de la règle soumises à un "ensemble de contraintes" (les prémisses de la règle).

Le système SNARK (Laurière& Vialatte, 1985) est le premier à être basé sur les méthodes issues de la résolution de problèmes de satisfaction de contraintes. [Dor 89]

Un algorithme d'instanciation, basé sur les notions de choix dynamique et de propagation de contraintes.

Pratiquement, cet algorithme est accompagné de la mise en oeuvre d'heuristiques. L'efficacité de cet algorithme passe par une structuration de la base de faits. [Dor 89].

4.3.1. Principes de l'algorithme d'instanciation

Le principe du mécanisme d'inférence dans certains algorithmes est de calculer les instanciations d'une seule règle à la fois [Dor 89].

Cela est très différent de ce qui est à la base de nombreux algorithmes de pattern matching, notamment celui basé sur le réseau RETE [Forg 82] décrit dès 1979 par Forgy. Celui-ci était justifié par la combinaison d'une contrainte sur le mécanisme d'inférence, une constatation sur le déroulement pratique de l'inférence :

- on veut que l'ensemble de toutes les instanciations de toutes les règles – appelé l'ensemble des conflits – soit déterminé avant chaque déclenchement de règle pour lui appliquer un critère de choix de l'instanciation à déclencher.

- le déclenchement d'une règle modifie – probablement – peu de faits par rapport à l'ensemble de la base de faits, et donc – probablement – peu de règles sont concernées par ces changements.

On peut donc espérer que l'application d'une règle modifie peu l'ensemble de conflits, et donc qu'une bonne part du travail effectué au cycle précédent ne soit pas à refaire.

Il s'avère que le moteur n'a pas les éléments nécessaires pour faire un « bon choix de règle ». Il est donc préférable de fournir à l'utilisateur un critère de choix de la règle à appliquer très simple , et cela évite bien des perversions , ce qui permet en retour de ne chercher les instanciations que pour la règle choisie.

Le deuxième argument tombe également en défaut dans la pratique. Il s'avère en effet fréquemment que le déclenchement d'une règle interdit quelques instanciations d'autres règles encore non choisies. [Dor 89]

4.3.2. Représentation du problème d'instanciation

Deux représentations graphique permettent de visualiser les opérations accomplies par un algorithme de pattern matching et de mieux juger de son efficacité potentielle : la représentation prémisse, et la représentation variable duale.

Dans chacune des deux représentations, on construit à partir des prémisses d'une règle un graphe (en général un hypergraphe) :

- dans le premier cas, on associe à chaque prémisse un noeud, et à chaque variable une – hyper – arête reliant les prémisses qui la contiennent.

- dans le second, on associe à chaque variable un noeud, et à chaque prémisse une – hyper – arête reliant les variables qu'elle contient. [Dor 89]

Les arêtes pourront être libellées par, respectivement, les variables et les prémisses.

4.4. L'inférence

L'inférence se fait classiquement en chaînage avant jusqu'à saturation de la base de faits.
Plusieurs conditions d'arrêt sont possibles, et paramétrables pour chaque règle par L'utilisateur.

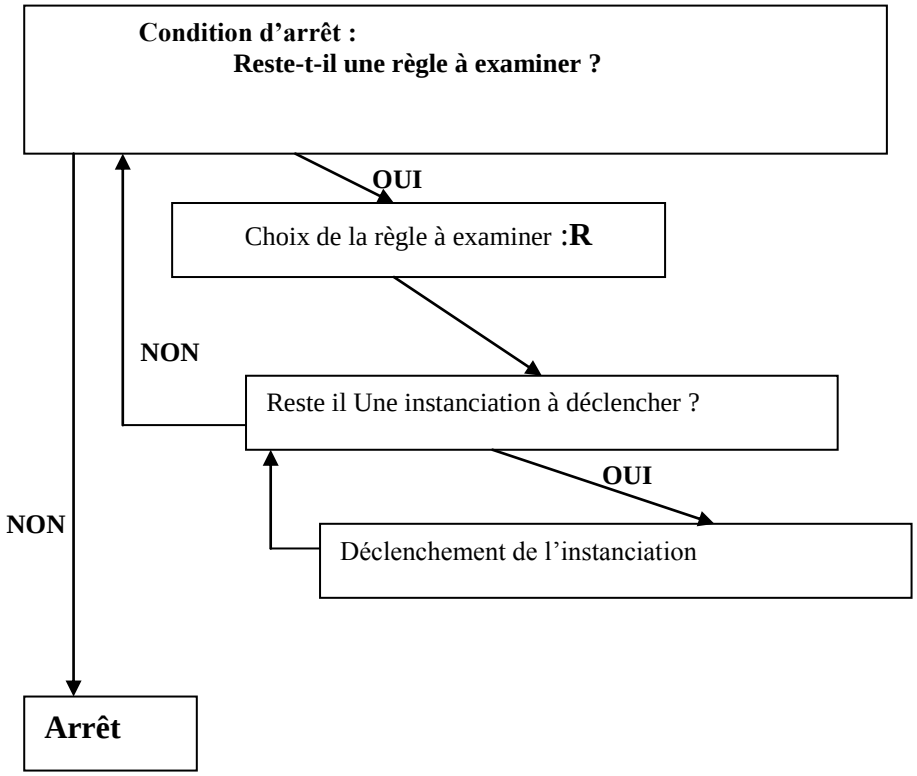


Figure 13 : Mécanisme d'inférence

4.5. Cycle d'inférence

Le cycle d'inférence comprend les étapes suivantes :

1. la première opération exécuté par le moteur d'inférence est d'instancier les variables ; c'est-à-dire de substituer les noms des classes par ceux des objets associées, puis évaluer chaque condition après l'autre.
   Une règle est candidate si toutes les prémisses sont vérifiées, et sera insérée dans une liste appelée liste de conflits.

2. Le choix de la règle à appliquer est fait par une condition très simple: il choisit d'appliquer la première règle applicable dans la liste des règles. On a évité tout algorithme de choix compliqué, car il s'avère à l'expérience qu'il est impossible d'obtenir un critère algorithmique de choix qui choisisse effectivement la "bonne" règle (cela nécessiterait une masse importante de connaissances sur les connaissances à traiter – des méta connaissances - que ne peut avoir un moteur d'inférence [Pitr 85]).

5.Conclusion

L'Ingénierie des connaissances :

- peut participer à une démarche d'explicitation globale des processus de traitement des connaissances.
- L'acquisition des connaissances a su devenir une Ingénierie des connaissances. Modélisant un objet et des processus complexes, elle a pour objectif de s'intégrer encore plus dans les ressources des programmes informatiques par la proposition de méthodes et d'outils reproductibles : l'Ingénierie des connaissances continue à produire des concepts et des méthodes reproductibles, évaluables, chiffrables pour modéliser à plusieurs niveaux, concevoir et de développer des systèmes spécialisés, des SBC.
- Elle a un rôle particulier à jouer pour construire des modèles et des outils favorisant et accompagnant la dynamique des connaissances.

Chapitre 3

Techniques des Arbres de Décision

1.Introduction

Les méthodes de classification ont pour but d'identifier les classes auxquelles appartiennent des objets à partir de certains traits descriptifs. Elles s'appliquent à un grand nombre d'activités humaines et conviennent en particulier au problème de la prise de décision automatisée.

La construction des arbres de décision à partir de données est une discipline déjà ancienne. Les statisticiens en attribuent la paternité à Morgan et Sonquist qui sont les premiers, ont utilisé les arbres de régression dans un processus de prédiction et d'explication (AID – Automatic Interaction Detection) [Morg,Sonq 63]. Il s'en est suivi toute une famille de méthodes, étendues jusqu'aux problèmes de discrimination et classement, qui s'appuyaient sur le même paradigme de la représentation par arbres [Morg,Messg,Thaid 73] ;[Kass 80].

On considère généralement que cette approche a connu son apogée avec la méthode CART (Classification and Regression Tree) de Breiman et al 84 décrite en détail dans une monographie qui fait encore référence aujourd'hui.

En apprentissage automatique, la plupart des travaux s'appuient sur la théorie de l'information. Il est d'usage de citer la méthode ID3 de Quinlan (Induction of Decision Tree – [Qui 79]) qui, lui même, rattache ses travaux à ceux de Hunt [Hunt 62).

Quinlan a été un acteur très actif dans la deuxième moitié des années 80 avec un grand nombre de publications où il propose un ensemble d'heuristiques pour améliorer le comportement de son système. Son approche a pris un tournant important dans les années 90 lorsqu'il présenta la méthode C4.5 qui est l'autre référence incontournable dès lors que l'on veut citer les arbres de décision (1993). Il existe bien une autre évolution de cet algorithme, C5.0, mais étant implémentée dans un logiciel commercial, il n'est pas possible d'en avoir le détail.

2. Représentation et interprétation des arbres de décisions

De façon générale, un arbre de décisions est un arbre dont les noeuds internes représentent des tests sur les attributs des données d'entrées, et dont les feuilles représentent des classes (concepts ou cas) correspondantes aux données [Nils 96]; Les tests des noeuds d'un arbre de décisions visent à couvrir les valeurs des attributs des données d'entrées de façon exhaustive. Chaque test peut porter sur une seule variable (c'est le cas en général) ou sur plusieurs. Les tests sur les valeurs peuvent être exclusifs ou être pondérés avec des probabilités.

L'interprétation d'un arbre de décisions correspond à celui d'une forme normale disjonctive : chaque chemin menant de la racine de l'arbre vers une feuille peut s'interpréter comme une conjonction de valeurs d'attributs ; les branches menant vers des feuilles dont la classe est similaire peuvent s'interpréter comme une disjonction. Des règles en forme normale disjonctive peuvent être extraites directement à partir des arbres de décisions [Lounis 06b] ; [Witt,lan,Frank 05]. Ces règles sont non-ambigües et insensibles à l'ordre mais peuvent être complexes et peuvent généralement être simplifiées automatiquement par des heuristiques.

2.1. Données d'entrées pour la construction et l'utilisation des arbres de décisions

Chaque donnée d'entrée est spécifiée par un ensemble d'attributs et est associée à une classe: cette classe est spécifiée à priori lors de l'apprentissage et doit être prédite à partir de l'arbre de décisions après apprentissage. Chaque attribut est spécifié par un nom et une valeur. Différents problèmes sont reliés au traitement des attributs [Lounis 06b] : (1) les attributs peuvent avoir des valeurs numériques, nominales ou les deux, (2) les valeurs d'attributs peuvent être manquantes, ou (3) les valeurs ou noms d'attributs peuvent être bruités (i.e. qu'ils peuvent contenir des erreurs). Les fonctionnalités supportées concernant les attributs varient selon les algorithmes de traitement d'arbres de décisions.

2.2.Attributs à valeurs numériques

Les attributs à valeurs numériques se retrouvent dans beaucoup de bases de données. Cependant, les premiers algorithmes reliés au traitement d'arbres de décisions n'ont d'abord bien supportés que les attributs

à valeurs nominales [Witt,lan,Frank 05]; les valeurs numériques étaient parfois traitées comme des valeurs nominales ce qui pouvait engendrer des arbres de grande taille. Dans les algorithmes supportant les attributs à valeurs numériques, les tests des noeuds d'un arbre de décisions portant sur des attributs numériques se font généralement en sélectionnant une valeur seuil permettant de séparer les données en deux ensembles.

Marsala et Bouchon-Meunier (1999) proposent une approche de logique floue dont les tests sur les valeurs d'attributs (aussi bien numériques que nominales) partitionnent l'ensemble des données selon une ou plusieurs valeurs floues.

2.3.Attributs à valeurs manquantes

Les attributs à valeurs manquantes sont fréquents dans les bases de données [Witt,Frank 05]; [Qui 89]. Quinlan (1989) présente une étude sur un algorithme dont les paramètres permettent de traiter de façon variée des bases de données contenant des valeurs manquantes.

Cette étude indique que prendre en considération le statut particulier des attributs à valeurs manquantes permet d'offrir une meilleure performance que de l'ignorer. Cette étude indique également qu'une classification qui combine différents résultats probables à partir des valeurs connues offre la meilleure performance parmi les approches étudiées.

2.4.Attributs à valeurs bruitées

De façon générale, les données peuvent être bruitées et elles peuvent contenir des erreurs au niveau des noms d'attributs ou de leurs valeurs. Les arbres de décisions peuvent répliquer ou non ces erreurs coller aux données) selon les algorithmes utilisés [Nils 96];[Lounis 06b]. Lors de l'apprentissage, un mécanisme d'élagage peut être utilisé afin de traiter les erreurs. Les données dont les attributs sont identiques mais dont les classes sont différentes peuvent être traitées de façon à sélectionner comme classe celle qui est la plus fréquente.

3. Induction d'arbres de décisions

3.1. Construction non incrémentale (batch)

La construction non incrémentale d'arbres de décisions consiste à construire l'arbre à partir d'un ensemble d'exemples de données spécifiées à priori. Il y a différents algorithmes de construction non incrémentale d'arbres de décisions (ID3, C4.5).

Une approche généralement utilisée consiste à construire l'arbre de façon descendante (top-down), récursive et gloutonne, à partir des exemples de données [Nils 96];[Lounis 06b]. À partir de la racine, l'algorithme utilise un critère de division afin de choisir un attribut-1 dont les valeurs permettent la meilleure répartition des données en différents regroupements; la motivation est de réduire la taille de l'arbre en choisissant d'abord l'attribut estimé comme étant le plus discriminatif.

Les données sont ensuite répartionnées selon les valeurs de cet attribut de manière à créer une branche pour chaque regroupement de données. Le processus s'applique récursivement pour chaque branche. Ce processus de construction s'arrête, pour une branche donnée, lorsqu'une condition d'arrêt est atteinte: les données regroupées à un noeud donné ne peuvent plus être (suffisamment) distinguées par les attributs restants. Certains algorithmes intègrent un processus d'élagage pendant ou après la construction de l'arbre.

3.2. Construction incrémentale

La construction incrémentale d'arbres de décisions permet la construction d'arbres de décisions puis la révision de ceux-ci à partir de nouveaux exemples d'apprentissage [Utg 89].

La motivation pour réviser un arbre existant plutôt que de le reconstruire est relié au coût inférieur de la révision en terme de temps de traitement. Il y a différents algorithmes de construction d'arbres de décisions. Utgoff (1989) présente deux techniques dont ID5R.

4.Élagage des arbres de décisions

Certains algorithmes de construction d'arbres de décisions permettent l'élagage [Male et al 96] ; [Espo et al 97]. Le but de l'élagage est de simplifier les arbres de façon à éviter de rendre les tests sur les attributs trop spécifiques aux données d'entraînement et non suffisamment généraux pour de nouvelles données.

L'élagage peut se faire après la construction de l'arbre (on parle alors de post-élagage) ou pendant la construction (on parle alors de pré-élagage). Le post-élagage est la méthode la plus utilisée à cause de la difficulté à déterminer quand arrêter la croissance de l'arbre pendant sa construction.

Des travaux présentent une étude portant sur 8 méthodes standard de post-élagage. Les méthodes présentées offrent différentes caractéristiques: (1) certaines méthodes appliquent l'élagage de façon descendante alors que d'autres le font de façon ascendante; (2) certaines méthodes utilisent pour l'élagage le même ensemble d'exemples d'entraînement que celui utilisé pour la construction de l'arbre, alors que d'autres utilisent des exemples d'entraînement qui sont distincts de ceux utilisés pour la construction de l'arbre; (3) certaines méthodes n'appliquent que des opérateurs d'élagage de branches alors que d'autres permettent le rattachement de branches.

Dans les travaux rapportés, ces méthodes de post-élagage ont été appliquées sur les arbres produits par le même algorithme de construction pour une douzaine de bases de données dans différents domaines variant en terme de nombre de classes, nombre et type d'attributs, taux de bruit dans les données, etc.

Les arbres élagués produits par ces méthodes ont ensuite été analysés aux niveaux du taux d'erreurs et de la taille des arbres: les taux d'erreurs des arbres élagués ont été comparés avec ceux des arbres non-élagués; les tailles des arbres élagués ont été comparées avec celles d'arbres élagués obtenus avec une méthode prouvée optimale et en utilisant des corpus similaires pour l'élagage (i.e. corpus distincts ou non du corpus de construction d'arbres).

Les conclusions des analyses effectuées indiquent que les méthodes d'élagage utilisant le même corpus d'entraînement que pour la construction d'arbres produisent généralement des taux d'erreurs moindres lors de tests que les méthodes utilisant un sous-ensemble distinct du corpus d'entraînement pour l'élagage; ceci est dû au nombre des données disponibles pour l'entrainement qui est souvent limité.

Les méthodes d'élagage permettent de produire des résultats lors de tests dont les taux d'erreurs sont significativement moindres ou comparables aux arbres non-élagués.

Certaines méthodes d'élagage ont tendance à trop élaguer alors que d'autres ont tendance à ne pas élaguer suffisamment.

5.Critères de division des données

Différents critères peuvent être utilisés afin de déterminer comment partitionner les données à partir des attributs. De façon générale, ces critères visent à minimiser la taille de l'arbre en sélectionnant d'abord

les attributs les plus discriminants [Lounis 06b]. Le reste de cette section décrit quelques critères de division et la Section 6 décrit des algorithmes utilisant quelques-uns uns de ces critères.

Ces critères s'appliquent aussi bien pour des valeurs nominales que numériques. Dans le cas de valeurs numériques, ces critères s'appliquent souvent sur des valeurs seuils déterminées en comparant les moyennes entre chaque paire de valeurs numériques consécutives.

Il est à noter que le partitionnement est généralement exhaustif pour les attributs nominaux alors qu'il est souvent partiel pour les attributs numériques; dans une branche donnée, un attribut nominal n'apparaîtra généralement qu'une fois alors qu'un attribut numérique sera souvent re-testé plusieurs fois avec des seuils différents.

Un des critères de division utilisé dans certains algorithmes (version originale de l'algorithme ID3) est celui du gain d'information. Cette métrique est basée sur la mesure d'entropie qui correspond au degré d'incertitude dans un système.

Dans le contexte des arbres de décisions, l'entropie correspond au nombre de questions booléennes (vrai/faux) nécessaires afin de déterminer une classe: l'entropie est proportionnelle au degré d'incertitude sur la valeur d'une classe.

Le gain en information est une mesure permettant de savoir comment un attribut permet de diminuer l'entropie (l'incertitude) concernant la valeur d'une classe en partitionnant les données avec cet attribut.

Un second critère de division utilisé dans certains algorithmes (algorithme C4.5) est celui de ratio de gain d'information. La métrique de gain d'information est biaisée pour les attributs ayant le plus grand nombre de valeurs (i.e. produisant le plus de branches).

Le ratio de gain d'information corrige le biais du gain d'information en prenant en considération le nombre de valeurs d'un attribut et la proportion de ces valeurs dans les données.

Un troisième critère de division utilisé dans certains algorithmes (algorithme CART) est celui du coefficient d'impureté Gini. Ce coefficient pour un noeud donné est déterminé en soustrayant, au nombre 1, la somme des carrés des fréquences de chaque classe. L'attribut produisant la partition ayant le plus bas coefficient est sélectionné.

6. Algorithmes d'induction d'arbres de décisions

6.1. ID3: algorithme fondateur

ID3 est un algorithme non-incrémental d'induction d'arbres de décisions développé vers 1983 par Ross Quinlan [Ham 06]. C'est le premier algorithme populaire a avoir été employé pour les arbres de décisions.

L'approche employée par ID3 se limite à la construction d'un arbre de décisions sans élagage. L'arbre est construit de façon non-incrémentale descendante gloutonne et la mesure du gain d'information est utilisée pour le partitionnement des données.

L'algorithme original d'ID3 utilise un critère de division (gain d'information) qui est biaisé pour les attributs ayant le plus grand nombre de valeurs.

- ID3 ne supporte pas bien les attributs numériques.
- ID3 ne supporte pas bien les valeurs manquantes (qui sont traitées comme des valeurs ordinaires).
- ID3 produit des arbres pouvant coller trop près aux données d'entraînement à cause qu'il n'utilise pas de technique d'élagage.

6.2.C4.5: algorithme populaire

C4.5 est un algorithme non-incrémental d'induction d'arbres de décisions développé vers 1993 par le développeur d'ID3, Ross Quinlan [Ham 06].

C4.5 offre plusieurs améliorations à ID3 en résolvant les limites d'ID3 mentionnées plus haut:

- C4.5 utilise le ratio du gain d'information comme critère de division (ce qui est une amélioration au gain d'information utilisé par ID3).
- Contrairement à ID3, C4.5 supporte les attributs numériques.
- Contrairement à ID3, C4.5 traite les valeurs manquantes de façon spécifique.
- Contrairement à ID3, C4.5 utilise une technique d'élagage (post-élagage).
- Finalement, C4.5 peut également produire des règles à partir d'arbres de décisions.

Comme pour ID3, l'arbre est construit de façon non-incrémentale descendante gloutonne. C4.5 est l'un des algorithmes d'induction d'arbres de décisions les plus connus et les plus utilisés aujourd'hui.

6.3. ID5R : algorithme incrémental

ID5R est un algorithme incrémental d'induction d'arbres de décisions [Utgoff 89]. L'algorithme révise l'arbre courant au besoin en réponse à de nouvelles données; l'arbre est révisé simplement lorsque la nouvelle donnée n'est pas bien classifiée.

ID5R utilise ID3 et peut produire les mêmes arbres qu'ID3 pour les mêmes données d'apprentissage. Le processus global de construction d'arbre (et de branches) d'ID5 est le même que celui d'ID3: processus descendant glouton utilisant la mesure du gain d'information comme critère de sélection du meilleur attribut pour la division.

Pour la révision de l'arbre, ID5R maintient des informations pertinentes sur les données d'entraînement au niveau des noeuds et des feuilles de l'arbre. Avec ID5R, un noeud interne contient un attribut test (tout comme ID3) et l'ensemble des autres attributs non encore utilisés. Chacun de ces attributs est associé au nombre de valeurs possibles de cet attribut.

Pour les révisions de l'arbre, ces informations permettent à ID5R de recalculer les gains d'information et de déplacer des noeuds de façon appropriée sans à avoir à tout reconstruire l'arbre. ID5R utilise une technique de remontage (pull-up) permettant de déplacer un attribut vers le haut sans avoir à réexaminer les données tout en gardant l'arbre consistant avec celles-ci.

6.4.Salammbô: algorithme de logique floue

Salammbô [Mars et al 99] est un algorithme non-incrémental d'induction d'arbres de décisions basé sur la logique floue.

Cet algorithme a été développé afin de mieux traiter les données dont les attributs contiennent des éléments numériques (continus) puisque ces données se partitionnent souvent moins bien que les données symboliques (discrètes). Les arbres de décisions induits par les algorithmes présentés précédemment dans cette section utilisent des tests disjoints portant sur des valeurs bien définies pour partitionner les données: cela résulte en des données dont l'appartenance à une partition est définie de façon booléenne (une donnée appartient ou n'appartient pas à une partition).

Salammbô et d'autres algorithmes flous comparables induisent des tests flous portant sur des valeurs floues: cela résulte en des données dont l'appartenance à une partition est définie de façon floue (i.e. permettant l'appartenance à une ou plusieurs partitions à des degrés divers).

Salammbô supporte différents critères de division: un de ceux supporté est basé sur la métrique du gain d'information où les fréquences sont remplacées par des mesures floues. Cependant, Salammbô ne supporte pas l'élagage.

Marsala et Bouchon-Meunier (1999) comparent les résultats expérimentaux obtenus avec Salammbô et C4.5 sur deux petits corpus de données contenant des attributs numériques. Les résultats rapportés indiquent que Salammbô semble offrir une meilleure classification que C4.5 sur ces corpus avec des taux d'erreurs inférieurs.

6.5.Algorithmes probabilistes

Les algorithmes de logique floue pour induire des arbres de décisions tels que Salammbô sont comparables aux approches probabilistes [Qui 93].

La motivation des approches probabilistes est de pouvoir supporter le caractère incertain de la classification. Dans les approches qui ne sont ni probabilistes et ni floues, les données sont classifiées de façon disjointe. Cependant, l'appartenance à une classe peut être incertaine à cause de plusieurs facteurs: bruit dans les données, données manquantes, valeurs numériques près de seuils utilisés par les critères de divisions.

[Qui 93] propose une approche probabiliste permettant à une feuille de représenter plusieurs classes selon différentes probabilités et à une observation donnée de pouvoir être mise en correspondance avec plusieurs

feuilles: une combinaison des probabilités assignées par les feuilles indique la classe d'appartenance probable de la donnée.

Chaque classe d'appartenance est assignée avec 3 paramètres de probabilités dont une estimation inférieure, une estimation centrale et une estimation supérieure pour qu'une donnée appartienne à la classe.

7. Avantages et inconvénients des arbres de décisions :

7.1. Avantages des arbres de décisions

Les arbres de décisions offrent plusieurs avantages [Bouk 06]:

• Production de règles claires et explicites; les règles sont facilement compréhensibles par pour les personnes après peu d'explications.

• Indication explicite des attributs qui sont les plus pertinentes pour la classification.

• Facilité du calcul au moment de la classification; les algorithmes appliquant les arbres de décisions sont simples et nécessitent peu de calcul.

• Bonne efficacité dans les domaines basés sur des règles; la correspondance avec une base de règles est simple.

• Capacité de manipuler des variables numériques et/ou nominales; certains autres types de représentations ne traitent que d'un type d'attribut (numérique ou nominale);

• Dérivation d'arbres compacts avec une grande capacité de classification lorsque les données se prêtent au groupement.

• Facilitation du traitement robuste de grande quantité de données en peu de temps; certains algorithmes supportent le bruit.

• Peu de prétraitement de données requis; certains algorithmes supportent les données non-spécifiées et/ou le bruit.

7.2. Inconvénients des arbres de décisions

Les arbres de décisions sont aussi liés à quelques désavantages [Bouk 06]:

• Manque de performance s'il y a beaucoup de classes; les arbres peuvent être très complexes et ne sont pas nécessairement optimaux.

• Sensibilité au manque de données; les arbres peuvent trop coller aux données (data overfitting) lorsque celles-ci ne sont pas assez représentatives et l'élagage peu ne pas entièrement résoudre le problème.

• Moins bonnes performances pour les prédictions portant sur des valeurs numériques.

• La construction et élagage des arbres de décisions nécessitent généralement beaucoup de temps calcul: le choix du meilleur partitionnement lors de la construction et la comparaison de sous-arbres lors de l'élagage sont souvent coûteux.

• La combinaison de différents attributs pour les tests n'est pas toujours bien traitée automatiquement; la plupart des algorithmes ne traitent que d'un attribut à la fois.

8.technique des arbres de décision

8.1.Introduction

Un des grands problèmes pour l'informatique c'est de retirer la connaissance des données. Pour résoudre ce problème, on essaie de classifier des objets dans le monde réel. A partir de la classification, on peut apercevoir la relation parmi des objets, c'est à dire, on obtient la connaissance.

Le processus de retirer la connaissance des données est dit. Extraction de Connaissances à partir de Données.

Le domaine .Fouille de données Ayant pour but de retirer des modèles des données est un pas dans ce processus. Il y a plusieurs techniques pour la classification mais on peut les diviser en deux groupes : L'apprentissage supervisé. Et l'apprentissage non supervisé.

La caractéristique utilisée pour cette division est le fait que l'objet a ou n'a pas l'échantillon à apprendre avant de classifier.

L'arbre de décision est une des techniques du groupe « L'apprentissage supervisé ». Elle est très populaire et vraiment pratique. Dans cette section, nous allons cités les méthodes pour construire les arbres de décision.

8.2.Algorithme de construction des arbres de décision :

Pour chaque échantillon d'enseignement, évidemment, il y a plusieurs arbres correspondants. Donc, on préfère quel arbre ? La réponse c'est qu'on aime l'arbre ayant la taille la plus petite possible. Car plus la taille d'un arbre est petite, plus cet arbre contient les meilleurs attributs.

Les meilleurs attributs, ce sont les attributs qui bien divisent les données, donc ce sont les attributs importants, significatifs et pertinents. Autre raison vient du principe.

Le rasoir d'Occam. [PERDO 99]]. On peut penser que chaque arbre correspond à une hypothèse. Le principe d'Occam dit que l'hypothèse plus simple, c'est à dire l'arbre ayant la taille plus petite, est préféré.

8.2.1. ID3

L'algorithme ID3 est créé par Quinlan en 1986 [TOM 96]]. Il utilise entropy pour mesurer le désordre des données. Cette mesure vient du théorème de Shannon [Shannon 48].
Shannon l'utilise pour but d'estimer la taille minimale pour coder un message.

❖ Le critère de division est la mesure est Information obtenue.

(S un ensemble contenant n sous-ensembles différents)

$$\text{Entropy(S)} = \sum_{1}^{N} - prob(\text{Si} / S) * \log 2 \, \text{Pr}ob(\text{Si} / S)$$

prob(A,B) = Nbre Elément A / Nbre Elément B

$$\text{Informationobtenue(S,A)} = \text{Entropy(S)} - \sum prob(\text{S} / S) * Entropy(s)$$

8.2.2. C4.5

L'algorithme C4.5 est créé par Quinlan en 1993 [Qui 93]. Il est une amélioration de ID3 et utilise quelques techniques pour essayer de dépasser les difficultés que ID3 rencontre.

Traiter la partialité : Tout d'abord, pour éviter la partialité d'attribut ayant plusieurs Valeurs, C4.5 fournit une nouvelle mesure Proportion obtenue à la place de Information obtenue. Premièrement, on va définir la quantité Information de division.

Cette quantité a la même formule de Entropy sur l'ensemble S. Mais le point différent c'est que les sous-ensembles Si formées par l'attribut A.

- **Problèmes :**

C4.5 et ID3 rencontrent des problèmes suivants [Kiang] :

✓ Division diagonale : Car cet algorithme utilise les lignes orthogonales et parallèles avec les axes donc, le résultat de division comprend des rectangles. C'est la raison pour laquelle, il rencontre des difficultés lors de le résultat demande la division diagonale.

✓ Densité des exemples dans quelques régions : Si la densité des données dans quelques régions est base, ou la mission de classification est essentiellement la probabilité.

✓ Nombre des régions fragmentées : Certes, si le nombre des régions fragmentées est grand, alors on doit avoir plus des données pour le bon résultat.

8.2.3. CART

Le nom complet de CART est arbre de classification et régression. Il est créé par Leo Breiman en 1984 [3]. CART utilise seulement la division binaire et donc, il construit seulement des arbres binaires (chaque noeud a deux branches). CART a deux façons de traiter les données.

Si la variable de classification est numérique, alors on fait la régression. Par contre, on fait la classification. C'est la raison pour laquelle il a le nom .arbre de classification et régression. Ensuite, pour résoudre le problème de overfitting, la technique cout-complexite´ est utilisée. D'ailleurs, CART introduit une nouvelle mesure pour mesurer une division, c'est la mesure gini-index. En _n, pour traiter les attributs manquants, la technique division de subrogation est utilisée.

- **Classification**

Dans le cas où la variable de classification est catégorique, on fait la classification. Le mécanisme est de même fac¸on que ID3, mais il y a deux points différents :

1. Le critère de division est la mesure Gini-index.
2. La division est binaire.

$$\text{Gini(S)} = \sum 1 - \left(prob(\text{Si} / S)\right)^2$$

- **Coupures obliques**

Jusqu'à maintenant, j'ai déjà présenté deux types de division qui peuvent .être utilisés pour les variables numériques de classification :

1. la division de variable unaire : X<= c
   Les coupures sont parallèles aux axes.
2. la division de combinaison linéaire $\sum_i a_i * x_I <= c$
   Les coupures sont obliques.

- **Problèmes** Quelques problèmes avec CART :

1. Comme ID3, CART préfère les variables ayant plusieurs valeurs. Il va choisir ces variables au début.
2. Il fait la recherche exhaustive, donc le coût de calcul est trop élevé

8.2.4. CHAID

L'origine de cet algorithme est AID (détection automatique d'interaction) qui est Présenté par Morgan et Sonquist en 1963 [Morgan 63]. AID est une méthode multiple et linéaire régression. Pour que les méthodes multiples et linéaire régression soient Correctes, trois conditions suivantes sont supposées.

