Titre: Title:	Potentiel et application des systèmes experts pour le transport urbain collectif
Auteur: Author:	Mohamed Thameur Souissi
Date:	1989
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Souissi, M. T. (1989). Potentiel et application des systèmes experts pour le transport urbain collectif [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/58283/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/58283/
Directeurs de recherche: Advisors:	
Programme: Program:	Non spécifié

UNIVERSITE DE MONTREAL

POTENTIEL ET APPLICATION DES SYSTEMES EXPERTS POUR LE TRANSPORT URBAIN COLLECTIF

par

Mohamed Thameur SOUISSI

DEPARTEMENT DE GENIE CIVIL

ECOLE POLYTECHNIQUE

MEMOIRE PRESENTE EN VUE DE L'OBTENTION

DU GRADE DE MAITRE ES SCIENCES APPLIQUEES (M.Sc.A.)

Avril 1989

ISBN 0-315-52712-9

Canada

SOMMAIRE

Vers la fin des années 80, les chercheurs en transport se sont intéressés à la nouvelle technologie des systèmes experts (S.E.). C'est ainsi que de nombreux prototypes sont actuellement à l'essai ou en développement.

Ce mémoire a pour objectif d'étudier le potentiel d'application des S.E. dans les différents secteurs du transport et de construire un prototype appliqué au transport urbain collectif (T.U.C.) pour illustrer les différents aspects de cette nouvelle technologie.

L'application retenue concerne un système d'information pour les usagers d'un réseau d'autobus. Il s'agit d'un prototype d'illustration où on utilise un réseau fictif de petite taille.

Une étude des différentes applications potentielles et des prototypes existants nous a permi de définir le S.E. d'information à construire, et de proposer des applications ultérieures dans le domaine du transport urbain collectif.

Le langage TURBO-PROLOG a été utilisé pour la construction du S.E. sur un micro-ordinateur IBM-AT. Ce langage permet de représenter les connaissances sous forme de règles de production et utilise comme technique de

recherche le chaînage arrière en profondeur d'abord. L'exécution sous ce langage est très rapide mais se fait aux dépens de l'espace de mémoire vive (RAM) disponible. Cette consommation excessive de RAM lors de la recherche d'un itinéraire limite le prototype construit à des réseaux de très petite taille.

A la lumière des quelques difficultés rencontrées dans ce travail et des limitations du système construit, nous proposons quelques applications possibles dans le domaine du T.U.C. que nous jugeons d'un potentiel élevé et d'un intérêt immédiat.

ABSTRACT

Many prototypes of Expert Systems (E.S.) are now being built to evaluate their application potential in the field of transportation. This thesis' object is to study the potential and application of this new technology to different sectors of the transportation field, and to construct a prototype of Expert system for transit. The main object of this prototype is to illustrate different aspects of the E.S. technique.

A review of the potential applications and the various existing prototypes helped us to choose and identify the system to build, and to suggest further applications that have high application potential in the transit sector.

TURBO-PROLOG is the programming language used in an IBM microcomputer environment. This language allows representing knowledge by mean of production rules using the "backward chaining in depth first" search strategy. This language offers a very quick execution time but it consumes a considerable amount of memory (RAM). This exagerated consumption of memory limits the prototype performance and limits its applications to very small networks.

Considering the difficulties we faced when building our prototype and considering its limitations, we suggest some applications in the transit sector of high potential and immediate interest .

REMERCIEMENTS

J'aimerais exprimer ma reconnaissance à tous ceux qui ont contribué à la réalisation de ce travail. Plus particulièrement mon directeur de recherche M. Robert CHAPLEAU pour son soutien et ses précieux conseils, mes collègues de la section du génie des transports pour leurs aides et encouragements continus, ma femme pour son soutien moral précieux, mes parents pour leur soutien matériel et pour les sacrifices qu'ils ont dû faire en mon absence, et le ministère de l'enseignement supérieur et de la recherche scientifique de la Tunisie pour la bourse d'étude qu'on m'a accordée.

TABLE DES MATIERES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SYMBOLES ET ABREVIATIONS

Symbole	Signification
C	Capacité.
D	Distance.
FHWA	Federal HighWay Administration.
HCM	HighWay Capacity Manual.
I.A.	Intelligence Artificielle.
Ing.	Ingénieur.
P.P.	Programme Procédural.
R.O.	Recherche Opérationnelle.
S.A.C.	Système d'Aide à la Connaissance.
S.B.C.	Système à Base de Connaissances.
S.E.	Système Expert.
S.E.B.C.	Système Expert à Base de Connaissances.
T.C.	Transport en Commun.
T.U.C.	Transport Urbain Collectif.
V	Vitesse.

INTRODUCTION

Les systèmes experts (S.E.) sont l'une des plus récentes technologies dans le domaine de la recherche. Issus du domaine de l'intelligence artificielle (I.A.), ils sont des outils informatiques qui visent à permettre l'emmagasinage et le traitement efficace des connaissances. Après plus de trois décennies d'existence, ce domaine continue à être un champ controversé à cause d'une performance plutôt irrégulière. En effet, alors que dans des domaines comme la médecine et l'exploration minière des S.E. très performants ont été développés depuis la fin des années 80, dans d'autres domaines on est encore à un niveau d'expérimentation modeste.

Dans le cadre du but ultime de la recherche en transport qui vise l'amélioration tangible de la qualité des services de transport en termes de productivité et de performance, les chercheurs ont étudié la possibilité de profiter de cette nouvelle technologie que sont les S.E.. L'intérêt porté à ces systèmes s'explique par le fait que le développement et la prospérité du domaine des transports dépend de l'efficacité avec laquelle les nouvelles technologies sont adaptées.

Après une étude préliminaire, les chercheurs sont arrivés à la conclusion que le meilleur moyen pour évaluer l'applicabilité des S.E. en transport est de développer plusieurs prototypes dans les différents secteurs tels que la circulation, le transport en commun (T.C.) l'entretien des infrastructures, le design de réseaux etc...

Les problèmes de transport sont en général complexes à cause des divers aspects qu'ils impliquent. Parmi ces aspect citons le comportement humain imprévisible, les considérations politiques et sociales souvent litigieuses, et les processus d'évaluation multicritère et de décision multiobjectif. Cette complexité du problème en fait un sujet difficile à analyser par des procédures algorithmiques explicites. Les S.E. par leur approche déclarative se proposent essentiellement de résoudre ce genre de problèmes.

Ce rapport porte sur les spécifications des S.E. appliqués au domaine des transports. Il décrit quelques applications potentielles et quelques prototypes existants, et présente un prototype de S.E. en transport.

Comme application au secteur du transport urbain collectif (T.U.C), nous nous proposons de construire un système d'information (renseignement) pour les usager d'un réseau d'autobus. Le but du prototype est d'illustrer et de mettre en évidence les différents aspects de la technique

S.E. tels que la modularité de la représentation des connaissances, la structure de contrôle, la justification du raisonnement, et le mode de conversation.

Pour développer ce système, dans une période de temps limitée, nous avons choisi de considérer un réseau fictif de petite taille, et d'utiliser le langage TURBO-PROLOG dans un environnement de micro-informatique (IBM-AT ou compatibles).

Les systèmes d'information s'attaquent au problème très important de la saturation des réseaux de transport. Le prototype proposé tente de remédier à ce problème en améliorant l'utilisation du service de T.C.. Il fournit l'information concernant les itinéraires les plus courts, les itinéraires les plus rapides, l'état de saturation du réseau, la vitesse locale de circulation, l'accès aux principaux monuments, et les lignes d'autobus à emprunter. En plus il permet de déterminer un chemin selon une évaluation multicritère flexible.

Une revue de la littérature nous a permit d'identifier les applications potentielles des S.E. en transport et les prototypes réalisés dans ce domaine. Cette étude nous a permit de définir le système à développer. La construction de ce système a été faite selon les étapes suivantes: 1. identification du problème, 2. conceptualisation, 3. formalisation des connaissances, 4. implantation du

programme, et 5. tests et amélioration du programme.

Une synthèse nous a enfin aidé à situer notre prototype par rapport à l'orientation actuelle de la recherche sur les S.E. et de proposer de nouvelles applications dans le secteur du T.C..

Ce rapport présentent les éléments de base nécessaires pour des applications ultérieures. Il aide à bien évaluer le potentiel d'application des S.E. à un problème spécifique, à bien choisir l'outil de développement du S.E., et à adopter la meilleure forme de représentation des connaissances.

CHAPITRE I

APPLICATIONS POTENTIELLES DES S.E. DANS LE DOMAINE DES TRANSPORTS.

I. INTRODUCTION:

l'émergence de nouvelles technologies en télécommunication, en automatisation et systèmes d'information présente des opportunités que les ingénieurs en transport doivent saisir pour maintenir les infrastructures existantes et pour planifier les développements futurs.

Ces technologies avancées permettent de réduire les coûts, d'améliorer la productivité et de réaliser de nouvelles applications à la hauteur des exigences de l'époque. En effet, les ingénieurs en transport croient, en général que la croissance et le développement de leur domaine dépendront de l'efficacité avec laquelle ces technologies avancées seront adoptées [Sinha, 1988]. En d'autres termes l'adoption de ces technologies doit s'inscrire dans le cadre de l'objectif ultime de la recherche en transport, soit la production d'innovations technologiques qui assureront l'amélioration **tangible** de la

qualité des services de transport en terme de **productivité** et de **performance** [Boyce, 1985].

Les S.E. (Systèmes Experts), ou S.E.B.C. (Systèmes Experts à Base de Connaissances), représentent l'une des plus récentes technologies qui trouve actuellement un potentiel d'application croissant dans le domaine de l'ingénierie des transports [Ritchie 1986, Yeh 1986, Bonsall 1986, etc...].

Ce premier chapitre mettra l'emphase sur le potentiel d'application de la technique des S.E. en transport. Nous débuterons par une présentation générale des S.E. et après une évaluation de l'applicabilité de cette technique dans les différents secteurs du champs des transports, nous conclurons par une revue des réalisations et des essais les plus intéressants relatifs à ce sujet.

II. LES S.E.: GENERALITES:

Issus du concept et du développement de l'I.A. (Intelligence Artificielle), les S.E. représentent des approches vers des outils informatiques permettant l'emmagasinage et le traitement efficaces des connaissances. Tels que définis par plusieurs auteurs, ce sont ***des programmes d'ordinateur dont le but est de répondre, dans des domaines bien délimités de***

connaissances, à des problèmes généralement considérés comme complexes et difficiles, et dont la performance se compare à celle d'un expert humain du domaine ou la dépasse [Pehlivanidis 1987].

Un S.E. est constitué principalement de trois composantes:

1. Le moteur d'inférence.
2. La base des connaissances.
3. Le langage d'expression.

Le **moteur d'inférence**, ou machine déductive, accomplit essentiellement deux rôles principaux. Premièrement, il combine (fait le chaînage) des groupes de règles pour inférer (dériver) des connaissances telles que les jugements, les plans, les preuves les décisions, etc... Deuxièmement, il rend compte de la manière dont les nouvelles connaissances ont été inférées (justification).

La **base de connaissances** contient deux éléments de base qui sont ***les faits*** (connaissances assertionnelles) généralement fournies par l'utilisateur pour décrire le problème, et ***les règles*** (connaissances opératoires) qui traduisent l'expertise de l'expert humain du domaine.

Enfin, le **langage d'expression** permet de faciliter l'expression la plus directe possible des règles par rapport à leur forme d'émergence chez les experts. Il doit faciliter la communication que ce soit avec l'utilisateur

(consultation) ou avec l'expert (instruction) [Farreny 1985].

Les différentes étapes de construction d'un S.E. sont les suivantes:

1. Identification des exigences.
2. Conceptualisation de la démarche de résolution.
3. Formalisation de l'expertise.
4. Implantation du programme.
5. Tests et amélioration du S.E..

En général, un S.E. appartient à l'une des catégories suivantes:

1. S.E. de diagnostic.
2. S.E. de prévision.
3. S.E. de consultation.
4. S.E. de contrôle et de surveillance.
5. S.E. de planification.
6. S.E. d'instruction.
7. S.E. d'analyse et d'interprétation.

Pour des détails supplémentaires concernant les S.E., le lecteur peut se référer à l'annexe B du présent mémoire.

III. EVALUATION DU POTENTIEL D'APPLICATION DES S.E. EN TRANSPORT:

Les S.E. ont un grand potentiel d'application pour les problèmes qui n'ont pas d'algorithmes explicites de résolution. Couramment, les problèmes "mal-définis" ou "mal-structurés", sont difficilement résolvables par des algorithmes, alors que certains experts humains du domaine en question, aidés par leur expérience et leur large connaissance, peuvent leur apporter des solutions acceptables et même optimales parfois.

Plusieurs problèmes réels sont de cette dernière catégorie et particulièrement les problèmes de transport qui impliquent **le comportement humain** assez complexe et imprévisible, **les considérations politiques et sociales** souvent litigeuses, et **les processus d'évaluation et de décision multicritère et multiobjectif** difficilement analysables par des procédures algorithmiques [Yeh,1987].

Les S.E. sont donc bons candidats pour les problèmes de transport. Cependant, pour qu'un S.E. soit justifié pour un problème donné, ce problème doit répondre au moins à l'un des critères établis par les chercheurs sur les S.E. appliqués au domaine des transports. Dans ce qui suit nous allons examiner ces critères qui ont été avancés comme justification de l'utilisation des S.E. pour résoudre des

problèmes de transport [Yeh,1987], [Pehlivanidis,1987].

Critère #1.

La majorité des problèmes de transport présentent des composantes physiques, sociales, politiques et de jugement complexes. Il n'existe pas de procédures algorithmiques satisfaisantes pour assister directement le processus de décision ou pour procéder à des évaluations multicritères adaptées à différentes situations [Pehlivanidis,1987].

Critère #2.

En général, on utilise pour les études en transport des données d'enquêtes qui peuvent être biaisées ou incomplètes. La validation de ces données est généralement faite par des experts du domaine.

Critère #3.

Les experts et spécialistes en transport ont besoin de l'appui des outils techniques tels que les ordinateurs et les livres de codes et de plusieurs sources de référence (données de zonage...) pour bien identifier et analyser le problème. Les connaissances de base exigées sont excessives. On n'a qu'à penser à la réalisation d'une enquête O-D et aux efforts qu'elle exige.

Critère #4.

Les modèles utilisés dans différents contextes doivent être recalibrés pour être utilisés pour le même problème mais dans un contexte différent. Que la variation soit spatiale ou temporelle, le modèle doit être recalibré.

Critère #5.

La vie d'une société dépend entre autre du système de transport en place ; et souvent des situations d'urgence se produisent (saturation, variation brusque de la demande de T.C., projets urgents de taille à courte échéance tel que la réparation d'une infrastructure très lourde, etc...). En général, les responsables prennent la décision qui est proposée par les cadres disponibles, qu'ils soient experts ou non, car ils n'ont pas de méthodes d'action à un terme si court.

Critère #6.

Un échec ou une faute de l'expert humain peut être fatale en transport car elle peut engendrer des accidents graves (pertes humaines et matérielles) ou encore engendrer une situation de crise financière à l'intérieur de la société de transport à cause des gros montants d'argent impliqués dans la majorité des projets de transport.

Critère #7.

En transport, l'expérience à montré que la seule forme de transfert des connaissances se situe au niveau de la période de formation en début de carrière de la personne engagée. Ensuite cette personne continue l'apprentissage grâce à ces expériences vécues. Un tel processus peut être très long. En plus l'instabilité des postes à cause des promotions à un poste souvent administratif plus haut fait que l'expertise accumulée n'est pas exploitée de façon optimale et que l'exercice de formation reste continuel

Critère #8.

Les experts en transports sont relativement rares et chers à payer quand on a besoin de leurs services. Et souvent un même problème nécessite la collaboration de différents experts qui ne sont pas nécessairement disponibles en temps et lieux voulus.

Critère #9.

Les connaissances essentielles pour l'étude de problèmes en transport sont en général frappées par le droit d'auteur. Elles restent donc la propriété presque exclusive des experts. Des fois ces connaissances ne sont tout simplement pas publiées. C'est le cas par exemple du savoir d'un répartiteur en T.C. ou d'un contrôleur ou

régulateur de la flotte.

Critère #10.

A long terme, un S.E. est sûrement payant. En effet, l'expertise ne sera plus perdue mais au contraire elle sera enrichie et améliorée. Cette expertise permettra d'améliorer la planification pour mieux ajuster l'offre à la demande et ainsi rentabiliser le système de transport en général. La minimisation du coût global du système de transport représente le gain qu'un ou plusieurs S.E. peuvent assurer à l'avenir.

Cette évaluation détaillée de l'applicabilité des S.E. au domaine des transports nous montre que le potentiel est très élevé pour que ces techniques deviennent prochainement des outils d'aide impressionnants aux spécialistes de transport.

IV. QUELQUES APPLICATIONS POTENTIELLES DES S.E. EN TRANSPORT:

Pour mieux présenter ces applications nous allons les classifier selon des catégories distinctes. Après une revue poussée de la littérature sur le sujet, les septs (7) classes suivantes ont été retenues [Bonsall,1986]:

1. Les S.E. de renseignement.
2. Les S.E. d'aide à l'analyse et l'interprétation.
3. Les S.E. de design.
4. Les S.E. de diagnostic et de prescription.
5. Les S.E. d'identification.
6. Les S.E. de contrôle et de surveillance.
7. Les S.E. de support à la politique.

IV.1. Les S.E. de renseignement:

Un système de renseignement efficace doit être capable de **tirer** du requérent ce qu'il veut/souhaite savoir, d'**accéder rapidement** à l'information pertinente/appropriée, et de la lui **transmettre** de façon **claire et succinte.** Il est évident que le système doit posséder une information valide dans le domaine en question.

Les S.E. répondent très bien aux exigences de ces systèmes. En effet, grâce à leur interface-usager en langue naturelle, ces S.E. démontrent une habileté intéressante pour saisir ce que l'usager demande. Ensuite, la réponse lui est transmise de la façon la plus appropriée puisqu'elle est formulée dans sa propre langue. Quant à l'accès à l'information, les bases de données souvent très larges ne poseront plus de problèmes majeurs grâce aux nouvelles machines d'I.A. à architecture parallèle et aux

techniques de recherches performantes telles que **l'unification** (mise en correspondance, filtrage, appariement) dans le langage PROLOG. Avec les recherches sur la méta-connaissance (voir annexe B), les S.E. seront capables de gérer leurs propres informations (mise à jours etc...). La qualité des renseignements qu'ils pourront offrir sera alors très appréciable.

Certains systèmes d'information basés sur des logiciels de traitement de base de données et sur des langages de programmation conventionnels existent déjà sur le marché et sont pour certains suffisants. Cependant, ces systèmes présentent un défaut majeur: la mise à jour des informations et l'accès au système (interface-usager) peuvent être si frustrants et ardus que de tels systèmes seront probablement ignorés dans un avenir proche [Bonsall, 1986].

Un système de renseignement peut être conçu pour une utilisation générale (tout le personnel), ou pour les innovateurs ou encore pour les personnes étrangères au domaine (ignorants, inéxpérimentés). Ceci fait que ces systèmes varient de systèmes reliés à des bases de données importantes (privilégiées) à des systèmes simples de conseil.

Parmi les applications possibles on cite:

1. les renseignements sur les voyages.

2. un guide de la route.

3. les renseignements sur les banques de données.

1.1 Renseignements sur les voyages:

Les principales caractéristiques de ce système sont les suivantes:

a. La capacité d'organiser de façon optimale les informations contenues dans la base de connaissances pour faciliter et accélérer d'avantage la recherche. Ceci permet d'offrir un meilleur service et de minimiser les coûts d'opération du système expert. Par exemple, par sa faculté d'apprentissage, le S.E. peut savoir que pendant les jours précédant des vacances la majorité des usagers sont intéressés à connaître les prix spéciaux de base des voyages. Il s'arrange alors pour mettre ce groupe de données à un endroit plus accéssible dans sa base de faits.

b. Possession d'un mode de conversation très évolué qui peut s'adapter aux diverses faiblesses de dialogue chez les usagers ignorant tout à propos du système. Ceci comprend les possibilités suivantes:

b.1. Le traitement de questions floues comme par exemple: "Quel train part avant ça?", où l'on fait

référence à une information antérieure.

b.2. Le traitement de propositions posées de façons différentes et qui sont identiques. Par exemple: "je veux arriver avant le coucher du soleil...", "je veux voyager le jour..." "je veux partir avant X heure..." etc...

b.3. L'analyse des choix pour donner des conseils. Par exemple: "ce train vous permettra d'arriver à 18h00, cependant si vous partez 5 minutes plus tôt dans le train X, vous arriverez à 17h30!".

b.4. L'explication d'une information non comprise en différentes façons alternatives. Par exemple, 6h00 PM ou 18h00 ou encore 100 pi ou 30 m ; etc...

b.5. La mise à disposition de l'usager de différents renseignements supplémentaires quand il le demande. Par exemple un voyageur peut demander s'il doit transférer pendant son voyage ou s'il aura accès à certains services.

c. La possibilité de combiner des informations en temps réel provenant de différentes sources et de façon simultanée. Cette possiblité est nécessaire pour les cas de retard, de changement de services sur une partie du réseau,

de rabais ou de variations de tarifs, etc...

1.2. Guide de la route:

Ce système a pour objet d'optimiser les déplacements sur un réseau de transport. On en trouve déjà quelques-uns que ce soit pour le T.C. ou pour le transport privé. Toutefois, ces anciens systèmes sont basés sur des algorithmes de calcul de chemin à coût minimum avec lesquels des amendements concernants les changements continuels de l'état du réseau et du service sont généralement longs et difficiles à effectuer.

Grâce à la séparation entre règles d'inférence et faits, les S.E. permettent d'effectuer ces amendements plus rapidement et plus facilement puisque seuls les faits touchés par la modification seront à amender.

D'autre part, les S.E. permettent de définir les critères d'optimalité d'une route en collaboration avec l'usager pour tenir compte de sa propre situation. Les critères tels que le temps, la distance à parcourir, le tarif, le paysage, et éventuellement les contraintes pour l'exclusion de certains tronçons ou liens sont ordonnés tel que l'usager le préfère. Ainsi le S.E. fournit à l'usager la route qui correspond mieux à sa demande puisqu'elle est évaluée selon ses propres jugements et non selon des

critères fixes et rigides.

Avec un S.E. l'usager reçoit une réponse dans sa propre terminologie. Par exemple le S.E. peut répondre: "suivez la 15 Nord..." ou "Prenez la 2ième sortie à droite, continuer tout droit, ensuite prenez la sortie 66 à droite".

Des informations supplémentaires peuvent aussi être fournies à l'usager sur demande. Il s'agit de renseignements concernant l'emplacement des stations de service par rapport à un point donné du réseau etc...

Quand un choix d'itinéraire est accepté par l'usager, le S.E. peut le commenter et proposer une meilleur solution. On peut alors avoir les commentaires suivants: "La route X est celle qui répond le mieux à vos critères, cependant, si vous acceptez de passer par le pont P, vous pouvez gagner M minutes ou K kilomètres". Ces commentaires ne doivent cependant pas être poussés à un point tel que la conversation devienne très lourde. Par exemple, le S.E. doit apprendre par l'expérience que peu importe le nombre de minutes ou de kilomètres sauvés, les usagers ne sont pas intéressés à passer par un pont pendant une heure de pointe. Alors il doit éviter de faire son commentaire.

1.3. Renseignements sur les bases de données:

Les problèmes de transport utilisent des quantités énormes de données pour améliorer la qualité des analyses et des jugements. Ceci est un incitateur à connaître toutes les informations déjà disponibles pour éviter des dépenses inutiles.

La difficulté qui précède l'utilisation d'une base de données se résume en trois points:

- On doit être conscient de l'**existence** des données.
- Ces données doivent être **pertinentes** et **appropriées** au problème à résoudre.
- Ces données doivent être **disponibles en temps et lieux opportuns.**

Il existe actuellement beaucoup de systèmes de gestion de base de données efficaces et suffisants pour certaines tâches mais souvent incomplets car ils ont l'habitude de fournir soit trop soit peu d'information. Les systèmes experts, grâce à leur module de conversation en langage naturel peuvent spécifier avec plus de précision l'information désirée par l'usager et la lui transmettre sous la forme qu'il préfère.

IV.2 Les S.E. d'aide à l'analyse et l'interprétation:

Ces systèmes permettent d'améliorer et la qualité et la vitesse de réalisation de différentes études de transport. Leur force vient du fait qu'ils peuvent contenir les connaissances de plusieurs experts du domaine en plus des rapports sur les expériences antérieures.

2.1. Aide concernant les procédures techniques et les méthodologies:

Il est évident que le choix de la technique d'analyse d'un problème dans n'importe quel domaine est une étape très importante et déterminante pour l'aboutissement de l'analyse. Le choix de la technique appropriée d'enquête ou des test statistiques pertinents ou des modèles efficaces lors d'une étude demande certainement une expertise respectable. Un S.E. est pour ce contexte d'un grand intérêt pour les raisons suivantes:

1. Un S.E. conserve dans sa base de connaissances une quantité importante d'informations concernant les méthodologies existantes. Contrairement à l'expert humain, il n'est pas sujet au risque d'en oublier une seule.

2. Un S.E. peut apprendre à faire le bon choix pour la bonne application et à fournir des explications pour

soutenir chacune de ses suggestions.

3. En plus de suggérer une procédure, le S.E. peut expliquer comment elle doit être utilisée.

4. Enfin, le S.E. peut tenir compte, en se basant sur son expérience, de procédures plus générales tout en donnant les coûts et les données additionnelles que son utilisation implique.

Pour illustrer les étapes à suivre lors d'une session de travail avec un S.E. d'aide à l'analyse et l'interprétation, nous allons considérer le cas d'une enquête O-D. Les étapes approximatives sont alors les suivantes:

1. L'usager donne des affirmations (hypothèses) concernant le budget d'enquête.

2. L'usager définit le niveau de confiance désiré.

3. L'usager fournit d'autres informations concernant les formes de distributions à échantillonner.

4. Le S.E. demande des informations additionnelles telles que la configuration du réseau actuel.

5. Le S.E. indique le nombre d'interviews de différents types à faire à des localités spécifiques et à des temps spécifiques.

6. L'usager peut alors commenter la suggestion du S.E. permettant à ce dernier d'améliorer son approche lors de la session subséquente.

Différents S.E. peuvent être construits pour la validation de données d'enquêtes, l'analyse de tendances de mobilité etc...

2.2 Aide à l'interprétation:

Une autre étape très importante lors d'une étude est l'interprétation des résultats d'analyse. C'est cette étape d'interprétation qui mettra en valeur l'importance de l'analyse effectuée. Une bonne analyse mal interprétée ne vaut pas plus qu'une mauvaise analyse.

Souvent pour savoir interpréter convenablement les résultats d'un modèle ou d'une technique donnée il faut les connaître à fond. Toutefois, pour faciliter l'accès à un modèle, il est souhaitable de disposer d'un système d'aide à l'interprétation. Un S.E. qui suggère une procédure d'analyse peut, sans avoir à l'exécuter, aider à interpréter ses résultats.

Certains modèles ont des sorties (outputs) très volumineuses. Un S.E. peut alors nous aider à accélérer l'interprétation en n'examinant que les paramètres significatifs pour le problème à l'étude. Certains chercheurs pensent qu'il serait souhaitable d'avoir une interface directe entre le S.E. et le modèle à exécuter. Il est alors possible de limiter l'intervention de l'usager

à l'introduction des données et à l'interprétation finale des résultats.

IV.3. Les S.E. de design:

Le design par sa définition la plus populaire signifie **un plan selon lequel un objet sera construit.** Il peut signifier différentes choses pour différentes situations. De toutes les définitions possibles ressort la suivante: ***«un arrangement d'objets pour accomplir une tâche selon certains critères»*** **[Bonsall, 1986].**

Le design assisté par ordinateur n'est pas nouveau. En effet la CAO et la DAO connaissent un grand essort grâce aux systèmes interactifs graphiques déjà disponibles sur le marché. Les chercheurs en I.A. essayent actuellement d'étendre le rôle des ordinateurs, via les S.E., pour rendre possible la sélection et l'examen de différents design alternatifs répondants à des critères préétablis.

Certaines composantes de S.E. peuvent s'avérer particulièrement intéressantes:

1. Un module intelligent agissant comme "périphérie" pour les procédures de design. Ce module pourrait par exemple traiter des informations présentées sous différentes formes.

2. Une base de règles facilement modifiable pour permettre l'introduction de nouvelles contraintes. Ainsi on réduit la marge des designs à produire.

3. Un module intelligent pour l'interprétation des sorties (outputs) de différentes procédures de design. Ce module évalue les faiblesses et les points forts du design produit.

Parmi les applications possibles en design on a:

1. le design des infrastructures de transport.
2. le design des réseaux de transport.
3. le design des horaires.
4. le design des questionnaires.

3.1 Design des infrastructures de transport:

Pour des infrastructures comme les ponts, les autoroutes les terminus (ports, aéroports, gares...) et les stationnements, un S.E. doit contenir dans sa base de connaissances des informations sur:

- les composantes de la structure.
- les interrelations de ces composantes.
- les critères à satisfaire globalement.
- et les indices de performance à calculer.

Un système de design doit normalement procéder selon les étapes suivantes:

* Définition du type d'infrastructure à designer.

* Spécification des paramètres majeurs de design tels que les limites budgétaires, la capacité de design, les dimensions maximales, la configuration du site, etc...

* Spécification des contraintes et outils spéciaux tels que la durée maximale de construction, la disponibilité de matériaux non coûteux à proximité du site etc...

* Designs incomplets faits par le système et qui sont des alternatives de designs possibles. Le planificateur doit alors les commenter aidant ainsi le S.E. à améliorer ultérieurement sa procédure de design.

* Choix du meilleur design par l'usager du S.E..

* Spécification d'informations supplémentaires pour finaliser le design retenu. Ici le S.E. va pousser le design à un niveau de détail plus élevé.

* Production et approbation du design final.

Certains chercheurs pensent que toute la procédure de design doit être conduite par le S.E., et que l'usager ne doit intervenir que pour fournir des informations manquantes ou nécessaires en cours de route.

3.2. Design des réseaux de transport:

En plus de considérer les caractéristiques du réseau, ces systèmes doivent en assurer un usage facile. Pour le faire, il est indispensable de jumeler l'exercice de design à une étude continue de la demande.

Parmi les applications possibles on peut considérer le design de réseaux de T.C. et le design d'une intersection par exemple. Des prototypes sont déjà en développement pour le design de réseaux de transport unimodal (Expert-Ufos) et la signalisation d'une intersection (Trali). Dans ce même contexte on peut penser à la possibilité de construire un S.E. pour l'aide à l'optimisation d'un réseau de T.C. et qui utilise les services d'analyse et de design de modèles tels que Madituc, Emme/2, Hastus etc...

3.3. Design des horaires:

Aussi bien en transport des marchandises qu'en transport des personnes, on a besoin de faire le design des tournées et des emplois du temps des chauffeurs pour répondre à la demande. Ce travail est généralement assuré par les répartiteurs. Le besoin pour une telle planification est plus fort dans le cas du transport des marchandises où les destinations et les quantités à

transporter changent beaucoup et de façon irrégulière. Le design de l'emploi du temps d'un chauffeur doit alors être refait très souvent.

Bien qu'il existe des algorithmes assez efficaces pour faire les calculs nécessaires, le fait que l'exercice soit répétitif amène les répartiteurs, sous des contraintes de temps, à recourir à une répartition basée sur des approximations (à vue de nez).

Un S.E. permet de faciliter la répartition quelle que soit la variation du service. En effet la facilité d'accès à la base des faits pour introduire les changements survenus permet au répartiteur d'économiser beaucoup de temps et de toujours fournir la meilleure façon de procéder.

3.4. Design des questionnaires:

En transport les enquêtes occupent une place privilégiée. Elles sont indispensables pour mener une étude de faisabilité ou de rentabilité. Elles doivent donc avoir une bonne forme, comporter toutes les questions nécessaires et être très claires et simples pour que la réponse des personnes enquêtées soit exempte de toute ambiguité. Un système d'aide au design de questionnaires d'enquêtes doit:

- saisir les spécifications concernant l'enquête.

- fournir différentes formes de questionnaires à partir de sa base de connaissances.

- fournir le contenu essentiel du questionnaire en se basant sur les expériences antérieures.

- permettre d'ajouter ou de retrancher des éléments du questionnaire proposé.

- produire la forme finale du questionnaire.

On profite ainsi des anciennes expériences d'enquêtes tout en ayant la possibilité d'améliorer la qualité du questionnaire.

IV.4. Les S.E. d'aide au diagnostic:

Le diagnostic est un terme auparavant appartenant à la médecine et qui désigne la détermination d'une maladie d'après ses symptômes. L'usage du mot "diagnostic" est maintenant étendu à tout travail visant à identifier un problème ou une situation anormale à partir de signes ou symptômes observés.

L'intérêt d'un S.E. de diagnostic en transport pour la réparation et l'entretien des chaussées, les évaluations environnementales, et l'étude de situations de crises telles que la saturation de la circulation a été souligné par plusieurs auteurs [Yeh,87], [Abrahamsohm,87],

[Bonsall,86]. Déjà quelques systèmes ont prouvé leur utilité dans ce domaine et en particulier pour la gestion des chaussées et l'étude de la saturation (voir le paragraphe V).

Certaines composantes sont nécessaires pour que le S.E. de diagnostic soit efficace:

1. la possibilité d'accepter des données (informations) en temps réel à partir de différentes sources pour prévenir et empêcher le problème de se produire. Il faut cependant que cette possibilité soit contrôlée pour assurer la validité et la cohérence des informations.

2. Le S.E. doit proposer des solutions rapidement. Ces solutions doivent être les plus économiques dans les circonstances qui prévalent.

3. Le S.E. doit pouvoir estimer la possibilité pour que le problème se produise après qu'une mesure ait été prise (ou avant).

4. Le S.E. doit avoir un module d'apprentissage qui lui permettra d'élargir sa base de connaissances. A chaque fois qu'un expert humain réagit à une situation de crise, le S.E. doit en déduire des règles pour améliorer son raisonnement à l'avenir.

5. Il est aussi nécessaire d'avoir des méta-règles pour mettre à jour et gérer les connaissances acquises. Ces

méta-règles ont la responsabilité d'éliminer toute contradiction, de compléter les faits manquants dès que c'est possible et de contôler l'ordre de priorité entre les régles de raisonnement.

Parmi les applications potentielles en diagnostic on retrouve:

- la sécurité routière.
- l'entretien et la réparation des chaussées.
- l'entretien et la réparation des équipements et des sturctures.

4.1 La sécurité routière:

En enregistrant des données sur l'historique des accidents les plus fréquents ou les plus dangereux, on peut à l'aide d'un S.E. évaluer le potentiel d'accidents par un exercice de diagnostic basé sur des données simples telles que l'état de l'infrastructure, sa géométrie, les conditions de circulation, et les conditions climatiques. Ces données sont alors des symptômes probables d'accidents qu'on veut empêcher. Le S.E. doit, selon son diagnostic signaler l'accident possible et formuler la mesure à prendre dans le contexte réel. Actuellement, un S.E. pour la sécurité routière est en développement au "University college of London (1986)" [Bonsall,1986].

4.2. L'entretien et la réparation des chaussées:

Dans un pays nordique comme le Québec où la neige est abondante, et où les variations de température sont très considérables, les chaussées (routes et stationnements) subissent beaucoup de dégats. Le problème de fissuration est une maladie chronique à laquelle les villes font face chaque année. Des milliers de dollars sont à chaque fois dépensés pour ne servir que pendant quelques mois.

Cet état démontre un intérêt très grand à toute tentative ou nouvelle technique qui peut assurer des économies dans ce domaine. La technologie des S.E. est une des façons, probablement la meilleure à venir, de sauver des dépenses inutiles.

Pour obtenir de meilleurs résultats, le S.E. nous aide à optimiser les interventions nécessaires. Pour cela, le S.E. doit contenir des données sur l'historique de la chaussée (son âge, les matériaux qui la constituent, l'épaisseur de chaque couche, les caractéristiques et conditions du sol, l'historique des interventions antérieures etc...), le débit cumulatif de la circulation locale (en unité de poids ou tel que prescrit par les codes en vigueurs), les conditions climatiques (à différentes localités et à différentes périodes de l'année), et d'autres données enregistrées telles que la porosité, le

compactage, la déformation etc... Il est aussi souhaitable que le S.E. soit lié directement aux divers senseurs installés sur la chaussée pour assurer un contrôle continu. Actuellement, un prototype de S.E. de cette catégorie est en développement à l'université de Leeds [Bonsall,1986].

4.3. L'entretien des équipements et des structures:

Il s'agit d'un S.E. identique au précédent sauf qu'il est appliqué aux équipements tels que les véhicules (camions autobus voitures de trains et de métros) et les installations diverses ; et aux structures telles que les ponts, les chemins de fer (trains et métros) et les quais. Ces structures et équipements nécessitent un entretien bien planifié afin d'augmenter leur durée de vie et leur rendement.

Le S.E. doit faire le diagnostic et proposer les mesures à prendre. Ces interventions proposées par le S.E. peuvent être le résultat de certaines règles d'inférence ou tout simplement une copie intégrale d'une intervention qui a eu lieu antérieurement et qui s'est avérée efficace.

IV.5. Les S.E. d'identification:

On pense dans ce cas à des systèmes qui accèptent des

informations à l'état brut pour en faire une description ou une formulation plus précise. Cette information brute peut être soit l'expression d'une proposition par un simple employé dans ses propres termes, ou une image décrite par un contour seulement. Le S.E. dispose alors de composantes de reconnaissance d'image en 2D ou en 3D et des composantes de compréhension du langage naturel.