1. La relation entre la variable de classification et les autres est linéaire.
2. Il n'y a aucune interaction sauf celles sont inclues dans le modèle.
3. Le résidu satisfait une forte supposition d'indépendance, de normale distribution et de constante variance.

CHAID (détection automatique d'interaction de X^2) est une amélioration de AID. En 1980, Kass a décrit cet algorithme dans un journal statistique [Kass 80].

Un point différent par rapport au AID, c'est que la variable de classification est Catégorique. S'il y a quelques variables de prédiction qui sont numériques, alors il faut qu'elles sont catégorisées avant de traiter.

Autre point différent, c'est que CHAID essaie de maximiser la diminution de signification de X^2. Le dernier point c'est que CHAID peut avoir des divisions multiples au lieu des divisions binaires.

$$\mathbf{X^2 = \sum (observé - souhaité)^2 / souhaité}$$

8.2.5. QUEST

QUEST s'appelle .Rapide, Impartial, Efficace, Statistique Arbre.. En 1997, Loh et Shih ont présenté cet algorithme dans le journal Statistica Sinica [WYin 97].

Les algorithmes qui utilisent la recherche exhaustive rencontrent toujours deux difficultés :

- .Le temps de calcul est trop élevé.
- .La division choisie est partiale.

QUEST est un effort de surmonter ces difficultés. Les notions principales de ce algorithme viennent de l'algorithme FACT [Vanich 86] ; [Vanich 88] . Dans beaucoup des algorithmes, la forme de division X <= c est trop utilisée. On cherche souvent la variable à diviser X et le point de division c en même temps. Mais l'approche de FACT est différente. FACT les traite séparément.

- Les variables catégoriques sont converties en variables numériques.
- à chaque noeud, on choisit la variable qui a la plus grande quantité F_statistique.
- Le point de division est trouvé par l'analyse de linéaire discrimination.

Un point important, c'est que FACT ne subit pas la division partiale si toutes les variables sont numériques. Ensuite, FACT utilise une technique pre-élagage, donc il obtient moins d'efficacité que CART qui utilise une technique post-élagage. Par rapport au FACT, QUEST a des points différents :

- Sélection de variable est négligeablement partiale.
- Posséder la simplicité de calcul de FACT.
- Permettre d'élaguer.
- Division binaire.

8.2.6. LMTD

LMTD est le nom abréviatif de Machine linéaire des arbres de décision. Il est présenté en 1991 par l'article de Utgoff et Brodley [Paul 91]. Les algorithmes classiques utilisent souvent les divisions unaires, c'est à dire, Chaque test à un noeud a seulement une variable dans la formule. Quelques fois, la relation existant parmi les exemples dépend de la combinaison de quelques variables. Dans ce cas, on a besoin d'un algorithme qui permet de découvrir cette combinaison. Cet algorithme doit .être capable de traiter les variables numériques et aussi les variables catégoriques.

Il peut traiter également les exemples qui sont imprécis ou dont la classe n'est pas correcte. LMTD a pour but de surmonter ces difficultés.

D'ailleurs, un test linéaire peut représenter les régions qui ne sont pas orthogonales. Plus exactement, une machine linéaire va diviser les exemples en des régions convexes.

L'algorithme LMTD construit les arbres de même façon d'autres algorithmes. Mais, au lieu de choisir un test unaire par une mesure comme Information obtenue, il forme une machine linéaire qui va être utilisée comme un test multi-variable à chaque noeud..

8.2.7. SLIQ

SLIQ est le nom abréviatif de Apprentissage supervisé dans QUEST [Manish 96]. Quest est un projet d'Extraction de Connaissances à partir de Données au centre de recherche Almaden de IBM. Au contraire des autres algorithmes qui lisent toutes les données à la mémoire d'ordinateur, SLIQ maintient seulement une partie des données à la mémoire, donc il peut traiter des données énormes qui peuvent avoir millions des exemples. Il peut traiter également les variables numériques et catégoriques.

Tout d'abord, SLIQ utilise une technique de trier avant pour réduire le coût d'évaluation des valeurs numériques.

Ensuite, un rapide algorithme de sous-ensemble est appliqué pour déterminer la division pour les variables catégoriques. Le critère utilisé ***Giniindex*** et une nouvelle méthode d'élagage basée sur le principe Taille minimale de description.

8.2.8. SPRINT

Le nom complet de SPRINT [Shafer 96] est Scalable Parallelizable Induction of Decision Trees en anglais. C'est un autre algorithme développé au centre de recherche Almaden de IBM. On peut dire que cet algorithme est une extension de SLIQ. Expérience dit que le temps perdu lors de construction des arbres est dans la phase de grandir. Donc, SPRINT ne développe pas une nouvelle technique pour élaguer, il utilise la technique de SLIQ. Par rapport à SLIQ, il y a deux points avancés :

1. SLIQ doit contenir une liste de classe dans la mémoire. Plus la taille de données augmente, plus celle de cette liste augmente. C'est une limite de SLIQ.
2. C'est facile à paralléliser SPRINT.

9.Elagage des arbres de décision

9.1. Introduction

On a vu Lors de la construction de l'arbre de décision que ça taille grandit de manière linéaire avec la taille de la base d'apprentissage, et lorsque l'ensemble d'apprentissage contient des données bruitées, ou qu'il ne contient pas certains exemples importants cause des erreurs dans l'arbre de décision.

Cependant ce problème est résolu par l'étape d'élagage qui consiste à supprimer les sous arbres superflus ou trop liés aux données, dans le but d'améliorer l'aspect prédictif de l'arbre d'une part, et réduire sa complexité d'autre part, Plusieurs techniques d'élagage [Mingers,1989] ont été proposées pour éviter le sur Apprentissage. On distingue deux approches principales : Le pré élagage et le post-élagage.

9.2.Pré Elagage :

Une solution simple consiste à cesser de diviser un nœud quand la puretés des points qu'il domine est non pas parfaite, mais suffisante .une fois sélectionné le meilleur attribut, on regarde si la valeur du critère de la division est inférieure à un certain seuil ; en pratique, ceci revient à admettre que, s'il existe une classe suffisante majoritaire sous un noeud, on peut considérer ce dernier comme une feuille et lui attribue la classe en question, selon le caractère de division utilisé.

Diverses heuristiques ont été proposées pour régler le seuil précédent, sa valeur peut d'ailleurs être variable selon le nœud où l'on se trouve ,dépendant de l'estimation de la probabilité a priori des classes,de l'estimation empirique de la difficulté à les séparer, etc.

Quinlan propose une technique appelé Chi-square pruning, citée dans [Crémilleux,1991, Quinlan, 1993, Kononenko, 1998] utilise le test du chi-deux pour déterminer si l'attribut sélectionné est pertinent ou non pour construire le sous-arbre. Autrement dit, on arrête le processus de construction sur un nœud lorsque tous les attributs candidats sont jugés indépendants de la classe. Le résultat de cette technique n'est pas tellement satisfaisant.

[Cestnik et al., 1987] et [Kononenko et al., 1984] utilisent des paramètres ad-hoc avec des seuils empiriques pour stopper la construction d'un sous-arbre si le nombre d'instances, l'estimation du meilleur attribut ou le taux d'erreurs de classement dans l'ensemble d'apprentissage sur le nœud courant devient assez petit.

Une autre idée a été introduite par [Bratko et Knononenko, 1987] qui vise à partitionner plusieurs fois l'ensemble d'instances associé au nœud courant en deux parties : une pour l'apprentissage et l'autre pour le test pour construire un seul niveau pour chaque partitionnement.

Ensuite la base de test est classée dans le sous-arbre construit dont la profondeur est 1 et le taux d'erreurs de classement est calculé. Ce processus est répété pour chaque partitionnement. Si le taux moyen d'erreur dans le sous-arbre est plus grand que le taux d'erreur dans le nœud courant, alors ce nœud devient une feuille et la construction de l'arbre est ainsi stoppée pour ce noeud.

Ces méthode présentent certains inconvénients, dont le principal est sont myopes (puisqu'elles ne prennent en compte qu'un caractère local à la feuille examinée), et peuvent de ce fait manquer un développement de l'arbre qu serait excellent, c'est pourquoi on leur préfère souvent des méthode d'élagage posteriori, une que l'arbre a été entièrement développé.

9.3.Poste Elagage :

Plusieurs méthodes d'élagage existent dans littérature telles que :

9.3.1. Méthode du coût complexité minimal (MCCP) :

L'idée qui a été introduite par [Breiman et al. 1984] est de construire une séquence d'arbres en utilisant une formulation dite complexité de coût minimale.

En fonction du coût de mauvaise classification et la taille de l'arbre, on dérive un ordre des arbres de complexité décroissante, commençant de l'arbre complet.

Cette séquence est récursivement crée en sélectionnant le dernier arbre dans la séquence (initialement, l'arbre complet) ; examinant chacun de ses sous arbres, et sélectionnant celui avec la moindre métrique de complexité de coût et faisant celui-ci le prochain sous arbre dans la séquence. Le processus s'arrête quand le sou arbre final est juste le noeud racine.

Une fois cette séquence est produite, on fait le choix d'un arbre optimal en fonction du critère suivant : on calcule l'erreur apparente (estimation de l'erreur réelle) de chaque sous arbre, celui qui minimise cette mesure, est choisi comme modèle final.

9.3.2. Méthode de l'erreur réduite (REP):

Au lieu de construire une séquence d'arbres puis choisir le meilleur d'entre eux comme dans la méthode précédente, Quillan a suggéré une méthode plus directe [Quillan, 1987b].

Cette méthode Consiste à estimer l'erreur réelle d'un sous arbre donné sur un ensemble d'élagage ou de test. Plus algorithmiquement, l'élagage se fait comme suit : Tant qu'il existe un arbre que l'on peut remplacer par une feuille sans faire croître l'estimation de l'erreur réelle alors élaguer cet arbre.

Cette technique donne un arbre légèrement 'conforme' dans le sens où certains exemples seront peut être mal classifiés.

9.3.3. Méthode de l'erreur pessimiste (PEP):

Afin de pallier les inconvénients de la méthode précédente, QUINLAN [Quillan, 1987] a proposé une stratégie d'élagage qui fait le recours à un seul ensemble de construction et d'élagage de l'arbre.

L'arbre est élagué en examinant le taux d'erreur à chaque noeud et assumant que le vrai taux d'erreur est considérablement pire. Si un noeud donné contient N enregistrements dans lesquels E parmi eux, sont mal classifiés, alors le taux d'erreur est estimé à E/N. Tous les sous-arbres sont examinés seulement une seule fois pour savoir s'ils

Doivent être élagués ou non. Cependant, les sous-sous-arbres des sous-arbres élagués ne sont jamais testés. Cette méthode est plus rapide que les autres parce qu'elle teste chaque sous-arbre une seule fois.

9.3.4. L'élagage basé sur le principe MDL

Le principe MDL (Minimum Description Length) [Rissanen, 1984] consiste à choisir parmi plusieurs théories celle dont le codage est minimal.

Ce principe est une approche naturelle pour éviter le problème d'overfitting. L'idée de base est qu'un sous-arbre est élagué si la longueur de description du classement de la base d'apprentissage dans tout l'arbre, augmentée de la Description Length de l'arbre, est supérieure à celle qu'on obtient si le sous-arbre est élagué.

Plusieurs méthodes ont été proposées pour élaguer un arbre de décision en utilisant ce principe. Nous trouvons ces travaux dans [Quinlan et Rivest, 1989].

9.3.5. Minimum error pruning

Niblett et Bratko [Niblett et Bratko., 1986] ont proposé une méthode pour trouver un seul arbre élagué qui devrait donner un taux d'erreur minimum lorsqu'on l'utilise pour classer des nouveaux cas. Le principe est Sur chaque noeud (non feuille) dans l'arbre, Niblett et Bratko calculent le taux d'erreur attendu (l'erreur statique) si le sous-arbre est élagué et remplacé par une feuille. Ensuite ils calculent le taux d'erreur attendu si le sous-arbre n'est pas élagué en utilisant le taux d'erreur sur chaque branche ; cette erreur, appelée erreur dynamique, est la moyenne pondérée de l'erreur de tous ses fils. Cette procédure est récursive. Si l'élagage d'un noeud conduit à une erreur statique supérieure à l'erreur dynamique de ce noeud, le noeud n'est pas élagué. Dans le cas contraire, quand l'erreur dynamique d'un noeud est supérieure à son erreur statique le noeud est élagué et remplacé par une feuille.

9.3.6. Méthode de la valeur critique

La stratégie d'élagage suivie dans cette méthode proposée par [Mingers, 1987, Mingers, 1989] consiste à spécifier une valeur critique et élaguer les noeuds qui n'arrivent pas à cette valeur, c'est-à-dire les noeuds dont la valeur de la mesure utilisée pour sélectionner l'attribut noeud ne dépasse pas la valeur critique. Cette méthode génère une séquence d'arbres élagués en augmentant la valeur critique.

Un seul arbre est choisi, de la même manière que dans la méthode du coût-complexité minimal. La valeur critique choisie dépend de la mesure utilisée pendant la construction de l'arbre. Plus la valeur critique est élevée, plus l'arbre élagué est petit.

Il peut arriver que les conditions d'élagage soient vérifiées par un noeud t mais pas par tous ses fils. Dans ce cas, on garde la branche T_t parce qu'elle contient des noeuds pertinents.

9.3.7. Error-based pruning :

L'idée est de cette méthode est la suivante : si N est le nombre d'instances parvenues à une feuille et E le nombre d'objets mal classés sur cette feuille, alors le taux d'erreur optimiste observé est E/N (observer E événements dans N essais).

Pour trouver la probabilité qu'une erreur se produise dans l'ensemble de cas couverts par la feuille, Quinlan a utilisé les limites de confiance de la distribution binomiale. Pour un niveau de confiance cf, on calcule la limite supérieure de cette probabilité U_{cf} (E,N) en utilisant les limites de confiance de la distribution binomiale.

C4.5 considère que Le taux d'erreur prédit sur une feuille (la probabilité qu'une erreur se produise) est égal à cette borne U_{cf} (E,N) où le niveau de confiance donné par Quinlan est 25%. Pour décider si on élague un sous-arbre ou pas on doit d'abord :

- Calculer le taux d'erreur prédit sur chaque feuille de ce sous-arbre en utilisant la limite supérieure de confiance de la distribution binomiale : **$U_{25\%}$(E;N).**
- Calculer le nombre d'erreurs prédites sur chaque feuille, qui est égal au nombre de cas sur cette feuille _ le taux d'erreur prédit : **N * $U_{25\%}$ (E,N).**
- Calculer le nombre total d'erreurs prédites sur toutes les feuilles de toutes les erreurs des feuilles) :

$$\sum N \times U_{25\%}(E, N).$$

- Calculer le nombre d'erreurs prédites sur le noeud père de ces feuilles ; qui est de cas sur ce noeud multiplié par le taux d'erreur prédit sur ce noeud.

- ❖ Comparer les valeurs obtenues lors des deux étapes précédentes pour prendre une concernant l'élagage du sous-arbre : si l'erreur donnée par le noeud père est inf l'erreur donnée par les feuilles alors on élague le sous-arbre de ce noeud père.

9.3.8. Indice de qualité

La plupart des méthodes d'élagage minimisent le taux d'erreur de classement [Crémilleux,2000]. L'objectif de cette méthode est de ne pas élaguer systématiquement les sous-arbres ayant un taux d'erreur égal au taux d'erreur de leur racine. [Fournier et Crémilleux, 2000] ont proposé une méthode d'élagage, appelée DI pruning, qui prend en compte la complexité des sous-arbres, et qui est capable de garder les sous-arbres avec leurs feuilles pour extraire des règles de décision pertinentes, même si cela n'améliore pas le taux de bon classement.

Cette méthode définit un indice de qualité qui est un compromis entre la profondeur de l'arbre et son impureté.

10. Règles issues des arbres de décision :

Un arbre de décision, une fois construit, peut être converti en un ensemble de règles équivalent, où chaque chemin, du noeud racine au noeud feuille, sera exprimé par une règle, en enregistrant les tests comme antécédents (prémisses), et la classe du noeud feuille comme conséquent (conclusion).

Ces règles doivent être par la suite, simplifiées. La conversion d'un arbre de décision en un ensemble de règles avant l'élagage, a les avantages suivants [Win92] :

- Elle permet de distinguer parmi les différents contextes, dans quel cas un noeud de décision est utilisé.

Puisque chaque chemin produit une règle distincte, la décision d'élagage concernant un test peut être prise différemment pour chaque chemin.

En revanche, si l'arbre celui même est élagué, les deux seuls choix seraient d'enlever le noeud de décision complètement, ou le maintenir tel qu'il est.

- Elle enlève la distinction entre les tests d'attributs qui se produisent près de la racine et les autres, qui se produisent près des feuilles.
- Elle améliore la lisibilité sans perte d'information.

11. Implémentation des arbres de décision

11.1. Introduction

Dans cette partie on va dérouler la construction d'un arbre de décision à partir d'un petit exemple, nous présentons la méthode C4.5 qui permet de répondre de manière cohérente à ces spécifications. Nous la mettons alors en oeuvre en utilisant notre Logiciel TreeClassifier développé à notre niveau qui permet la construction, la classification, l'élagage et la génération des règles de productions.

11.2. Exemple de base de donnés '' Risque Routier'' :

11.2.1. Construction de l'arbre de décision :

Le principe de construction des arbres des décisions est de choisir le meilleur critère de division et/ou de segmentation des attributs pour avoir un partitionner des individus que l'on représente sous la forme d'un arbre de décision. L'objectif est de produire des groupes d'individus les plus homogènes possibles du point de vue de la variable à prédire. Il est d'usage de représenter la distribution empirique de l'attribut à prédire sur chaque sommet (noeud) de l'arbre.

Pour plus appréhender, nous allons prendre et dérouler un exemple choisi arbitrairement ces données sont alimentées d'une façon aléatoire d'un coté et on même temps on a respectés une homogénéités des données. Le fichier est composé de 21 observations, il s'agit d'expliquer le comportement des individus par

rapport à la présence d'un risque routier {Risque, non Risque} à partir des prévisions météorologiques, type de Route, Age du Chauffeur,et en dernier le type de véhicule (voir le tableau suivant).

ordre	Outlo	Type	Age	Route	Risqu
1	Rain	camio	20	Depart	Oui
2	suny	camio	45	Depart	Non
3	Rain	camio	32	Auto	Non
4	suny	camio	18	Natio	Oui
5	suny	camio	32	Natio	Oui
6	Rain	camio	20	Natio	Oui
7	suny	camio	68	Natio	Non
8	suny	camio	45	Auto	Non
9	Rain	camio	20	Natio	Oui
10	Rain	camio	32	Depart	Oui
11	Rain	camio	45	Natio	Non
12	Rain	Famili	68	Auto	Non
13	suny	Famili	18	Auto	Non
14	Rain	Famili	18	Depart	Oui
15	Rain	Famili	68	Depart	Oui
16	suny	Famili	20	Auto	Non
17	suny	Sport	18	Natio	Oui
18	Rain	Sport	18	Auto	Oui
19	Rain	Sport	18	Depart	Oui
20	Rain	Sport	45	Depart	Oui
21	Rain	Sport	25	Depart	Oui

Figure 14 : Tableau Echantillon Statistique Risque routier

L'arbre de décision correspondant est décrit ci-dessous (Figure 13).

- Le premier sommet est appelé la « racine » de l'arbre. Il est situé sur le premier niveau. Nous y observons la distribution de fréquence de la variable à prédire « Risque ». Nous constatons qu'il y a bien 21 observations, dont 13 « oui » (il y a un risque) et 5 « non ».

- La variable « Route » est la première variable utilisée ; on parle de variable de segmentation. Comme elle est composée de 3 modalités {Auto, Départemental, National}, elle produit donc 3 sommets enfants.

- La première arête (la première branche), à gauche, sur le deuxième niveau, est produite à partir de la modalité « Auto » de la variable « Route ». Le sommet qui en résulte couvre 6 observations correspondant aux individus {3,8,12,13,16,18}, la distribution de fréquence nous indique qu'il y a 5 « Risque = Non » et 1 « Risque = oui ».

- La seconde arête, au centre, correspond à la modalité « Depart. » de la variable de segmentation « Route » ; le sommet correspondant couvre 8 observations correspondant aux individus {1,2,10,14,15,19,20,21}, la distribution de fréquence nous indique qu'il y a 1 « Risque = Non » et 7 « Risque = oui ».

- La troisième arête, (la dernière branche), à droite, sur le deuxième niveau, est produite à partir de la modalité « National » de la variable « Route ». Le sommet qui en résulte couvre 7 observations, la distribution de fréquence nous indique qu'il y a 5 « Risque = Non » et 2 « Risque = oui ».

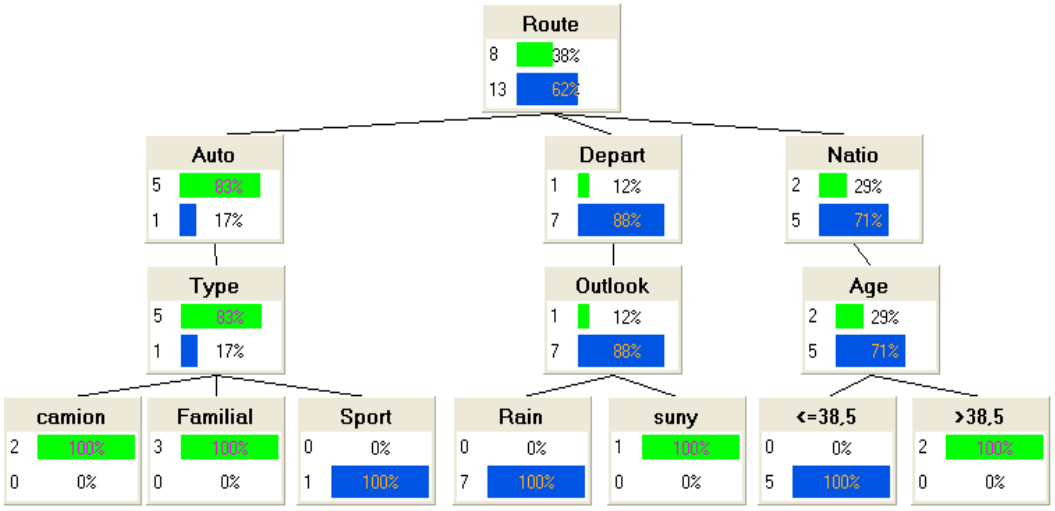


Figure 14 Arbre de Décision ‘‘risque routier’’

11.2.2. Choix d’une variable de segmentation :

La plupart des méthodes d’induction d’arbres s’appuient sur le même procédé, pour chaque variable candidate, nous réalisons le partitionnement des observations et nous calculons un indicateur de qualité ; la variable retenue sera alors celle qui optimise cet indicateur. Les méthodes diffèrent selon la mesure utilisée.

11.2.3. Traitement des variables continues

Plaçons nous maintenant sur le sommet le plus à Droite sur le 3ème niveau de l’arbre. Il couvre 7 individus et a été segmenté à l’aide de la variable « Age », le seuil de coupure utilisé étant « 38.5 ». Ce résultat est la conséquence de deux tâches élémentaires :

1. Sélectionner la meilleure valeur de coupure pour chaque variable continue ;
2. Sélectionner globalement la meilleure segmentation en comparant la pertinence de tous les descripteurs : les descripteurs discrets et les descripteurs continus qui ont été découpés en 2 intervalles.

11.2.4. Choix du point de coupure :

La première opération consiste à déterminer le meilleur point de coupure pour les variables continues. Nous considérons le cas du découpage binaire. Ceci n’est pas limitatif dans la mesure où il est possible de reconsidérer la même variable sur un sommet situé plus bas dans l’arbre et initier une autre discrétisation avec une valeur seuil différente.

Le choix du seuil de discrétisation doit être cohérent avec la procédure de sélection des variables de segmentation ; il paraît donc naturel de faire intervenir dans le choix de la borne le gainRatio de C4.5 qui sert à évaluer les partitions.

Le procédé est alors relativement simple pour un descripteur continu X : il s’agit dans un premier temps de trier les données selon les valeurs de X, puis tester chaque borne de coupure possible entre deux valeurs de la variable en calculant le gainRatio de C4.5

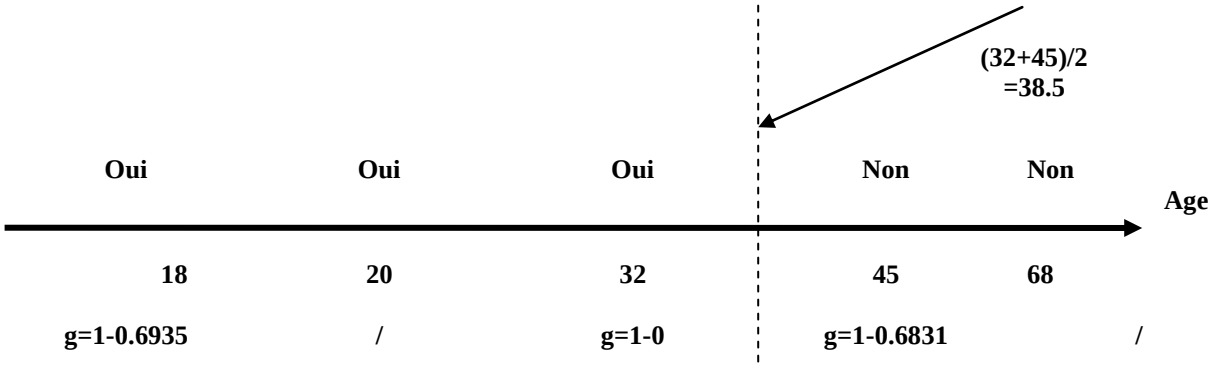


Figure 16 Choix de point de coupure

11.3. Décision :

Le processus de génération des règles de production est la phase la plus simple dans les arbres de décisions, chaque chemin reliant une feuille à la racine de l'arbre peut être lu comme une règle du type attribut valeur « Si prémisse… alors Conclusion… ».

Lorsque la feuille est pure, en lui attribuer la conclusion correspond à la seule modalité, et lorsque la feuille contient plus d'une 1 modalité en lui attribuer la conclusion correspondant à la modalité qui présente l'effectif le plus grand, Cette règle, qui semble de bon sens, repose sur des fondements théoriques bien établis. En effet, la distribution de fréquences visible sur la feuille est une estimation de la probabilité conditionnelle d'appartenance à chaque étiquette de la variable à prédire.

Dans l'exemple, toutes les feuilles étant pures, nous pouvons très facilement déduire 7 règles (Tableau 2).

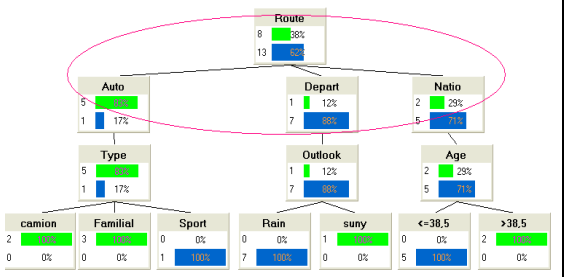
0	*SI Route IN Auto AND Type IN camion ALORS NON*
0	*SI Route IN Auto AND Type IN Familial ALORS NON*
0	*SI Route IN Auto AND Type IN Sport ALORS OUI*
0	*SI Route IN Depart AND Outlook IN Rain ALORS OUI*
0	*SI Route IN Depart AND Outlook IN suny ALORS NON*
0	*SI Route IN Natio AND Age IN <=38,5 ALORS OUI*
0	*SI Route IN Natio AND Age IN >38,5 ALORS NON*

Figure 17 : Tableau des règles de production

11.4. ELAGAGE :

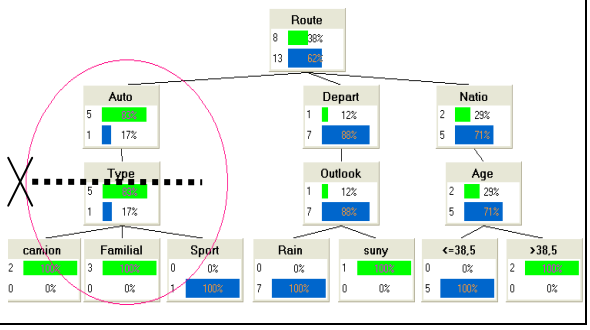
La méthode appliquer est : Error-based pruning de Quinlan qui utilisé les limites de confiance de la distribution binomiale. Pour un niveau de confiance ***cf25%*** (voir Chapitre3).

❖ 1ier Niveau :

Nœud	C1 – C2	N	Alpha	ERREUR	Arbre
Route	8 - 13	21	0,47865748	**10,0518072**	
1-Auto	5 - 1	6	0,38947964	2,33687782	
2-Depart	1 - 7	8	0,3026998	2,42159843	
3-Natio	2 - 5	7	0,48609686	3,40267801	
			TOTAL E. fils	**8,16115427**	

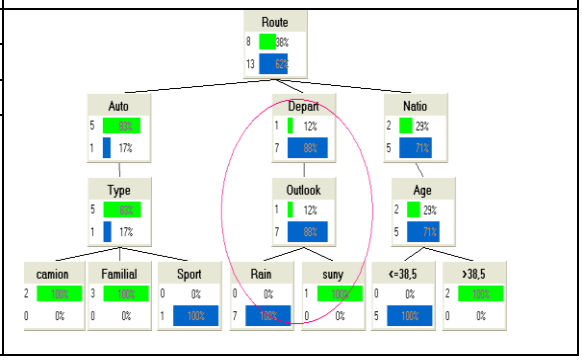
=>Décision Non Elaguer

❖ 2em Niveau / 1ier Arête à gauche:

Nœud	C1 – C2	N	Alpha	ERREUR	Arbre
Auto	5 - 1	6	0,38947964	**2,33687782**	
1-Camion	2 - 0	2	0,5	1	
2-Familial	3 - 0	3	0,37003946	1,11011839	
3-Sport	0 - 1	1	0,75	0,75	
			TOTAL E. fils	**2,86011839**	

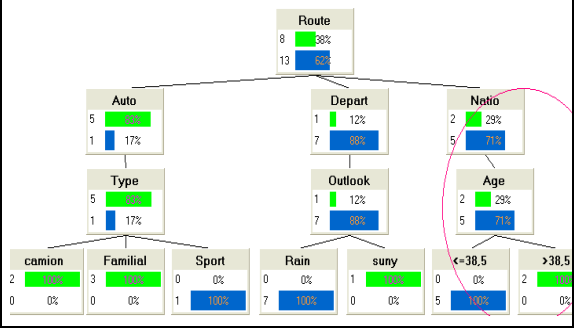
=> Décision Elaguer

❖ **2eme Niveau 2eme Arête à gauche:**

Nœud	C1 – C2	N	Alpha	ERREUR	Arbre
Depart	1 7	8	0,3026998	**2,42159843**	
1-Rain	0 7	7	0,17966461	1,25765228	
2-Suny	1 0	1	0,75	0,75	
			TOTAL E. fils	**2,00765228**	

=> Décision Non Elaguer

❖ **2eme Niveau 3eme Arête à gauche:**

Nœud	**C1 – C2**	**N**	**Alpha**	**ERREUR**	**Arbre**
National	2 5	7	0,48609686	**3,40267801**	
Age>38.5	0 5	5	0,242141724	1,21070862	
Age<=38.5	2 0	2	0,5	1	
			TOTAL E. fils	**2,21070862**	

=> Décision Non Elaguer

11.5. Résultat d'arbre Elagué :

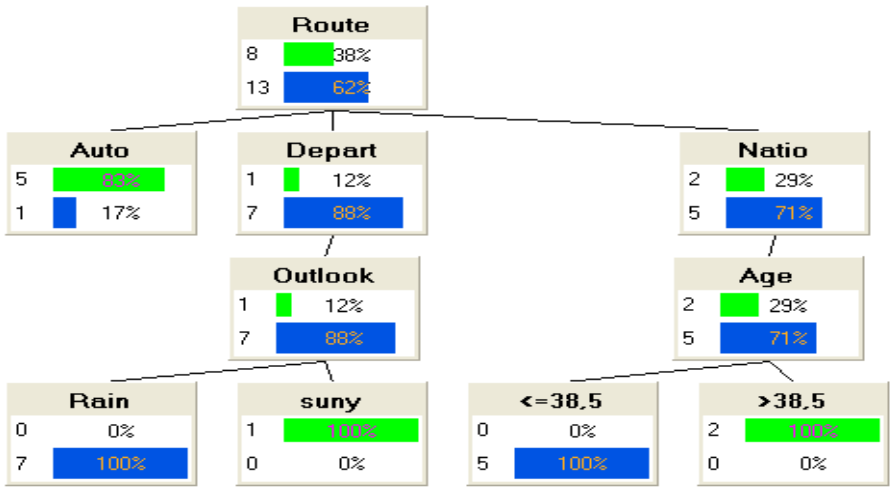


Figure 18 Arbre de Décision ''risque routier'' élagué

12. Conclusion

Nous avons présenté dans ce chapitre une méthode d'apprentissage automatique qui est une des plusieurs méthodes de systèmes d'apprentissage. Nous avons montrés les différents algorithmes de constructions, la procédure d'élagage, les avantages et les désavantages de chaque.