Ce système peut être utilisé pour des inventaires très importants où les standards ne sont pas nécessairement acquis par les employés, ou encore pour identifier des situations qui se répètent ou qui se produisent pour la première fois.

Par exemple, en examinant les peletons de circulation à un endroit donné on peut s'apercevoir d'un problème (qui s'est déjà produit auparavant) avant qu'il ne prenne de l'ampleur.

Cependant, la justification de ces S.E. d'identification n'est pas encore solide car le rapport avantages/efforts-requis n'est pas assez élevé [Bonsall,86].

IV.6. Les S.E. de contrôle et de surveillance:

Les S.E. de contrôle ont la particularité d'utiliser des données directes (en temps réel) pour améliorer la

performance d'un système en opération. Leur intérêt croît avec la complexité du système à contrôler. Parmi les applications possibles en transport on a:

- la surveillance et le contrôle de la circulation.
- la surveillance et le contrôle du trafic aérien.
- le contrôle des enquêtes.

6.1. Surveillance et contrôle de la circulation:

En général, pour assurer ce rôle, le S.E. doit posséder les connaissances suivantes:

1. la géométrie du réseau.
2. l'occupation des terrains avoisinants (obstacles etc...).
3. Description ponctuelle ou continue des peletons de circulation.
4. Les files d'attentes des automobiles aux intersections.
5. La signalisation en vigueur: feux, etc...
6. Les codes et règlements qui régissent le design tels que le HCM (Highway Capaity Manuel) etc...
7. Des données historiques sur le réseau: différents problèmes reliés à l'utilisation des liens et des intersections par les automobilistes.

Le S.E. doit intervenir à tous les niveaux. En particulier:

a. il reçoit les données disponibles et en commande d'autres lorsqu'il le juge nécessaire. Par exemple il peut demander qu'un comptage soit effectué à un endroit où il n'y a pas de senseurs.

b. il analyse la situation et demande des précisions à l'opérateur.

c. Enfin il déduit l'action à entreprendre et en fournit la justification. Il peut ainsi proposer l'installation de feux de circulation à une intersection où la sécurité des automobilistes et des piétons est sérieusement menacée.

6.2 La surveillance et le contrôle du trafic aérien:

Jusqu'à présent, le trafic aérien a suscité plus d'intérêt pour l'application des S.E. que le trafic terrestre.

Un S.E. peut assister le contrôleur aérien dans certaines tâches comme [Gosling,1987]:

1. la détection de situations de risques de collisions.

2. la suggestion de manoeuvres anti-collisions.

3. la prévention de situations de saturation dans les aéroports et les couloirs de navigation.

4. la préparation, à l'avance, des informations

concernant les mouvements des avions.

5. la gestion de l'affichage sur l'écran qui sert à la surveillance.

6. les conseils concernant les procédures de services aux vols.

7. la vérification des équipements de sécurité.

8. l'assistance à la formation des cadres.

9. la détection de situations climatiques inattendues.

Certains projets pilotes ont été lancés aux E.U. par la "MITRE Corporation". Au Royaume Uni la "Civil Aviation Authority" a déjà sérieusement considéré l'application des S.E. pour la régulation du trafic aérien pour minimiser les retards et pour planifier l'exploitation commune des espaces aériens.

Même si les S.E. ne remplacent pas entièrement un contrôleur, ils sont pour lui une aide et un conseiller très efficace.

6.3. le contrôle des enquêtes:

Un S.E. peut sauver beaucoup d'efforts inutiles lors d'une enquête à grande échelle. Le principe consiste à établir un lien entre le S.E. et des instruments lui transmettant périodiquement des informations sur le domaine en question. En analysant ces données et connaissant le

niveau de précision désiré, le S.E. peut alors décider quelles informations supplémentaires sont nécessaires, quand et où on peut les obtenir. Son rôle principal reste cependant d'optimiser la procédure d'enquête et de la rendre plus productive. La justification des efforts requis pour construire ce genre de S.E. reste encore à développer.

IV.7. Les S.E. de support des politiques:

En général, les S.E. sont vus comme des aides à la décision. Les caractéristiques essentielles qui en font des systèmes efficaces dans ce domaine sont:

1. La capacité de mémoriser plusieurs facteurs (variables) et de les traiter en même temps.

2. La capacité de justifier la décision.

3. La consistence.

4. La génération d'une multitude de stratégies alternatives. Etc...

Parmi les applications possibles on a:

- les décisions multiobjectifs et les évaluations multicritères.

- le traitement de l'incertitude.

- le contrôle de la consistence des politiques.

7.1 Les décisions multiobjectifs:

Il s'agit essentiellement de l'évaluation de différentes alternatives établies selon des critères multiples. Par exemple l'optimisation d'un réseau de T.C. se fait par une série de décisions qui considèrent à chaque fois plusieurs critères.

Un S.E. dont la base de connaissances comporte le savoir-faire de plusieurs experts permet d'aboutir à la meilleure solution en un temps réduit.

Le sujet de l'analyse multicritère a été abordé par les spécialistes des S.E.. En particulier, un S.E. appliqué aux méthodes de surclassement a été expérimenté [Pasche,87].

7.2. Le traitement de l'incertitude:

Le calcul d'incertitude peut être fait de façon très satisfaisante par divers logiciels d'analyse statistique. Cependant, un S.E. sait en plus quand déclencher une estimation quelconque selon le besoin. D'autre part, quand des informations (données) sont manquantes il peut proposer une stratégie pour les reconstituer tel que l'aurait fait un expert en statistiques. Dans le cas d'une expérience tout à fait nouvelle, il peut assister l'usager pour

déterminer la meilleur façon de remédier au problème qui se pose.

Ces systèmes sont particulièrement utiles pour des situations d'urgence ou de courtes échéances.

7.3. Le contrôle de la consistence des politiques:

Un S.E. pour assurer la consistence des politiques du preneur de décisions est le moyen le plus objectif et attentif qu'on peut proposer.

Pour les décisions en transport, on rentre dans la base des connaissances du S.E. les données concernant:

1. l'esthétique.
2. l'environnement.
3. les transferts lors d'un voyage.
4. la sécurité.
5. le financement. etc...

Une fois que ces standards sont connus par le système, il procède à une évaluation de différents projets pour proposer l'alternative qui convient le mieux à chacun d'eux compte tenu des standards préétablis.

Le S.E. peut aussi servir de consultant. L'usager lui propose une décision pour vérifier si elle respecte les règlements et les contraintes de design. Il est ensuite possible d'avoir une conversation entre le S.E. et l'usager

pour corriger ou améliorer la décision proposée.

Les avantages d'un S.E. de contrôle des politiques sont les suivants:

1. il est objectif.
2. il n'oublie rien.
3. il justifie toutes ses décisions.
4. il ne tolère pas de contradictions.

V. QUELQUES APPLICATIONS DE S.E. EN TRANSPORT:

Des prototypes de S.E. ont été réalisés pour les domaines suivants:

1. Le transport aérien.
2. La gestion des chaussées.
3. La circulation.
4. Les considérations d'environnement et de sécurité.
5. Le design des réseaux de transport.
6. L'entretien des infrastructures.

Dans ce chapitre nous allons effectuer un survol rapide de ces différentes réalisations.

V.1. Le transport aérien:

Jusqu'à maintenant, il n'y a pas eu de S.E. opérationnels pour le contrôle du trafic aérien

[Gosling,87] Cependant, le domaine a été sondé par plusieurs chercheurs pour en déterminer le potentiel et les stratégies possibles. L'utilisation des **shells** de S.E. est l'approche la plus populaire. Aux E.U., l'augmentation considérable des vols à certains aéroports à obligé la FAA (Federal Aviation Administration) à encourager l'automatisation de ses procédures de contrôle. Les S.E. sont l'une des techniques d'automatisation considérées.

Parmi les prototypes testés, on peut mentionner le S.E. pour l'assignation des retards de vols (delay assignment) qui a été testé pour les arrivées des vols aux aéroports d'Atlanta et de Georgia.

Exemple de règles d'assignation des retards:

1. ***Si*** *l'origine du vol n'est pas située aux E.U.*

   ALors *le vol ne peut pas être retardé.*

2. *Toutes autres choses étant égales, les avions les plus grands ont priorité sur les plus petits.*

3. ***Si*** *un vol a subit un temps de retard supérieur à la moyenne au mois précédent,* ***alors*** *il a priorité sur les vols qui ont subi un temps de retard inférieur à la moyenne pendant la même priode (heure) de temps.*

V.2 La gestion des chaussées:

Les types de S.E. les plus valables pour la gestion des chaussées sont:

1. Les S.E. de diagnostic, pour la réparation et l'entretien des chaussées.

2. Les S.E. de renseignement, pour la consultation des normes et des articles des différents codes et règlements.

Les autres applications possibles comme le design et la construction sont plutôt procédurales et présentent un intérêt modeste pour les S.E. [Pehlivanidis,1987].

Parmi les réalisations faites jusqu'à présent on trouve le S.E. SCEPTRE développé en Californie [Ritchie,87] et le S.E. ROSE développé à waterloo [Hajek,1987].

SCEPTRE est un conseiller interne et un outil d'analyse qui évalue la déformation de la surface de la chaussée ainsi que d'autres données pour identifier et recommander les stratégies de réparation ou d'entretien faisables. Il fonctionne sur micro-ordinateur. A la figure 1.1. un réseau partiel d'inférence du S.E. SCEPTRE est présenté.

SCEPTRE considère dix (10) stratégies de base:

1. ne rien faire.

2. remplir les fissures.

3. couvrir de goudron.

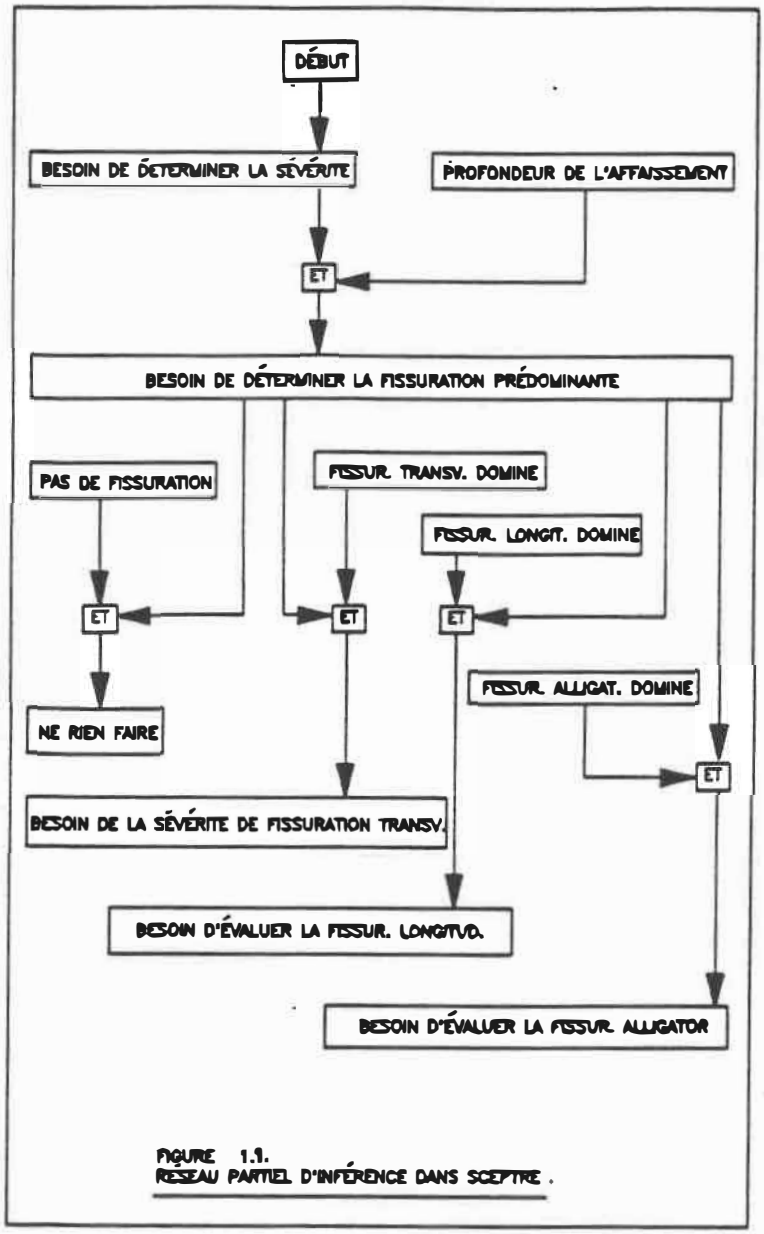


FIGURE 1.1.
RÉSEAU PARTIEL D'INFÉRENCE DANS SCEPTRE.

4. couche de friction ou d'adhérence.

5. couche de gravier.

6. double couche de gravier.

7. couche mince de béton bitumineux (=<.10 pi).

8. couche moyenne de béton bitumineux (.10 à.25 pi).

9. couche épaisse de béton bitumineux (>=.25 pi).

10. enlever et remplacer.

Pour la sélection de la stratégie, SCEPTRE considère six (6) facteurs de base:

a. le type de déformation à la surface.

b. la quantité (densité) de déformations.

c. la sévérité de la déformation.

d. la performance du pavage en place (taux de détérioration).

e. le niveau du trafic.

f. le climat.

Le système a été développé en utilisant l'environnement (Shell) EXSYS [EXSYS, 1985] qui permet d'appeler et d'exécuter des programmes externes. Ceci a permi d'utiliser différentes techniques pour procéder à des analyses de rentabilité.

SCEPTRE fait actuellement partie d'un S.E. plus complet appelé PARADIGM. Ce dernier a été conçu en regroupant trois S.E.:

1. SCEPTRE (Surface Condition Expert for Pavement Rehabilitation).

2. OVERDRIVE (Overlay Design Heuristic Advisor).

3. Deux systèmes similaires pour l'optimisation des stratégies et designs produits par SCEPTRE et OVERDRIVE. Ils font surtout de la programmation en nombres entiers pour les études de rentabilité.

A la figure 1.2. on retrouve la structure de PARADIGM [Ritchie,1987].

ROSE est un S.E. qui donne des recommendations pour refaire ou sceller les chaussées d'asphalte dans les régions froides ("ROuting and SEaling"). Il utilise des données traduites par 41 variables relatives à la performance de la chaussée, à son âge, et aux différents types de fissurations possibles. Il contient 360 règles qui traduisent une expertise dérivée des plus récentes recherches et expérimentations.

ROSE a été développé avec un shell (outil de S.E.) appelé EXSYS (version 3.0) et possède des interfaces avec d'autres programmes comme SAS, FOCUS, et VS-FORTRAN.

Son architecture est illustrée à la figure 1.3.. ROSE utilise comme donnée, la condition actuelle de la chaussée et fournit comme résultat un ensemble de recommandations concernant les mesures à entreprendre durant l'année en cours. La seule limitation quant à la pertinence de ses

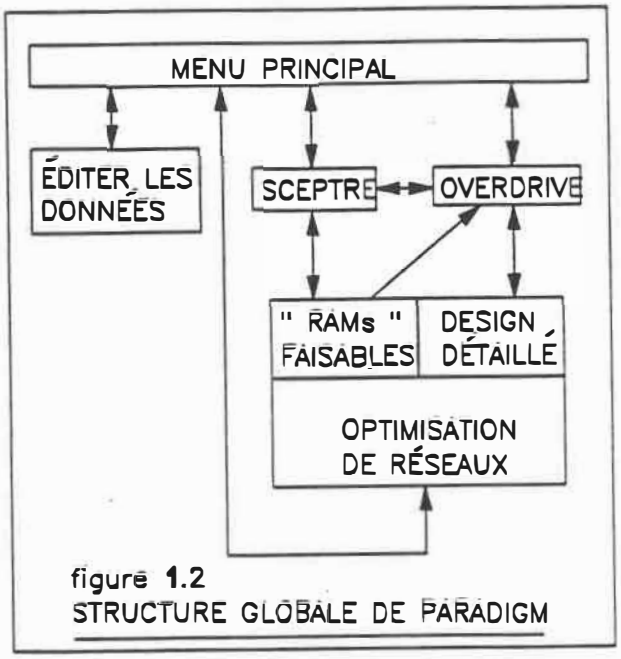


figure 1.2
STRUCTURE GLOBALE DE PARADIGM

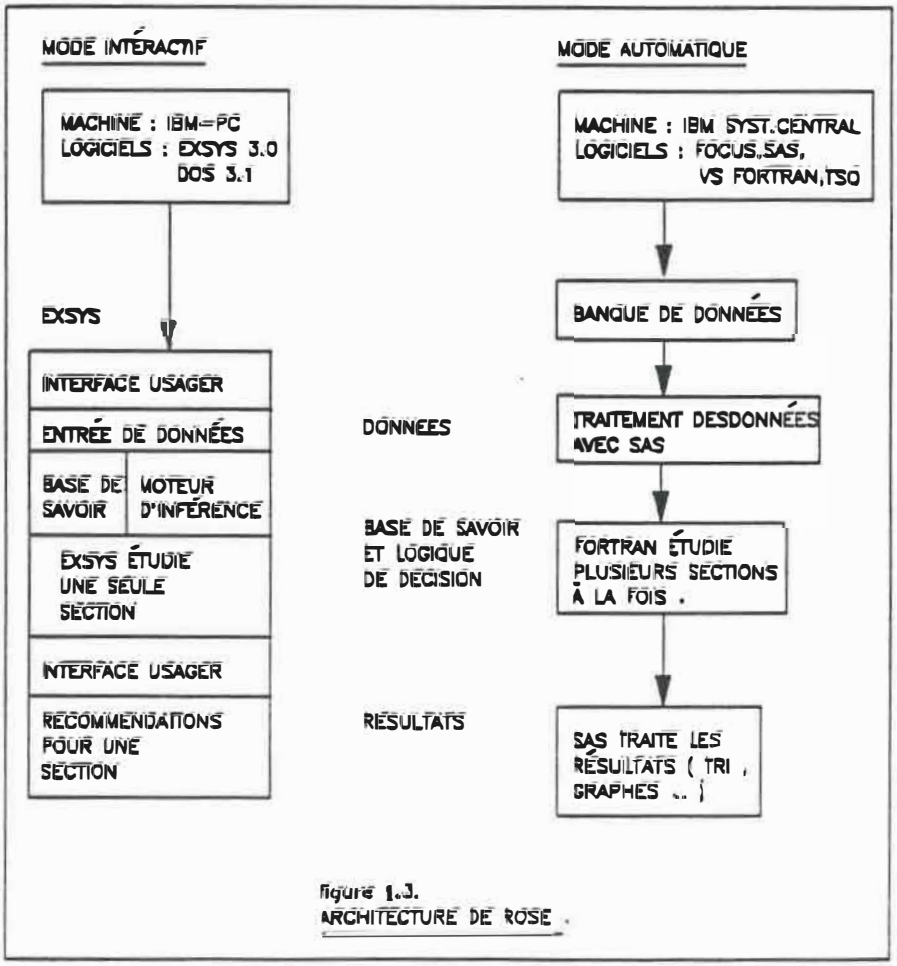


figure 1.3.
ARCHITECTURE DE ROSE .

résultats reste l'exactitude des données de départ sur la condition actuelle de la chaussée.

Ce S.E. a l'avantage de pouvoir justifier tout besoin de données et de permettre à l'usager de changer ou réviser facilement une partie des données ou des règles d'inférence. Il procède aussi à une évaluation du coût de l'intervention qu'il propose.

V.3 La circulation:

Des S.E. ont été construit pour le traitement de la saturation (congestion), l'évaluation des croisements autoroutes-chemins de fer, la signalisation routière, l'aide à l'analyse, le traitement du signal de virage à gauche, et le design d'une intersection. Ces systèmes accomplissent, en général, des tâches de diagnostic, de contrôle, de surveillance, d'analyse et de design.

3.1. Traitement de la saturation:

En France, un groupe de l'INRETS a développé un S.E. pour le traitement de la saturation. Il s'agit en fait d'un système centralisé de régulation équipé d'un ordinateur dont les fonctions sont de gérer les transmissions de données entre le poste central et le site, d'analyser le

trafic, et de sélectionner les plans de feux adéquats [Foraste,1986]. La saisie des données en temps réel est rendue possible grâce à des capteurs répartis sur le réseau capables de détecter la présence des automobiles et de rendre compte de l'état de chaque tronçon. En plus, le poste central est équipé d'écrans de contrôles sur lesquels il est possible de sélectionner des images provenant d'un réseau de caméras vidéo. L'organisation générale du S.E. est présentée à la figure 1.4..

3.2. Evaluation des croisements autoroutes-chemins de fer:

Un prototype de S.E. a été développé pour l'étude des croisements entre autoroutes et chemins de fer en milieu rural en Virginie (E.U.). Dans ce système, les langages utilisés sont Pascal et Lisp.

A la figure 1.5. un exemple de diagramme de relations causales utilisé par ce système est présenté.

La sortie (résultats) du système ont la forme décrite à la table 1. [Faghri,1987].

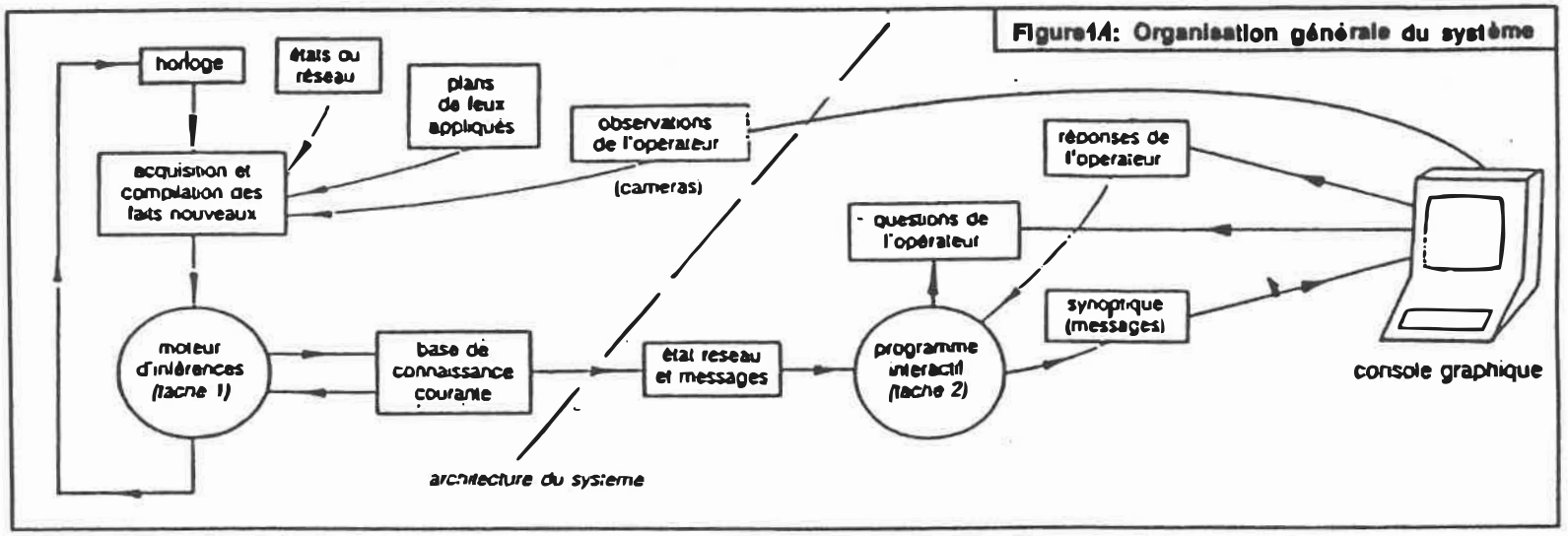


Figure1A: Organisation générale du système

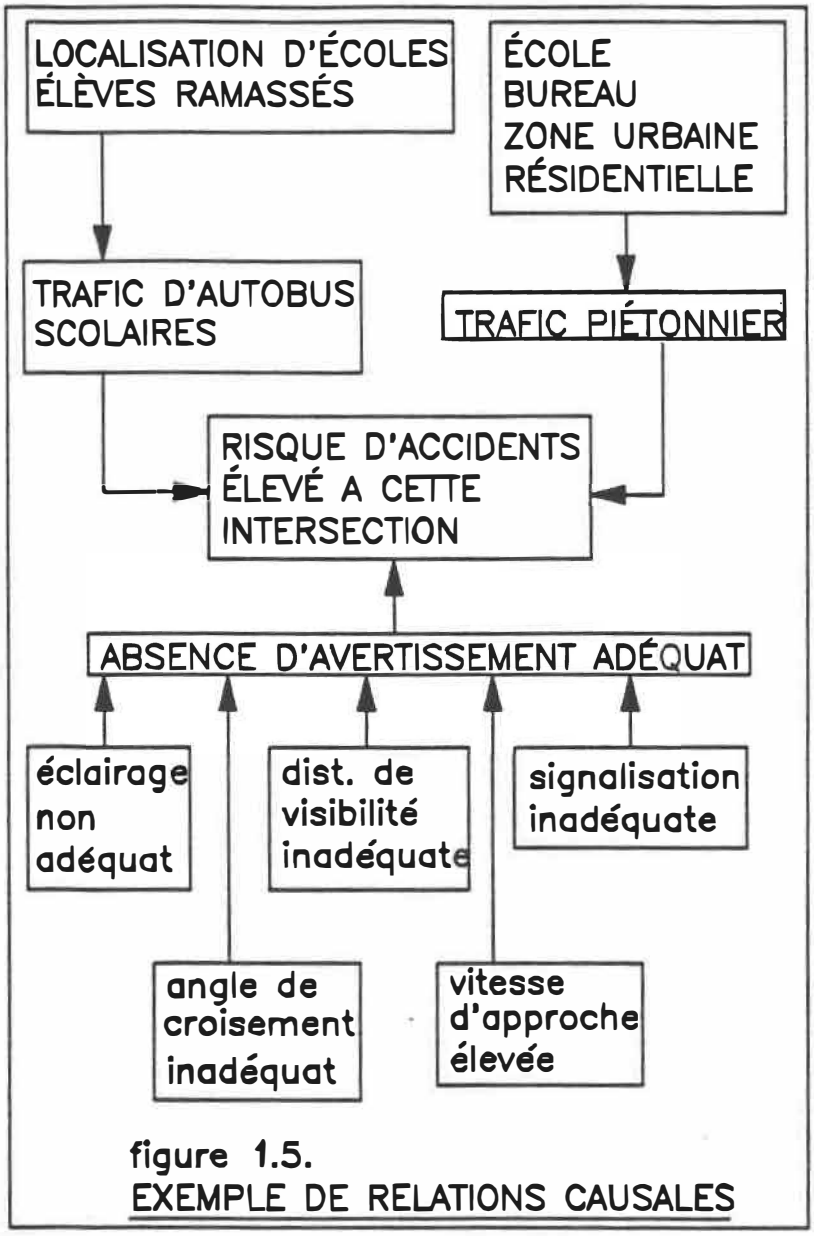


figure 1.5.
EXEMPLE DE RELATIONS CAUSALES

no du croisement	rang	diagnostic	action
554633EA	1	éclairage inadéquat.	augmenter l'éclairage de 10 %.
634592FG	2	trafic piétonnier très élevé.	remonter le système d'alarme.

Table 1. Forme des résultats fournis par le S.E. d'évaluation des croisements autoroutes-chemin de fer.

Les facteurs considérés dans l'évaluation d'un croisement ont été déterminés et ordonnés par les ingénieurs de la connaissance et l'expert du domaine. Huit (8) facteurs ont été retenus dans l'ordre suivant:

1. distances de visibilité disponibles.
2. trafic d'autobus scolaires.
3. les transporteurs de matières dangereuses.
4. le danger.
5. le renforcement des lois et règlements.
6. les centres pour personnes âgées et handicapées.
7. l'activité pietonnière.
8. la visibilité pendant la nuit (l'éclairage).

Le schéma du mécanisme opérationnel du prototype est le suivant (figure 1.6.).

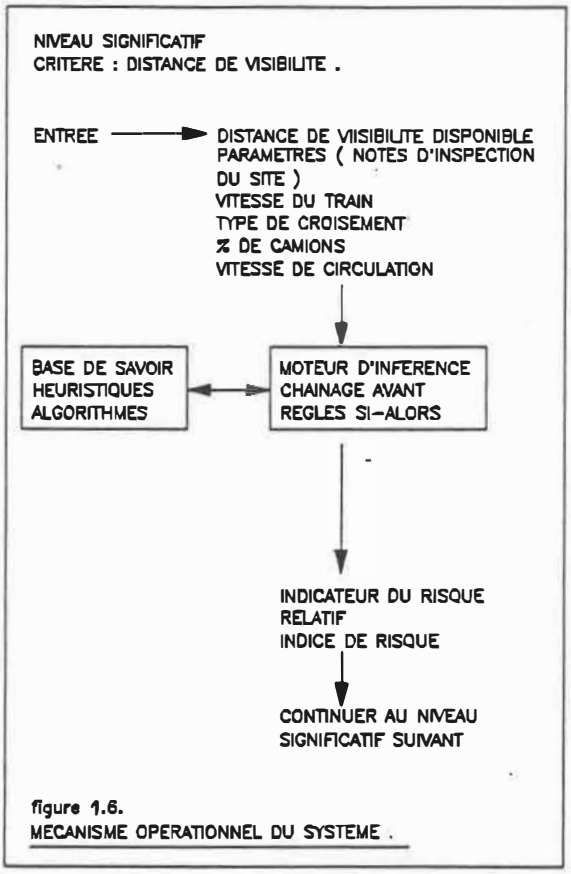


figure 1.6.
MECANISME OPERATIONNEL DU SYSTEME .

Les décisions recommandées par le S.E. ont été déjà approuvées par l'expert ayant participé au développement du système. Il reste cependant à l'évaluer par d'autre experts et pour des contextes différents.

3.3 Signalisation d'une intersection:

Un prototype de S.E. a été construit pour le design de la signalisation d'une intersection isolée. Dans un contexte réel l'interaction entre différentes intersections est très importante pour des fins de synchronisation etc... Le S.E. reste donc limité de ce point de vue.

Ce prototype a été nommé TRALI. Il n'énumère pas toutes les solutions possibles pour l'opération du signal mais suit un cheminement analogue à celui d'un ingénieur de circulation, plus directe, pour aboutir à une bonne alternative de design du premier coup. Les principales tâches accomplies par TRALI sont présentées à la figure 1.7.

TRALI a été écrit à l'aide de l'environnement OPS5 [Forgy, 1981]. Sa structure générale est présentée à la figure 1.8..

Les auteurs de TRALI estiment qu'il n'est pas encore prêt pour les applications réelles. L'évaluation des indices de mérite est imcomplète et sa vitesse d'exécution reste relativement faible [Zozaya, 1987].

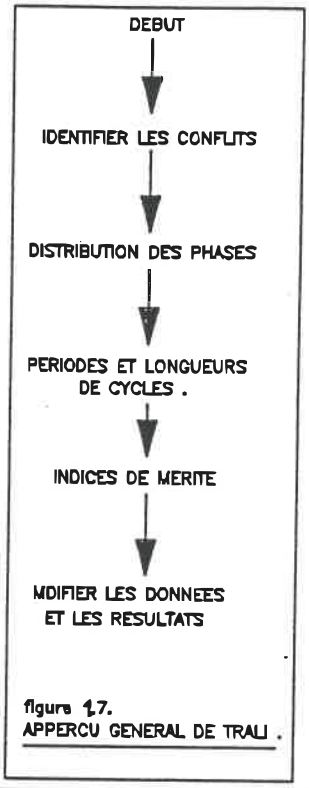


figure 1.7.
APPERCU GENERAL DE TRALI .

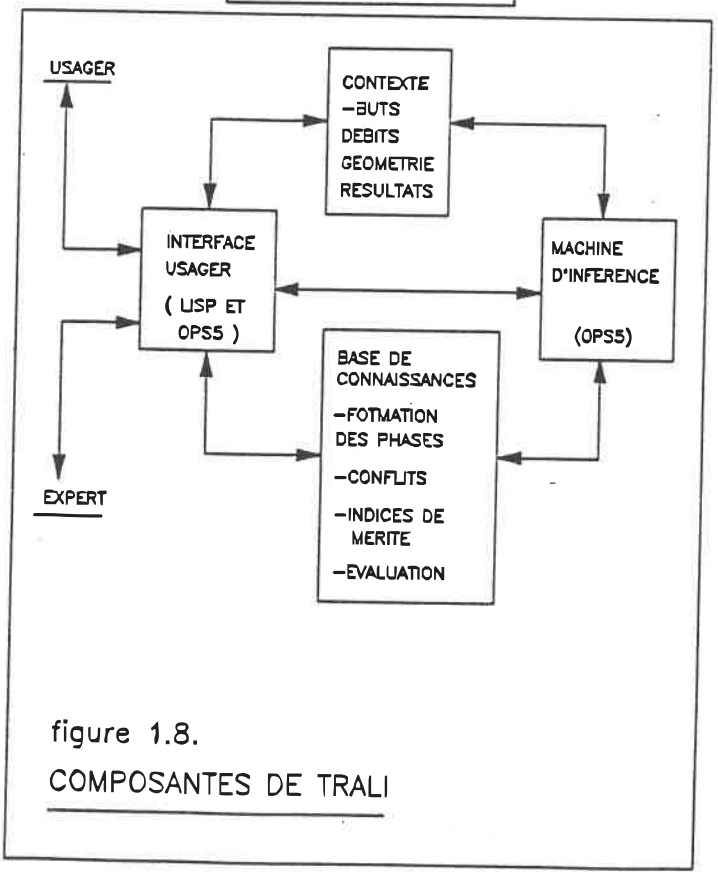


figure 1.8.
COMPOSANTES DE TRALI

3.4 Aide à l'analyse:

Un prototype de S.E. a été développé à l'institut de transport du Texas (TTI) pour aider l'ingénieur en circulation à faire un choix judicieux de logiciels informatiques (TTI-FHWA expert system).

La force de ce S.E. réside d'une part dans la banque d'information sur les logiciels existants dont il dispose, et d'autre part dans l'arbre de décision qu'il utilise pour évaluer la pertinence d'un logiciel pour une application spécifique.

La banque d'information concernant les logiciels a été construite à partir des spécifications commerciales des logiciels supportés par la FHWA (Federal HighWay Administration). Parmi les variables qui ont servi à la description des logiciels on trouve: le numéro de la dernière version disponible, le nom de l'auteur du logiciel les fonctions accomlpies par le logiciels, les exigences informatiques (hardware), contact pour assistance technique, projets futurs etc...

Le Shell INSIGHT 2+ [LFR, 1986] a été utilisé pour écrire le S.E. et l'exécuter sur micro-ordinateur IBM [Ching-Ping, 1987].

3.5 Traitement du signal de virage à gauche:

Un prototype de S.E. pour le choix de la phase de signalisation du virage à gauche a été construit à l'institut de transport du Texas (TTI). Ses objectifs sont de maximiser le niveau de service, de minimiser les retards aux approches et de réduire le nombre d'accidents associés au virage à gauche. Le choix fait par le système est basé sur un ensemble de lignes directrices qui guident la sélection de la phase [Ching-Ping, 1987].

Trois outils de S.E. (shells) ont été utilisés en parallèle pour des fins d'évaluation subséquente. IL s'agit des environnements PD PROLOG, TURBO PROLOG et INSIGHT 1.

Le processus de design comprend les étapes suivantes:

1. Extraire l'information de base (identification).
2. Définir les facteurs déterminants.
3. Définir les objectifs et les buts.
4. Définir les contraintes d'analyse.
5. Ecrire le programme.
6. Documenter le programme (commentaires internes).

3.6 Design d'une intersection:

"Intersection Advisor" est un S.E. pour le design et le réaménagement d'une intersection. Il a été construit à

l'Université de la Caroline du Nord à l'aide du shell M.1. [Teknowledge, 1985].

Ce système propose des modifications à la géometrie d'une intersection à fin d'améliorer son opération. L'expertise qui a été utilisée est tiré du HCM et de l'expérience des ingénieurs de circulation. Chaque proposition est obtenue en considérant les débits de circulation, la géometrie de l'intersection, le plan de signalisation, et l'étude de capacité.[A.Bryson,1987].

V.4. Considérations d'environnement et de sécurité:

Un système de design a été construit pour la conception de barrières anti-bruit et d'autres S.E. de contrôle ont été conçus pour le transport de matières dangereuses.

Nous parlerons ici du S.E. CHINA (Computerized HIghway Noise Analyst) développé à l'université de Louisville pour le design des barrières anti-bruit. CHINA est capable de communiquer avec deux procédures algorithmiques de design écrites en Fortran. Il exécute ces modèles d'analyse, interprète leurs résultats, et décide si ces résultats sont valables. S'ils ne le sont pas, il détermine de nouveaux paramètres d'entrée et réexécute les mêmes modèles jusqu'à ce qu'un résultat satisfaisant soit obtenu.

Lors de cet exercice CHINA agit comme un collègue de l'expert ou comme un conseiller pour le simple usager [Harris 1987].

CHINA a été écrit en langage FRANZ LISP en 1983 [Cohn 1988]. Actuellement, une version de CHINA est en développement à l'aide du shell EXSYS pour l'utilisation sur micro-ordinateurs IBM-compatibles.

V.5 Design des réseaux:

A l'université de Washington un prototype de S.E. à été construit pour le design d'un réseau optimal de transport. Il s'agit du système EXPERT-UFOS développé à l'aide de l'environnement M.1. Ce S.E. a été conçu pour le design d'un réseau de transport unimodal (l'automobile uniquement). La partie design est assurée par le programme UFOS qui comprend les composantes suivantes [Shieng-I Tung,1987]:

1. UFOS1: éditeur de réseaux.
2. UFOS2: affectation d'équilibre.
3. UFOS3: graphisme (interactif).
4. UFOS4: évaluation multicritère.