Les arbres de décision restent très importantes et sont largement utilisées. Elles font partie de la classe des méthodes dites non paramétriques et sont issues de l'Intelligence artificielle. Les méthodes basées sur les arbres de décision sont des méthodes dites symboliques (la procédure produite sous forme de règles), Il existe d'autres méthodes que celles présentées et les méthodes présentées possèdent de nombreuses variantes.

Si l'on souhaite que les procédures de classification puissent être compréhensibles par l'utilisateur, on s'orientera vers les méthodes à base d'arbres de décision. Sinon le choix reste ouvert (exemple : réseaux de neurones.). Ces derniers sont des méthodes dites adaptatives (on procède par ajustement de coefficients) et la procédure produite est de type ``boîte noire" (résultats non compréhensibles par l'utilisateur).

Chapitre 4

Présentation & Modélisation

1.Introduction

Le transfert de connaissance d'un expert en vue d'être traitée par une machine est une opération très délicate, c'est d'ailleurs pour cela que plusieurs formalismes de modélisation de cette connaissances, c'est trouver une structure globale qui articule et assure la cohésion des connaissances traitées en vue de résoudre une classe de problèmes dans un domaine donné.

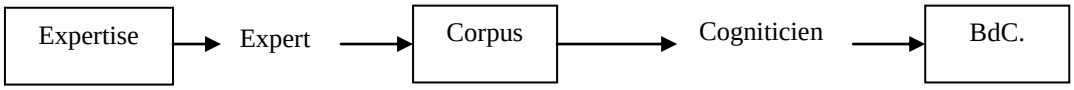


Figure 19 Cycle de Modélisation d'une Base de connaissance

- Corpus : description contextuelle en langage naturelle d'une expertise quelconque.
- BdC. : projection d'une expertise dans un formalisme appelé le langage de représentation de connaissances (LRC).

2.Le langage UML

En terme d'analyse et de modélisation objet, UML (Unified Modeling Language) est aujourd'hui un standard incontournable, UML est un langage standard conçu pour l'écriture de plans d'élaboration de logiciels.

Il peut être utilisé pour visualiser, spécifier, construire et documenter les artefacts d'un système à forte composante logicielle.

UML est adapté à la modélisation de système, depuis les systèmes informatiques d'entreprise jusqu'aux applications distribuées basées sur le web, en passant par les systèmes temps réel embarqués. C'est un langage très expressif qui couvre toutes perspectives nécessaires au développement puis au déploiement des systèmes.

La notation UML [OMG 77] constitue une étape importante dans la convergence des notations utilisées dans le domaine de l'analyse et la conception objet puisqu'elle représente une synthèse des méthodes les plus utilisées : OMT (Object Modeling Technique), Booch et OOSE (Object-Oriented Software Engineering) [94]. Depuis la première version d'UML, le standard de l'OMG (Object Management Groupe) n'a pas cessé d'évoluer.

Cependant la refonte majeure d'UML est le standard UML2.0 adopté en août 2003. UML2.0 représente un pas réel pour supporter la croissance des logiciels actuels, d'une part par le support de la nouvelle vision de l'OMG à savoir MDA (Model Driven Architecture), et d'autre part par le support des nouvelles technologies en particulier l'approche par composants logiciels.

Depuis ses premières versions, le standard UML est caractérisé par sa sémantique définie par une approche de méta-modélisation. Un méta-modèle est la définition des constructions et des règles de création des modèles.

2.1. Les diagrammes

La notation UML est décrite sous forme d'un ensemble de diagrammes. La première génération d'UML (UML1.x), dé_nit neuf diagrammes pour la documentation et la spécification des logiciels. Dans UML2.0 Superstructure , quatre nouveaux diagrammes ont été ajoutés : il s'agit des diagrammes de structure composite (Composite structure diagrams), les diagrammes de paquetages (Packages diagrams), les diagrammes de vue d'ensemble d'interaction (Interaction overview diagrams) et les diagrammes de synchronisation (Timing diagrams). Ils sont regroupés dans deux classes principales :

- **Diagrammes statiques** : ils regroupent les diagrammes de classes, les diagrammes d'objets, les diagrammes de structure composite, les diagrammes de composants, les diagrammes de déploiement, et les diagrammes de paquetages.

- **Diagrammes dynamiques :** les diagrammes de séquence, les diagrammes de communication (nouvelle appellation des diagrammes de collaboration d'UML1.x), les diagrammes d'activités, les machines à états, les diagrammes de vue d'ensemble d'interaction, et les diagrammes de synchronisation.

Dans notre cas, nous nous sommes intéressés à quatre types de diagrammes : les diagrammes de cas d'utilisation, de classes, de séquence, et d'activité. Nous décrivons brièvement ces diagrammes. Les diagrammes de cas d'utilisation, de séquence, et d'activité sont choisis pour leur possible utilisation pour la conception de la chorégraphie et de l'orchestration des services et les diagrammes de classes pour leur possibilité de représenter la structure interne.

2.1.1. Diagrammes des cas d'utilisation

Les cas d'utilisation (use cases) représentent l'apport principal de Ivar Jaccobson et sa méthode OOSE à UML. L'objectif des cas d'utilisation est l'expression des besoins en termes de services que doit assurer le système.

Les diagrammes de cas d'utilisation dans UML définissent deux concepts principaux : les acteurs, et les cas d'utilisation. Un acteur est une entité extérieure au système qui peut initier l'un de ses cas d'utilisation. Un cas d'utilisation est une fonctionnalité offerte par le système.

2.1.2. Diagrammes des classes

Le diagramme de classes permet de définir la structure de toutes les classes qui constituent un système. Une classe nommée est définie par des attributs et des méthodes.
Les classes constituent les briques de base les plus importantes d'un système orienté objet.
Une classe est la description d'un ensemble d'objets qui partagent les mêmes attributs, les mêmes opérations, les mêmes relations et la même sémantique. Une classe implémente une ou plusieurs interfaces.

L'intérêt des diagrammes des classes sont de rassembler les données utilisées par le système dans des entités (classes).
Définir les opérations qui permettent de manipuler ces données (uniquement que lorsque qu'elles sont nécessaires).

De réaliser une vision des éléments statiques du système, c'est à dire de recenser les parties des structures qui ne se modifieront pas avec le temps.
Mettre en œuvre les concepts objets (héritage).

2.1.3. Diagrammes de séquences

Les diagrammes de séquences montrent les interactions qui surviennent dans une séquence de temps. En particulier ils montrent la participation d'objets dans les interactions et les messages qu'ils échangent dans un intervalle de temps. Ils ne montrent pas les associations entre les objets.
Un diagramme de séquence a les caractéristiques suivantes.

- Une dimension verticale qui représente le temps, une dimension horizontale qui représente différents objets.
- Le temps s'écoule de haut en bas et de gauche à droite.
- Les messages échangés entre objets sont représentés horizontalement par une ligne terminée par une flèche.

Les diagrammes de séquences Permettre de mieux comprendre le fonctionnement du système, de modéliser la vie des objets dans le temps et leur chronologie, et de représenter les interactions, les communications (par messages) entre objets.

2.1.4. Les diagrammes d'activités

D'après les diagrammes d'activités font partie des cinq diagrammes d'UML utilisés pour la modélisation des aspects dynamiques des systèmes. Un diagramme d'activités est principalement un organigramme qui montre le flot de contrôle d'une activité à l'autre. Ils sont utilisés pour modéliser les aspects dynamiques d'un système. Il s'agit de la modélisation des étapes séquentielles et/ou concurrentes dans un processus.

2.2. Les Avantages d'UML

Les principaux points forts de UML sont :

- UML est un langage formel et normalisé, il permet un gain de précision et un langage de stabilité.
  ce qui encourage l'utilisation d'outils.
- UML est un support de communication performant, il cadre l'analyse et il facilite la compréhension de représentations abstraite complexes.
- UML est polyvalent et sa souplesse en font un langage universel.
- Sa notation graphique permet d'exprimer visuellement une solution objet.
- L'aspect formel de sa notation limite les ambiguïtés et les incompréhensions.
- Son aspect visuel facilite la comparaison et l'évaluation de solutions.

3. Le langage de représentation des connaissances ExpertRule

Notre langage de représentation est un langage déclaratif utilisant le formalisme objet avec les règles de production (Langage Hybride) qui permet de représenter deux types de connaissances :

3.1. Connaissances descriptives :

Un fait est représenté sous forme de liste de couples (attribut, valeur), ou attribut désigne une entité du domaine étudié.

Les modèles sont organisés sous forme de classe afin d'inclure un nombre important d'information de caractères prototypiques utiles à l'usager.

3.2. Connaissances Opératoires :

La base de règle est constitue par un ensemble de règles repérés par leur noms (numéro) et elle sont saisies dans un ordre quelconque.

3.3. caractéristique du langage ExpertRule:

1. les classes sont à héritage simple.
2. les objets représentent les données du système.
3. les attribut sont les caractéristiques d'une classe.
4. les règles représentent le comportement.
5. une base d'objets formés de couples (Attribut, Valeur).
6. les valeurs associées aux attributs peuvent être : chaînes de caractères, numériques (entier ou réel) ou instances d'une classe.
7. les attributs sont soit simples ou multiple, c'est-à-dire qu'ils peuvent posséder une ou plusieurs valeurs au cours d'une session.
8. les attributs peuvent être demandables ou calculables.

3.4. Les connaissances descriptives :

3.4.1. les classes

Une classe est comparable à une structure, et contient des données (connaissances génériques) qui servent à décrire le domaine conceptuel qu'à l'utilisation des cers connaissances.

3.4.2. Les Objets

Un objet est une spécificité d'une classe donnée (instance), il est désigné anonymement par un $, suivi du nom de la classe de l'objet ou directement par son nom.

3.4.3. les Règles

Les règles de production fournissent un formalisme des connaissances adaptées à l'expression de nombreux problèmes de l'intelligence artificielle.

Une règle de production est tout fragment de savoir exprimer sous forme : S ->A

Ou S décrit un situation et A est l'action a entreprendre.

3.4.4. la classification

La classification se faite, à partir des arbres, se sont des fonctions qui font l'objet d'une classification d'un attribut d'objet, via un processus d'extraction des connaissances à partir données ECD.