La base de connaissances a été synthétisée à partir d'un exercice de laboratoire fait par un groupe d'étudiants. Les critères d'évaluation des différents

scénarios sont les suivants:

1. le coût d'opération total.
2. le rapport V/C (V: volume ; C: capacité).
3. le temps moyen de voyage.

Le rôle du S.E. EXPERT-UFOS est de conduire l'exercice d'optimisation d'un réseau de base en proposant des stratégies de simulation avec UFOS. Pour accélérer l'exécution le langage C a été choisi pour la communication avec les fonctions externes. La structure globale du S.E. est la suivante:

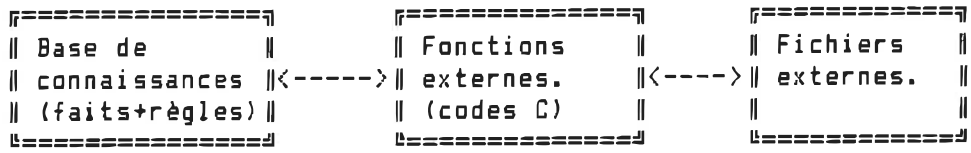


figure 1.9. Communication dans EXPERT-UFOS.

Ce prototype basé sur un exercice de laboratoire a pour but de montrer la faisabilité et le potentiel d'applications des S.E. pour le design des réseaux de transport. Le résultat a été concluant puisque EXPERT-UFOS a fourni le meilleur design, compte tenu des critères d'évaluation retenus, et en moins de cycles que les 76 étudiants qui ont participé à l'expérience.

Souvent, le design d'un réseau de T.C. nécessite une évaluation multicritère. Le S.E. développé à l'Ecole Polytechnique Fédérale de Lausanne pour l'analyse multicritère à l'aide de la méthode du surclassement, peut être utilisé pour une application au design de réseaux de T.C. (figure 1.10.).

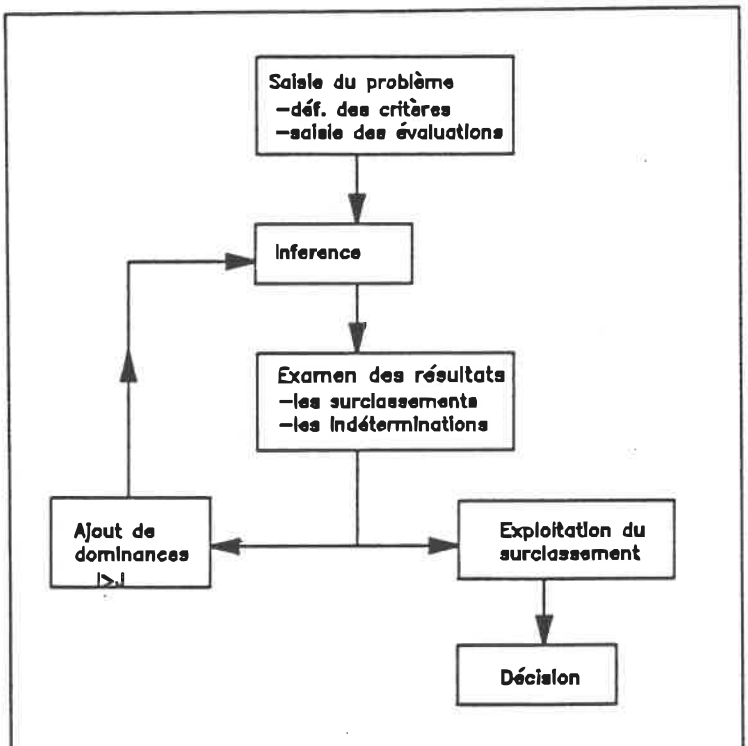


Fig. 1.10 Processus d'analyse du S.E. pour l'analyse multicritère

V.6 L'entretien des infrastructures:

En plus des opérations d'entretien des chaussées dont nous avons discuté dans un paragraphe séparé, il y a eu des

prototypes de S.E. pour l'entretien d'infrastructures lourdes comme les ponts par exemple.

Un S.E. pour la peinture d'un pont a été construit au MIT. Son objectif est de produire des stratégies pour l'exercice de peinture et d'en minimiser le coût. Il a été nommé "bridge PIARS" (Painting Identification And Ranking System) [McNeil, 1987].

L'environnement GEPSE (General Engineering Problem Solving Environment) écrit en langage C a servi de shell pour ce S.E. [Chebayeb, 1986]. Bridge PIARS est constitué de deux parties. La première identifie les stratégies possible pour la peinture du pont en se basant sur huit (8) caractéristiques du pont à peindre. La deuxième partie évalue le coût associé à chaque stratégie. La conversation avec le système est amicale grâce à un système de menus à choix multiples.

Nous venons de survoler rapidement quelques applications décrites dans la littérature. D'autres S.E. ont été mentionnés dans certaines publications, sans toutefois que des détails en soient révélés. Ces S.E. sont en général commercialisés et leur structure interne reste donc inaccessible au public.

Parmi les S.E. réalisés ou en cours de réalisation au Canada on peut mentionner les suivants [Magwood, 1987]:

a. S.E. pour le transport des matières dangereuses.

(CDT: Centre de développement des transports).

b. Affectation des chauffeurs réservistes.

(Ministère des transports et des communications de l'Ontario).

c. S.E. pour l'analyse de la maintenance à bord des locomotives.

d. Navigation maritime (Transport Canada).

e. Gestion des horaires de travail des chauffeurs de la commission des transport en commun de Kitchener.

f. S.E. pour les tournées d'autobus (CRT).

Comparativement aux applications potentielles, il est clair que les réalisations actuelles restent très modestes et il y a toujours un champ vaste pour des expériences supplémentaires. En fait, la meilleure façon de prouver le potentiel des S.E. appliqués aux problèmes de transport est de construire des prototypes qui soient très performants. Parmi les défis à relever il y a la question du financement de ces systèmes, et la limitation des moyens techniques de l'I.A. (machines de traitement de symboles).

VI. CONCLUSION ET DISCUSSION:

Lors de cette réflexion sur le potentiel d'application des S.E. aux problèmes de transport nous avons énuméré et discuté de quelques applications possibles en design, analyse et interprétation, renseignements, diagnostic, identification, contrôle, opération, et support des politiques. Il peut y avoir d'autres applications possibles ayant un potentiel élevé dans des domaines que nous n'avons pas étudiés.

Comme la majorité des problèmes auxquels font face les professionnels en transports exigent un savoir spécialisé, du raisonnement, de l'expérience et du jugement pour déterminer des stratégies, la majorité des chercheurs croient, en général que le potentiel est élevé pour que les techniques de S.E. deviennent des outils utiles pour les planificateurs et ingénieurs de transport [Yeh, 1987].

En plus des critères retenus pour la justification de l'utilisation d'un S.E. il faut insister sur les points suivants:

1. La technologie des S.E. sera considérée utile en transport seulement si elle s'inscrit dans la cadre d'une approche qui affronte la totalité du problème et non seulement une partie [Bonsall, 1986].

2. Pour être jugé important, un problème candidat à l'approche S.E. doit se produire **fréquemment** et impliquer beaucoup de **ressources** [Abrahamsohm, 1987].

3. L'utilisation du S.E. doit engendrer une minimisation nécessaire des coûts pour le propriétaire [Abrahamsohm, 1987].

La réussite des applications d'I.A. et des techniques de S.E. en transport, comme en tout autre domaine, dépend de la **facilité** avec laquelle les professionnels adoptent et utilisent ces techniques. Malheureusement, il existe actuellement plusieurs barrières pour la compréhension de ces systèmes. On peut en mentionner les suivants:

1. Plusieurs termes sont utilisés pour dire la même chose, et inversement parfois le même terme signifie différentes choses pour différentes personnes.

2. Les concepts sont différents et la façon de penser à propos d'un même problème est différente d'une personne à l'autre .

3. Le matériel d'illustration disponible semble souvent trivial ou inapproprié.

Heureusement, des textes deviennent de plus en plus disponibles pour aider à démystifier et mieux expliquer ce sujet. En plus, les conférences de plus en plus nombreuses sur ce sujet devraient conduire à une certaine standardisation de la terminologie utilisée par tout le

monde [Pavel,1987].

Jusqu'à récemment, certains arguments ont été retenus contre les S.E.:

1. Les S.E. actuels ne permettent pas de résoudre facilement des problèmes basés sur le bon sens et pour lesquels il est difficile de fabriquer des règles.

2. En général, les S.E. actuels ne connaissent pas la limite de leur savoir. D'après le philosophe Socrates, la preuve de la vraie sagesse est de savoir quand ne pas chercher ou demander l'expertise. Les S.E. ne disposent pas de cette sagesse.

3. Un autre argument, plus ou moins valable aujourd'hui découle du précédent. La base de connaissance des S.E. est généralement très large et le temps de recherche est très long. Ce problème sera résolu grâce aux nouvelles machines à architecture parallèle [Abrahamsohm, 1987].

IL est souvent facile de tomber dans l'erreur de demander trop à un S.E.. Ce dernier n'a dépassé la performance de l'homme que pour des tâches qu'on a pu définir de façon très stricte et précise. Le plus souvent, le S.E. ne fait que reproduire la performance de l'homme dans une partie restreinte de son domaine d'expertise [Bonsall,1986].

Pour simplifier le travail de construction des S.E., des shells fonctionnant sur micro-ordinateurs ont été développés dans certains champs du génie civil tels que les constructions, batiments etc... Ces shells sont cependant limités puisqu'ils sont performants pour les buts pour lesquels ils ont été construits et ne peuvent pas, en général, servir pour d'autres buts. Par exemple un shell construit pour faire du design ne peut pas être utilisé pour des systèmes de renseignement.

Les applications en transport sont très exigeantes au niveau de la technologie des S.E.. Elles demandent beaucoup d'effort de celui qui construit le S.E. pour les raisons suivantes:

1. La quantité de calcul numérique requise par la majorité des problèmes de transport est très importante. En plus plusieurs règles et relations doivent être fabriquée pour que le système soit complet.

2. Comme un système devient désuet dès qu'un autre **plus rapide et suffisant** devient disponible, la technologie de S.E. doit améliorer sa vitesse d'exécution non seulement pour les applications en temps réel mais de façon générale [Bonsall,86].

Le plus grand défi, probablement, qui se pose à ceux qui développent des S.E. c'est d'améliorer la capacité du S.E. d'apprendre à partir de " ses propres expériences ".

Bien que des exemples d'illustration très simples (parfois écrits en Basic sur micro-ordinateurs) existent, presque dans tous les textes d'introduction à l'I.A., l'introduction de techniques d'apprentissage automatique dans les applications sérieuses demeure extrèmement difficile et généralement non disponible dans les logiciels courants [Bonsall,1986].

Il est aussi nécessaire de développer une expertise concernant la procédure d'acquisition des connaissances et de l'incorporer dans le S.E.. Des études à cet effet commencent à émerger et certains chercheurs travaillent actuellement sur des systèmes de transfert de l'expertise de l'expert vers le S.E. [Boose, 1986]. Actuellement, il est surprenant de constater que peu de travail a été fait dans ce domaine. Une des conséquences de ce manque est alors que le développement d'un S.E. ne doit pas être laissé au spécialiste en I.A. à lui seul. La connaissance du sujet en question est indispensable pour identifier les questions à poser, à qui et dans quel ordre les poser.

Les S.E. en transport doivent être considérés comme des outils complémentaires aux techniques déjà existantes. Le but ne doit pas être de produire des systèmes indépendants (isolés) mais plutôt de produire des systèmes dans lesquels les techniques de S.E. sont utilisées là où elles sont bonnes et les procédures

algorithmiques sont utilisées là où elles sont bonnes.

Enfin, l'évaluation du mérite global d'une technologie comprend les points suivants:

1. Le mérite d'innovation.
2. Le mérite d'adoption.
3. Le mérite opérationnel.
4. Le mérite du marché.

A la table 3.2. une évaluation qualitative du mérite des S.E. appliqués au T.C. est présentée. Pour une évaluation plus concrète et concluante, tous les chercheurs s'entendent sur le fait que le développement de prototypes relatifs à des applications sérieuses est nécessaire [Bonsall, 1986], [Abrahamsohm, 1987], [Yeh, 1987] etc...

TABLE 2.: Evaluation du mérite des S.E. appliqués au T.C.

mérite d'innovation.	1. les S.E. permettent l'utilisation d'heuristiques. La disponibilité de solutions probabilistiques enrichi les décisions des professionnels de T.C.. La pression compétitive conduira à l'acceptation de ces systèmes. 2. la séparation entre données et raisonnement facilite les corrections et la construction de prototypes. 3. les solutions inférentielles sont "humanisées" dans le sens qu'elles sont de plus en plus proches du raisonnement humain. 4. le dialogue avec l'usager permet une interaction très large. 5. le manque de professionnels en I.A. (ingénieurs de la connaissance) est une contrainte majeure lors de l'implantation d'un S.E. à large échelle.
mérite d'adoption.	1. La disponibilité des shells et d'autres outils prêts à utiliser rend les S.E. attirants pour les applications à petites échelles. 2. Les interfaces en langue naturelle vont offrir une façon intéressante et non pénible pour introduire les ordinateurs aux petites et moyennes entreprises de T.C. 3. La baisse des prix des outils de rassemblement des données, des micro-ordinateurs de plus en plus puissants, et des shells de S.E. vont favoriser les applications de S.E..

(Table 2. suite)

	4. Les S.E. complexes exigent des machines d'I.A. puissantes qui fonctionnent pour des buts spéciaux. Ces machines sont coûteuses et volumineuses.
mérite opérationnel.	1. La promesse de l'expertise distribuée est très intéressante pour le secteur des T.C. 2. Les shells à bas prix, réutilisables, offrent une alternative économiquement intéressante aux solutions de logiciels commercialisés. 3. Les S.E. sont disponibles pendant les 24 heures. 4. L'expertise est préservée. Certains professionnels du T.C. partent prochainement en retraite. 5. Des grands S.E. peuvent permettre des gains sur les coûts de personnels et d'équipement.
mérite du marché.	.. Les gains au niveau de la productivité par les S.E. sont très importants pour les opérateurs du T.C.

Source: [Abrahamsohm,1987], (adaptation).

CHAPITRE II

S.E. DE RENSEIGNEMENT POUR LE T.U.C.

I. INTRODUCTION:

Dans ce chapitre il sera question de la construction d'un prototype de S.E. sur micro-ordinateur pour le T.U.C.. L'application concerne un système de renseignement (Information) pour les usagers d'un réseau de transport en commun en milieu urbain. S'agissant d'un prototype d'**illustration** essentiellement il nous a fallu choisir un réseau **fictif** pour simplifier l'application.

Les systèmes d'information en transport sont vus comme une composante indispensable pour remédier au problème de saturation des réseaux et pour améliorer l'utilisation des services de transport collectif (T.C.) [Boyce,1988]. Les sociétés et agences de transport en commun devraient donc voir ces programmes d'information de l'usager comme une partie intégrante de l'effort global de "marketing", qui peuvent engendrer des bénéfices en terme d'accroissement de la mobilité et d'amélioration de la perception publique de la valeur des services de T.C. [Fruin,85].

Parmi les innovations technologiques, les S.E. pourraient offrir des façons intéressantes d'éliminer certains problèmes des systèmes de transport saturés ou probablement stagnants. Ceci dégage un potentiel assez élevé qui nous a poussé à retenir comme application de S.E., un système d'information (renseignement) pour les usagers d'un réseau d'autobus.

Pour le prototype qu'on se propose de construire, le réseau est représenté par les éléments suivants:

- des liens (tronçons).
- des rues (artères).
- des monuments (batiments, places, etc...).
- et des lignes d'autobus.

Le prototype a pour tâche d'informer les usagers sur l'utilisation du service de transport afin d'optimiser son fonctionnement.

Les étapes de construction du S.E. sont: l'identification des exigences du problème, la conceptualisation, la formalisation des concepts, l'implantation du programme, et les tests et améliorations.

Une évaluation du prototype nous permettra ensuite d'illustrer certains aspects de l'approche S.E. tels que: la modularité de la représentation des connaissances, la capacité de justifier le raisonnement et les résultats, le degré de flexibilité du mode de conversation, la structure

de contrôle et l'efficacité d'exécution du S.E..

II. CONSTRUCTION DU S.E.:

II.A Identification du problème:

A.1. Buts:

On se propose de construire un système d'information qui permet de:

- informer l'usager sur l'utilisation du sol (adresses descriptions, etc...)
- informer l'usager sur l'utilisation du système de transport en commun (détermination des itinéraires et de leurs différents paramètres caractéristiques).
- modifier facilement et en temps réel les données relatives à l'opération du système de transport (contraintes temporaires relatives à la circulation...).
- dialoguer amicalement avec l'usager (système de menus à choix multiples).
- tenir à jour l'état de saturation du réseau.
- valider les informations rentrées par l'usager.

A.2. Principaux aspects à considérer:

Le prototype doit avoir le comportement d'un agent d'information *expérimenté* et *intelligent*. Mais avant tout, il doit posséder un mode de conversation très simple, précis et évolué. Les principaux aspects nécessaires pour assurer une bonne qualité du système de renseignement sont:

1. la facilité et la flexibilité de la conversation.

2. la suffisance et la validité de l'information relative à l'opération du système de transport.

3. la suffisance et la validité de l'information relative au réseau routier et à l'utilisation du sol.

4. la maîtrise des différentes techniques (formules, algorithmes, heuristiques...) nécessaires pour les calculs et les estimations éventuels tels que le calcul du chemin le plus court, l'évaluation du temps total de voyage, la considération des contraintes d'inaccessibilité à une partie du réseau routier etc...

5. la possession d'un mécanisme de contrôle et de validation des informations quelle que soit leur provenance.

6. la flexibilité d'exploitation du S.E. pour éviter toutes les étapes de recherche ou de calcul inutiles et accéder le plus directement possible à l'information voulue.

7. la facilité de modification des informations et connaissances assertionnelles (faits) et opératoires (règles).

8. la possibilité d'utiliser des critères propres à chaque usager pour évaluer l'optimalité d'un itinéraire.

9. la possibilité d'obtenir des informations supplémentaires concernant un itinéraire qu'on vient de déterminer.

10. la présence d'une aide à l'utilisation du système à chacune des étapes jugées relativement complexes pour l'usager.

A.3. Ressources disponibles:

Les principales ressources impliquées dans la construction du prototype de S.E. sont les suivantes:

1. Micro-ordinateur IBM ou IBM-compatible (640K).

2. TURBO-PROLOG: Langage de programmation logique développé par la compagnie Borland International Inc.

3. Données de départ relatives à un réseau routier et un réseau de transport par autobus fictifs.

4. Expertise: Documentation relative au transport en commun en milieu urbain et à l'application des S.E.. Notons ici la collaboration du personnel de la section des transports en tant qu'experts du domaine des T.U.C..

II.B. Conceptualisation:

Maintenant que le problème est défini avec ses buts, ses différents aspects et les ressources disponibles, il s'agit d'identifier les concepts clés, les relations de connaissances les types de données disponibles, les hypothèses, les contraintes et les stratégies. Ensuite, le schéma fonctionnel du processus de résolution sera tracé.

Nous avons choisi une application fictive pour réduire le temps nécessaire à la collecte des diverses données dont on aura besoin pour cet exercice. Le réseau de la figure 2.1. sera donc utilisé. Pour le caractériser les informations suivantes sont nécessaires:

* les liens (tronçons).
* les rues (artères).
* les lignes d'autobus.
* les monuments (immeubles, parcs, services etc...).

Plusieurs variables caractérisent chacune de ces informations. Pour notre application, nous nous sommes limités aux suivantes:

tronçon(1er noeud, 2ième noeud, nom de la rue, longueur du lien vitesse moyenne de circulation sur le tronçon, liste des autobus qui desservent ce lien, nature du paysage).

artère(nom de la rue, liste des noeuds).

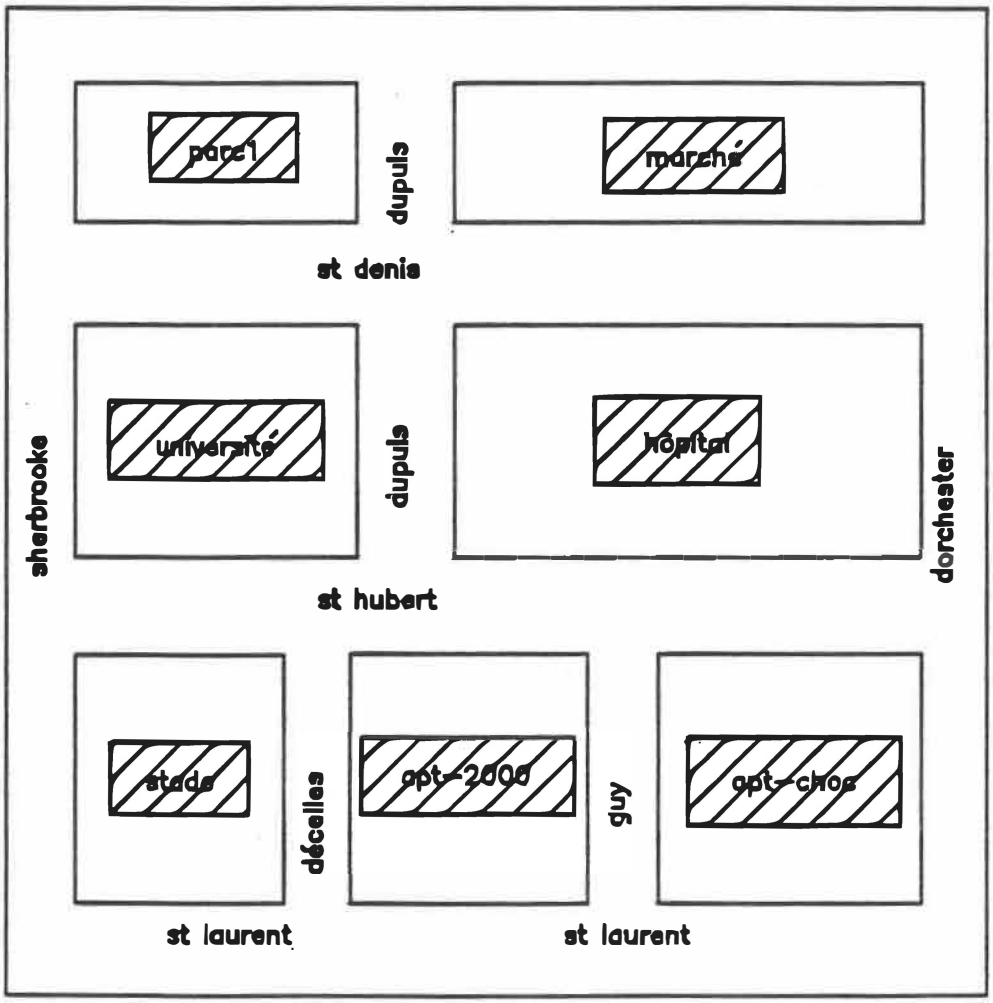


figure 2.1. Réseau fictif utilisé dans le prototype

ligne(nom de la ligne, liste des noeuds de parcours).

monument(nom du monument, liste des noeuds d'accès).

En plus de ces données physiques du réseau, d'autres informations font partie de la base des faits comme l'état de saturation, la liste de critères qu'on peut utiliser pour évaluer un itinéraire, la liste des contraintes qu'on peut imposer sur les itinéraires, la liste de paysages reconnus par le système, et l'identification des usagers privilégiés.

Pour représenter ces éléments certaines variables sont utilisées:

saturé(rue, 1er noeud du lien saturé, 2ième noeud du lien saturé, pourcentage de saturation).

critères(liste des critères d'évaluation).

les_contraintes(liste des numéros de contraintes).

les_paysages(liste des numéros de paysage).

identification(nom de l'usager, son code d'accès).

Le système accomplit les tâches suivantes:

1- détermination d'un itinéraire:

- selon la distance minimale de parcours,
- selon le temps minimum de voyage,
- avec élimination de tous les transferts,
- ou selon une combinaison de critères.

2- Accès direct à des informations partielles telles que:

- la ligne de transport à emprunter,
- la longueur d'un parcours donné,
- la vitesse moyenne sur un tronçon particulier,
- les zones saturées,
- le paysage local pour un tronçon donné,
- et les adresses des monuments.

3- Mise à jour des informations par:

- ajout,
- annulation,
- ou modification d'informations.

Pour chacune de ces tâches, quelques concepts de base doivent être établis pour une formalisation ultérieure. Lors de la détermination d'un itinéraire, les principales opérations seront les suivantes:

B.1. La recherche d'un itinéraire:

Les variables caractéristiques d'un itinéraire sont le noeud de départ, le noeud de destination, l'itinéraire (liste de noeuds) et la longueur. La recherche se fait selon la règle suivante:

Un point est un itinéraire vide de longueur nulle.

Il existe un itinéraire I entre O et D de longueur L

si il existe un lien (O,Z) de longueur L1

et il existe un chemin entre Z et D de longueur L2.

la longueur du chemin entre O et D est L=L1+L2.

Comme on peut le constater, cette recherche est basée sur la **récursivité** qui est une technique très utilisée en programmation logique. Très brièvement, cette technique ressemble un peu à une **boucle** en programmation traditionnelle comme dans les langages FORTRAN, PASCAL, BASIC etc... La récursivité consiste à un appel d'une régle à l'intérieur d'elle même.

B.2. Détermination du chemin le plus court:

Comme la longueur du chemin est l'un de ses arguments, la détermination du plus court chemin devient simple du point de vue procédure. L'approche est la suivante:

. déterminer tous les chemins possibles,

. comparer les longueurs de chemins déterminés,

. et retenir le chemin de plus courte longueur.

Cette façon de procéder est peu pratique quand il s'agit de grands réseaux routiers. La difficulté vient du fait que la récursivité impliquée lors de la détermination de chaque itinéraire par TURBO-PROLOG consomme beaucoup

d'espace mémoire.

B.3. Détermination de l'itinéraire le plus rapide:

Le temps de voyage n'étant pas un argument intrinsèque à la définition de l'itinéraire, il faut tout d'abord le calucler pour chacun des itinéraires. La suite est identique au calcul du chemin le plus court en remplaçant la variable «distance» par la la variable «temps total de voyage».

Le temps de voyage est composé du temps de parcours, du temps d'attente et du temps de marche. Le calcul du temps de voyage total se fait selon la procédure suivante:

. déterminer l'itinéraire.

. tester la présence d'autobus.

. calcul du temps d'attente: Intervalle de service pendant l'heure de pointe divisé par deux.

. calcul du temps de parcours: utiliser la vitesse moyenne de circulation pour le parcours en autobus et la vitesse de marche (5 Km/h) pour le parcours à pieds.

. comparer les temps de voyage de tous les itinéraires.

. et retenir le chemin à temps de voyage minimum.

B.4. Elimination des transferts:

Pour avoir un itinéraire sans transfert d'un autobus à un autre, il faut qu'il y ait une ligne d'autobus qui dessert tous les noeuds de cet itinéraire. Ceci se traduit par le processus suivant:

. déterminer tous les itinéraires possibles.

. déterminer les lignes d'autobus directes entre l'origine et la destination.

. trier les itinéraires possibles pour retenir le plus court et la ligne qui le dessert.

Une des possibilités qui n'a pas été étudiée dans ce travail est de minimiser le nombre de transferts sans nécessairement les éliminer.

B.5. Détermination multicritère d'un itinéraire.

Le système propose à l'usager un ensemble de critères qui peuvent être considérés lors de la détermination des chemins. Le processus à suivre est le suivant:

. pondération des critères: essentiels, importants, facultatifs ou inutiles, avec les degrés d'importance respectifs 3, 2, 1 et 0.

. spécifications des différents paramètres de ces critères ou choix de certaines valeurs limites d'estimation

par l'utilisateur.

. détermination des itinéraires possibles.

. test de satisfaction des critères essentiels.

. tri des itinéraires et maintient de ceux qui répondent aux critères essentiels.

. test de satisfaction des autres critères (importants et facultatifs) et calcul du degré d'importance de chaque itinéraire. Ce degré est obtenu en multipliant le nombre de critères facultatifs par 1, le nombre de critères importants par 2 et le nombre de critères essentiels par 3. Notons ici qu'on parle de critères qui sont satisfaits par l'itinéraire en question.

. tri des itinéraires retenus et de leurs indices d'importance respectifs.

. détermination du meilleur itinéraire: celui qui possède l'indice d'importance le plus élevé.

. détermination d'un sous ensemble d'informations additionnelles sur l'itinéraire retenu comme sa longueur, son temps de voyage total et la liste des critères qu'il satisfait.

B.6. Informations partielles directes sur le déplacement:

L'usager a la possibilité d'éviter de passer par toutes les étapes de la détermination d'un itinéraire et de demander des informations partielles directement. Le système procède souvent par la vérification de certains faits pour répondre à la requète. Les informations disponibles sont les suivantes:

. Détermination de la ligne de transport à emprunter: le système détermine directement le chemin le plus court, consulte la base des faits concernant les lignes d'autobus pour en suggérer une au demandeur.

. détermination de la longueur d'un parcours: le système donne la longueur la plus courte pour aller de l'origine à la destination. Donc il effectue uniquement un calcul du chemin le plus court.

. détermination de la vitesse moyenne de circulation sur un tronçon: ceci se fait par une simple consultation de la base des faits concernant les liens. Rappelons que la vitesse constitue un des arguments qui définissent un tronçon de rue.

. détermination des zones saturées: Dans la base des faits nous avons des clauses qui concernent la saturation. Le système en fait une lecture et renvoie le résultat.

. détermination du paysage local: cette information est aussi contenu dans la description de chaque lien. Le système procède par consultation de la base de faits pour la retrouver.

. détermination des adresses des monuments: il ne s'agit pas d'adresses postales mais plutôt d'une liste de noeuds d'accès représentant des coins de rues. Cette information est aussi présente dans la base des faits et le système procède par simple consultation de cette base.

B.7. Mise à jour de la base des faits:

Cette opération consiste soit à **ajouter**, **annuler** ou **modifier** une information. Donc le système doit permettre de faire la modification nécessaire que ce soit pour un élément (prédicat) en entier ou pour un sous ensemble de paramètres (arguments) de ce prédicat.

Pour une modification quelconque, le système affiche les possibilités et l'usager fait son choix. L'information

à modifier étant déterminée (par la détermination du prédicat impliqué), le système affiche une autre liste de possibilités de modifications à l'intérieur de ce prédicat. Ceci étant complété (argument à changer connu), le système reçoit l'ordre de l'usager pour procéder au changement (annulation ajout ou remplacement).

B.8. Règles d'ordre général:

Certaines règles d'ordre général sont utilisées assez fréquemment tout au long du programme. Les principales sont les suivantes:

* test d'appartenance d'un élément à une liste:

- un élément appartient à une liste s'il en est la «tête»*.

- un élément appartient à une liste de «tête» quelconque et de «queue»* Y s'il appartient à cette «queue».

* détermination de la longueur d'une liste c.a.d du nombre de ses éléments. Cette règle est très importante

---

*N.B. la «tête» d'une liste est son premier élément, et sa «queue» est le restant de la liste.

parce qu'elle nous permet de connaître la longueur de la liste lorsqu'elle varie selon les applications. Elle permet aussi d'établir des critères d'arrêt pour certaines situations en plus de la possibilité de calculer des probabilités sans connaître préalablement la taille de l'échantillon (la liste). Son calcul se fait par la règle suivante:

- une liste vide a une longueur nulle.

- une liste de «tête» quelconque et de «queue» Y a une longueur Z si Y a une longueur L et Z=L+1.

* détermination du plus petit ou du plus grand élément d'une liste pour les différents tris et comparaisons en cours de route. Voici les règles respectives:

. Un élément est le plus petit d'une liste lorsqu'il est son élément unique.

. R est le plus petit élément d'une liste Z

si R est le plus petit élément de tout sous ensemble de Z.

- Un élément est le plus grand d'une liste lorsqu'il est son élément unique.

- Un élément R est le plus grand élément d'une liste Z

si R est le plus grand élément de tout sous ensemble de Z.

Certaines hypothèses de travail ont été établies pour

faciliter l'application. Par exemple pour les lignes d'autobus nous nous sommes limités aux données de l'heure de pointe uniquement sans considérer les données de la période hors pointe ni les périodes de service. Pour les monuments nous avons omis de considérer les heures de service. Avec une horloge dans le S.E. ces données de temps pourraient améliorer l'exploitation des informations disponibles.

Le système s'adresse à la fois à l'usager du service et à l'opérateur, dans un cadre académique restreint.

Cette conceptualisation du problème à résoudre peut se traduire par un **schéma fonctionnel** qui servira de point de départ pour l'étape suivante de la **formalisation.** Une esquisse de ce schéma est présentée à la figure 2.2..

II.C. Formalisation du savoir:

La conceptualisation étant complétée par l'établissement du schéma fonctionnel, nous sommes maintenant prêts à l'étape que certains appellent de " traduction " des éléments de connaissances en TURBO-PROLOG. Cette traduction est en général connue sous le nom de ***formalisation du savoir.***

Certains prérequis sont indispensables pour la bonne compréhension de la présente section. Pour les lecteurs qui

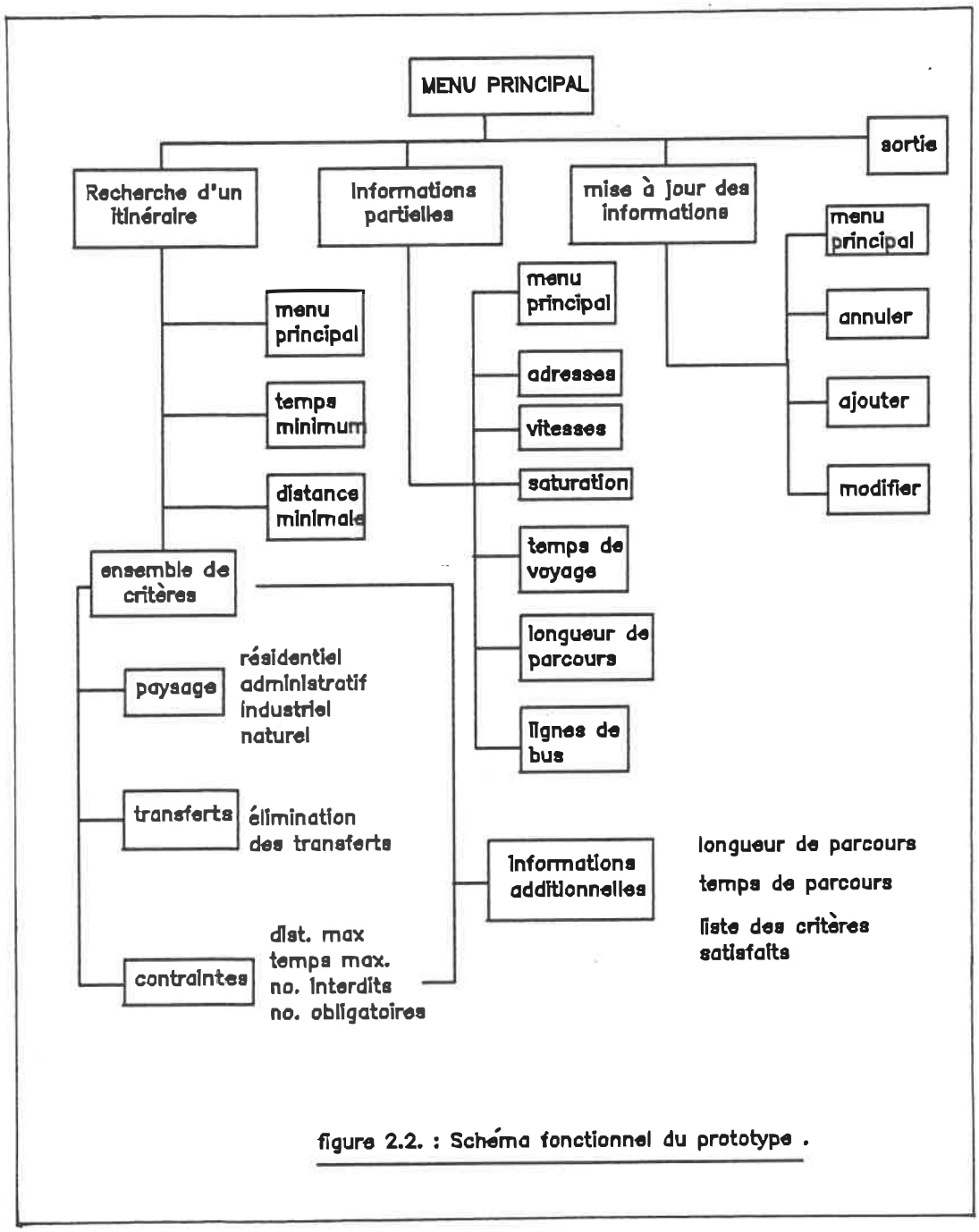


figure 2.2. : Schéma fonctionnel du prototype .

ne sont pas familiers avec les langages de programmation logique nous suggérons de consulter les références 4, 5, 24, 39 et 60.

Nous traiterons ici trois points en particulier:

- la représentation des faits.
- la représentation des règles.
- et la déclaration des domaines et des prédicats.

C.1 Représentation des faits:

Un fait est représenté par un prédicat (nom) suivi d'une série d'arguments (paramètres) entre parenthèses:

prédicat (argument1,argument2,...,argumentN).

La formalisation des différents éléments de la base des faits est la suivante:

1. tronçon ("noeud1","noeud2","rue",longueur,vitesse, ["bus1","bus2",....,"busN"],"type de paysage").

Le nom de chaque noeud est composé par les premiers syllabes des deux rues en intersection. Il s'agit d'un choix de représentation et non d'une contrainte car le système accèpte tous les noms que l'usager donne.