4. Application d'UML sur notre Langage ExpertRule

4.1.Diagramme de cas utilisation Expert

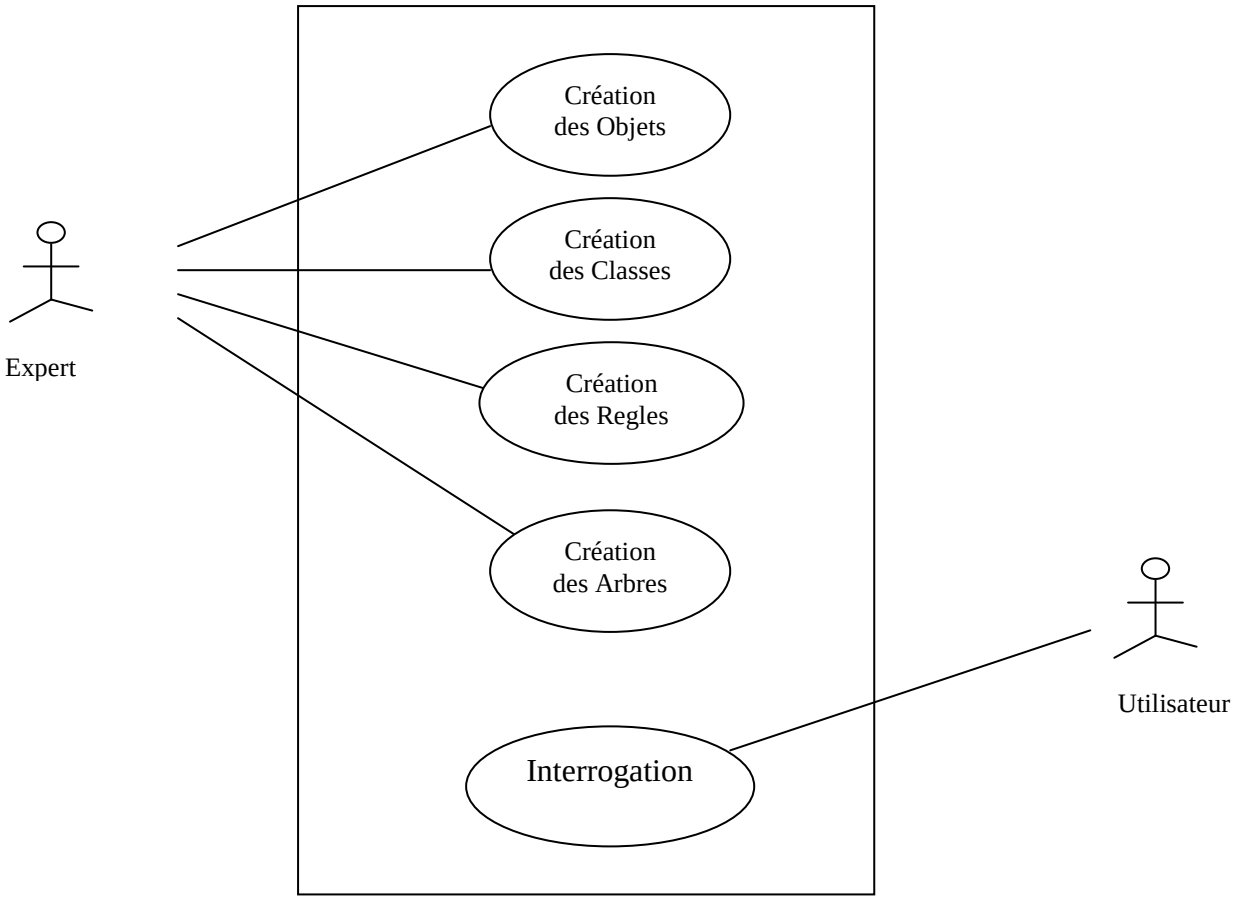


Figure 20 : Diagramme de cas utilisation Expert

4.2.Diagramme état transition Expert

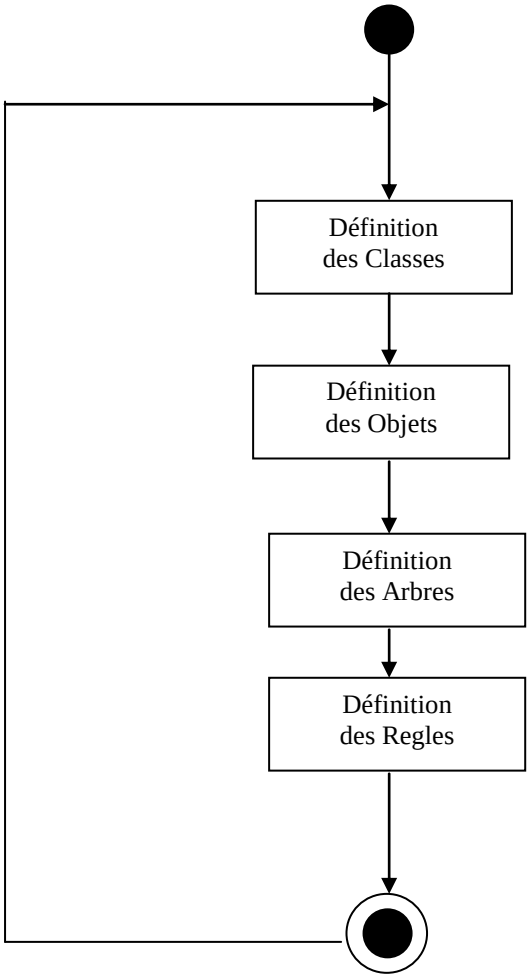


Figure 21 : Diagramme état transition Expert

4.3.Diagramme état transition utilisateur

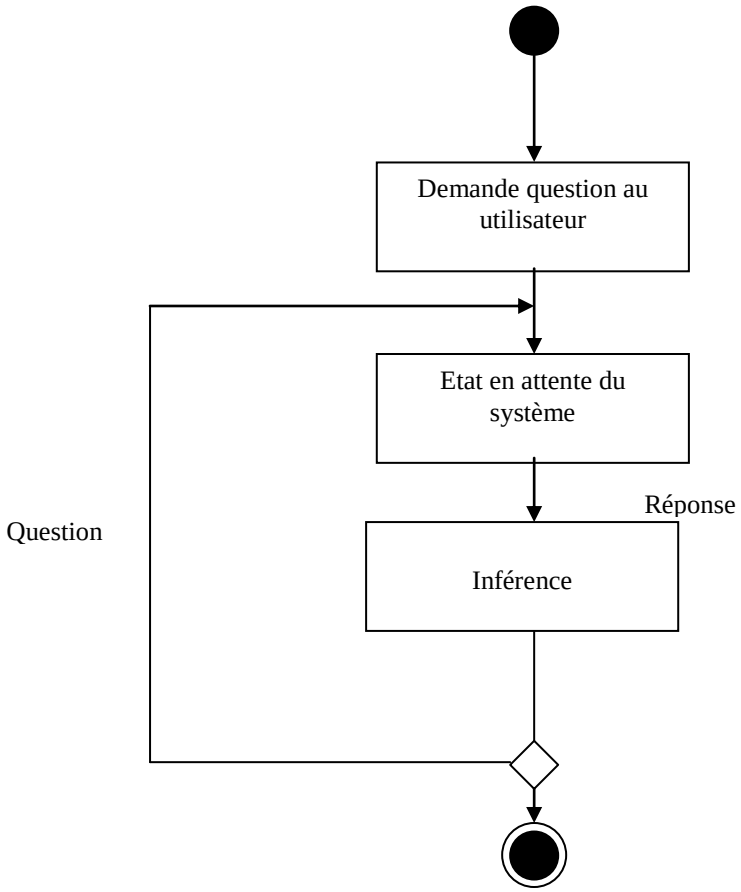


Figure 22: Diagramme état transition Utilisateur

4.4.Diagramme de classe Partie TClasse et Tobjet

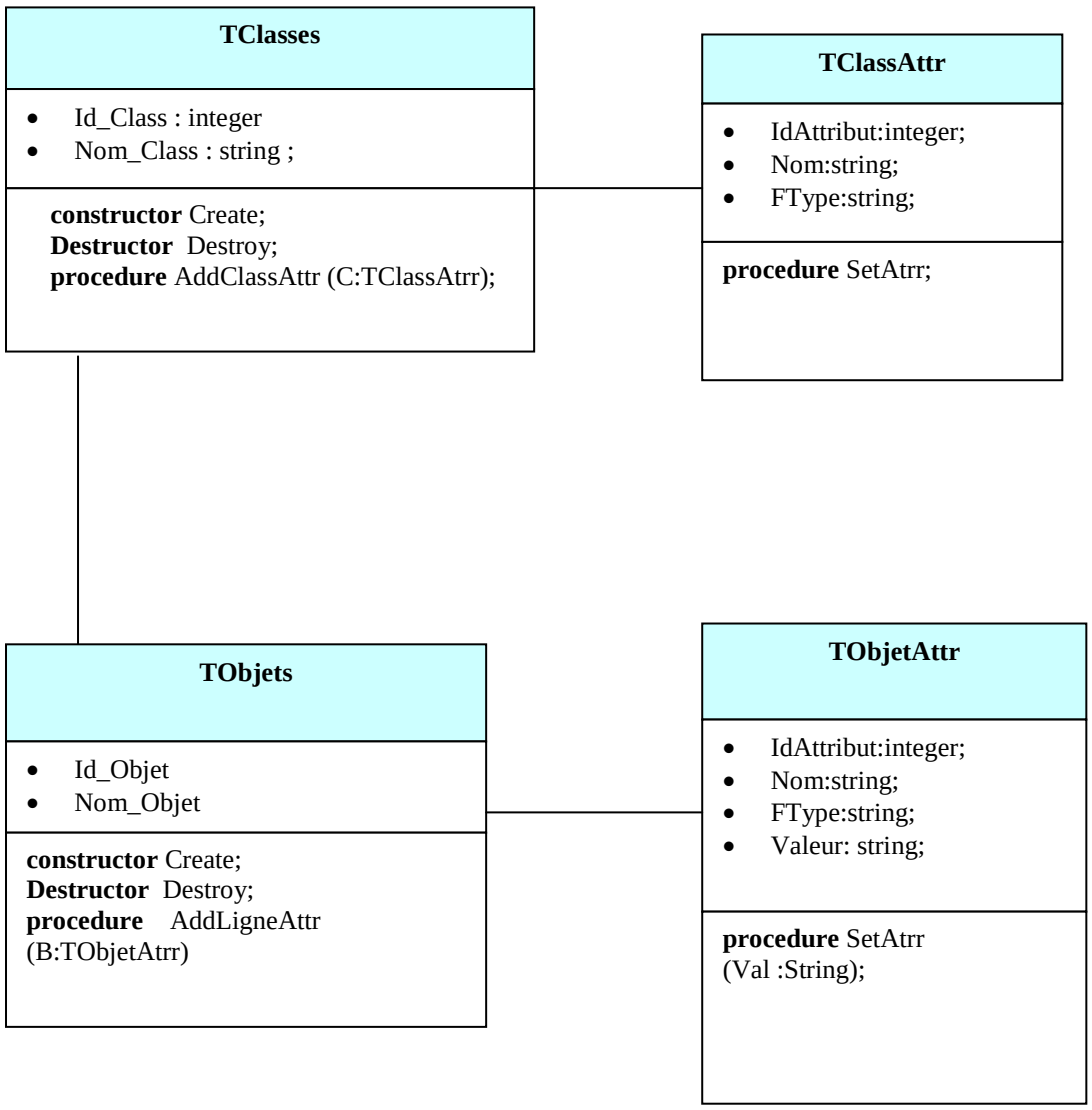


Figure 23 : Diagramme de classe Partie TClasse et TObjet

4.5.Diagramme de classe Partie Texpert

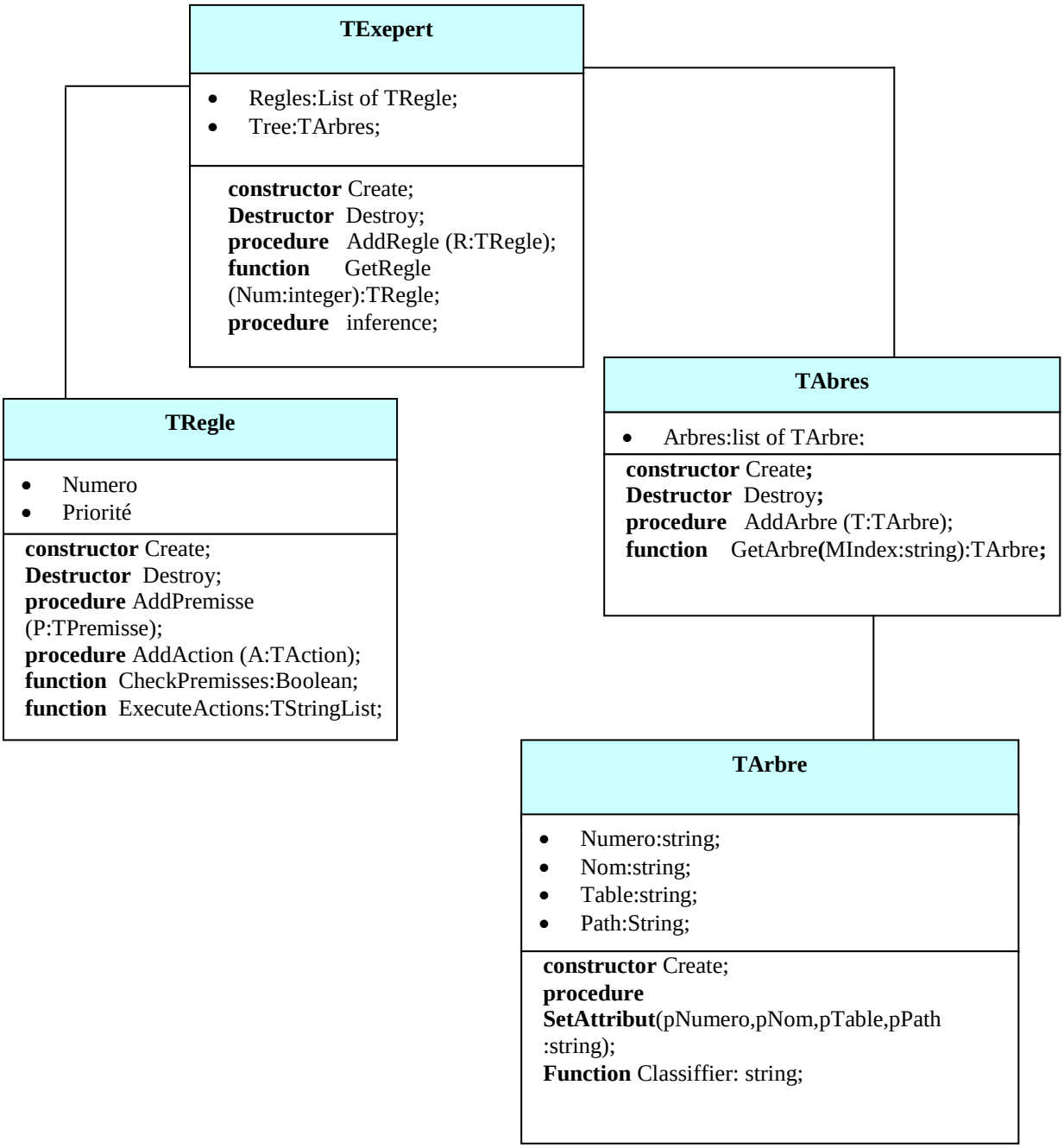


Figure 24 : Diagramme de classe Partie TExpert

4.6.Diagramme de classe Partie Règles

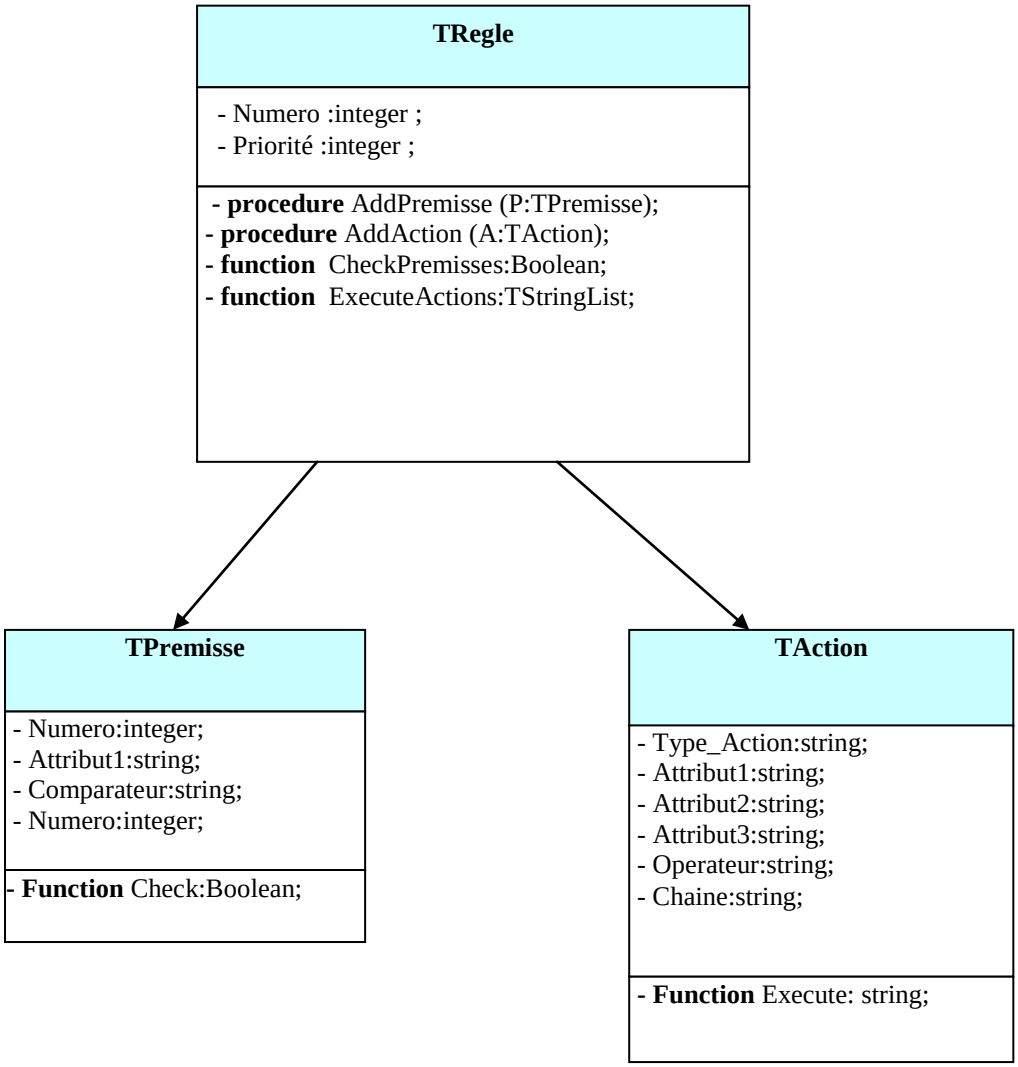


Figure 25 : Diagramme de Classe Représentant les Regèles

4.7. Diagramme de classe Global

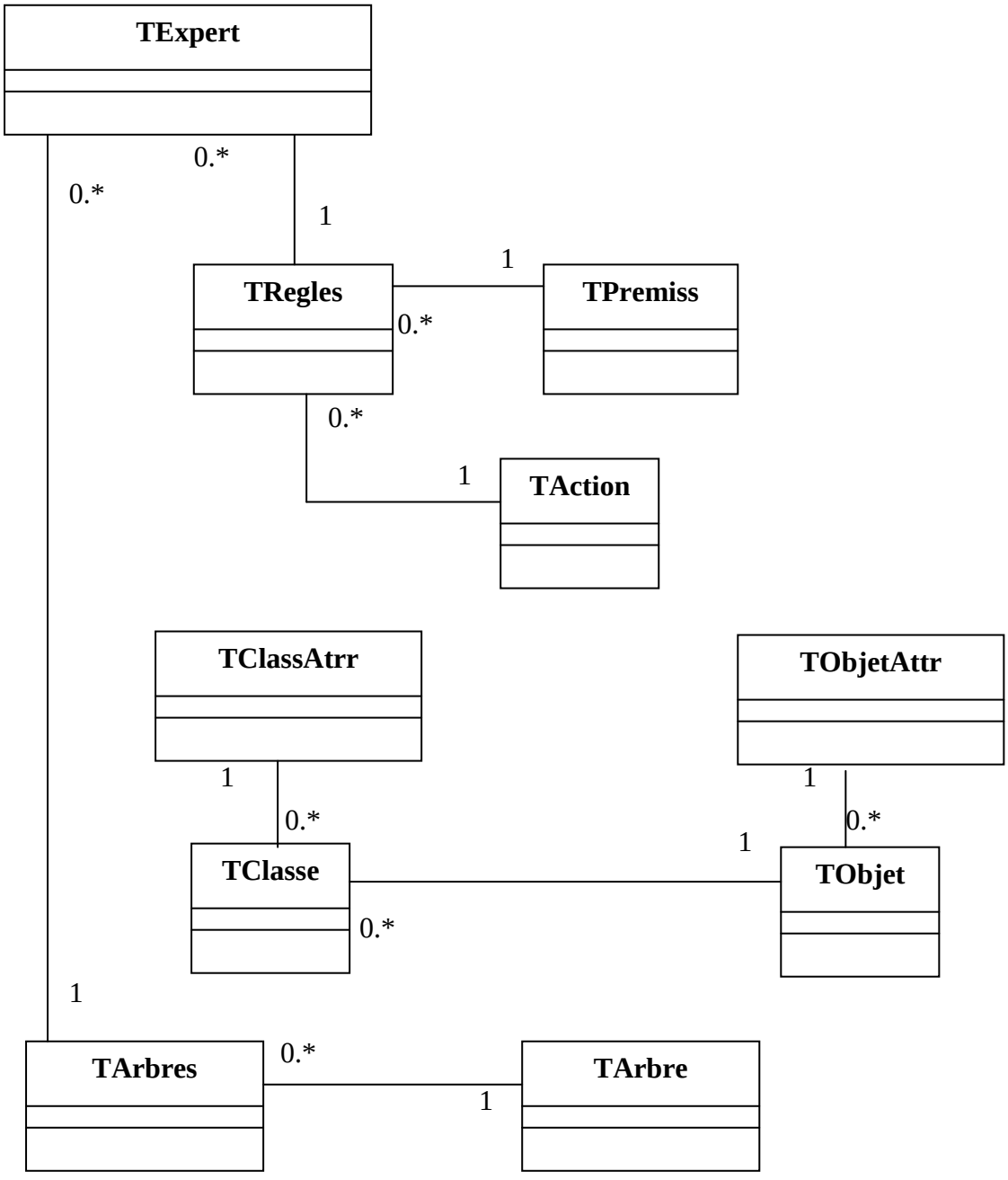


Figure 26 : Diagramme de Classe Représentation Global

4.8. Diagramme de Séquence Création Règles

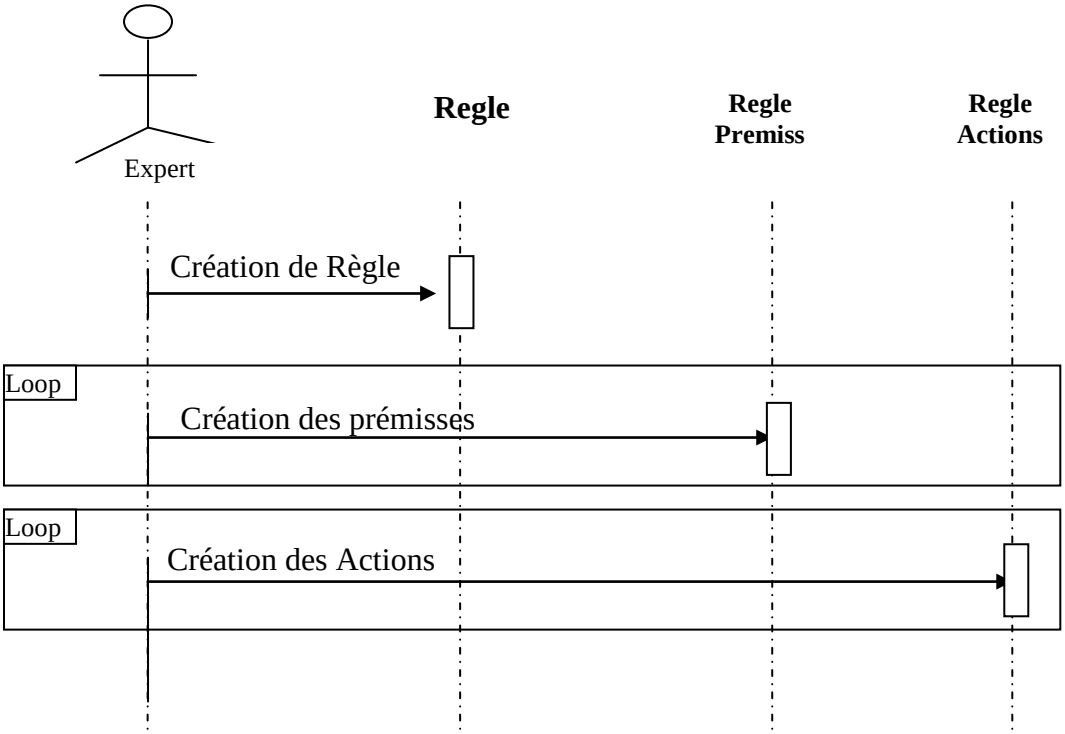


Figure 27 : Diagramme de Séquence Création Règles

4.9. Diagramme de Séquence Création Arbre

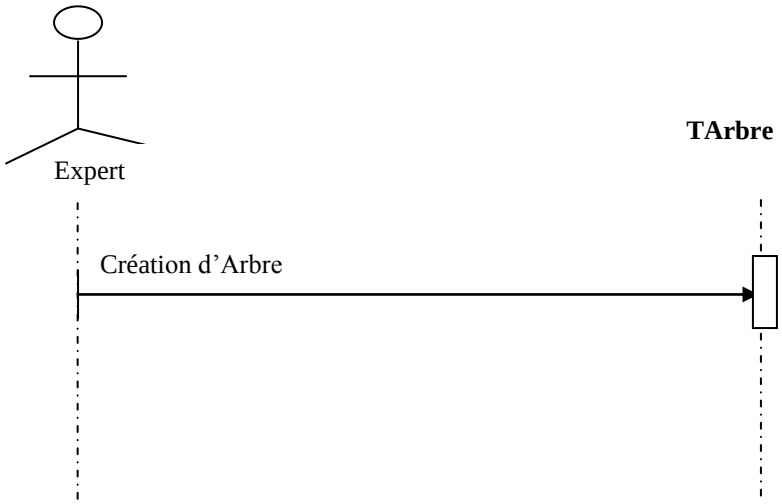


Figure 28 : Diagramme de Séquence Création Arbre

4.10. Diagramme de Séquence Création de Classe

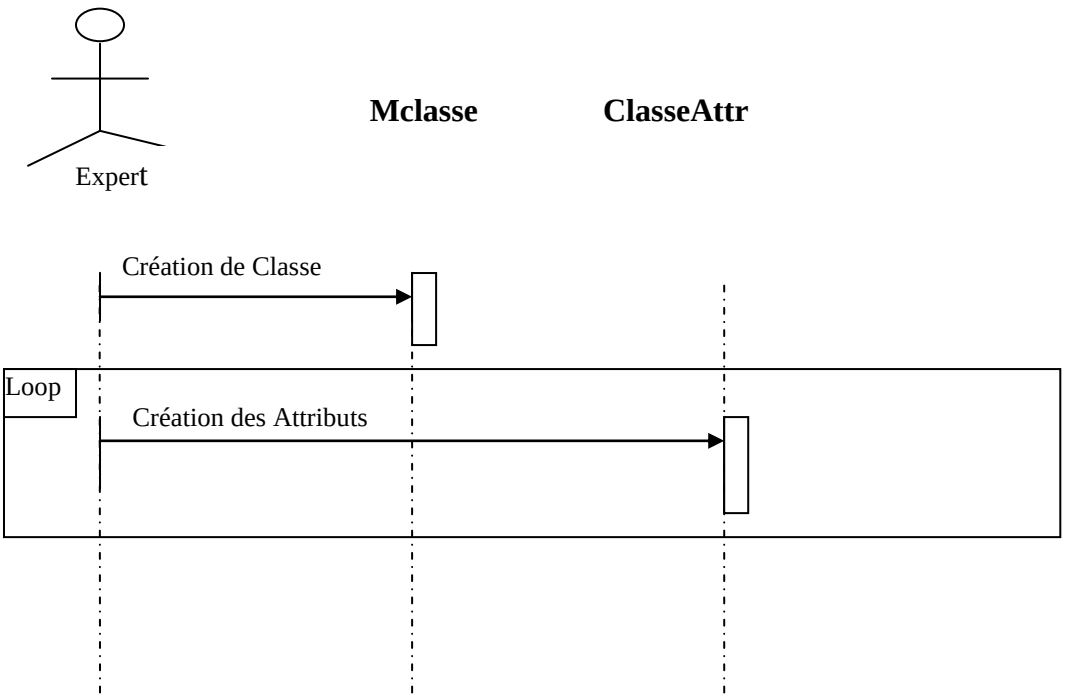


Figure 29 : Diagramme de Séquence Création de classe

4.11. Diagramme de Séquence Création des Objets

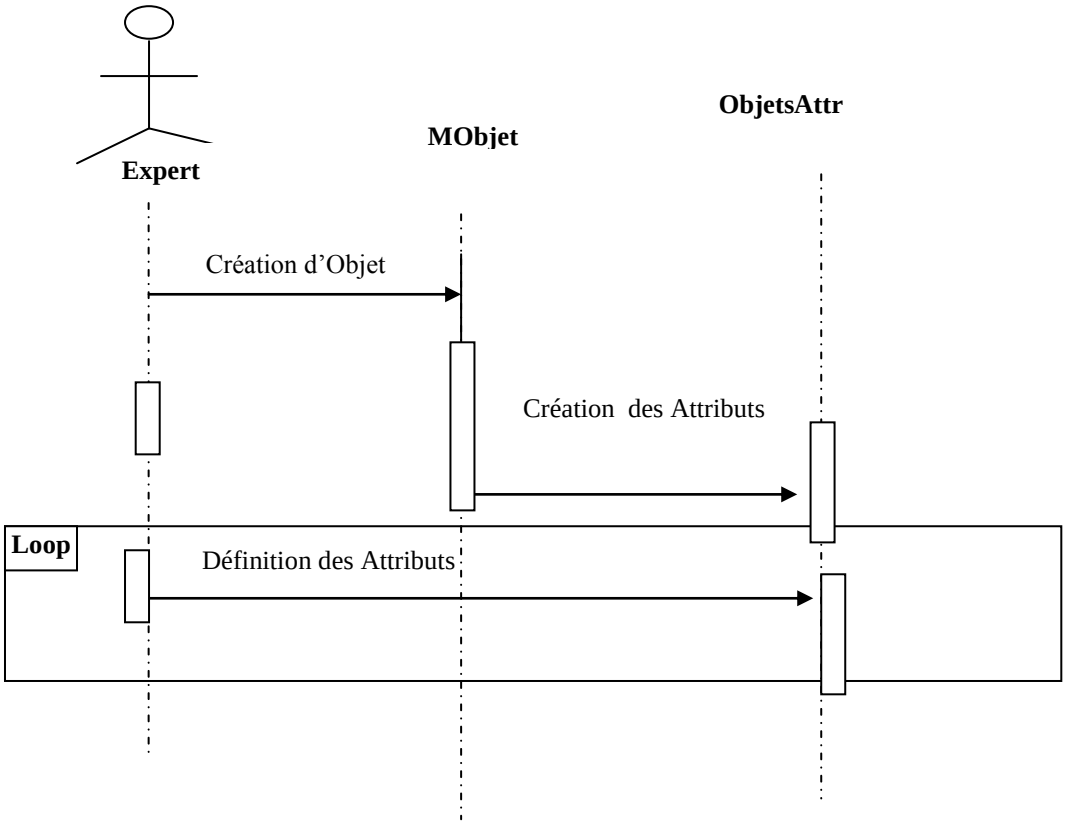


Figure 30 : Diagramme de Séquence Création des Objets

5.Modélisation de la base Aide au voyageur :

Aider un voyageur ou lui donner des conseils est une opération très délicate vu la diversité des voyageurs (Age, sexe, classe, religion…), le but de voyage (tourisme, affaire, loisir…), les moyens de transport (voiture personnel, transport public,train, bus, avion…), lieu de départ et d'arrivée nationale ou internationale,par exemple les conseils pour les voyageurs accompagnés de leurs familles, enfants, ou animaux, avec/sans bagages, suivant les saisons de voyage été,hivers …, jour de fêtes et jours férie.

Modéliser une base de connaissances qui prend en charge les situations cité ci-dessus, pose des difficultés d'enchaînement et de sélection des règles de productions pour assurer un ordre de question pertinent.

On veut créer une base de connaissances qui répond aux besoins des voyageurs, selon leurs situations (financières, pays, temps …), et de leurs offrir les services tels que : Réservation de Vols, hôtels, train, et location de Véhicule.

Et pour cela, on a opté pour le modèle suivant :

- 1 la classe voyageur (Age, en Famille, Nombre d'enfant, Nationalité, avec ou sans bagage, avec ou sans animal).
- 2 la classe Réservation (Ville de départ, ville de destination, heurs de départ, heurs arrivée, distance, le risque, prix).
- 3 la classe Véhicule (nom, type, prix, ville, Libre).
- 4 la classe hôtel (nom, ville, prix, ville, nombre chambre Libre).
- 5 la classe Recherche (type pour un domaine [VOYAGE ; HOTEL ; VOITURE]).

5.1. Application d'UML sur la base Aide aux Voyageurs

5.1.1. Diagramme de cas utilisation voyageur

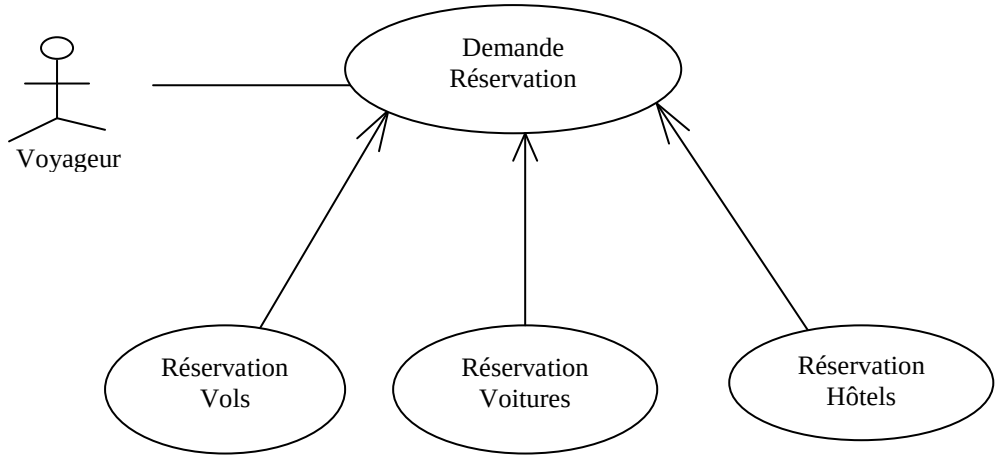


Figure 31 : Diagramme de cas utilisation Voyageur

5.1.2. Diagramme de cas utilisation Expert

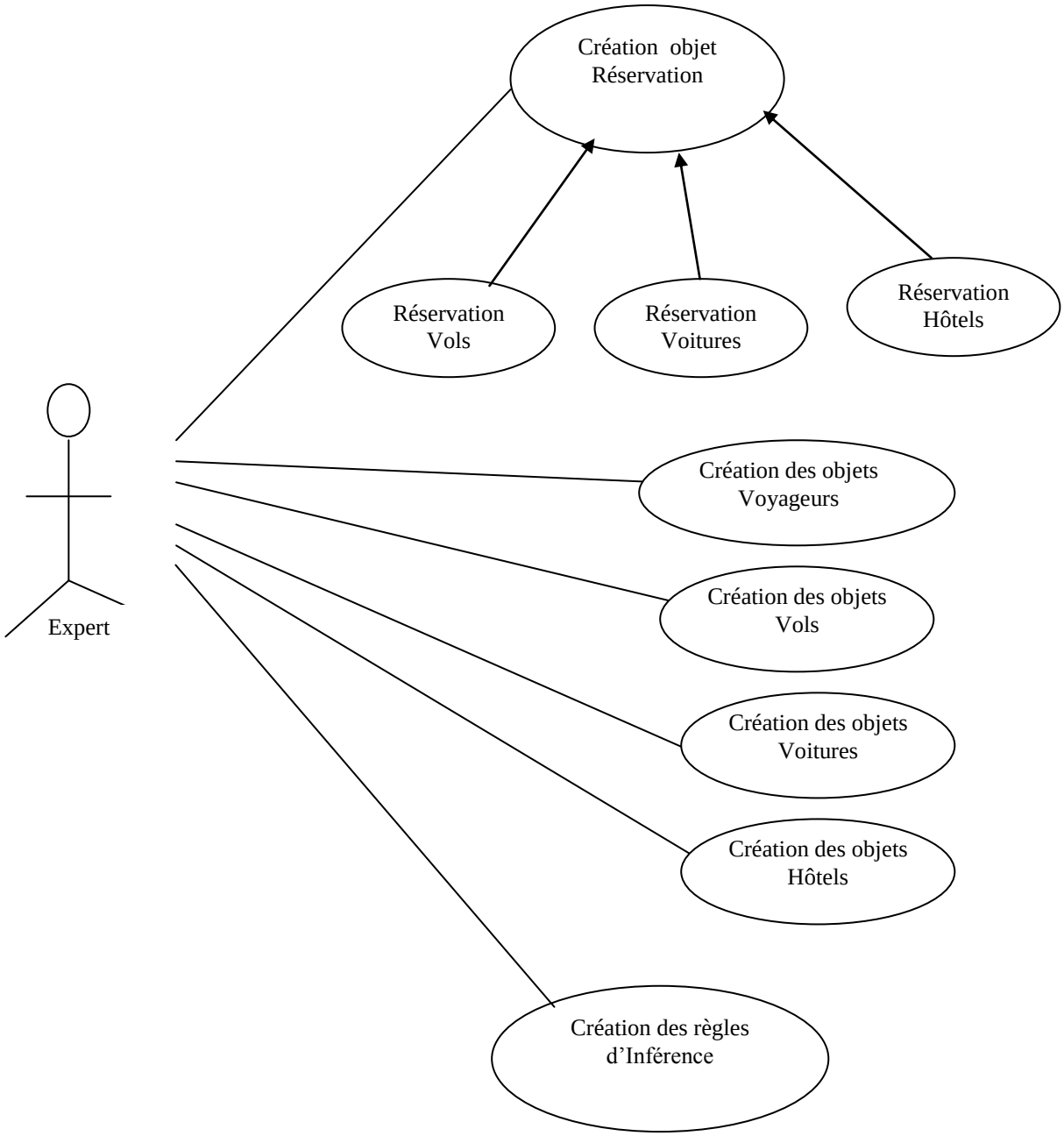


Figure 32 : Diagramme de cas utilisation Expert (Aide Voyageur)

5.1.3. 8.3 Diagramme état transition Expert

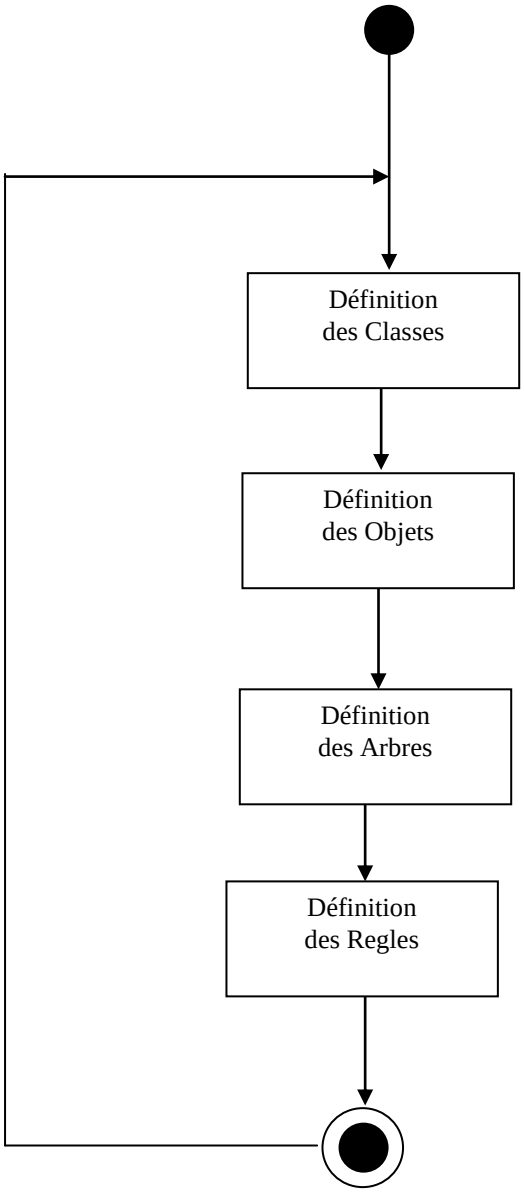


Figure 33 : Diagramme état transition Expert (Aide Voyageur)

5.1.4. Diagramme état transition Réservation Voyageur

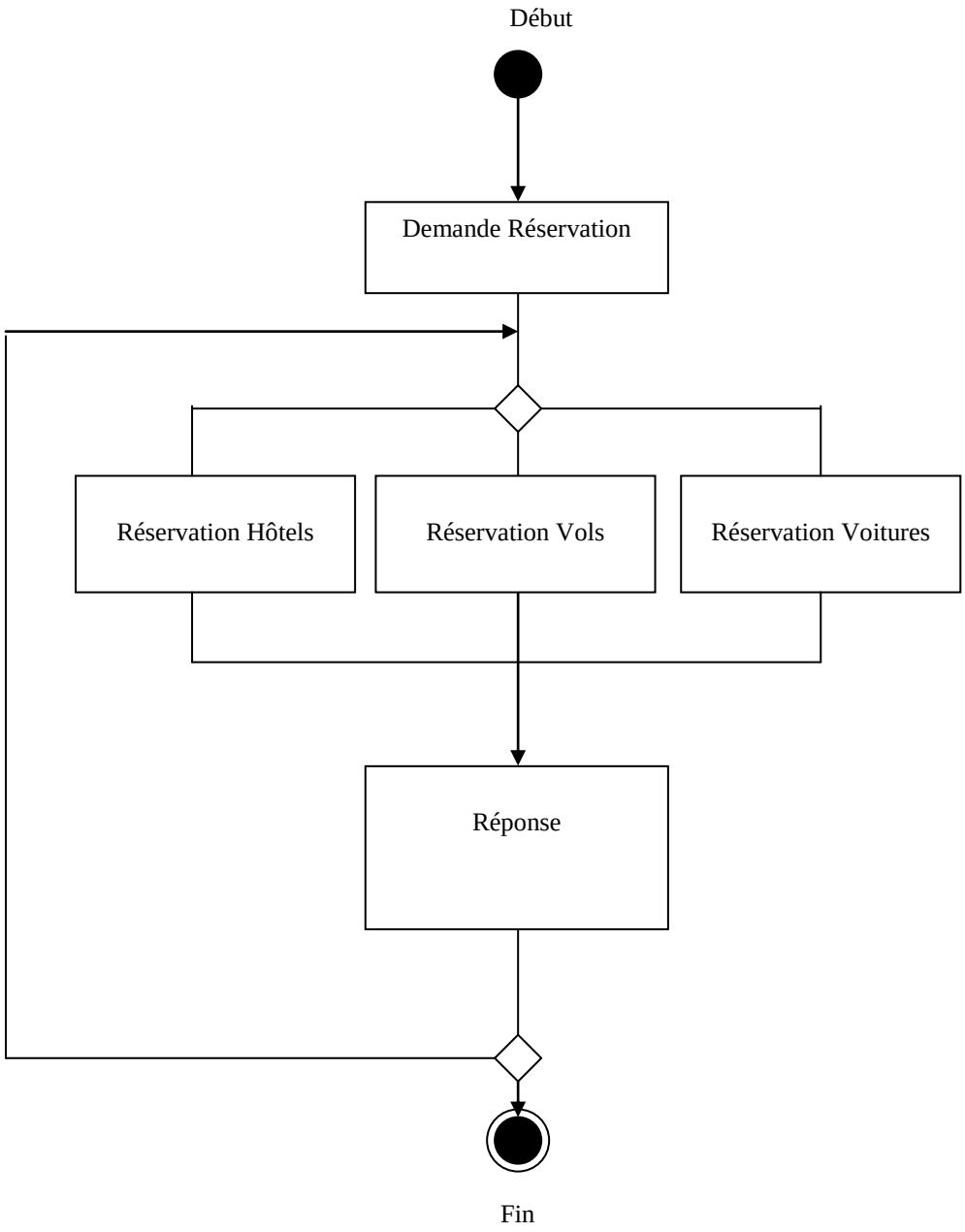


Figure 34 : Diagramme état transition Voyageur

5.1.5. Diagramme de Séquence Expert Création des Objets

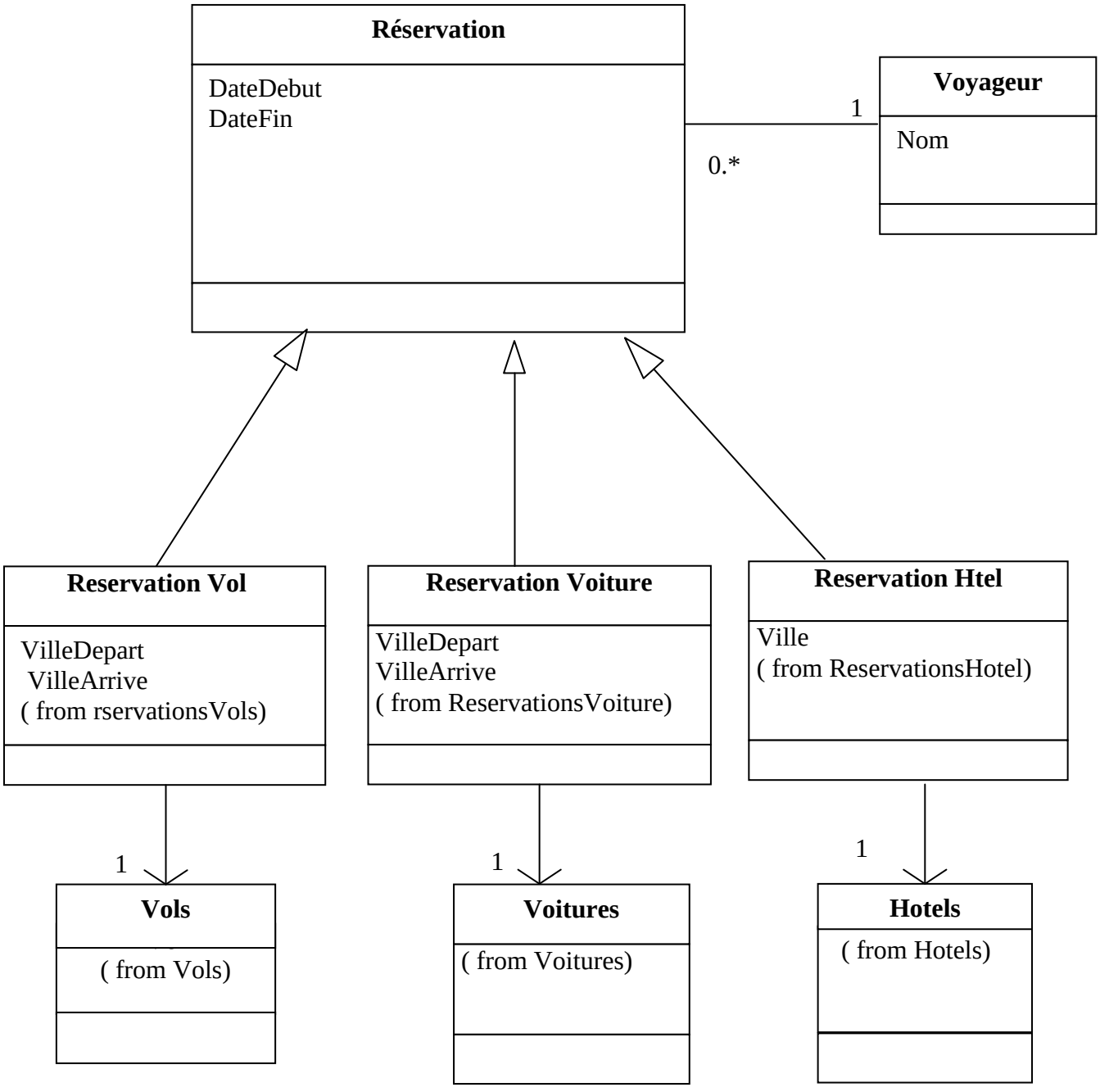


Figure 35 : DIAGRAME DE CLASSE Aide Voyageur

5.1.6. Diagramme de Séquence Création des Objets

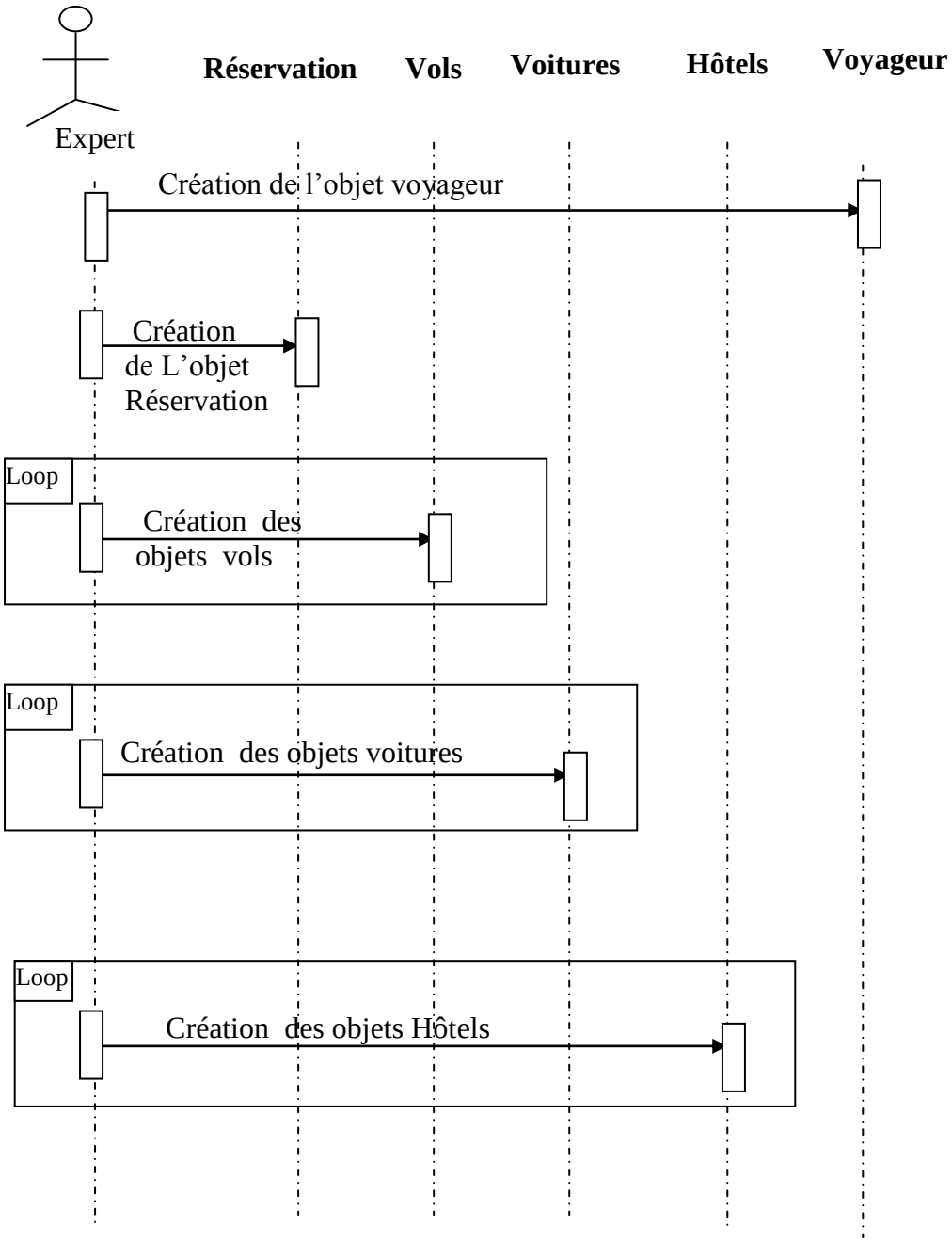


Figure 36 : Diagramme de Séquence Création des Objets

5.1.7. Diagramme de Séquence Voyageur

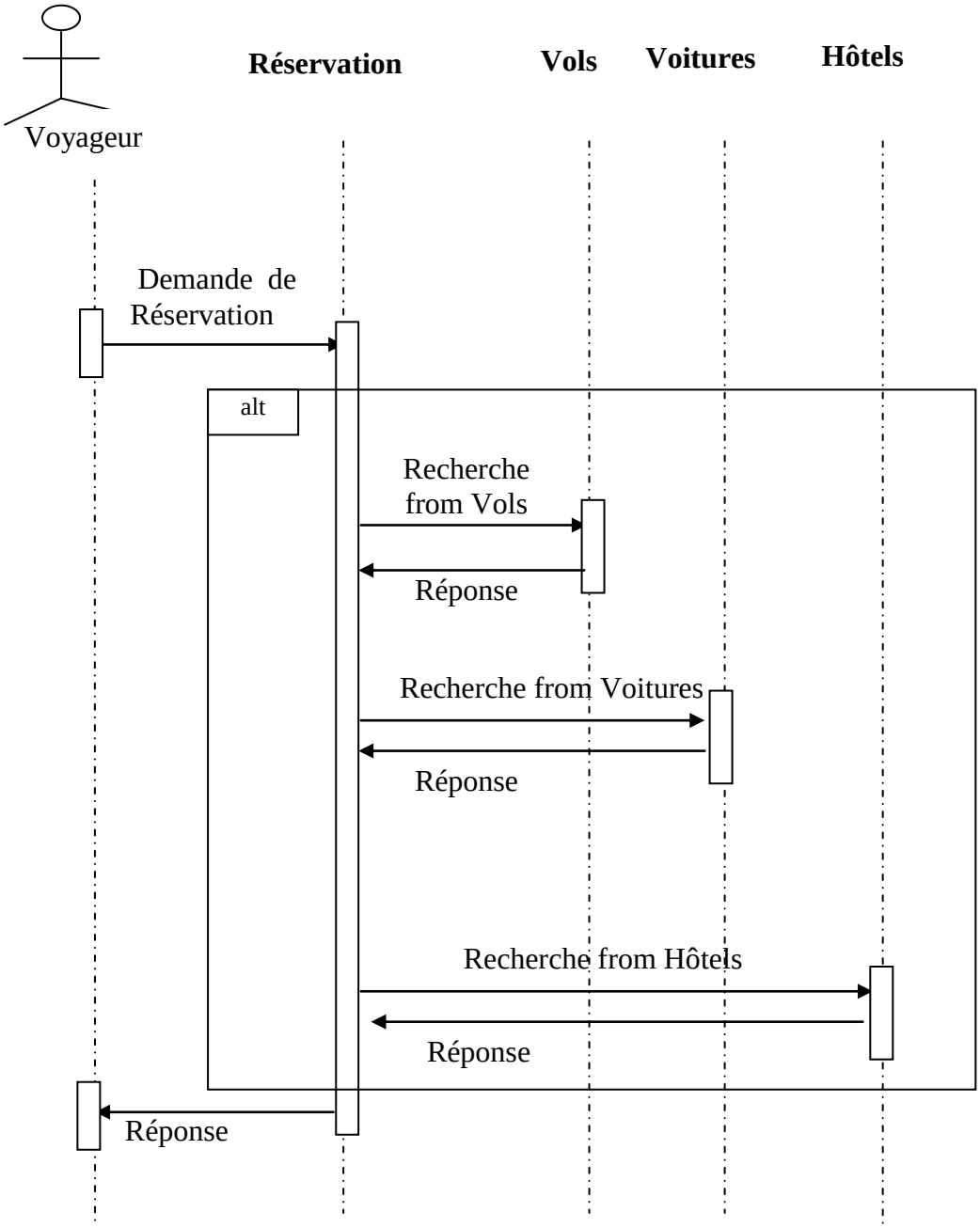


Figure 37 : Diagramme de Séquence Voyageur

6.Application de ExpertRule

On modélise la base de connaissance par une structure de connaissance selon notre langage ExpertRule. La structure de notre langage basé sur le stockage des connaissances dans une base de données relationnel (Microsoft Access) nous a permet d'éviter les traitement et contrôle de syntaxe et la génération de fichier compiler qui sera ultérieurement utilisé pour l'inférence, et de l'autre coté ça facilite la saisie des classes, des objets, et les règles.