2. artère("rue",["noeud1","noeud2",...,"noeudN]).

3. ligne("nom de ligne",["noeud1","noeud2",...,"noeudN"], fréquence en heure de pointe).

Dépendamment de la forme du trajet d'une ligne, le parcours peut ou non être représenté par une seule clause. Dans le cas d'un parcours en boucle on utilise une seule clause où la suite de noeuds de la ligne commence et se termine par le terminus de la boucle. Exemple:

```
ligne("essalam",["bar-lin","bar-cot","cot-dor","dor-sim",
"sim-bar","bar-lin"],3).
```

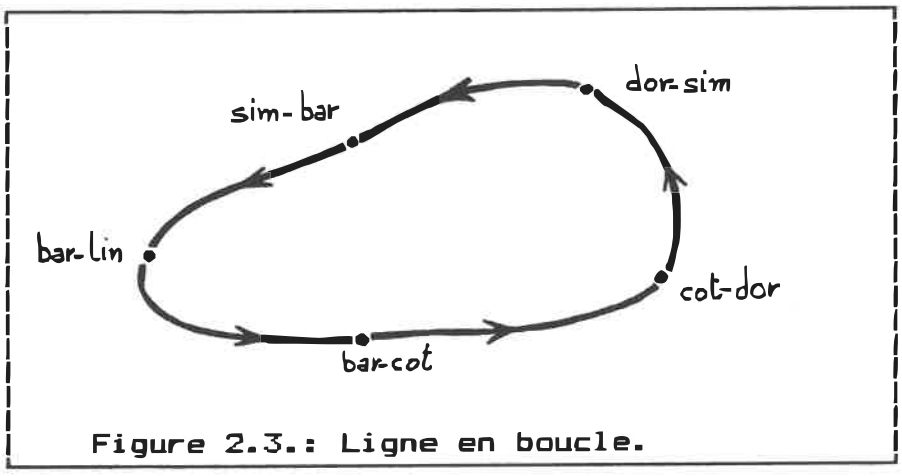


Figure 2.3.: Ligne en boucle.

Dans le cas d'un parcours aller-retour sans boucles, deux clauses sont nécessaires, une pour chaque sens. Exemple:

```
ligne("amana",["bar-lin","bar-cot","cot-dor","dor-sim",
                "sim-bar"],3).
```

```
ligne("amana",["sim-bar","dor-sim","cot-dor","bar-cot",
                "bar-lin"],3).
```

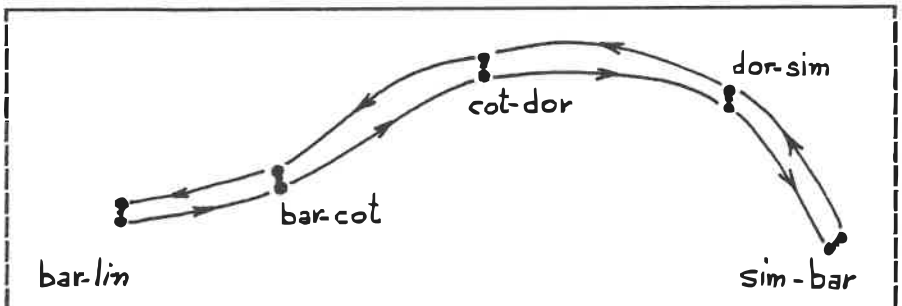


Figure 2.4.: ligne aller-retour.

Dans le cas d'une ligne aller-retour ayant une boucle à l'intérieur deux clauses sont nécessaires. Exemple:

```
ligne("elhana",["bar-lin","bar-cot","bar-dec","dec-cot",
"cot-dor","dor-dup","dup-bar","bar-dec"],4).
    ligne("elhana",["bar-dec","bar-cot","bar-lin"],4).
```

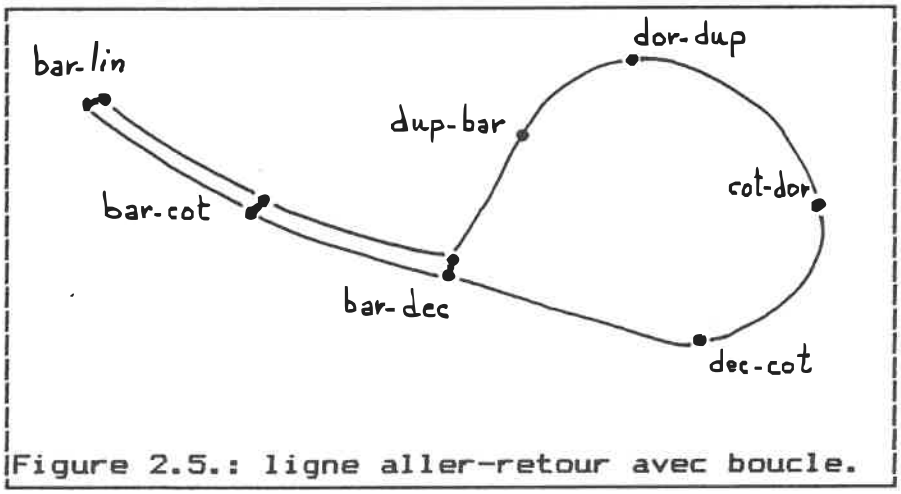


Figure 2.5.: ligne aller-retour avec boucle.

4. monument("nom de monument",["noeud1","noeud2",...,
"noeudN"]).

Le nombre de noeuds d'accès varie d'un monument à l'autre. Chaque noeud d'accès représente l'intersection de deux rues adjacentes au monument.

5. saturé("nom de rue","noeud1","noeud2",pourcentage).

La saturation est décrite pour chaque lien de façon indépendante. Cette approche peut être modifiée pour permettre de représenter la saturation sur une partie de la rue composée de plusieurs liens par une seule clause. Les noeuds «noeud1» et «noeud2» représenteraient alors les limites de la zone saturée et non les limites d'un lien saturé.

6. critères(["critère1","critère2",...,"critèreN"]).

7. les_contraintes([1,2,...,N]).

A chaque numéro correspond une contrainte prédéfinie dans le système. Ces contraintes sont explicitement présentées à l'usager lors de l'exécution et les numéros correspondent aux choix que ce dernier peut faire parmi la liste qui lui est présentée. Actuellement, les choix disponibles sont:

a. interdiction de passer par certains noeuds.

b. obligation de passer par certains noeuds.

c. distance maximale de parcours.

d. temps maximum de parcours.

8. les_paysages([1,2,...,N]).

Chaque type de paysage correspond à un numéro implicite dans le système. Les choix disponibles actuellement sont: 1 résidentiel.

2 industriel.

3 administratif.

4 naturel.

C2. Représentation des règles:

Une règle est une *clause* composée par une *conclusion* et un ensemble de *conditions*. La signification logique d'une règle est la suivante: *la conclusion (prémisse) est vraie si les conditions (antécédents) sont vérifiées.*

Dans ce paragraphe nous nous limiterons à quelques règles du prototype construit. Pour mieux assimiler la technique de formalisation il est indispensable de se référer chaque fois à la conceptualisation correspondante.

a. Recherche d'un itinéraire:

Soit O l'origine du déplacement et D sa destination. La détermination d'un chemin entre O et D se fait par la règle suivante:

```
recherche_itinéraire(O,O,_,[],0).
recherche_itinéraire(O,D,V,[Z/ITIN],LONG):-
        tronçon(O,Z,_,LONG1,_,_,_),
        not(appartient(Z,V)),
        recherche_itinéraire(Z,D,[Z/V],ITIN,LONG2),
        LONG=LONG1+LONG2.
```

En TURBO-PROLOG le symbole «:-» signifie **«Si»**.

b. Détermination de l'itinéraire le plus court:

En ayant les mêmes données précédentes on obtient le plus court chemin par la règle suivante :

```
plus_court_itinéraire(O,D,I,LONG):-
   findall(L,recherche_itinéraire(O,D,[O],_,L),K),
   plus_petit(K,LONG),
   recherche_itinéraire(O,D,[O],I,LONG).
```

c. Détermination de l'itinéraire le plus rapide:

Toujours en considérant les mêmes données l'itinéraire le plus rapide est déterminé comme suit :

```
plus_rapide_itinéraire(O,D,[O],I,L):-
findall(T,temps_total(O,D,[O],I,T),TOUT),
plus_petit(TOUT,K),
temps_total(O,D,[O],ITIN,K).
```

Le temps total de voyage est déterminé par les règles suivantes:

d. Calcul du temps total de voyage:

```
temps_total(O,D,[O],I,TEMPS):-
          itinéraire_temps(O,D,[O],I,TPARC),
          attente([O/I],TATT),
          TEMPS=TPARC+TATT.
```

Le temps de parcours s'obtient par :

```
itinéraire_temps(O,O,_,[],0).
itinéraire_temps(O,D,V,[Z/ITIN],TPARC):-
          tronçon(O,Z,_,_,_,_,_),
```

```
                not(appartient(Z,V)),
                temps(O,Z,T1),
                itinéraire_temps(Z,D,[Z/V],ITIN,T2),
                TPARC=T1+T2.

temps(X,Y,T):-      tronçon(X,Y,_,D,V,BUS,_),
                     not(vide(BUS)),
                     T=60*D/V,!.
temps(X,Y,T):-      tronçon(X,Y,_,D,_,BUS,_),
                     vide(BUS),
                     T=60*D/5,!.
vide([]).
```

Le temps d'attente s'obtient par :

```
attente([_],0).
attente([X/[Y/Q]],A):-
                     tronçon(X,Y,_,_,_,BUS,_),
                     not(vide(BUS)),
                     attendre(BUS),
                     findall(L,fréquence(_,L),FREQ),
                     plus_petit(FREQ,W1),
                     A=W1/2, !
                     ;
                     attente([Y/Q],A),!.
```

```
attendre([]).
attendre([X/Q]):-  ligne(X,_,F),
                   not(fréquence(X,_)),
                   assertz(fréquence(X,F)),
                   attendre(Q)
                   ;
                   attendre(Q).
```

---

e. Appartenance d'un élément à une liste:

On défini une règle pour tester l'appartenance d'un élément à une liste, ensuite grâce au principe de la récursivité on utilise cette règle pour tester l'appartenance d'un ensemble d'éléments à une même liste:

```
appartient(X,[X/_]).
appartient(X,[_/Y]):-  appartient(X,Y).

appartiennent([],_).
appartiennent([T/Q],H):-      appartient(T,H),
                              appartiennent(Q,H).
```

---

f. Plus petit ou plus grand élément d'une liste:

Soit [X,Y|Z] une liste d'éléments quelconques dont on ignore ou non le nombre d'éléments, et soit R le plus petit ou le plus grand élément de cette liste. R est déterminé ainsi:

```
plus_grand([X],X).
plus_grand(X,Y/Z],R)   :-      X≥Y,
                                plus_grand([X/Z],R)
                                ;
                                X<Y,
                                plus_grand([Y/Z],R).

-------------------------------------------------------

plus_petit([X],X).
plus_petit([X,Y/Z],R):-        X≤Y,
                                plus_petit([X/Z],R)
                                ;
                                X>Y,
                                plus_petit([Y/Z],R).
```

Ce sont là quelques exemples de formalisation des différents concepts du processus de résolution. Les commentaires inclus dans le programme sont de nature à

faciliter la visualisation du formalisme qui y est utilisé.

C.3. Déclaration des domaines et des prédicats:

Il s'agit d'une particularité de TURBO-PROLOG qui exige de spécifier la composition de chaque prédicat et la nature de chacun de ses arguments (réel, entier, symbôle etc...).

Deux sections différentes sont prévues à cette fin:

- **domains** pour la nature de l'argument.
- **predicates** pour la définition du prédicat.

Pour illustrer le principe considérons un prédicat quelconque. Soit:

saturé("rue","n1","n2",90). La déclaration se fait ainsi:

```
domains
rue = symbol
n1=string
n2=string
p = integer
predicates
saturé(rue,n1,n2,p)
```

Cette façon de déclarer les prédicats et les arguments a l'avantage d'assurer une très grande transparence du

programme. Cependant quand il s'agit d'un programme qui dépasse quelques dizaines de prédicats, elle devient très lourde. Une façon de simplifier la déclaration consiste à déclarer le type d'argument à l'intérieur de la définition du prédicat. On obtient alors:

```
predicates
saturé(symbol,string,string,integer).
```

Et la section «domains» n'est plus nécessaire. Cette simplification ne peut pas être utilisée pour déclarer les éléments d'une liste. Soit le prédicat:

```
ligne("autobus",["n1","n2","n3"],5).
```

La déclaration se fait nécessiarement en deux étapes, l'une dans «domains» et l'autre dans «predicates»:

```
domains
     lnoeuds = string*
predicates
     ligne(string,lnoeuds,real).
```

Le symbôle "*" dans «string*» signifie que «lnoeuds» est une liste d'éléments de type «string».

II.D. Implantation du programme:

Cette étape consiste à rassembler tous les éléments de savoir formalisés dans un programme exécutable. Les principales parties de ce travail sont:

1. Une première écriture du programme.

2. L'établissement de la structure de contrôle.

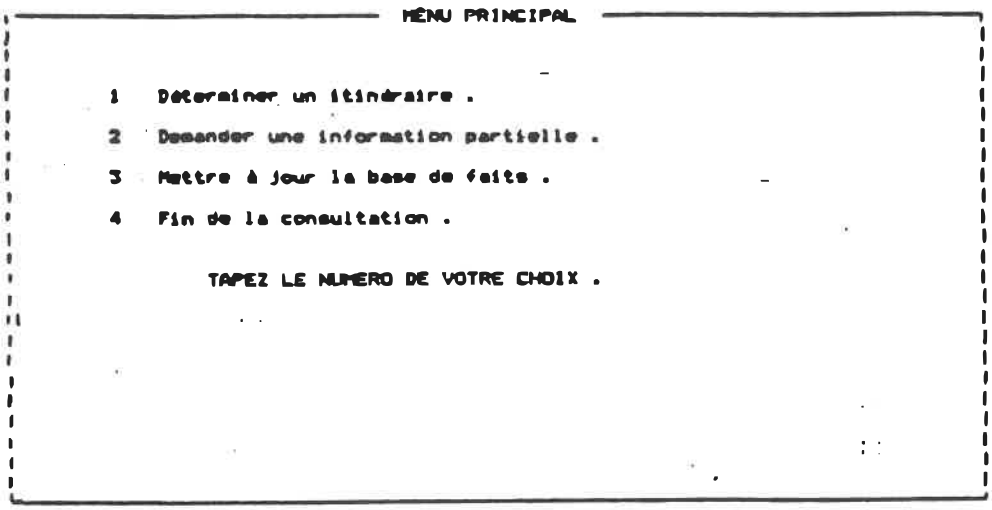
3. La conception de l'interface usager.

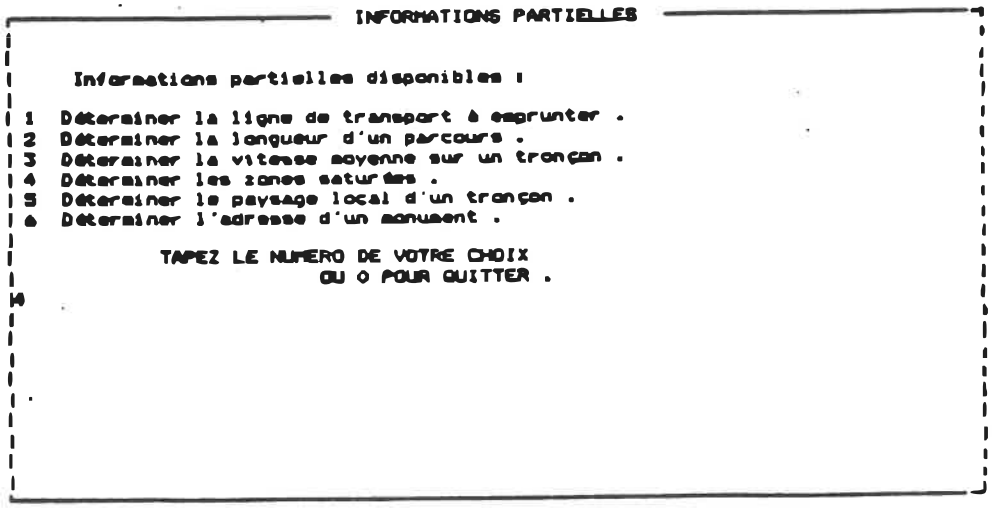
4. Et la compilation du programme.

La première écriture du programme est le regroupement des éléments de connaissances formalisés ensemble. Il faut assurer la cohérence entre toutes les règles, éviter la redondance, et supprimer toute contradiction dans le système.

La structure de contrôle comprend deux composantes: une interne et une autre externe. Le contrôle interne est assuré par le moteur d'inférence de TURBO-PROLOG. Il décide de l'ordre de priorité entre les différentes règles, gère la mémoire lors de l'exécution et exécute toutes les règles. Quand au contrôle externe, il fait partie du programme. Il valide les données fournies par l'usager, contrôle l'accès à certaines fonctions du prototype et gère les résultats obtenus.

L'interface-usager utlisée dans notre S.E. fonctionne par menus à choix multiples. Un menu principal présente à l'usager les différentes opérations faisables et des sous-menus le guident à chaque étape. Comme exemple nous présentons à la figure 2.6. une série de menus utilisés par le prototype.