6.1. Les Classes

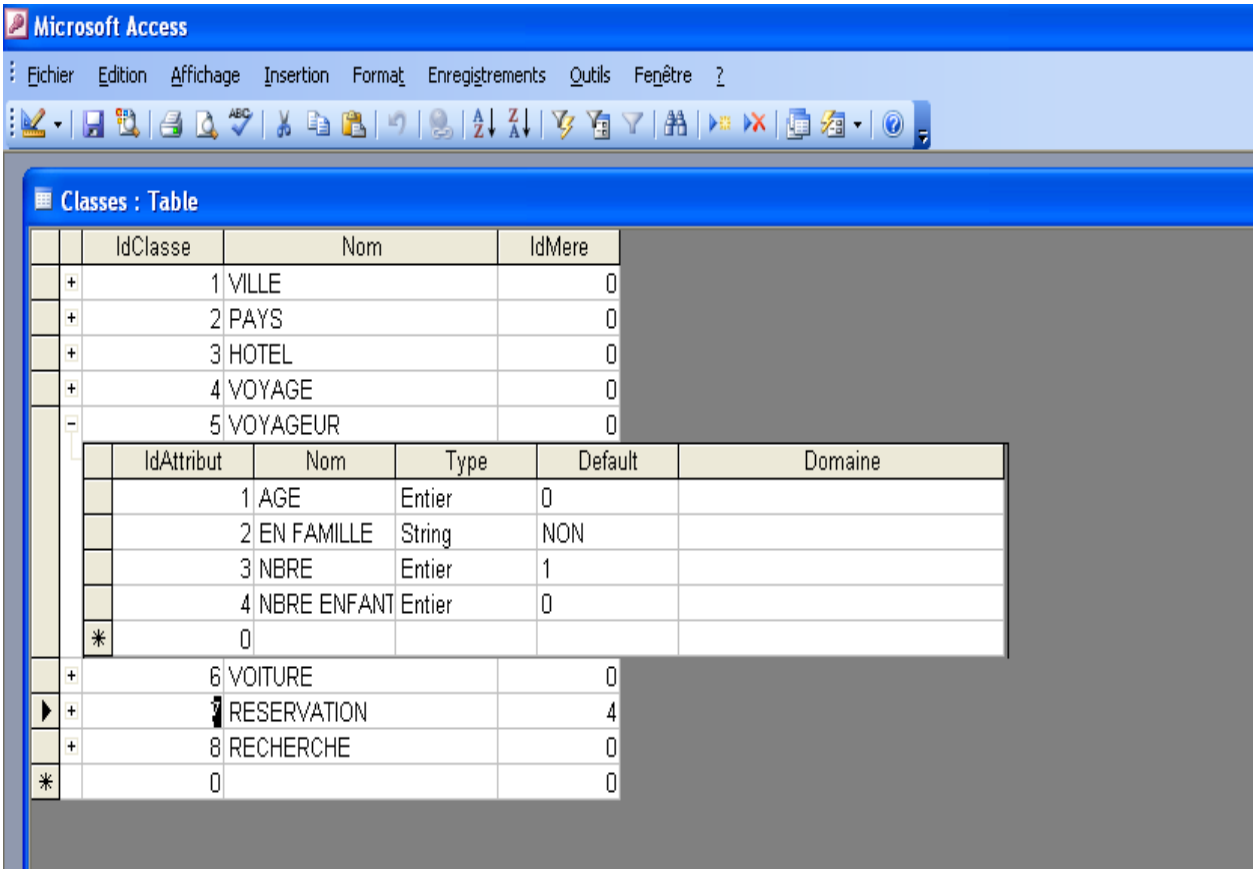
CLASSE = Ville ATTRIBUT :
Nom :STRING ; DEFAULT :NULL ;
pays :STRING ; DEFAULT :NULL ;

CLASSE = HOTEL ATTRIBUT :
Nom :STRING ; DEFAULT :NULL ;
VILLE:STRING ; DEFAULT :NULL ;
CHAMBRE_LIBRE :ENTIER ; DEFAULT :NULL ;
PRIX : ENTIER ; DEFAULT :0 ;

CLASSE : RECHERCHE ATTRIBUT :
TYPE :STRING ; DEFAULT :NULL ;
DOMAINE [VOYAGE ;HOTEL ;VOITURE] ;

CLASSE : VOYAGEUR ATTRIBUT :
AGE:STRING ; ENTIER ; DEFAULT :0 ;
EN FAMILLE :STRING ; DEFAULT :NULL ;
NBRE :ENTIER ; DEFAULT :NULL ;
NBRE ENFANT : ENTIER ; DEFAULT :0 ;

CLASSE : RESERVATION ATTRIBUT :
VILLE1 :STRING ; STRING ; DEFAULT :NULL ;
VILLE1:STRING ; STRING ; DEFAULT :NULL ;
HEURS1 : ENTIER ; DEFAULT :0 ;
HEURS2 : ENTIER ; DEFAULT :0 ;
MOYEN :STRING ; DEFAULT :NULL ;
HOTEL:STRING ; DEFAULT :NULL ;
VEHICULE:STRING ; DEFAULT :NULL ;
PRIX :ENTIER ; DEFAULT :NULL ;
RISQUE : ENTIER ; DEFAULT :NULL ;



IdClasse	Nom	IdMere
1	VILLE	0
2	PAYS	0
3	HOTEL	0
4	VOYAGE	0
5	VOYAGEUR	0

IdAttribut	Nom	Type	Default	Domaine
1	AGE	Entier	0	
2	EN FAMILLE	String	NON	
3	NBRE	Entier	1	
4	NBRE ENFANT	Entier	0	
0				

6	VOITURE	0
7	RESERVATION	4
8	RECHERCHE	0
0		0

FIGURE 38 : SCHEMA DE LA TABLE CLASSES SOUS MSACCES

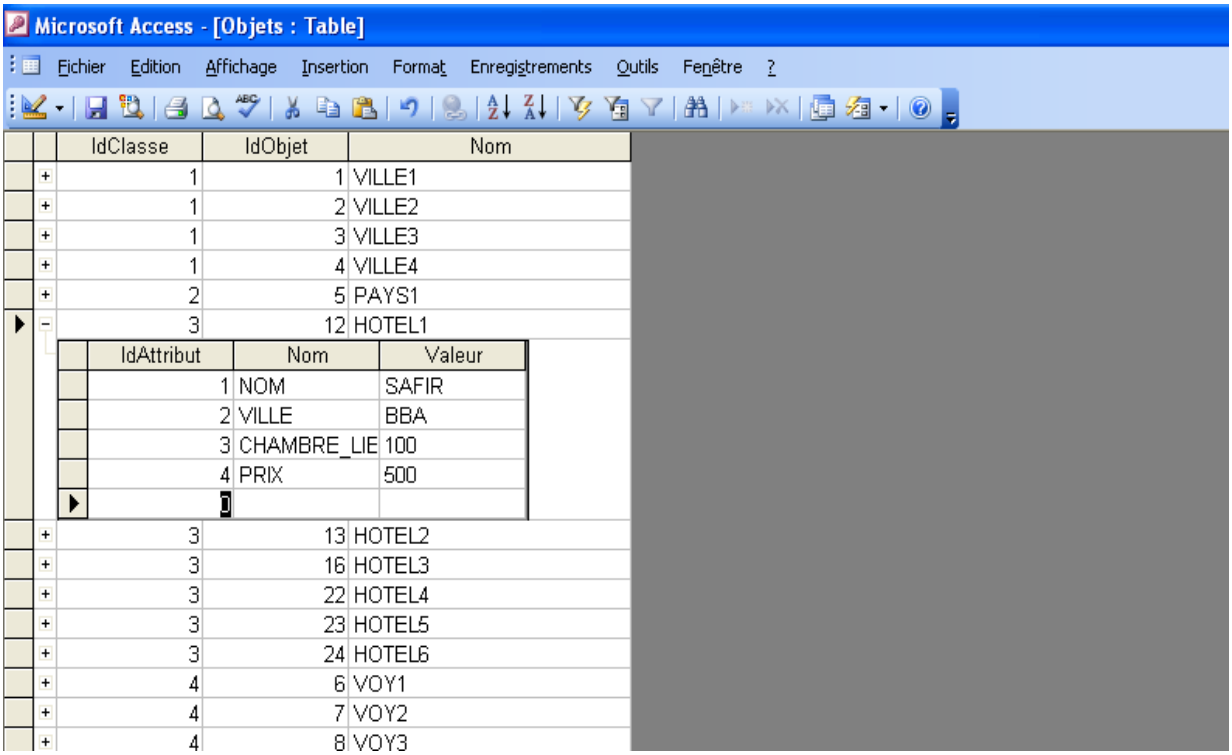
6.2. les Objets

OBJET : *VILLE1 est_un VILLE :*
Nom =”BBA” ;
pays =”ALGER” ;

OBJET : ***VILLE2 est_un VILLE :***
Nom =”SETIF” ;
pays =”ALGER” ;

OBJET : *HOTEL1 est_un HOTEL :*
Nom =”TERGUI” ;
VILLE=”BBA” ;
CHAMBRE_LIBRE=11 ;
PRIX=2000 DA ;

OBJET : *HOTEL2 est_un HOTEL :*
Nom =”SETIFIS” ;
VILLE=”SETIF” ;
CHAMBRE_LIBRE=20 ;
PRIX=2500 DA ;



IdClasse	IdObjet	Nom
1	1	VILLE1
1	2	VILLE2
1	3	VILLE3
1	4	VILLE4
2	5	PAYS1
3	12	HOTEL1

IdAttribut	Nom	Valeur
1	NOM	SAFIR
2	VILLE	BBA
3	CHAMBRE_LIE	100
4	PRIX	500

IdClasse	IdObjet	Nom
3	13	HOTEL2
3	16	HOTEL3
3	22	HOTEL4
3	23	HOTEL5
3	24	HOTEL6
4	6	VOY1
4	7	VOY2
4	8	VOY3

FIGURE 39 : LA TABLE OBJETS ET SOUS MSACCES

6.3. les Règles

Règle N° 6

SI
VOYAGEUR.AGE = 0
ALORS
Ecrire -> ENTREZ VOTRE AGE SVP :
Lire -> VOYAGEUR.AGE
Ecrire -> AGE SAISIE EST : VOYAGEUR.AGE

Règle N° 7

SI
RECHERCHE.TYPE = VOYAGE
ALORS
Ecrire -> ENTREZ VOTRE VILLE DE DEPART...
Lire -> RESERVATION.VILLE1
Ecrire -> VILLE DE DEPART EST : RESERVATION.VILLE1

Règle N° 8

SI
RESERVATION.VILLE1 <> NULL
ALORS
RESERVATION.VILLE2 := LisVoyage.Lines/ RESERVATION.VILLE1
RESERVATION.HEURS1 := LisVoyage.Lines/ HEURS1
RESERVATION.HEURS2 := LisVoyage.Lines/ HEURS2
RESERVATION.VEHICULE := LisVoyage.Lines/ MOYEN
Ecrire -> RECHERCHER UN VOYAGE...

Regle N° 9

SI
RESERVATION.VILLE2 <> NULL
RECHERCHE.TYPE = VOYAGE
ALORS
Ecrire -> VOYAGE PLANIFIER
Ecrire -> VILLE DE DESTINATION EST : RESERVATION.VILLE2
Ecrire -> HEURS DEPART ESET : RESERVATION.HEURS1
Ecrire -> HEURS ARRIVE EST : RESERVATION.HEURS2
Ecrire -> MOYEN TRANSPORT EST : RESERVATION.VEHICULE

Regle N° 10

SI
RESERVATION.VILLE2 = NULL
ALORS
Lire -> RESERVATION.VILLE2
Ecrire -> VILEE DE DESTINATION RESERVATION.VILLE2

Regle N° 11

SI
RESERVATION.HEURS1 = 0
RESERVATION.VILLE2 <> NULL
RECHERCHE.TYPE = VOYAGE
ALORS
Ecrire -> ENTREZ VOTRE HEURS DE DEPART...
Lire -> RESERVATION.HEURS1
Ecrire -> HEURS DEPART EST : RESERVATION.HEURS1

Regle N° 14

SI
RESERVATION.VEHICULE = NULL
RESERVATION.VILLE2 <> NULL
RECHERCHE.TYPE = VOITURE
ALORS
RESERVATION.VEHICULE := LisVoiture.Lines/ RESERVATION.VILLE2
Ecrire -> VEHICULE A ALLOUER : RESERVATION.VEHICULE
Lire -> RECHERCHE.TYPE

Regle N° 15

SI
RESERVATION.RISQUE = NULL
RESERVATION.VILLE1 <> NULL
RESERVATION.VILLE2 <> NULL
ALORS
Ecrire -> RECHERCHE RESULTAT VIA ARBRE DE DECISION
Ecrire -> --- R15 ---
RESERVATION.RISQUE := #Risques.Risque

Regle N° 16

SI
RESERVATION.RISQUE = oui
ALORS
Ecrire -> IL EST POSSIBLE QUE VOUS RISQUER DANS CE VOYAGE

Regle N° 17

SI
RESERVATION.RISQUE = non
ALORS
Ecrire -> LE TAUX DE RISQUE EST BAS

Chapitre 5

Implémentation & validation

1. Introduction

Le travail proposé consiste principalement à réaliser un générateur des systèmes experts dans la plupart des domaines, offre aux utilisateurs spécialisés dans le domaine (des Experts) la possibilité d'actualiser les bases de connaissances pour apporter une grande souplesse aux multiples complexités. Il offre aussi la possibilité d'interrogé des bases de données pour extrait des connaissances pour l'utiliser dans un processus d'inférence.

Pour ces raisons on a réalisé un éditeur de connaissance (Edc) professionnelles, avec l'intégration des bases de données relationnelles via le moteur des bases de données « MicroSoft Access », qui se devise en trois éditeurs principaux :

- ***Editeur des classes (EdC).***
- ***Editeur des objets (EdO).***
- ***Editeur des règles (EdR).***

En a rajouter un module de classification qui permet de classifier et d'extraire des connaissances a partir de données.

Notre application a été développe dans un environnement Windows XP, avec le langage de programmation « Delphi 7 ».

2.But du système

ExpertTreeRule est un moteur d'inférence qui permet de développer rapidement des systèmes experts, c'est-à-dire un système d'aide à la décision dans des domaines relativement structurés (dépannage, diagnostic médical simple, ...). Il est d'ordre 1+ et utilise principalement la technique du chaînage avant. Ce logiciel possède de nombreuses possibilités de fenêtrage, de chaînage de bases de connaissance, d'utilisation de données diverses, etc.

Interroger une base de connaissance Résultante d'un algorithme extraction des connaissances E.C.d. au moment de l'inférence est la partie la plus efficace, cela permet d'identifier et de classer un Attribut d'un objet dans un domaine précis, par l'appel de module de classification TreeClassifier, ce dernier fait l'objet d'un processus d'apprentissage automatique qui extrait des connaissances à partir des données, la méthode utilisé est C4.5 de quinlan 96.

Le processus d'extraction des connaissances a partir de données est une arme à double tranche, d'un coté si la base de connaissance extraite traite le sujet entièrement, on va l'utilisé comme une base de connaissance initial, et dans le cas où la base de connaissance extraite traite ou représente une partie du Sujet Global, le système expert interroge seulement cette Base pour classifier un attribut D'objet dans ça base de règles au moment de l'inférence.

3.L'environnement d'ExpertTreeRule

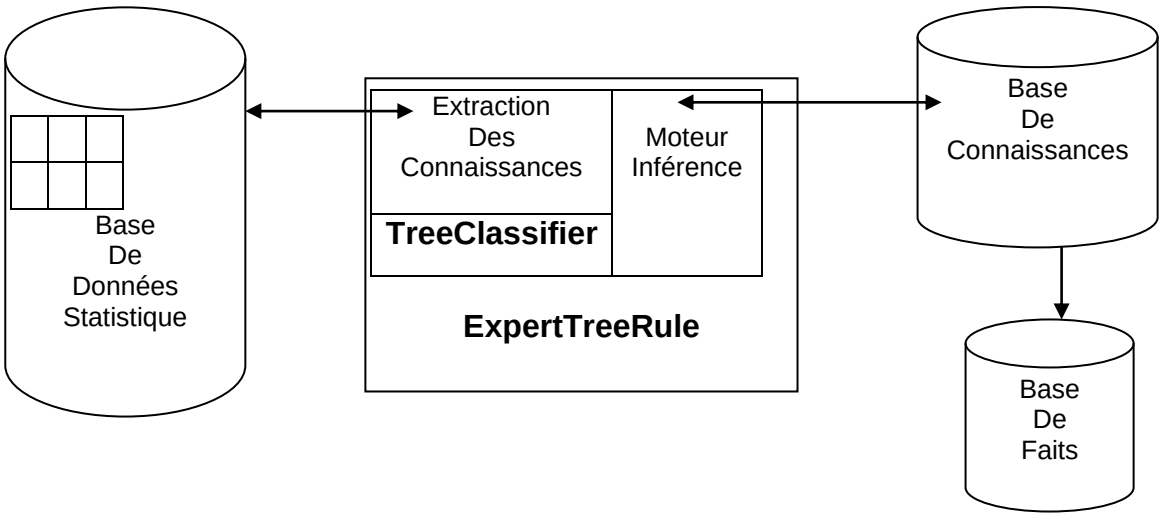


Figure 40 : Schéma général de *ExperTreeRule*

3.1. Structure de Base de Données :

ExpertTreeRule stock et enregistre les Faits et les Base de règles dans une base de données relationnelle MsAcces, les données sont directement enregistrées et encodées dans les tables, Cela évite les erreurs et de saisie d'un coté et diminue les traitements et les contrôles de l'autre coté (Analyse syntaxique, la compilation et la génération des fichier comme c'est le cas dans la plupart des système expert).

3.2. Langage de développement :

Le logiciel est développé sous Delphi 7, vu que Delphi est un langage compilé de haut niveau à types stricts qui gère la conception structurée et orientée objet. Basé sur Objet Pascal, ces qualités sont la lisibilité du code, une compilation rapide et l'utilisation d'unités multiples permettant une programmation modulaire.

Delphi propose des caractéristiques spéciales qui gèrent le modèle de composant et l'environnement RAD de Borland.

La plupart des développeurs utilisant les outils de développement Borland, écrivent et compilent leurs programmes dans l'environnement de développement intégré (EDI).

L'accès aux bases de données en utilisant les technologies DAO, ADO , ainsi que la création de contrôles ou objets ActiveX.

4.Architecture de notre système Expert

4.1. Stockage des Données

En représente ici la structure des tables dans Microsoft Access, nous avons crées sept tables qui permettent le stockage des classes, des objets et les règles.

4.1.1. Les Classes :

Les classes sont définit dans la table CLASSES qui contienne les champs IdClasse : 'Identifiant de la classe ', Nom= le Nom de la classe ', et IdMere : 'Identifiant de la classe Mère pour l'héritage ', relier avec la Table CLASSEATTRIBUTS pour définir les attribut d'une classe et qui contienne les champs IdClasse='Identifiant de la classe où appartient l'attribut', idAttribut= 'Identifiant de l'attribut', MIndex ='clé pou indexer les Attribut ', Nom= 'Nom de l'Attribut ', et Type = 'Type de donnée de l'Attribut :string,integer… '.

4.1.2. Les Objets :

Les Objets sont définit par la table OBJETS qui contienne les champs IdClasse 'Identifiant de la classe ', IdObjet 'Identifiant de l'objet', Nom= le Nom de la classe ', relier avec la Table OBJETATTRIBUTS pour définir les attribut d'un OBJET et qui contienne les champs IdClasse, Idobjet, idAttribut, MIndex, Nom, Type ,Vlaeur = ' désigne la valeur de l'attribut '.

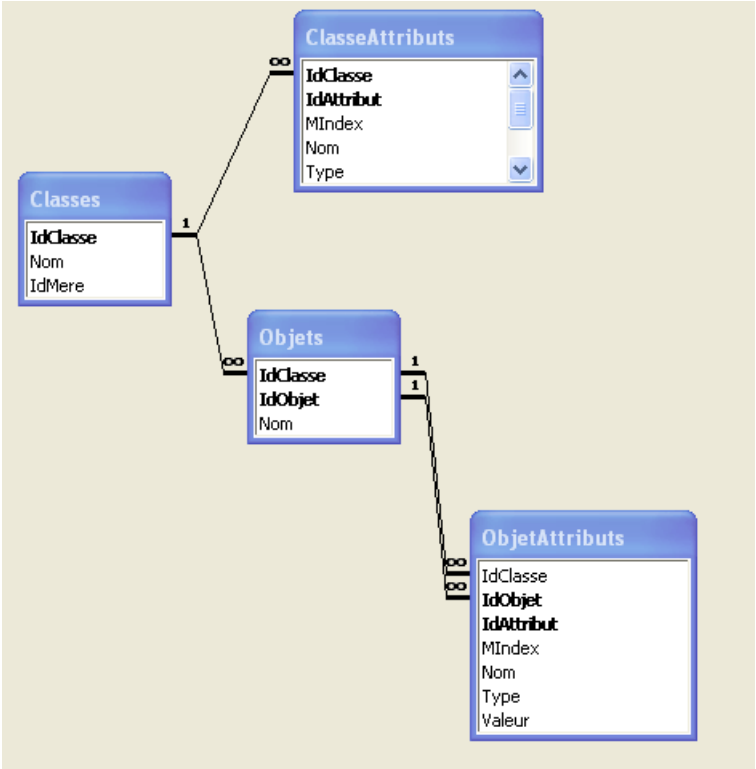


Figure 41 : schéma des tables : classes_ClasseAttribut, objets_ObjetAttributs.

4.1.3. Les Règles :

Les règles sont définit par la table REGLES qui contienne les champs Idregle : Numero de la regle ', Priorite 'l'ordre de declenchement relier 1 avec la Table REGLEPREMISSES pour définir les Premisses d'une REGLE et qui contienne les champs Idregle , MIndex1 'adresse de 1er Attribut', MIndex2 'adresse de 2eme Attribut' , OP = ' opérateur de comparaison : =,>,<…'.
Et 2 avec la Table REGLEACTIONS pour définir les Actions d'une règle et qui contienne les champs Idregle , MIndex1 'adresse de 1er Attribut à Lire ou à Modifier', MIndex2 'adresse de 2eme Attribut' , MIndex3 'adresse de 3eme Attribut', OP = ' opérateur Arithmétique : = +,-,/,*..', Action 'désigne le type de l'action (Lire,Affectation,Recherche,Affichage)', chaine 'le Texte à Afficher'.

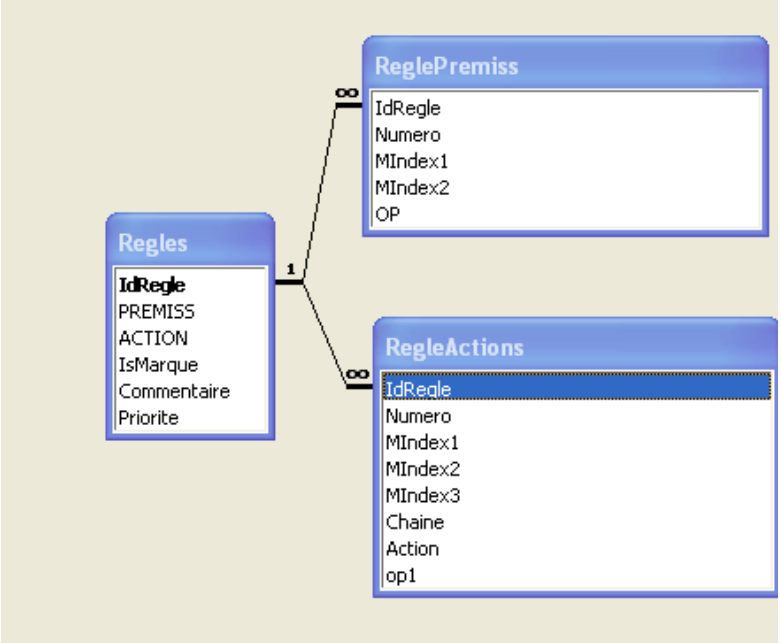


Figure 42 : schéma des tables : Regles_RegleActioins_ReglePremisses.

5.Syntaxe de notre Langage:

Ici on présentes les mots réservés, et les différentes formes de règles supportées dans notre langage ExperTreeRule.

5.1.Les Symboles :

+ addition de deux expressions
- soustraction de deux expressions
* multiplication de deux expressions
/ division de deux expressions
< plus petit que
<= plus petit ou égal
> plus grand que
>= plus grand ou égal
<> différent .

5.2. Les règles : (RULE)

Les règles satisfont à la forme générale IF prémisse then conclusion.

5.2.1. La partie prémisse

La partie prémisse est constituée d'une suite de conditions Chaque condition est de la forme: <Objet1.Propriete> <comparateur> <valeur>/<Objet2.Propriete>.

Exemple

1) Voyage1.Moyen= ‘ Voiture ’. de forme <Objet1.Propriete> <comparateur> <valeur>
Ou :
2) Voyage1.Moyen= Reservation.Moyen. . de forme <Objet1.Propriete> <comparateur> <Objet2.Propriete>.

5.2.2. La partie Action :

La partie Action est constituée, du point de vue syntaxique, d'une suite de CLAUSES. Chaque CLAUSE implique une action. Les trois principales clauses d’actions sont:

- La formulation d'un but : syntaxe : Objet.Attribut=$classe.propriete

  Exemple: Voyage1.Hotel=$ Ville.Hotel : Recherche l’objet Ville dans tous les objets de classe Ville dont la propriété Hotel égal à la propriét hotel de l’objet Voyage1.

- La modification d'un fait : syntaxe : <Objet1.Propriete> := <Objet1.Propriete>/ / <Valeur><operateur> <Objet1.Propriete> / <Valeur>

  Exemple: Voyage1.ville1 := ‘’ BBA ‘’.
  Reservation.Prix :=Voyage1.Prix+Hetel6.Prix.

- L’affichage : syntaxe : Ecrire <Constatant Texte > <Objet.propriete>.

  Exemple: Ecrire ->’Ville de depart est :’ Voyage1.ville1.

- Les Listes : syntaxe : <NomListe><Classe> <Parametre>

  Exemple:LisVoiture.Lines/Véhicule /RESERVATION.VILLE2 : Rechercher dans les objets Voiture de classe véhicule où leur propriétés ville égal à la propriété Ville2 de l’objet RESERVATION.

6. Implémentation sous Delphi :

Dans ce qui suit on va présentés quelques exemple de déclaration des classes et des méthodes de ExpertTreeRule dans notre code source écrit en langage Delphi.

6.1. Les classes

```
Type TPremisse=class
NumeroRegle:integer;
ord:Integer;
Attribut1:string;
Valeur1:string;
Attribut2:string;
Valeur2:string;
Comparateur:string;
end;

Type TAction=class
Numero:integer;
ord:Integer;
Attribut1:string;
Valeur1:string;
Attribut2:string;
Valeur2:string;
Attribut3:string;
Valeur3:string;
Operateur:string;
Type_Action:string;
Chaine:string;
FType:string;
end;

Type TRegle=class
Numero:integer;
Premisses:Tlist; { Liste des Premisses}
Actions: Tlist;  { Liste des Actions}
end;

Type TRegles=class
Regles:TList; { Liste des Regles}
Tree:TArbres;      // ensemble des mini expert E.C.D
end;

Type TArbre=class  // declaration des Classes : mini expert E.C.D
Numero:string;
Nom:string;
Table:string;
Path:String;
End;

Type TObjetAtrr=class
IdClasse:Integer;
IdObjet:Integer;
IdAttribut:integer;
MIndex:string;
Nom:string;
FType:string;
Valeur:string;
```

```
End;

Type TObjet=class
IdClasse:Integer;
IdObjet:Integer;
Nom:string;
ListAttr:Tlist;
end;
```

6.2. Les méthodes

```
procedure  TRegles.Inference;
    var i:integer;    R:TRegle;
    begin
     for i:=0 to Regles.Count-1 do
     begin
     R:=Regles[i];
     if R.CheckPremisses then
     R.ExecuteActions;
     end;
    end;

function TRegle.CheckPremisses:Boolean;
    var i:integer;     LocP:TPremisse;     flag:boolean;
    begin
    flag:=true;
     for i:=0 to  Premisses.Count-1 do
     begin
     LocP :=  Premisses[i];
     flag:=flag and LocP.Check;
     end;
    end;

function TPremisse.Check: Boolean;
    begin;
    SetValeur(1);
    SetValeur(2);
    Check:= CompareAttr(Valeur1 , Valeur2,Comparateur,FType);
    end;


Function TArbre.Classiffier:string;
    var ps:PChar;
    begin
    ps:=Pchar(Table);
    Result:=Classifier(ps);
    end;
```

7. Exemple base de règles : Aide au voyageur

Regle N° 7

SI
RECHERCHE.TYPE = VOYAGE
ALORS
Ecrire -> ENTREZ VOTRE VILLE DE DEPART...
Lire -> RESERVATION.VILLE1
Ecrire -> VILLE DE DEPART EST : RESERVATION.VILLE1

Regle N° 8

SI
RESERVATION.VILLE1 <> NULL
ALORS
RESERVATION.VILLE2 := LisVoyage.Lines/ RESERVATION.VILLE1
RESERVATION.HEURS1 := LisVoyage.Lines/ HEURS1
RESERVATION.HEURS2 := LisVoyage.Lines/ HEURS2
RESERVATION.VEHICULE := LisVoyage.Lines/ MOYEN
Ecrire -> RECHERCHER UN VOYAGE.

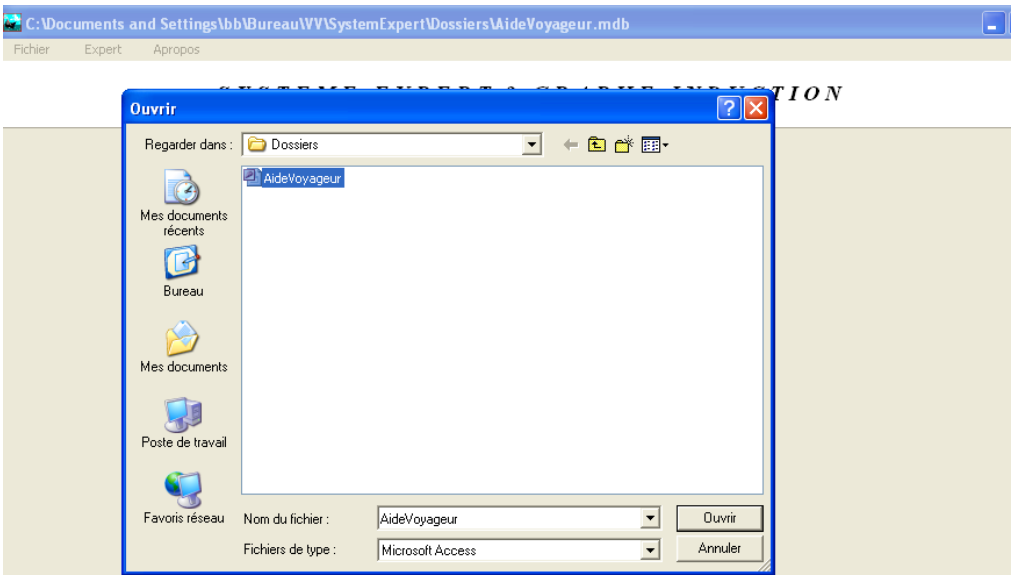
Regle N° 9

SI
RESERVATION.VILLE2 <> NULL
RECHERCHE.TYPE = VOYAGE
ALORS
Ecrire -> VOYAGE PLANIFIER
Ecrire -> VILLE DE DESTINATION EST : RESERVATION.VILLE2
Ecrire -> HEURS DEPART ESET : RESERVATION.HEURS1
Ecrire -> HEURS ARRIVE EST : RESERVATION.HEURS2
Ecrire -> MOYEN TRANSPORT EST : RESERVATION.VEHICULE

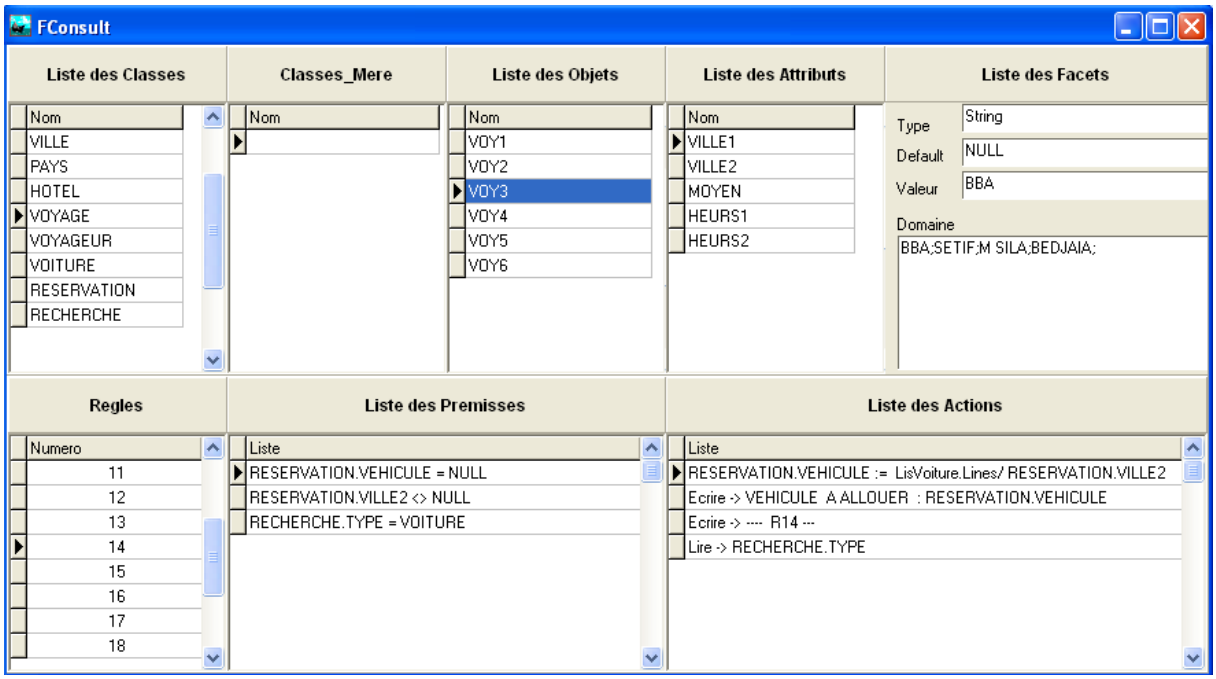
8.Interface

8.1. Présentation des fenêtres principale du logiciel ExpertTreeRule :

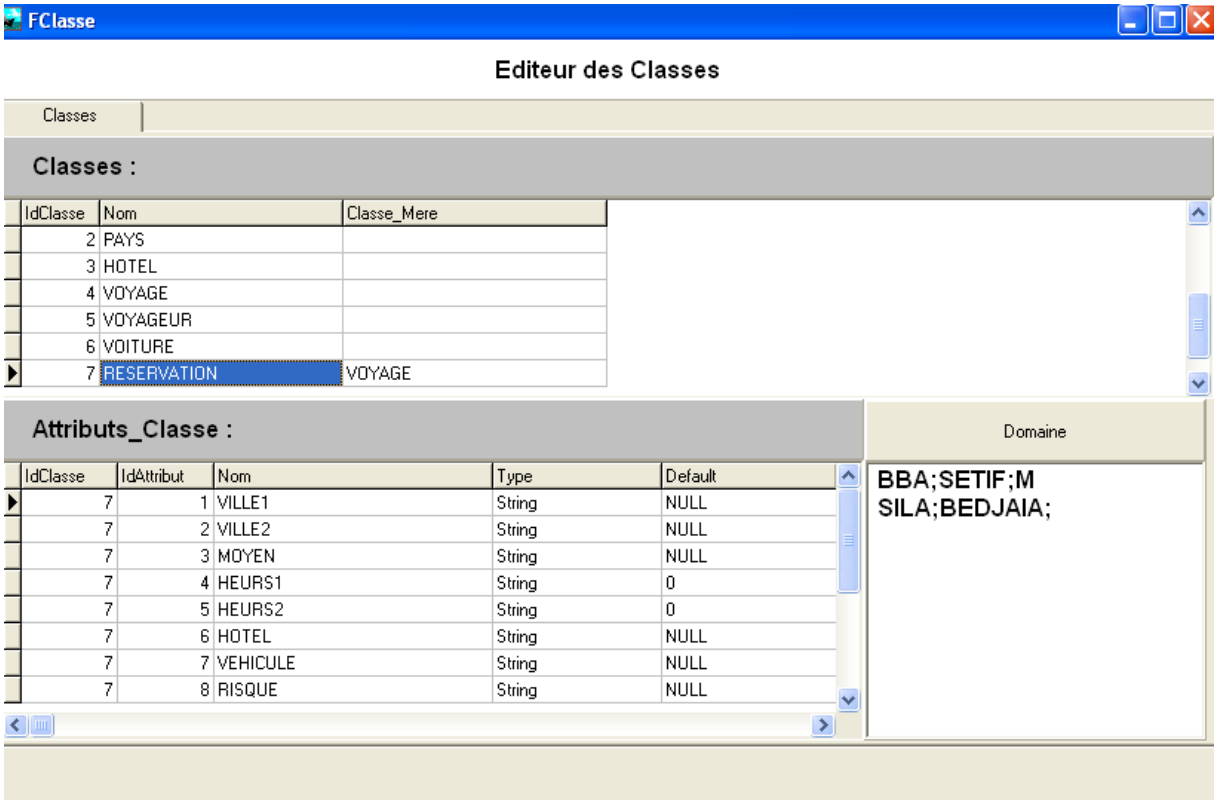
Fiche principal, boite de dialogue pour connecter a une base de données.



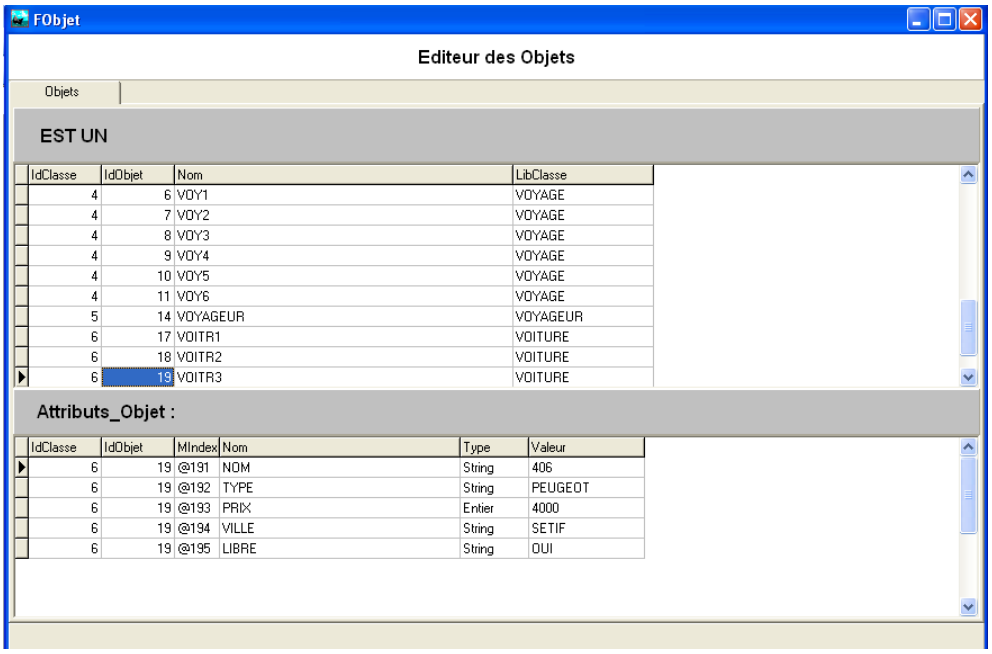
- **Editeur de connaissance :**



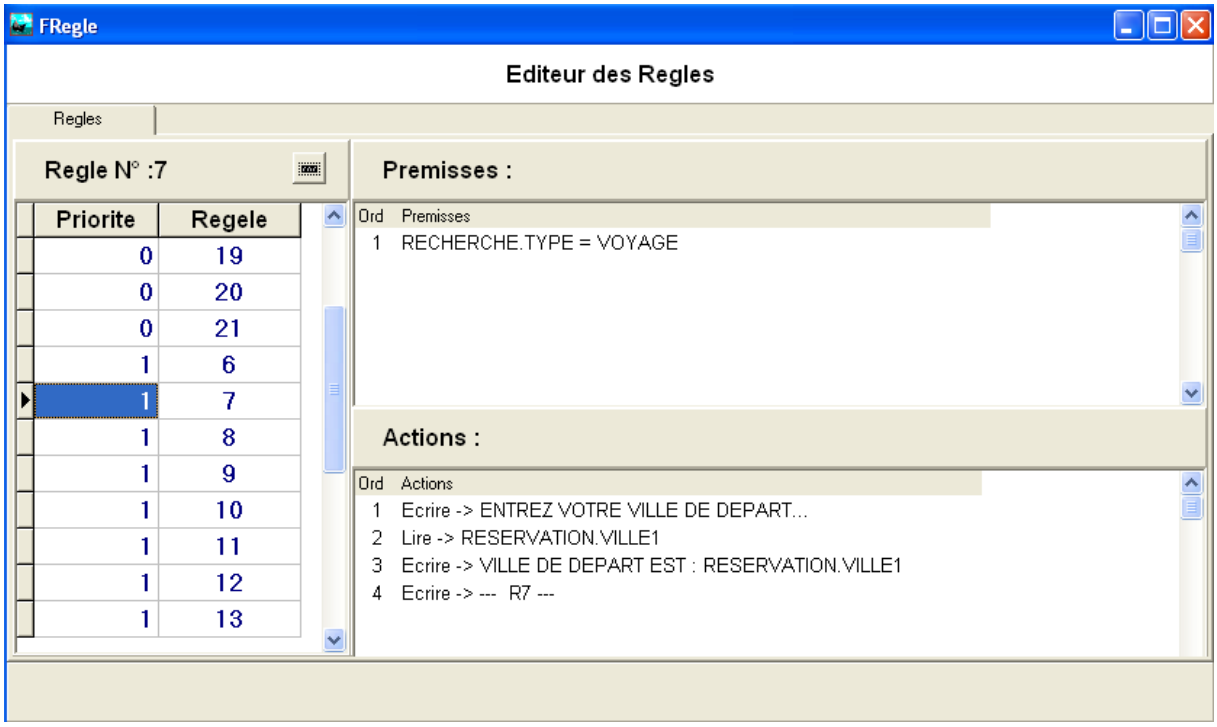
- **Saisie et Modification des classes :**



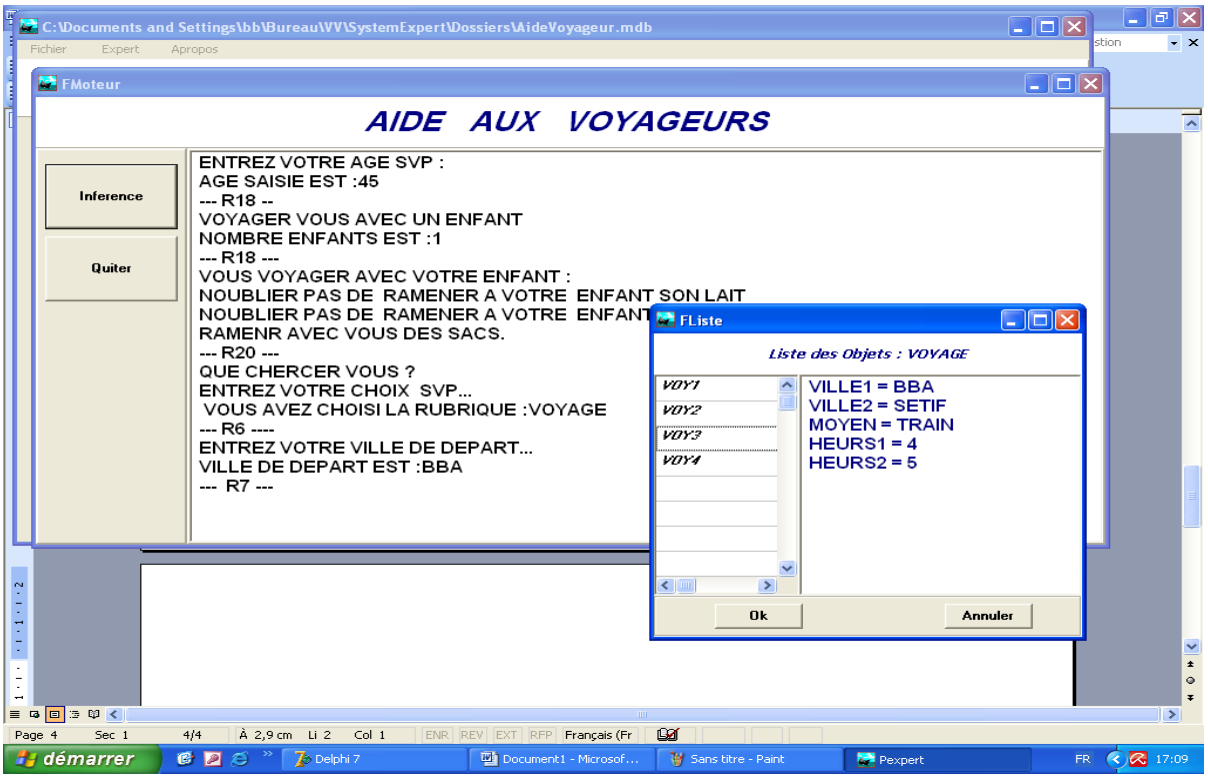
Saisie et Modification des Objets :



- **Saisie et Modification des Regles :**

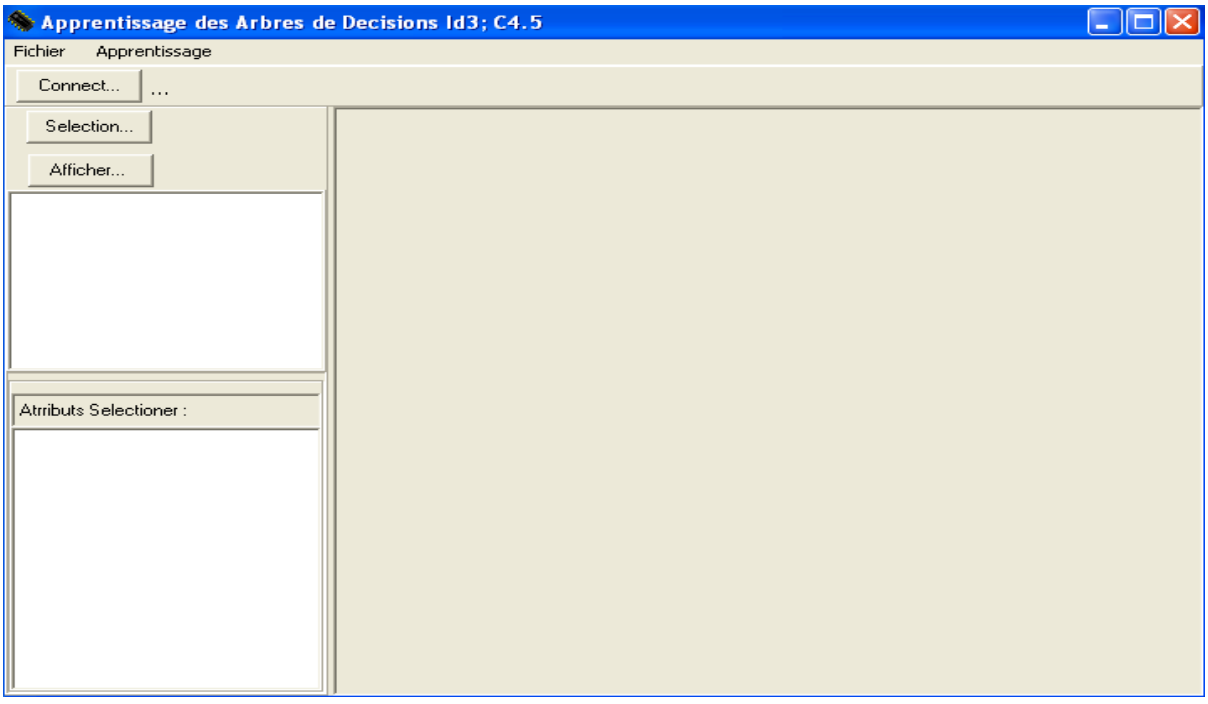


- **Fiche exécution de l'inférence :**

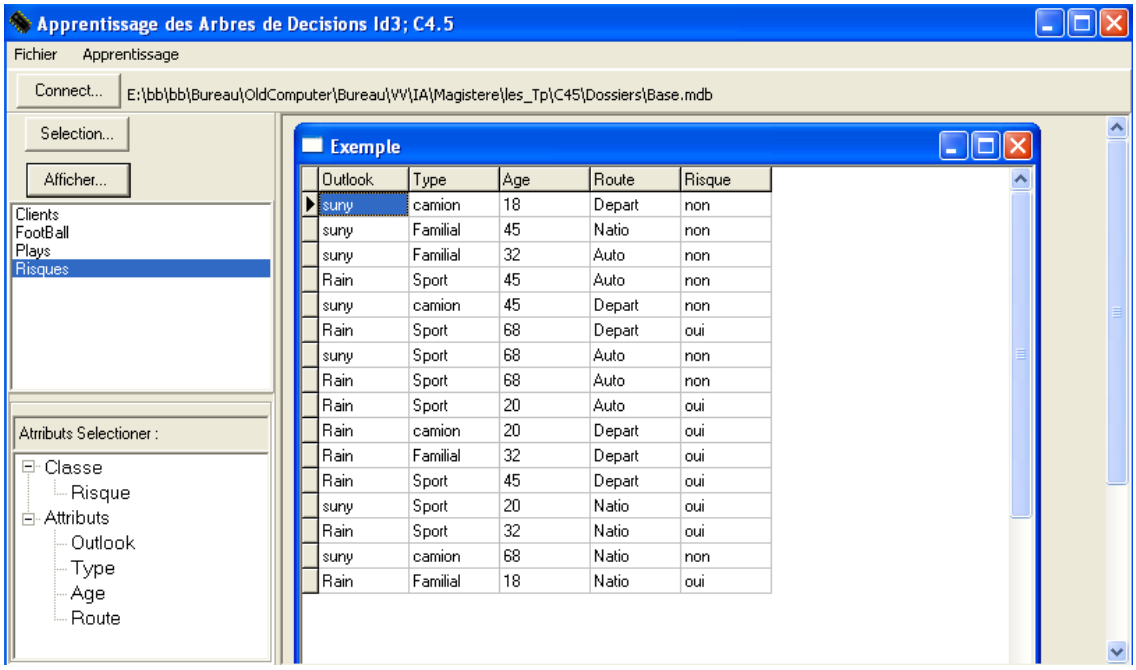


9. Présentation des fenêtres principale du logiciel TreeClassifier :

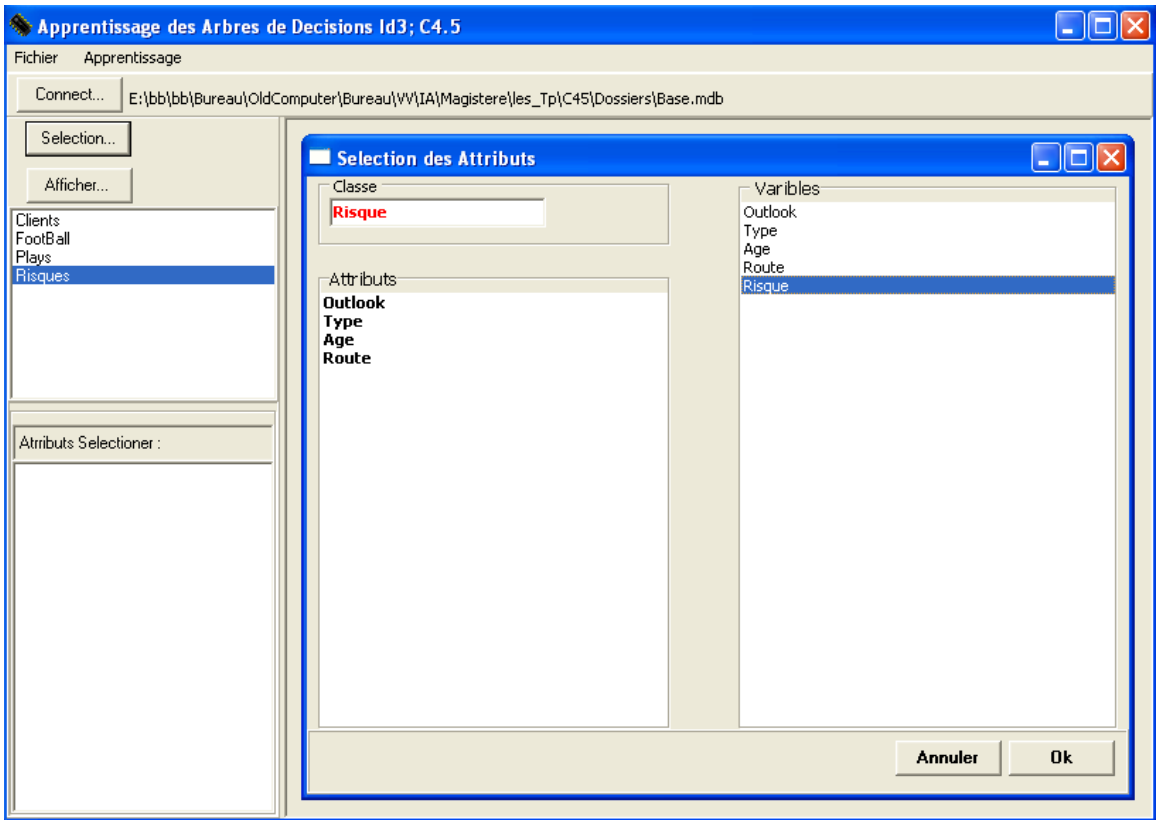
9.1. Fiche principal, boite de dialogue pour connecter a une base de données.



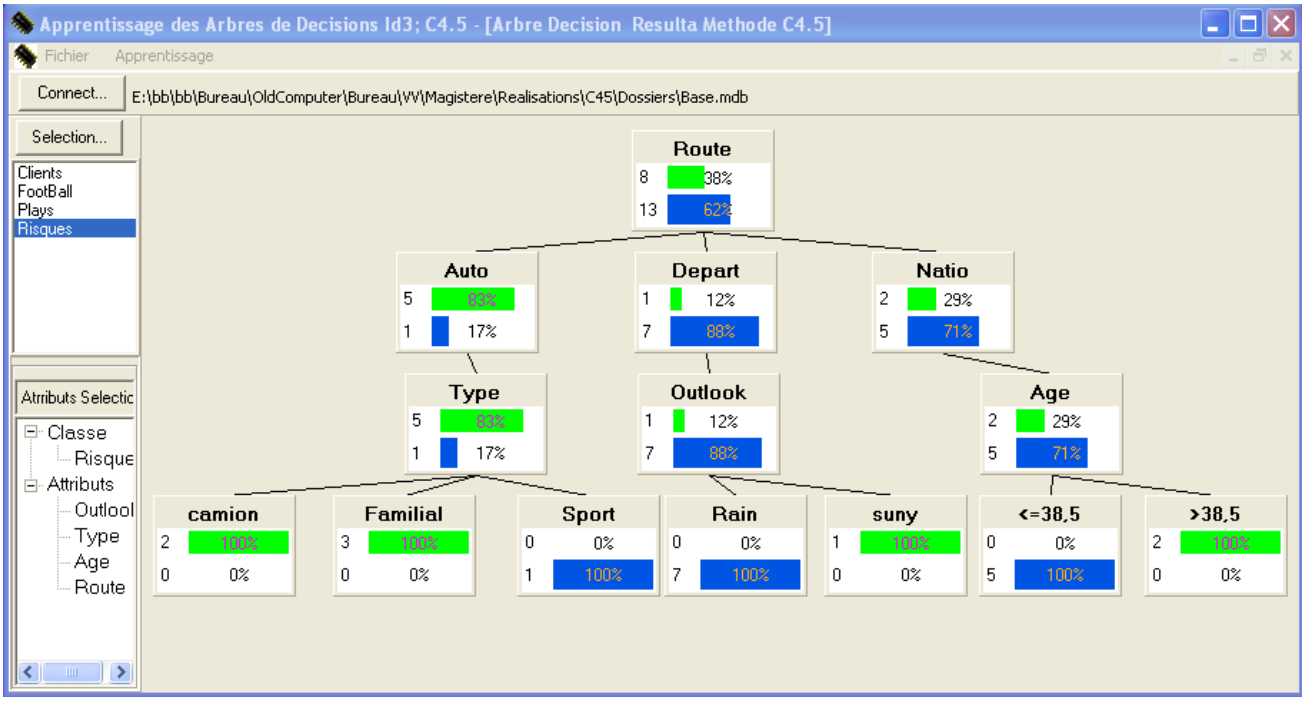
9.2. Sélection et choix de base de données statistique.



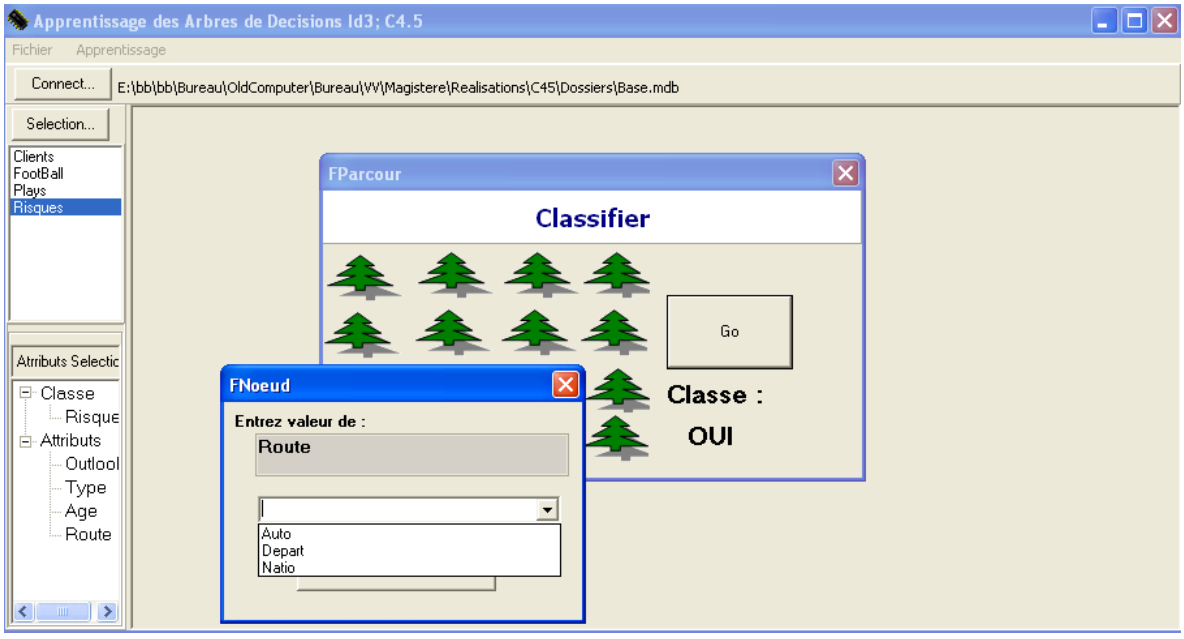
9.3. Sélection des attributs et de Classe pour lancer l'algorithme d'apprentissage .



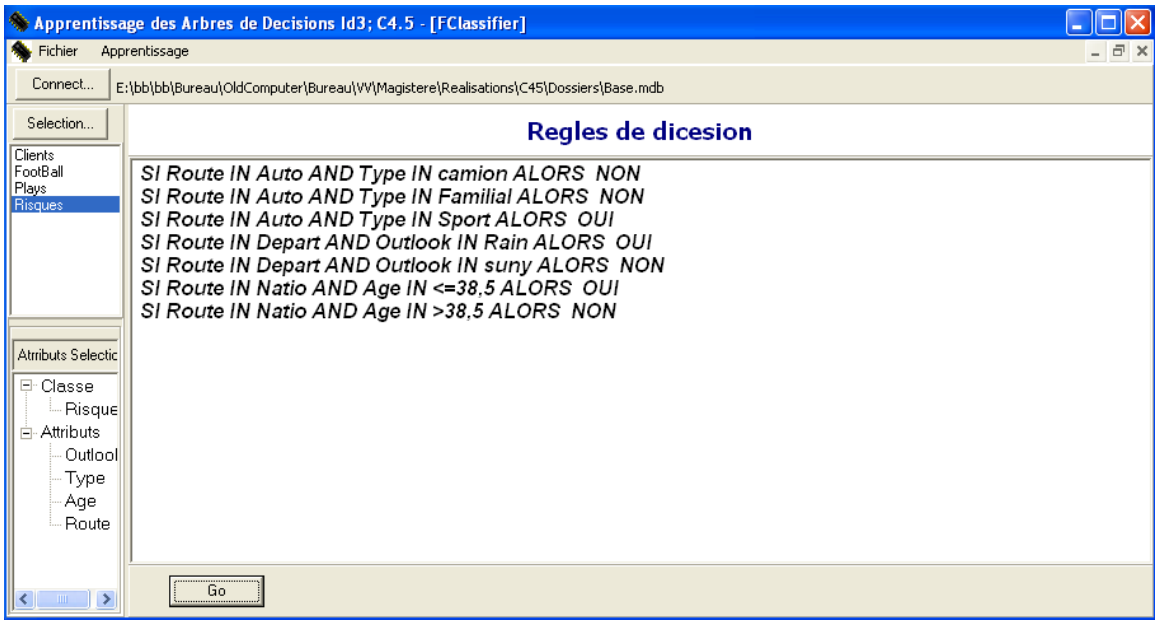
9.4. Exécution de l'algorithme d'apprentissage des arbres de décision C4.5 .



9.5. Exécution de l'algorithme de classification (parcours d'arbre) .



9.6. Génération des règles de Décisions.



Conclusion & perspectives

Les systèmes Expert sont avant tout des logiciels qui simulent le raisonnement d'un expert, ils puisent donc son expérience dans un domaine précis.

Les systèmes experts dits de première génération tentaient de capturer le savoir-faire d'un expert. Cette première vague, si elle a permis des réalisations remarquables, n'a pas donné les résultats escomptés.

Dans le but de dépasser les limitations rencontrées lors de la construction des premiers systèmes à base de connaissances, les systèmes experts de seconde génération appelés aussi « Shell » ont permis un certain nombre d'avancées significatives.

Les principaux travaux qui ont contribué à cette nouvelle génération de systèmes à base de connaissances, sont la construction de systèmes hétérogènes d'une part, qui combinent différents types de connaissances, représentations et méthodes de résolution ; d'autre part la volonté de décrire le processus de résolution à un meilleur niveau d'abstraction «knowledge level».

Les apports de ces différents travaux en ce qui concerne l'acquisition des connaissances, les explications, la robustesse et la réutilisabilité.

De nombreux systèmes experts ont étés implantés avec succès pour résoudre des problèmes concrets. Ce qui est plutôt bon signe du point de vue de l'IA, néanmoins, les grandes difficultés rencontrées pendant l'extraction des connaissances des experts, cependant, leur formalisation forment peut être un point faible difficilement contournable dans les Systèmes Experts.

La représentation Mixte (Objet et Règles de production), la représentation des connaissances qui Intègre des objets et des règles de production est une discipline ancienne. La base de faits d'un système de règles est un modèle d'une situation concrète modélisée, considérés comme des propriétés d'objets constituant le monde modélisé. Les objets apparaissent comme des entités fermées accessibles seulement de l'extérieur par leur interface.

Un des points importants concernant l'intégration de règles et d'objets est la possibilité d'écrire des règles sur des objets quelconques, sans imposer de contraintes sur leur superclasse.

L'inférence peut effectivement être considérée comme une extension naturelle de la programmation par objets ordinaires. L'exécution d'un programme objets effectue une série de modifications d'un ensemble d'objets. De la même manière, l'exécution d'un programme de règles en chaînage avant est un cycle de modifications de la base de faits.

L'apprentissage par induction reste toujours une des principales méthodes étudiées dans le domaine de l'apprentissage automatique. Dans cette approche, on cherche à acquérir des règles générales qui représentent les connaissances obtenues à partir d'exemples. L'algorithme d'apprentissage par induction reçoit un ensemble d'exemples d'apprentissage et doit produire des règles de classification, permettant de classer les nouveaux exemples. Le processus d'apprentissage cherche à créer une représentation plus générale des exemples, selon une *méthode de généralisation de connaissances*. Ce type de méthodes est aussi appelé apprentissage de concepts ou bien acquisition de concepts.

La compilation des techniques cité précédemment nous à permet de réalisé un système Expert puisant, point de vue qu'il est composé de plusieurs modules intégrés : une représentation Hybride (Objet & Règles de production), un moteur inférence d'ordre un (1) manipulant ces objets, un module d'apprentissage par induction (C4.5 de Quinllan), un Module de classification pour classifier les attributs d'objets, et module interrogation de base de données.

Notre système expert ***ExpertTreeRule*** utilise les techniques d'extraction de connaissances à partir de données (E.C.D), en vue de constituer une base de connaissances initiales. Ou interroger une base de connaissance Résultante d'un algorithme extraction des connaissances (E.C.d.) au moment de l'inférence par l'appel de module de classification TreeClassifier, cela permet d'identifier et de classer un Attribut d'un objet dans un domaine précis.

TreeClassifier fait l'objet d'un processus d'apprentissage automatique qui extrait des connaissances à partir des données, la méthode utilisé est C4.5 (Quinlan 96), Le processus d'extraction des connaissances a partir de données est une arme à double tranche, d'une part on peut utilisé la base de connaissance extraite à l'initialisation d'une base de connaissance global de système expert, et d'autre part on peut l'utilisés comme des bases de connaissances secondaires stockés dans des bases de données qui servent a la classification des attributs d'objets dans la base de règles au moment de l'inférence.

Perspectives, comme c'est la première version de ExpertTreeRule, il y a des insuffisances qu'on peut les améliorés on l'utilisant les techniques existantes. Si on peut dire, il y a une infinité de techniques et des méthodes concernant les systèmes expert et l'apprentissage automatique, le principal but est d'améliorés notre moteur d'inférence et d'enrichir notre système par des nouveaux type d'apprentissage.

On première phase on veut améliorer l'ordre d'inférence on passant à d'ordre 2, amélioré les techniques d'inférence par l'utilisation l'algorithme basé sur le réseau RETE [Forgy, 1982]. Attacher des valeurs d'incertitude aux prémisses des règles. Le raisonnement à partir de connaissances incertaines ou imprécis.

Améliorer notre représentation objets par une interface graphique plus souple, implémentation des méthodes dans les classes, objets.

Renforcer l'éditeur de règles par un éditeur de texte puisant qui permet l'implémentation des objets, des procédures, et des fonctions.

Enrichir les techniques d'apprentissage, par la création d'une bibliothèque, on implémente les différentes méthodes d'apprentissage connus : Réseau de neuraux, apprentissage analogique, apprentissage à partir des cas,…etc.

Bibliographie

Aus 2000 N. Aussenac & J. Charlet Ingénierie des connaissances et modélisation

Auss 89 N. Aussenac. Conception d'une méthodologie et d'un outil d'acquisition des connaissances expertes. *Thèse de doctorat en Informatique*. Université Paul Sabatier de Toulouse. Octobre 1989.

BAC 01 Bachimont B, Modélisation linguistique et modélisation logique des ontologiies : l'apport de l'ontologie formelle, In Actes de la conference IC 2001, pages 309-327, Grenoble, 2001.

BART 32 BARTLETT, F. (1932). Remembering. Cambridge Press, Cambridge.

BOUG 01 Boughazala I,Zacklad M, Matta N, Gestion des connaissances dans une entreprise étendue :Mémoire d'entreprise et système d'information coopératifs interentreprises. Revue Extraction des connaissances et apprentissage (Actes de la conférences Extraction et gestion des connaissances, Hermès, 1(1-2) :259-270,2001.

Bouk 06 Boukadoum, Mounir (2006). *Évaluation de variables hydro-électriques: une comparaison de différentes approches d'apprentissage automatique*. Notes de séminaire en informatique cognitive, Université du Québec à Montréal, 23 février.

Bratko et Knononenko, 1987 Bratko, I. et Knononenko, I. (1987). Learning diagnostic rules from incomplete and noisy data,. *Interactions in Arti_cial Intelligence and Statistical Methods*.

Bre 96 Breiman L., *Bagging Predictors*, Machine Learning, 24, 123-140, 1996.

Breim et al 84 Breiman L, Friedman J., Olshen R., Stone C., *Classification and Regression Tree*, California: Wadsworth International, 1984.

Breiman 84 Leo Breiman, Jerome H. Friedman, Richard Olshen, and Charles Stone. Classi- _cation and Regression Trees. Wadsworth and Brooks, Monterey, CA, 1984.

Breiman *et al.*, 1984 Breiman, L., Friedman, J. H., Olshen, R. A. et Stone, C. J. (1984). *Classi_cation and regression trees*. CA : Wadsworth International Group.

BROOKS BROOKS, R. (1991). Intelligence Without Reason. In *IJCAI*, pages 569–595.

Brunet 91 E. Brunet, C. Toussaint et M. Toyé. Ascopa et Sadse: exemples d'application industrielle de la méthode KADS. In *Avignon-91, Vol. 1*, pp 393-406.

BUCH 83 BUCHANAN et al. (1983). Constructing an expert system. In HAYES-ROTH, F., WATERMAN, D., et LENAT, D., (eds.), *Building Expert Systems*. Addison-Wesley, New York.

burgun01 burgun01mappingUMLS Anita Burgun and Olivier Bodenreider. Mapping the UMLS Semantic Network into general ontologies. *Proceedings of the AMIA Annual Symposium*. pp 81-85. 2001.

CAL 99 Callot M,Oldham K,Stockes M, Godwin, Brimblke R, Klein R,Sellini F,Merceron F, Danino D, Methodologie and tools oriented to knowledge based ingeneering applications, Rapport public 2.0,december 1999.

Cestnik et al., 1987 Cestnik, B., Knononenko, I. et Bratko, I. (1987). Assistant 86 : A knowledge elicitation tool for sophisticated users. In I. Bratko and N. Lavrac, editors, Progress in Machine Learning, Sigma Press, Wilmslow, UK, pages 31-45.

chand 99 B. Chandrasekaran, J. R. Josephson and V. R. Benjamins. What are ontologies and why do we need them? *IEEE Intelligent Systems*. 14(1):20-26. 1999.

CHAR, DERM 85 CHARNIAK, E. et McDERMOTT, D. (1985). Introducution to Artificial Intelligence. Addison- Wesley, Reading, MA.

Cho 99 Chouraqui E,Caussanel J, Information et Connaissances : quelles implications pour les projets de capitalisation de connaissances. Document numerique, 3(3-4) : 101-119, 1999.

Clan 85 W. J. Clancy. Heuristic classification. Artificial Intelligence. 27(3) :289 – 350.Decembre 1985

Crémilleux, 2000 Crémilleux, B. (2000). Decision trees as a data mining tool. Computing and Information Systems, pages pp. 91-97.

De Greef 85 P. De Greef, J. Breuker. A case study in structured knowledge acquisition. In *Proc. of the 9th IJCAI*, Los Angeles, CA, pp 390-392, August 1985.

Dieng 90 Dieng R, Méthodes et Outils d'Acquisition des connaissances, Rapport technique 1319 INRIA, Novembre 1990.

Dor 89 Jean-Luc DORMOY**,** Amélioration de l'efficacité du pattern matching dans le langage à base de règles BOOJUM, 1989.

ERM 93 ErmineJL, Genie Logiciel et genie cognitive pour les systeme a base de connaissances ,aspects, methologie(vol1) ,etude de cas ()volume2, technique et documentation – Lavoisier 93

Espo et al 97 Esposito, Floriana; Donato Malerba; & Giovanni Semeraro (1997). A Comparative Analysis of Methods for Pruning Decision Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 476-491.

Ferber 89 Ferber, J. (1989). Objets et agents : une étude des structures de representation et de communications en Intelligence Artificielle, These d'Etat, Paris 6, Paris.

Forg 82 C.L. Forgy. RETE, a fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence Vol. 19.

Fournier et Crémilleux, 2000 Fournier, D. et Crémilleux, B. (2000). Using impurity and depth for decision trees pruning. Second International ICSC Symposium on Engineering of Intelligent Systems (EIS'00), pages pp. 320-326.

GRU 93 Gruber T, A translation approach to portable ontology specification, JKnowledge Acquisition , 5: 199-220,1993.

guarino95a Nicola Guarino and Pierdaniele Giaretta. *Towards very large knowledge bases: knowledge building and knowledge sharing*. Chapter: Ontologies and knowledge bases: Towards a terminological clarification. pp 25-32. IOS Press, Amsterdam. 1995.

guarino97a Nicola Guarino. Understanding, building and using ontologies: a commentary to 'using explicit ontologies in kbs development'. 1997.

Ham 06 Hamilton, Howard J. (2006). *Decision Trees*. Lecture notes (Knowledge Discovery in Databases), University of Regina.

Hunt 62 Hunt E.B., *Concept Learning: An Information Processing Problem*, Wiley, 1962.

Karbach 91 W. Karbach, A. Voss, R. Schukey, U. Drouwen. Model-K: Prototyping at the knowledge level. In *proceedings Expert Systems 1991,* Avignon, France, pp 501-512.

KASS 00 Kassel G, Abel M-H, Barry C, Boulitrea P, Construction et exploitation d'une ontologie pour la gestion des connaissances d'une équipe de recherche, In Actes des journées francophones d'Ingénierie des connaissances IC 2000, pages 251-260, Toulouse, 2000.

Kass 80 Kass G., *An exploratory technique for investigating large quantities of categorical data,* Applied Statistics, 29(2), 119-127, 1980.

Kass 80 G. V. Kass. An exploratory technique for investigating large quantities of categorical data. Applied Statistics, 29 :119.127, 1980.

Kiang Melody Y. Kiang. A comparative assessment of classi_cation methods. Decision Support Systems, 35 :441.454.

Kod 96 Kodratoff Y, Leçons d'Apprentissage symbolique Automatique, Edition Cépadues, Toulouse, 19996.

Laur 76 Jean-Louis Laurière. Un langage et un programme pour résoudre et énoncer des problèmes combinatoires: ALICE. Thèse d'Etat présentée à l'Université Paris 6, mai 1976.

LAUR 82 J.-L. Laurière. Représentation et utilisation des connaissances. *Techniques et science informatique* 1(1):25-42 & 1(2):109-133, 1982

Lounis 06a Lounis, Hakim (2006a). *Apprentissage automatique*. Notes de cours (Séminaire sur l'apprentissage automatique), Programme de Doctorat en Informatique Cognitive, Université du Québec à Montréal.

Lounis 06b Lounis, Hakim (2006b). *Arbres de décision*. Notes de cours (Séminaire sur l'apprentissage automatique), Programme de Doctorat en Informatique Cognitive, Université du Québec à Montréal.

Male et al 96 Malerba, Donato; Floriana Esposito; & Giovanni Semeraro (1996). A Further Comparison of Simplification Methods for Decision-Tree Induction, *Learning From Data: Artificial Intelligence and Statistics* V, D. Fisher and H. Lenz, eds., Lecture Notes in Statistics. Berlin: Springer, no. 112, pp. 365-374.

MALV 94 MalVacheP, Eichenbaum CH,et Prieur P, La Maîtrise du retour d'expérience avec la méthode Rex,Performances Humaines et techniques. Dossier retour d'expérience 69,1994.

Manish 96 Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. Sliq : A fast scalable classi_er for data mining. In Proceedings of the 5th International Conference on Extending Database Technology, pages 18.32. Springer-Verlag, 1996.

Mars et al 99 Marsala, Christophe; & Bernadette Bouchon-Meunier (1999). An Adaptable System to Construct Fuzzy Decision Trees. *Proceedings of the North American Fuzzy Information Processing Society Conference*, NAFIPS'99 , New York, United States, pp. 223-297.

Mingers, 1987 Mingers, J. (1987). Expert systems-rule induction with statistical data. The Journal of the Operational Research Society., 38(1):39-47.

Mingers, 1989 Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. Machine Learning, 4(2):227-243.

MINSKY 75 MINSKY, M. (1975). A Framework for Representing Knowledge. In WINSTON, P., (ed.), The Psychology of Computervision. McGraw-Hill, New York.

Morg,Messg,Thaid 73 Morgan J., Messenger R., *THAID-a sequential analysis program for the analysis of nominal scale dependent variables*, Survey Research Center, U of Michigan, 1973.

Morg,Sonq 63 Morgan J., Sonquist J.A., *Problems in the Analysis of Survey Data, and a Proposal*, Journal of the American Statistical Association, 58:415-435, 1963.

Morgan 63 J. N. Morgan and J. N. Sonquist. Problems in the analysis of survey data, and a proposal. Journal of the American Statistical Association, 58 :415.434, 1963.

New 72 NEWELL, A. et SIMON, H. A. (1972). *Human Problem Solving*. Prentice-Hall, London.

Niblett et Bratko., 1986 Niblett, T. et Bratko., I. (1986). Learning decision rules in noisy domains. Research and Development in Expert Systems III, pages 24-25.

Nils 96 Nilsson, Nils J. (1996). Decision Trees. Chapter 6 of *Introduction to Machine Learning*. Draft notes available at: http://ai.stanford.edu/people/nilsson/mlbook.html .

Paul 91 Paul E. Utgoff and Carla E. Brodley. Linear machine decision trees. Technical Report UM-CS-1991-010, , 1991.

PERDO 99 Pedro Domingos. The role of occams razor in knowledge discovery. Data Min. Knowl. Discov., 3(4) :409.425, 1999.

Pitr 85 Jacques Pitrat. Utilisation des connaissnces déclaratives. Cours fait à l'école Internationale d'informatique de l'AFCET, Aix-en-Provence, juillet 1985. Publications du LAFORIA, Université Paris 6.

Qui 87 Quinlan, J. R. (1987b). Simplifying decision trees. Int. J. Man-Mach. Stud., 27(3):221-234.

Qui 89 Quinlan, John Ross (1989). Unknown attribute values in induction. *Proceedings of the 6th International Workshop on Machine Learning*, Ithaca, New York, United States. pp. 164-168.

Qui 93 J. Ross Quinlan. C4.5 : programs for machine learning. Morgan Kaufmann Publishers Inc., 1993.

Quinlan et Rivest 89 Quinlan, J. R. et Rivest, R. L. (1989). Inferring decision trees using the minimum description length principle. Inf. Comput., 80(3):227-248.

Ragel et Cremilleux, 1998 Ragel, A. et Cremilleux, B. (1998). Treatment of missing values for association rules. In Paci c-Asia Conference on Knowledge Discovery and Data Mining, pages 258-270.

Rissanen, 1984 Rissanen, J. (1984). Universal coding, information, prediction, and estimation. IEEE Transactions on Information Theory, 30(4):629-636.

Sam 59 Samuel A,L, Some studies in machine learning using the game of checkers, IBM journal of research and development – vol 3, n°3 -1959.

SCH, ABEL 77 SCHANK, R. C. et ABELSON, R. (1977). Scripts, Plans, Goals and Understanding. Erlbaum, Hillsdale.

Schreiber 88 G. Schreiber, J. Breuker, B. Bredeweg, and B. Wielinga. Modelling in KBS Development. In *Proceedings of the 2nd European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW 88)*, pp 7.1–7.15, Bonn, RFA, June 1988.

Shafer 96 John C. Shafer, Rakesh Agrawal, and Manish Mehta. Sprint : A scalable parallel classi_er for data mining. In Proceedings of the 22th International Conference on Very Large Data Bases, pages 544.555. Morgan Kaufmann Publishers Inc., 1996.

Shannon 48 C.E. Shannon. A mathematical theory of communication. Bell System Technical Journal, 27 :379.423 and 623.656, July and October 1948.

Shaw, Gain 89 MLG Shaw B.R. Gaines. Knowledge Acquisition : some foundation. Manual methods and current trends, lu procceding of the 3rd Europenn

SWA 97 Swartout B, Patil R, Knight K, Russ T , Towards distributed use of large-scale ontologies, In Spring Symposium Series on Ontological Engineering, Pages 138-148, Stanford University, CA, 1997.

TIX01 Tixier B, la problématique de la gestion des connaissances, le cas d'une enterprise nde développement informatique boncaire, Rapport de Rechercjhe n° 01.9, IRIN, Université de Nantes, 2001.

Tolman 51. TOLMAN, E. C. (1951). A Psychological Model. In PARSONS, T., (ed.), Toward a General Theory of Action. Harper, New York.

TOM 96 Tom M. Mitchell. Machine learning, chapter 3. McGraw Hill, New York, US, 1996.

Utg 89 Utgoff, Paul E. (1989). Incremental Induction of Decision Trees. *Machine Learning*, vol. 4, 161-186.

Vanich 86 N. Vanichsetakul. Tree-Structured Classi_cation via Recursive Discriminant Analysis. PhD thesis, Department of Statistics, University of Wisconsin, Madison, 1986.

Vanich 88 N. Vanichsetakul and Wei-Yin Loh. Tree-structured classi_cation via generalized discriminant analysis (with discussion). Journal of the American Statistical Association, 83 :715.728, 1988.

Vial 85 Michèle Vialatte. Description et applications du moteur d'inférence SNARK. Thèse de l'Université Paris 6, mai 1985.

WAT 85 WATERMAN, D. (1985). *A Guide to Expert Systems*. Addison-Wesley, New York.

Win92 P. WINSTON (1992). Artificial Intelligence. Addison Wesley.

Witt,Frank 89 Witten, Ian H.; & Eibe Frank (2nd edition, 2005). Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann.

WYin 97 Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classi_cation trees. Statistica Sinica, 7 :815.840, 1997.

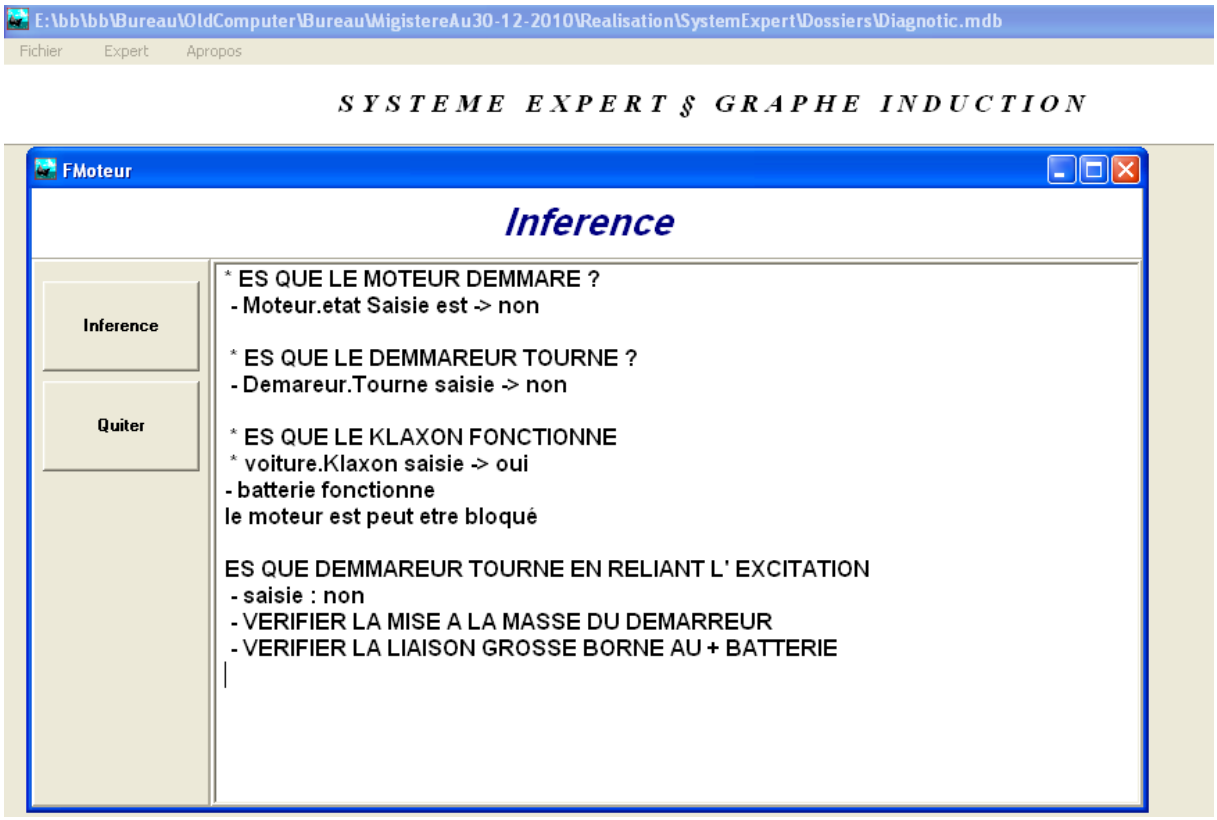
Annexe

ANEXE 1 : modélisation et implémentation d'une base de connaissance :

Diagnostic à la panne d'un véhicule.

1. Représentation de la fiche Inférence durant l'exécution du moteur d'inférence

Affichage de l'interface utilisateur, ici on montre une séquence d'exécution de règles de production, le dialogue utilisateur & machine, de la base diagnostic au pane véhicule.



2. Représentation des classes :

La modélisation de dossier diagnostic a donné selon notre modélisation les classes suivantes :

1- Voiture ; 2- Moteur ; 3- Démarreur ; et 4-Lanceur. ces classes sont représentés hiérarchiquement comme suit :

Voiture ->Moteur->Démarreur
->Lanceur.

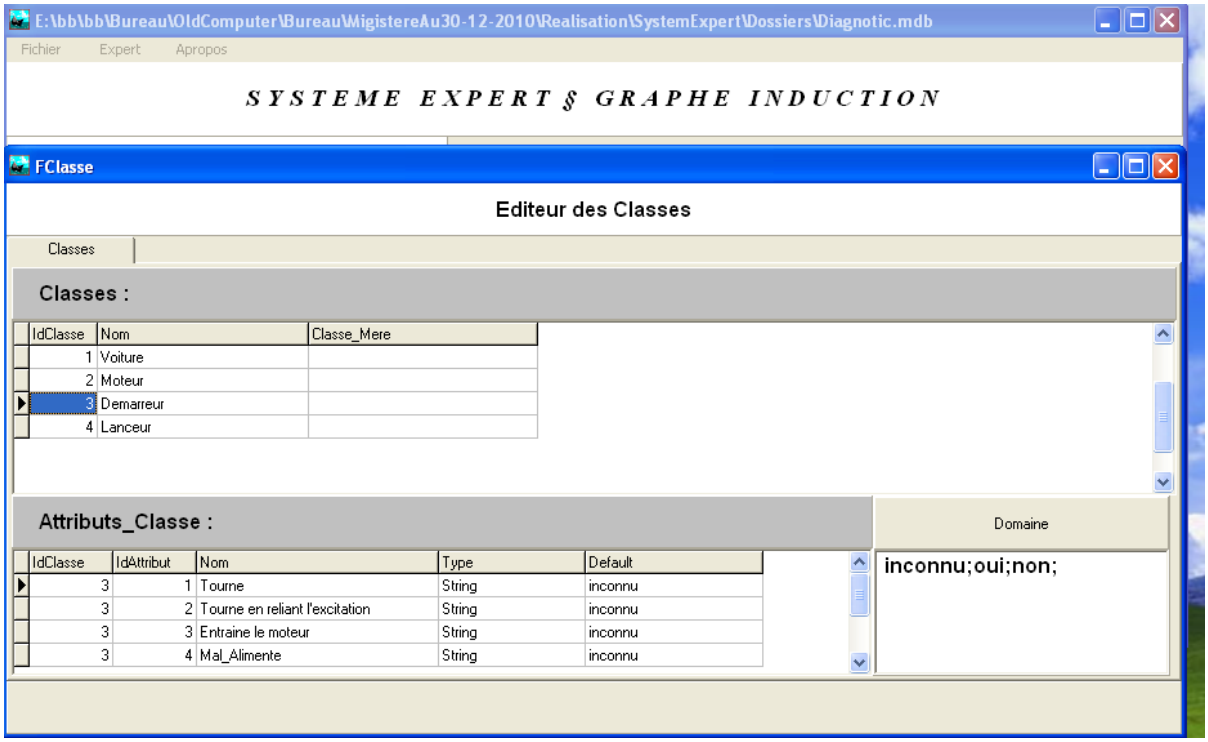
Fin.

Les classes sont définies et implémentées comme suit :

Classes.Nom	Classeproprités	Type	Domaine
Voiture	*Vitesse en prise*	*String*	*oui;non;inconnu;*
Voiture	*Batterie_Fonctionne*	*String*	*oui;non;inconnu;*
Voiture	*Batterie_Fonctionne*	*String*	*oui;non;inconnu;*
Voiture	*Moteur*		
Moteur	*Demare*	*String*	*oui;non;inconnu;*
Moteur	*demarreur*		
Moteur	*lanceur*		
Demarreur	*Tourne*	*String*	*inconnu;oui;non;*

Classes.Nom	Classeproprités	Type	Domaine
Demarreur	*Tourne en reliant l'excitation*	*String*	*inconnu;oui;non;*
Demarreur	*Entraine le moteur*	*String*	*inconnu;oui;non;*
Demarreur	*Mal_Alimente*	*String*	*inconnu;oui;non;peut-etre;*
Demarreur	*bloque*	*String*	*inconnu;oui;non;peut-etre;*
Lanceur	*bloque*	*String*	*inconnu;peut-etre;oui;non;*

- **La partie implémentation des classes :**

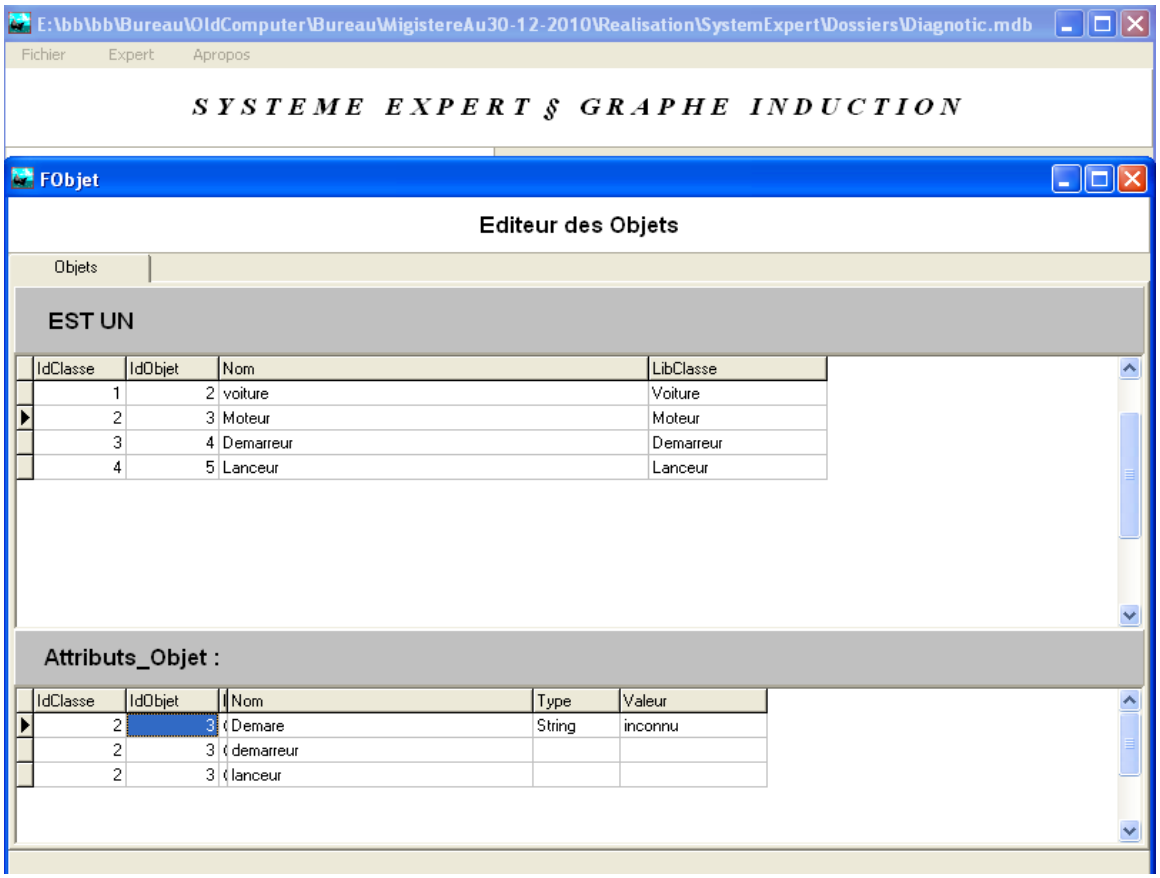


3. Représentation des Objets :

- **Création et définition des objets :**

IdObiet	*Obiets*	*Attributs*	*Valeur*
2	***voiture***	***Vitesse en prise***	***inconnu***
2	***voiture***	***Batterie_Fonctionne***	***inconnu***
2	***voiture***	***Klaxon_Fonctionne***	***inconnu***
2	***voiture***	***Moteur***	
3	***Moteur***	***Demare***	***inconnu***
3	***Moteur***	***demarreur***	
3	***Moteur***	***lanceur***	
4	***Demarreur***	***Tourne***	***inconnu***
4	***Demarreur***	***Tourne en reliant l'excitation***	***inconnu***
4	***Demarreur***	***Entraine le moteur***	***inconnu***
4	***Demarreur***	***Mal_Alimente***	***inconnu***
4	***Demarreur***	***bloque***	***inconnu***
5	***Lanceur***	***bloque***	***inconnu***

- **Implémentation et saisie des objets :**

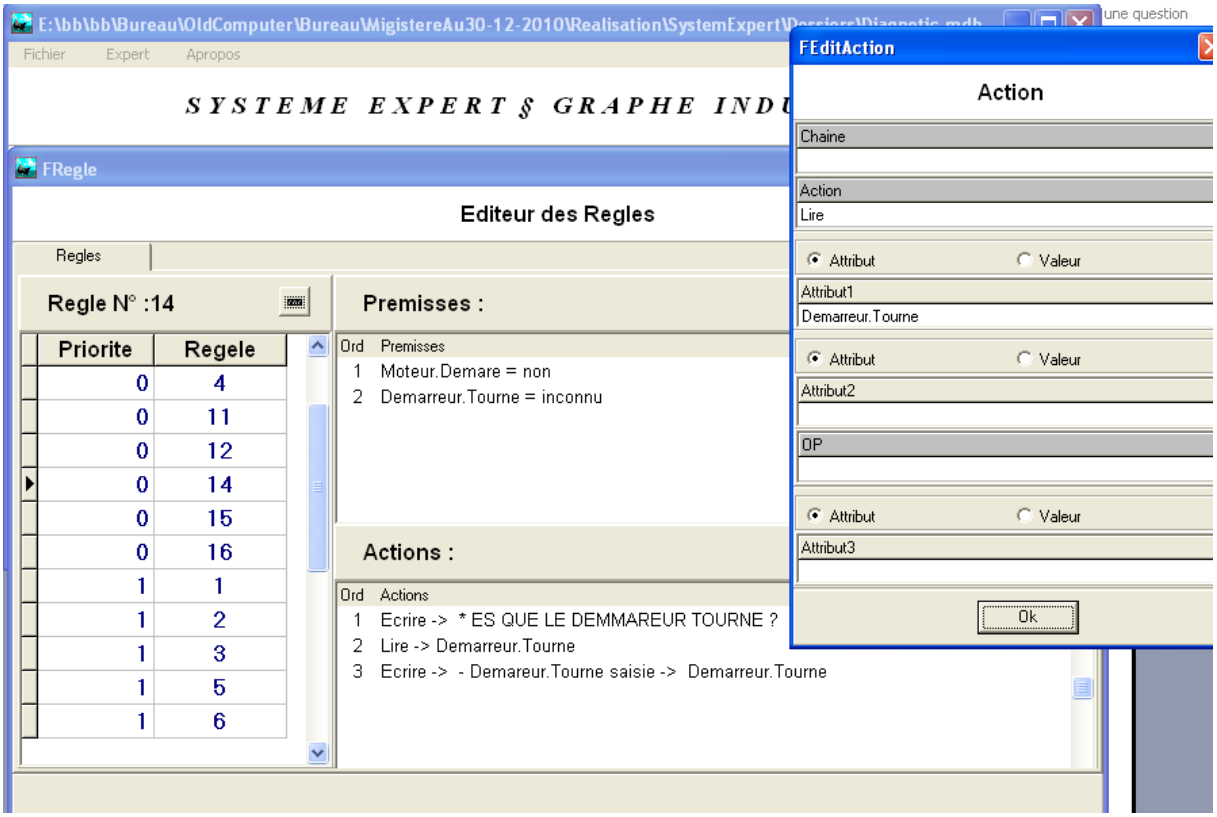


4. Extrait de Règles de production de la base de connaissance diagnostic

Id Règle	*Priorité*	*Numéro*	*texte*
1	*1*	*1*	*SI Demarreur.Tourne = non*
1	*1*	*2*	*SI voiture.Batterie_Fonctionne = oui*
1	*1*	*1*	*Ecrire le moteur est peut etre bloqué*
1	*1*	*2*	*Action Moteur.Demare := non*
1	*1*	*3*	*Action Demarreur.bloque := peut_etre*
1	*1*	*4*	*Action Demarreur.Mal_Alimente := peut_etre*
2	*1*	*1*	*SI Moteur.Demare = non*
2	*1*	*2*	*SI voiture.Vitesse en prise = oui*
2	*1*	*1*	*Ecrire * mettre au point mort **
3	*1*	*1*	*SI Moteur.Demare = non*
3	*1*	*2*	*SI voiture.Vitesse en prise = non*
3	*1*	*1*	*Ecrire - segments colles. mettre de l'huile par les orifices de bougies*
3	*1*	*2*	*Ecrire - rupture interne d'organe. demontage moteur*
4	*0*	*1*	*SI Moteur.Demare = non*
4	*0*	*2*	*SI voiture.Vitesse en prise = inconnu*
4	*0*	*1*	*Ecrire * ES QUE VITESSE EN PRISE ?*
4	*0*	*2*	*Action voiture.Vitesse en prise Lire*
4	*0*	*3*	*Action voiture.Vitesse en prise Ecrire VITESSE EN PRISE SAISIE EST ->*
5	*1*	*1*	*SI Demarreur.Mal_Alimente = peut_etre*

Id Règle	Priorité	Numéro	texte
5	1	2	SI Demarreur.Tourne en reliant l'excitation = inconnu
5	1	1	Ecrire ES QUE DEMMAREUR TOURNE EN RELIANT L' EXCITATION
5	1	2	Action Demarreur.Tourne en reliant l'excitation Lire
5	1	3	Action Ecrire Demarreur.Tourne en reliant l'excitation - saisie :
6	1	1	SI Demarreur.Mal_Alimente = peut_etre
6	1	2	SI Demarreur.Tourne en reliant l'excitation = non
6	1	1	Ecrire - VERIFIER LA MISE A LA MASSE DU DEMARREUR
6	1	2	Ecrire - VERIFIER LA LIAISON GROSSE BORNE AU + BATTERIE
7	1	1	SI Demarreur.Mal_Alimente = peut_etre
7	1	2	SI Demarreur.Tourne en reliant l'excitation = oui
7	1	1	Ecrire - le lanceur est peut-etre bloque
7	1	2	Ecrire - verifier la liaison cle de contact a l'excitationverifier
7	1	3	Action Lanceur.bloque := peut_etre
8	1	1	SI Lanceur.bloque = peut_etre
8	1	1	Ecrire - verifier la liaison cle de contact a l'excitation
8	1	2	Ecrire - demonter le lanceur et degriper/verifier le pignon
9	1	1	SI Demarreur.bloque = peut_etre
9	1	2	SI Lanceur.bloque = peut_etre
9	1	1	Ecrire - verifier si demarreur bloque (axe grippe/bobinages HS)
10	1	1	SI Moteur.Demare = oui
10	1	2	SI Demarreur.Tourne = oui
10	1	3	SI Lanceur.bloque = inconnu
10	1	1	Ecrire - lanceur est peut-etre bloque
10	1	2	Action Lanceur.bloque := peut-etre
11	0	1	SI Demarreur.Entraine le moteur = non
11	0	2	SI Demarreur.Tourne = oui
11	0	3	SI Lanceur.bloque <> peut_etre
11	0	1	Ecrire - la couronne/le pignon de demarreur sont uses
12	0	1	SI Moteur.Demare = inconnu
12	0	1	Ecrire * ES QUE LE MOTEUR DEMMARE ?
12	0	2	Action Moteur.Demare Lire
12	0	3	Action Moteur.Demare Ecrire - Moteur.etat Saisie est ->
14	0	1	SI Moteur.Demare = non
14	0	2	SI Demarreur.Tourne = inconnu
14	0	1	Ecrire * ES QUE LE DEMMAREUR TOURNE ?
14	0	2	Action Demarreur.Tourne Lire
14	0	3	Action Demarreur.Tourne Ecrire - Demareur.Tourne saisie ->
15	0	1	SI Demarreur.Tourne = non
15	0	2	SI voiture.Batterie_Fonctionne = inconnu
15	0	1	Ecrire * ES QUE LE KLAXON FONCTIONNE
15	0	2	Action voiture.Klaxon_Fonctionne Lire
15	0	3	Action voiture.Klaxon_Fonctionne Ecrire * voiture.Klaxon saisie ->
16	0	1	SI voiture.Klaxon_Fonctionne = oui
16	0	1	Ecrire - batterie fonctionne
16	0	2	Action voiture.Batterie_Fonctionne := oui

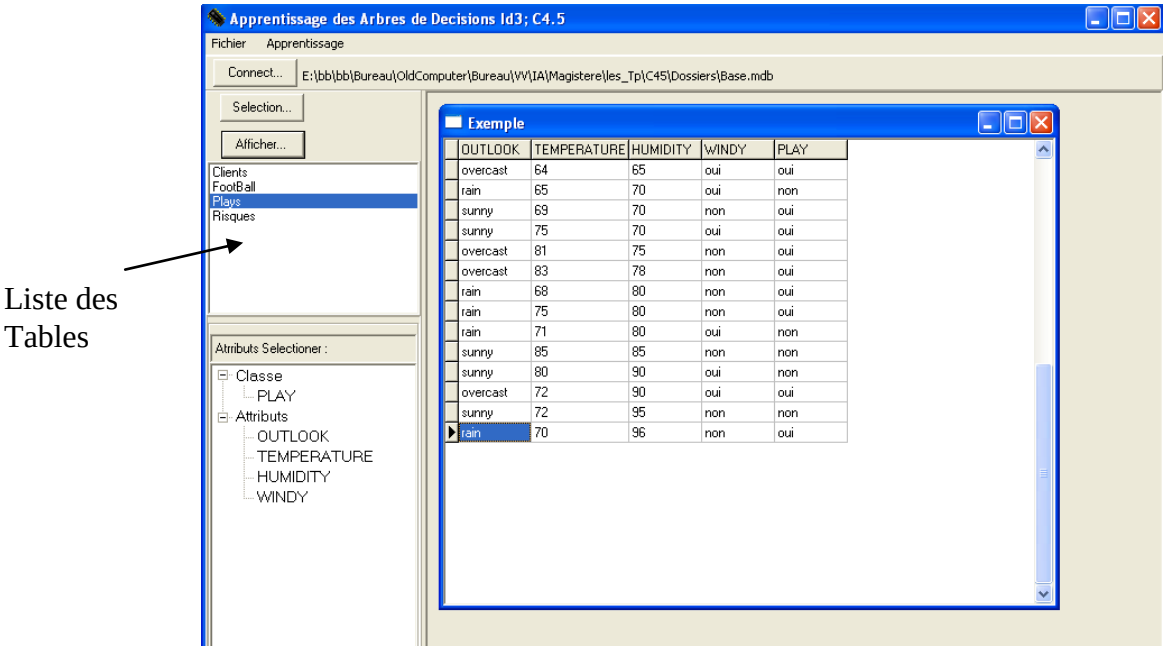
- **Implémentation et saisie des regèles de productions**



ANEXE 2 : Application des arbre de décision sur l'exemple jouer au football.

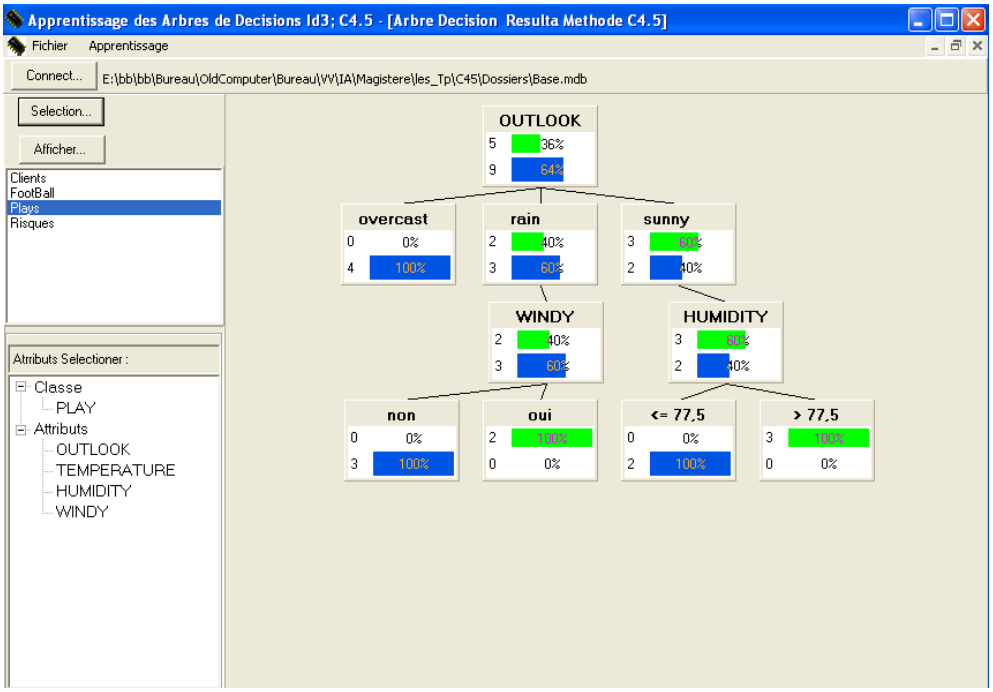
1. l'échantillon d'apprentissage

la première phase est de connecter a la base données statistique, le liste des tables est apparu a gauche de la fenêtre, juste en bas on trouve les attributs en sélection avec la classe choisie.



Liste des Tables

2. Application de l'algorithme d'apprentissage C4.5



3. Code source de l'algorithme C4.5 sous DELPHI 7

```
procedure TFMain.Algoc45(LocData:TAdoDataSet;LocNode:TTreeNode);
var i,Indice,IndexTree,Count:integer;
   LocArrayGain:TCold;
   LocArrayClass:TColi;
   Colones:TCols;
   Min:Double;
   Colone:string;
   MyTreeNode1,MyTreeNode2:TTreeNode;
   LocAdo:TadoDataSet;
   LocAdoAnalyse,ResteData,ResteAnalyse:TadoDataSet;
   LabelNode,ValNode,TextTree:String;
begin
Colones:=GetColoneAtrribut(LocData);
LocArrayGain:=GetArrayfGainC45(LocData);
Count:=Length( LocArrayGain);
Min:=1;
for i:=0 to Count-1 do
 if Min > LocArrayGain[i] then
  begin
   Min:= LocArrayGain[i];
   Indice:=i;
  end;
Colone:=Colones[Indice];
UpdateColNumToStr(LocData,Colone);
LocAdoAnalyse:=ColAnalyse(LocData,Indice);
LocArrayClass:= GetArrayClass(LocAdoAnalyse);
with TreeView1.Items do
 begin
  MyTreeNode1 :=AddChild(LocNode,Colone);
  FArbrePanel.InserLigneTree(LocNode,MyTreeNode1,Min,LocArrayClass);
  LocAdoAnalyse.First;
//----------------
 while not (LocAdoAnalyse.Eof) do
  begin
    LabelNode:=LocAdoAnalyse.Fields[0].FieldName;
    ValNode:=LocAdoAnalyse.Fields[0].Value;
   if (SameValueClass(LabelNode,ValNode,LocData)) or (Length(Colones)=1) then
    begin
     MyTreeNode2:=CreerFeuille(MyTreeNode1,LabelNode,ValNode,LocData);
     ResteData:=FiltreData(LabelNode,ValNode,LocData);
     ResteAnalyse:=ColAnalyse(ResteData,Indice);
     LocArrayClass:= GetArrayClass(ResteAnalyse);
     FArbrePanel.InserLigneTree(MyTreeNode1,MyTreeNode2,99,LocArrayClass);
     end
   else
    begin
     MyTreeNode2:=CreerNoeud(MyTreeNode1,ValNode);
     LocAdo:=RestData(LocData,LabelNode,ValNode);
     ResteAnalyse:=ColAnalyse(LocAdo,Indice);
     LocArrayClass:= GetArrayClass(ResteAnalyse);
     FArbrePanel.InserLigneTree(MyTreeNode1,MyTreeNode2,Min,LocArrayClass);
     AlgoC45( LocAdo ,MyTreeNode2 );
    end;//--else if
  LocAdoAnalyse.Next;
  end;
 end;
end;
```

ANEXE 3 : PRESENTATION DES ALGORITHMES

1. Algorithme d'inférence :

Procedure *ChaınageAvant (F,R)*
Q ← faits de d´epart F
while *Q n'est pas vide* ***do***
q ← premier(Q), Q ← reste(Q)
ajouter q à la base de données
if *q est une solution* ***then*** *imprimer q*
for *chaque règle r de la base de règles R* ***do***
if *q est dans conditions(r)* ***and*** *conditions(r) est contenu dans la base de données*
and non *conclusion(r) est dans la base de donnees then*
ajouter conclusion(r) à en queue de Q
end if
end for
end while

Figure 01: Algorithme pour un moteur d'inférence en chaînage avant.

2. Algorithme construction d'arbre de decision :

procedure *construirearbre* ***(n : noeud, d : donnees, ll : methode)***
appliquer ll sur d pour trouver le critere de division ***;***
diviser ***n*** en utilisant le critere de division ***;***
k nombre des sous noeuds de ***n ;***
if k > 0 ***then***
fabriquer ***k*** sous-noeuds ***n1…….nk*** de ***n ;***
diviser d en ***d1……dk ;***
for i:= 1 ***to k do***
construirearbre ***(ni, di, ll) ;***
end for
end if
end procedure

Figure 02: algorithme construction arbre

3. Algorithme id3;

```
Procedure id3 (n : noeud, d : donnees)
 if tous les elements du d sont a meme classe  then
     exit ;
          end if
   for chaque attribut a do
     calculer informationobtenue(d;a) ;
          amax ← argmaxa(informationobtenue(d;a) ;
          diviser n en utilisant amax ;
          k ← nombre des sous-noeuds de n ;
               if k > 0 then
               fabriquer k sous-noeuds n1…….. nk de n ;
               diviser d en d1……..dk ;
              for i :=1 to k do
                   id3 ( ni , di , ll ) ;
              end for
     end if
  end for
end procedure
```

Figure 03: algorithme id3

4. Algorithme d'élagage;

```
     procedure  elaguer (arbre)
      racine la racine du arbre ;
      if racine est une feuille then
      return erreur(racine) ;
     end if
      erreurnoeud ← erreur(racine) ;
      erreurarbre ← 0 ;
     for all branche i  de l' arbre do
     erreurarbre ← erreurarbre + elaguer(branche i) ;
     end for
      branche ← le branche ayant les exemples les plus ;
     erreur meilleur branche ← elaguer(branche) ;
     if erreur noeud <= erreurarbre et erreurmeilleurbranche then
               arbre ← racine ;
               return erreurnoeud ;
               else
                  if erreur meuilleur branche <= erreur arbre then
                  arbre ←branche ;
                   return erreur meilleur branche ;
                  end if
     end if
     return erreur arbre ;
      fin
```

Figure 04: algorithme d'élagage.

ANEXE 4 : Loi Binomial & pessimiste error pruning.

1. Loi Binomial

La distribution binomiale est couramment utilisée dans les statistiques dans une variété d'applications. Une expérience binomiale est procédée dans lequel:

- L'expérience consiste à **n** essais identiques.
- Il n'y a que deux résultats à chaque essai de l'expérience. Un des résultats est généralement considéré comme un **succès,** et l'autre comme un **échec.** Le succès et l'échec sont des termes un mauvais choix de la terminologie puisque l'attribution des résultats est arbitraire et il n'y a pas d'incidence que les résultats sont bons ou mauvais.
- La probabilité de succès à un essai est **p** et ne change pas tout au long de l'expérience. La probabilité d'un échec est **q = 1 - p** et également ne pas changer au cours de l'expérience.
- Les tirages sont indépendants.
- La **variable aléatoire binomiale** est le nombre de succès dans les essais **n.**

$$\text{Loi Binomial} = C_N^E p^E (1-p)^{N-E}$$

1.1. Intervalle de confiance binomial

Il existe plusieurs façons de calculer un intervalle de confiance pour une proportion binomiale. L'intervalle approximation normale est la formule la plus simple, et celui présenté dans les classes statistiques les plus élémentaires et les manuels scolaires. Cette formule, toutefois, est basée sur une approximation qui ne fonctionne pas toujours bien.

Plusieurs formules sont disponibles concurrents plus performants, en particulier pour les situations avec une taille réduite de l'échantillon et une proportion très proche de zéro ou un. Le choix de l'intervalle dépend de combien il est important d'utiliser un simple et facile d'expliquer l'intervalle par rapport au désir d'une meilleure précision.

1.2. Intervalle de confiance Exact

Les lacunes dans l'approximation normale (1- Précision souffre quand n*p <5 ou n* (1-p) <5 ; 2- Calcul pas possible lorsque p = 0 ou p = 1) ont été traitées par Clopper et Pearson quand ils ont développé la méthode de Pearson-Clopper qui est communément appelé la "confiance exact Interval " Au lieu d'utiliser une approximation normale, l'inverse exact CI bilatéral binomial test unique à l'alpha désiré. Plus précisément, le CI est exacte gamme de $\mathbf{p}_{\mathbf{lb}}$ à $\mathbf{p}_{\mathbf{ub}}$ qui satisfait aux conditions suivantes.

$$\sum_{k=0}^{k} \binom{n}{k} {p_{UB}}^k (1-p_{UB})^{n-k} = \frac{\alpha}{2}$$

$$\sum_{k=x}^{n} \binom{n}{k} {p_{LB}}^k (1-p_{LB})^{n-k} = \frac{\alpha}{2}$$

La proportion de la population se situe dans la gamme $\mathbf{p}_{\mathbf{lb}}$ à $\mathbf{p}_{\mathbf{ub}}$ où:
$\mathbf{p}_{\mathbf{lb}}$ est l'intervalle de confiance borne inférieure
$\mathbf{p}_{\mathbf{ub}}$ est l'intervalle de confiance borne supérieure
n est le nombre d'essais
k est le nombre de succès dans les essais n
α est l'intervalle de confiance désiré.

Bien que la méthode normale rapprochement est facile à apprendre et à comprendre. Bien que la proportion de la population se situe dans la gamme $\mathbf{p_{lb}}$ à $\mathbf{p_{ub}}$, le calcul de ces valeurs est non négligeable et pour la plupart nécessite l'utilisation d'un ordinateur. Les équations ci-dessus sont basées sur la fonction de distribution binomiale cumulative (cdf).

La distribution bêta peut être utilisée pour calculer la cdf binomiale, et donc une manière plus commune pour représenter le binôme CI est exacte en utilisant les équations ci-dessous.

$$p_{ub} = 1 - \text{BetaInv}\left(\frac{1-\alpha}{2}, n-k, k+1\right)$$

$$p_{lb} = 1 - \text{BetaInv}\left(1 - \left(\frac{1-\alpha}{2}\right), n-k+1, k\right)$$

2. Post élagage des arbres de décision

- Pour un noeud donnée, soit **N** le nombre d'exemples arrivant en ce noeud.
- Soit, parmi ces **N** exemples, **E** le nombre d'exemples mal classées (dans la suite de l'arbre).
- Le taux d'erreur **E / N** est manifestement optimiste.
- Trouver une valeur plus pessimiste de la probabilité d'erreur en ce noeud.

2.1. Estimation pessimiste de l'erreur

- Soit **p** la " vraie" probabilité d'erreur en ce noeud.
- Soit **E/N** la probabilité optimiste constatée en ce noeud.
- Quelle est la valeur maximale de **p** qui fait qu'une erreur de **E/N** ne soit pas rare ?
- Quelle est la valeur maximale de **p** telle que obtenir **E/N** erreurs s'observe dans au moins **25%** des cas ?

2.1.1. utilisation de l'intervalle de confiance de la Loi Binomial

- Soit **p** la probabilité de mal classer un exemple.

Probabilité de rencontrer **E** erreurs pour **N** exemples (Loi binomiale) = $C_N^E p^E (1-p)^{N-E}$

- Calculer l'intervalle de confiance pour $C_N^E p^E (1-p)^{N-E}$ =0.25
- **p** est l'estimation pessimiste de l'erreur.

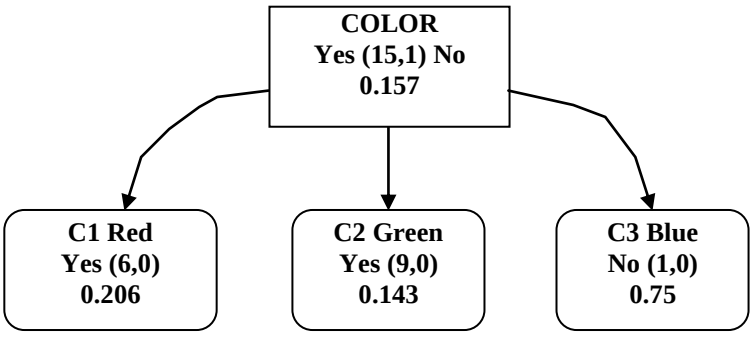
$$p_{lb} = 1 - \text{BetaInv}\left(1 - \left(\frac{1-\alpha}{2}\right), n-k+1, k\right)$$

2.1.2. Test et Implémentation

Exemple : si on considérer le tableau suivant qui contient 16 ligne observé sur l'apparition des jouets, nous voulons savoir s'ils font rire (amusant) ou pas

COLOR	MAX_PLAYER	FUN
red	2	yes
red	3	yes
green	2	yes
red	2	yes
green	2	yes
green	4	yes
green	2	yes
green	1	yes
red	2	yes
green	2	yes
red	1	yes
blue	2	no
green	2	yes
green	1	yes
red	3	yes
green	1	yes

On obtient l'arbre de décision suivant.



- **Calcule de p** (l'estimation pessimiste de l'erreur).

1-beta	beta,Inv	α	k+1	n-k
0.20629954	*0.79370046*	0.25	1	6
0.14275599	*0.85724401*	0.25	1	9
0.75	*0.25*	0.25	1	1
0.15961075	*0.84038925*	0.25	2	15

Syntaxe sous Microsoft Excel
=BETA.INVERSE (α ;n-k ;K+1)

On a :

L'estimation pessimiste de l'erreur du feuille 1 est $6 \times U25\%(0 ; 6) = 6 \times 0{,}206$.

L'estimation pessimiste de l'erreur du feuille 2 est $9 \times U25\%(0 ; 9) = 9 \times 0{,}143$.

L'estimation pessimiste de l'erreur du feuille 3 est $1 \times U25\%(0 ; 1) = 1 \times 0{,}75$.

Le total des estimations pessimistes de l'erreur pour les feuilles 1,2 et 3 est :

$$6 \S 0{\cdot}206 + 9 \S 0{\cdot}143 + 1 \S 0{,}75 = 3{\cdot}273.$$

Tandis que

L'estimation pessimiste de l'erreur du nœud racine est : $16 \times U25\%(1 ; 16) = 16 \times 0{:}157 = 2{,}512$.

Comme

L'estimation pessimiste de l'erreur du nœud racine est inférieure à la somme des estimations pessimistes de l'erreur pour les feuilles 1,2 et 3 **(2·512 <3·273).**

Alors

L'arbre sera élagué on une seul feuillue de classe= **YES.**

Références *: b Clopper, C. et Pearson, S. L'utilisation de confiance ou de limites de confiance illustré dans le cas du binôme. Biometrika 26: 404-413, 1934.*

Titre : Interrogation d'une Base de Données à partir d'une Base de connaissances dans un Environnement de Développement de connaissances.

Résume : Les systèmes Expert sont avant tout des logiciels qui simulent le raisonnement d'un expert, il puise donc son expérience dans un domaine précis.

De nombreux systèmes experts ont été implantés avec succès pour résoudre des problèmes concrets. Ce qui est plutôt bon signe du point de vue de l'IA, néanmoins, les grandes difficultés rencontrées pendant l'extraction des connaissances des experts, cependant, leur formalisation forment peut être un point faible difficilement contournable dans les Systèmes Experts.

Pour soulever ces contraintes on a créé un système expert utilisant les techniques d'extraction de connaissances a partir de données (E.C.D), en vue de constituer une base de connaissances initiales d'une part. Et d'autre part on veut classifier les attributs d'objets au moment de l'inférence du système expert par l'appelle de module de classification à ces bases de connaissances extraites par un algorithme apprentissage (E.C.D.) des arbres de décision.

Mots clés : système expert, Extraction de connaissance, datamining, apprentissage automatique.

Title: Querying a Database from a base of knowledge in a development environment for knowledge.

Abstract: Expert systems are essentially software programs that simulate the reasoning of an expert, so he draws his experience in a specific area.

Many expert systems have been successfully implemented to solve real problems. What is a good sign in terms of AI, however, major difficulties encountered during the extraction of expert knowledge, however, their construction form may be a weakness difficult to avoid in expert systems.

To lift these constraints were created using an expert system technique for extracting knowledge from data (KDD), to form an initial knowledge base of a share. And secondly we want to classify the attributes of objects at the inference of the expert system module called the classification of these knowledge bases extracted by a learning algorithm (EDC) of decision trees.

Key words: expert system, knowledge extraction, data mining, machine learning.

عنوان المذكرة: الاستعلام عن قاعدة بيانات من قاعدة المعرفة في مجال التنمية للمعرفة.

ملخص: النظم الخبيرة هي أساسا البرامج التي تحاكي المنطق خبير، لذلك فهو يستمد خبرته في مجال محدد. وقد نظم العديد من الخبراء نفذت بنجاح على حل المشاكل الحقيقية. ما هو علامة جيدة من حيث لمنظمة العفو الدولية، ومع ذلك، واجه صعوبات كبيرة خلال استخراج المعرفة من الخبراء، ومع ذلك، قد شكلها البناء يكون من الصعب تجنب الضعف في النظم الخبيرة.

لرفع هذه القيود تم إنشاؤها باستخدام تقنيات نظام خبير لاستخراج المعرفة من البيانات (ECD)، لتشكل قاعدة المعرفة الأولية للسهم. وثانيا نريد أن تصنيف سمات العناصر في الاستدلال من الخبراء وحدة ما يطلق عليه نظام تصنيف هذه القواعد المعرفة المستخرجة بواسطة خوارزمية التعلم (تنمية الصادرات) من أشجار القرار.

كلمات المفتاح : نظام خبير، واستخراج المعرفة والبيانات والتعدين ، والتعلم الآلي.