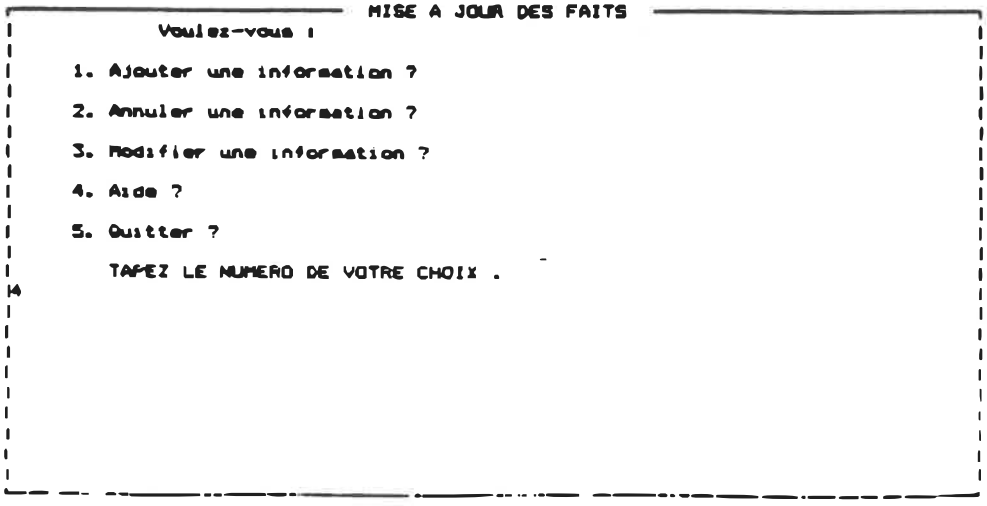


Figure 2.6. : Exemples de menus du prototype .

```
MODALITE DE RECHERCHE

En fonction de quel critère voulez-vous
que la recherche soit effectuée ?

   1  distance minimale de parcours .
   2  temps minimum de voyage .
   3  aucun transfert .
   4  combinaison de critères .

TAPEZ LE NUMERO DE VOTRE CHOIX
4
```

```
RECHERCHE

  Point de départ ?   cher-hub
  Point de destination ?   dor-dem
Les critères de choix d'itinéraires sont :

  1   éliminer les transferts .
  2   bonne qualité du paysage .
  3   imposer des contraintes à l'itinéraire .
        PRESSEZ <ENTER>
```

```
PONDERATION DES CRITERES
Quel poids attribuez-vous au critère :

éliminer les transferts ?
```

```
ECHELLE DE PONDERATION
Les critères d'évaluation d'un
itinéraire sont pondérés selon
l'échelle suivante :

POIDS        SIGNIFICATION

  3          critère essentiel .
  2          critère important .
  1          critère facultatif .
  0          critère inutile .

       PRESSEZ <ENTER>
```

Figure 2.6. : (suite) .

```
CONTRAINTES
Quelle contrainte voulez-vous imposer ?

  1 interdiction de passer par un/des noeud(s) ?
  2 obligation de passer par un/des noeud(s) ?
  3 distance maximale de parcours ?
  4 temps maximum de parcours ?

      ENTREZ UN PAR UN LES NUMEROS DE VOTRE
      CHOIX . A LA FIN TAPEZ 0 .
```

```
CONTRAINTES
? 3
? 4
? 0
```

```
Quelle contrainte voulez-vou

  1 interdiction de passer p
  2 obligation de passer par
  3 distance maximale de par
  4 temps maximum de parcour

      ENTREZ UN PAR UN LES
      CHOIX . A LA FIN TAPE
```

```
LONGUEUR MAXIMALE
Longueur maximale tolérée ( km ) ?
```

```
MISE A JOUR DES FAITS

  NOM DE L'USAGER ?      soulssi

  CODE D'ACCES ?         soul00163

  ACCES AUTORISE POUR :  soulssi
                CODE :   soul00163

      PRESSEZ <ENTER>
```

Figure 2.6. :(suite) .

La compilation se fait automatiquement avec TURBO-PROLOG. On a quatre options possibles pour compiler le programme:

1. Pour une exécution sous TURBO-PROLOG (Memory).
2. Pour produire un fichier objet (OBJ file).
3. Pour une exécution indépendante (EXE file).
4. Pour exécution comme projet indépendant composé de plusieurs modules (Project).

La première et la deuxième option sont utilisées surtout avant que la version du programme ne soit définitive. Ce dernier peut ou non renfermer une section «goal».

La troisième option est utilisée avec la version finale d'un programme qui n'est pas très long. On doit nécessairement inclure une section «goal» dans le programme. Une fois que la compilation est faite l'usager n'aura plus à charger TURBO-PROLOG pour exécuter son programme.

La dernière option est semblable à la précédente sauf qu'elle s'applique pour les programmes composés de différents modules et qui sont généralement très long. L'un des principaux avantages de cette compilation est de séparer physiquement la base des faits du reste du programme. Le programme doit aussi avoir une section «goal» et TURBO-PROLOG n'est pas nécessaire lors de l'exécution.

L'étape de compilation nous permet de corriger toutes les erreurs de syntaxe ou de logique dans le programme. TURBO-PROLOG permet de procéder aux corrections très facilement grâce à ces indications très claires concernant chaque erreur.

Nous présentons à l'annexe C une copie du programme compilé.

II.E. Tests et améliorations:

Cette étape est la plus importante car elle est la seule qui permet de visualiser le résultat de toutes les étapes précédentes. On doit faire des tests sur la base des faits et sur les différentes règles de raisonnement. Ceci nous permet de voir si l'information a été correctement représentée et si les différents raisonnements sont logiques. L'étape suivante consiste à évaluer la performance du système à l'aide d'une série de statistiques concernants les résultats. Cette évaluation doit nous permettre d'améliorer le programme et de connaître ses limites.

Pour notre système, la difficulté majeure est le manque de mémoire vive lors de l'exécution. En effet, la recherche des itinéraires consomme beaucoup d'espace mémoire à cause de la récursivité d'une part et du nombre

élevé de liens dans le réseau d'autre part. Des chiffres précis concernant l'efficacité d'exécution sont présentés plus loin dans ce chapitre. De même les principales améliorations suggérées sont présentées à la conclusion.

III. EVALUATION DU PROTOTYPE DE S.E.:

Le but de cette section est de sortir les arguments qui pourraient être retenus pour soutenir l'utilisation des S.E. pour différentes applications en transport. La démarche consiste à *illustrer chacun des principaux aspects de la technique S.E. et à dégager ses avantages par rapport aux techniques de programmation habituelles.*

III.A. Modularité du programme:

L'un des avantages de la programmation logique est son aspect déclaratif (modulaire). Cette modularité nous permet de représenter différentes connaissances de façon indépendante. Ainsi on a la possibilité d'effectuer avec beaucoup de facilité toute modification souhaitée sans avoir à reconsidérer la totalité du programme. Les prédicats prédéfinis de TURBO-PROLOG «retract», «asserta» et «assertz» permettent de retrancher ou d'ajouter de l'information sans aucune obligation de revoir le reste du

programme. L'opération de modification peut être interactive pour simplifier la tâche de l'utilisateur. Dans le prototype discuté ici, un module de mise à jour interactif a été construit pour cette fin.

Pendant la programmation, TURBO-PROLOG présente un processus de correction (déboggage) très facile. En effet les composantes du programme sont conçues de façon indépendante de sorte que pour une correction le programmeur doit très rarement examiner des procédures d'énormes tailles. En plus, la fonction «trace» de TURBO-PROLOG lui permet de suivre le cheminement du raisonnement étape par étape, lui permettant ainsi de localiser très rapidement l'erreur.

Un autre atout non négligeable de la modularité est le fait de permettre l'entrée des informations " en vrac ". En effet, chaque fait est représenté par un prédicat suivi d'une suite d'arguments, sans aucune contrainte de format. Enfin, au niveau de l'exécution du programme, la modularité nous assure un nombre variable de possibilités d'exploitation des différentes règles. En d'autres mots, le même prédicat permet d'avoir une multitude d'informations. La combinaison de plusieurs prédicats est une façon d'augmenter encore ces possibilités.

La séparation entre les faits et les règles de raisonnement nous permet de structurer l'exploitation de la

connaissance. Elle offre les possibilités suivantes:

- renseignement sur les connaissances assertionnelles (faits) uniquement.
- renseignement sur les connaissances opératoires (règles) uniquement.
- ou renseignement utilisant toutes les connaissances.

Pour illustrer cet aspect modulaire du prototype, considérons l'exemple du prédicat:

recherche_itinéraire(O,D,V,I,L).

où:

O = Origine du déplacement.

D = Destination du déplacement.

V = noeuds Visités (lors du calcul).

I = Itinéraire.

L = Longueur de l'itinéraire.

Lors de l'exécution du S.E. sous TURBO-PROLOG les arguments O,D,V,I,et L peuvent être soit des entités connues (constantes) soit des entités variables (à déterminer). Nous exposons si dessous quelques possibilités d'exploitation de ce prédicat. Deux situations peuvent se produire:

- l'origine et la destination sont constantes.
- l'origine ou la destination est à déterminer.

Lorsque l'origine("o") et la destination ("d") sont fixées on peut demander les informations suivantes:

a. déterminer tous les itinéraires (suites de noeuds) possibles entre "o" et "d".

recherche_itinéraire("o","d",["o"],I,_).

b. déterminer tous les itinéraires entre "o" et "d" ainsi que leurs longueurs respectives.

recherche_itinéraire("o","d",["o"],I,L).

c. vérifier l'existence d'itinéraires entre "o" et "d".

recherche_itinéraire("o","d",["o"],_,_).

d. déterminer, s'il en existe un, le/les itinéraires entre "o" et "d" et qui ont une longueur déterminée *I*.

recherche_itinéraire("o","d",["o"],I,50).

e. vérifier l'existence d'itinéraires entre "o" et "d" ayant une longueur déterminée *I*.

recherche_itinéraire("o","d",["o"],_,50).

f. déterminer les itinéraires entre "o" et "d" ne passant pas par un ensemble de noeuds prédéterminés { "e","f","g" } ainsi que leur longueurs respectives.

recherche_itinéraire("o","d",["o","e","f","g"],I,L).

g. déterminer les itinéraires possibles entre "o" et "d" passant obligatoirement par un ensemble de points {k,l}.

recherche_itinéraire("o","d",["o"],I,L),

appartiennent(["k","l"],I).

h. déterminer les itinéraires possibles entre "o" et "d" ayant une longueur inférieur/supérieur à une valeur donnée.

recherche_itinéraire("o","d",["o"],I,L), L =< 40.

ou

recherche_itinéraire("o","d",["o"],I,L), L >= 40.

La liste des possibilités est encore plus longue lorsque on considère la conjonction avec différents prédicats de comparaison, de test, de calcul etc...

Lorsque l'origine ou la destination est libre (à déterminer ou quelconque) on peut aussi obtenir différentes informations. Par exemple:

* déterminer tous les itinéraires partant de "o" vers tous les autres noeuds du réseau.

recherche_itinéraire("o",X,["o"],I,L).

De façon analogue aux exemples précédents, on peut poser une multitude de questions en introduisant différents

prédicats de comparaison, de tests et de calcul tout en gardant la destination libre (X).

III.B. Justification du raisonnement:

L'un des avantages principaux d'un S.E. est sa capacité d'expliquer ou d'exposer le raisonnement utilisé pour obtenir un résultat donné. Avec TURBO-PROLOG, la justification proprement dite doit être programmée par le concepteur du S.E. et ce, pour chacune des opérations faites par le système. Il s'agit là d'une tâche délicate qui demande un temps considérable. Avec les contraintes de temps qui prévalent pour le présent travail, il n'est pas possible de concevoir un tel module. Cependant, grâce à TURBO-PROLOG, ce problème est en partie résolu. La fonction «trace» nous permet de retracer le schéma de résolution complet que ce soit pour la totalité du programme ou pour une règle ou prédicat particulier. Ce retraçage ("tracing") montre toutes les opérations telles que la vérification d'un fait (vrai/faux) l'appel d'une règle (exécution), l'instanciation (affectation) des variables, la livraison des résultats etc...

Ce prédicat (trace) a cependant un défaut. Il n'est pas possible de faire des retraçages sommaires. A chaque fois il faut passer par toutes les étapes, une par une.

Ceci n'est pas très pratique quand on a des banques de faits assez larges ou des règles qui font usage de plusieurs prédicats. On ne peut donc pas dire que «trace» remplace le module de justification du prototype car il s'agit d'une justification plutôt mécanique et non intelligente.

Ne s'agissant pas d'un S.E. de diagnostic, cette fonction reste suffisante pour notre prototype. En pratique, parmi les S.E. les plus connus par leur système de justification puissant on trouve le système **MYCIN** (voir chapitre I).

III.C. Mode de conversation:

L'idéal pour un S.E. est d'avoir une interface-usager en langue naturelle. C'est le cas des grands S.E. comme MYCIN. En fait, l'interface elle même constitue un S.E. de conversation dans un domaine bien délimité. Dans notre cas, les limites de temps ne permettent absolument pas de réussir un tel exploit. Nous avons donc opté pour une forme de conversation assez simple et amicale, soit les menu à choix multiples. L'utilisateur doit à chaque fois taper le numéro de son choix parmi les possibilités qui lui sont offertes. Ce mode de conversation, bien que simple limite considérablement les possibilités d'exploitation du

S.E. en limitant le nombre et les genres de questions que l'utilisateur veut tester. Cependant, lorsque exécuté sous TURBO-PROLOG la conversation est plus flexible puisque le nombre de but qu'on peut donner au programme reste non limité. L'utilisateur n'est alors pas obligé de passer par le système de menus pour communiquer avec le système.

Dans les menus que nous avons définis, nous utilisons un dialogue simple et bref pour éviter toute confusion et s'assurer d'une clarté maximale. Lorsque jugée nécessaire, une aide a été ajoutée pour aider l'usager à faire un usage adéquat du système.

III.D. Structure de contrôle:

Au premier chapitre, nous avions vu que le S.E. est composé essentiellement du **moteur d'inférence, du langage d'expression** et de la **base des connaissances.** La structure de contrôle est la composante principale du moteur d'inférence. Elle détermine la technique de recherche, établit l'ordre de priorité entre les différentes règles, gère l'espace de travail etc...

Pour notre prototype, la structure de contrôle est celle contenue dans TURBO-PROLOG. Elle est caractérisée par les éléments suivants:

D.1. Stratégie de recherche:

TURBO-PROLOG utilise le **chaînage arrière** selon une recherche **en profondeur** d'abord.

Le chaînage arrière signifie partir de la conclusion d'une règle (prémisse) et vérifier ses conditions de vérité (les antécédents). Si tous les antécédents sont vrais (exécutés sans échec) alors le système déduit que la conclusion est un fait vrai pour la suite de l'exécution.

La recherche en profondeur d'abord signifie que le système essaye à chaque fois de prouver de façon complète une règle en exécution et ne passe à la suivante qu'après un echec.

Pour illustrer cette stratégie de recherche considérons les cas suivants:

Soit la base de faits composée des clauses suivantes:

```
lien(a,b,5).
lien(b,c,10).
lien(a,c,12).
```

et considérons une règle de calcul de chemin:

```
chemin(X,Y,L):- lien(X,X,L).
chemin(X,Y,L):- lien(X,K,L1),
                chemin(K,Y,L2),
                L=L1+L2.
```

La première clause nous dit qu'un **chemin** existe entre

deux points s'il existe un **lien** entre eux. La deuxième dit qu'un **chemin** existe entre X et Y s'il existe un **lien** entre X et un point quelconque K et qu'il existe un **chemin** entre ce point K et la destination Y.

Supposons qu'on pose la question suivante au programme:

chemin(a,c,L).

qui se traduit par:

Y'a-t-il un chemin entre a et c et quelle est sa longueur L?

TURBO-PROLOG cherche le premier prédicat «chemin». Il trouve: chemin(X,Y,L):- lien(X,Y,L). Il procède à l'instanciation suivante: X=a et Y=c. La règle devient alors:

```
chemin(a,c,L):- lien(a,c,L).
```

Il cherche le prédicat: lien(a,c,L) et trouve: lien(a,c,12). Il fait alors l'instanciation finale: L=12. L'exécution est réussie, il essaye d'autres possibilités pour trouver toutes les réponses possibles. Le cheminement est alors le suivant:

```
chemin(a,c,L):- lien(a,K,L1),
                chemin(K,c,L2),
                L=L1+L2.

lien(a,b,5).

K=b, L1=5.
```

```
chemin(b,c,L2):- lien(b,c,L2).
lien(b,c,L2).
lien(b,c,10).
L2=10.
L=L1+L2.
L=15.
```

TURBO-PROLOG livre alors le résultat suivant:

```
L=12
L=15
2 solutions.
```

D.2. Utilisation des connaissances:

L'exécution d'une règle sous TURBO-PROLOG se fait par une série de consultations de la base des faits. En fait, le seul dialogue possible entre deux règles se fait à travers la vérification d'un ensemble de faits (vrais/faux). Lorsque la base de faits est grande, cette façon de communiquer représente un inconvénient puisque le temps d'exécution est alors considérablement augmenté.

Avec TURBO-PROLOG, la notion d'ordre (priorité) persiste toujours. La priorité est implicitement déterminée par l'ordre d'écriture de la règle. Cette façon d'établir la priorité n'est pas intelligente puisque l'ordre dans lequel le concepteur du S.E. écrit les règles ne concorde

pas toujours avec l'ordre de priorité logique propre à chaque cas étudié.

L'utilisation de la disjonction «ou» n'est pas très efficace. Pour éviter une mauvaise interprétation par le programme du «ou» dans une même règle on doit la réécrire pour chaque éventualité. Ceci conduit à une certaine redondance dans le programme.

IV. CONCLUSION:

Tous les spécialistes des S.E. s'entendent sur le fait que l'étape de représentation des connaissances est la plus importante de toutes les étapes de construction d'un S.E.. Le choix de représentation a des conséquences directes sur l'efficacité du système construit. Le concepteur doit choisir une représentation très simple et légère des faits et doit fabriquer des régles qui les exploitent de façon intelligente. Il est donc nécessaire de faire ce choix de représentation en fonction du type de savoir à manipuler. Une des limites du présent travail est le fait d'avoir décidé du langage à utliser dès le départ et avant même de définir précisement l'application. Cette décision se justifie par le fait que le temps alloué à ce travail ne permettait pas d'étudier plusieurs outils de S.E. pour en faire un choix convenable.

Aujourd'hui plusieurs outils de S.E. (Shells) existent sur le marché et sont très intéressants lorsque le concepteur du S.E. n'est pas spécialiste en I.A.. Les spécifications de ces outils sont, en général assez complètes pour guider l'acheteur au bon outil pour son application. Il serait donc intéressant de tester un ou plusieurs de ces outils pour évaluer leurs performances pour des applications telles que la notre.

Un problème majeur a été rencontré lors de la construction du prototype de renseignement. L'insuffisance de l'espace de mémoire vive (RAM) limite le S.E. à des réseaux très petits. Ce problème peut être évité temporairement en optimisant au maximum la gestion de la mémoire interne. Cependant il n'est pas éliminé de façon définitive. Il s'agit là d'un défaut du langage utilisé. En effet, TURBO-PROLOG consomme beaucoup de mémoire vive lors de l'exécution. Cette consommation excessive de mémoire donne une exécution très rapide mais limite la taille des problèmes à résoudre. Jusqu'à la version 1.1 du langage, ces problèmes persistaient encore.

A part les difficultés mentionnées ci-haut, le prototype illustre bien les différents aspects de la technique S.E.. La conversation est assez simple, la validation est présente et les résultats sont obtenus très rapidement en général.

CHAPITRE III.

SYNTHESE

Pour conclure notre travail, il est important de faire une esquisse des différents enseignements à retenir. Ces enseignements doivent être discutés dans le cadre général de la problématique des S.E. pour formuler des recommandations d'applications plus intéressantes de S.E. dans le domaine des transports.

Le présent chapitre est donc un premier arrêt sur le chemin de la mise en application des S.E. dans le domaine des transports où on évalue le pas déjà franchi pour mieux préparer la suite.

I. EVOLUTION DE L'APPROCHE S.E.:

L'I.A. et particulièrement les S.E. continuent à être des thèmes controversés chez plusieurs chercheurs. Les critiques les plus sévères leur sont encore adressées et ce, malgré les divers prototypes conçus et déjà en opération. Les spécialistes de S.E. répondent à ces critiques en cherchant des moyens de plus en plus évolués pour répondre aux préoccupations de leurs clients. Leur tâche n'est pas facile puisque de nombreux problèmes sont

encore à l'origine d'un écart considérable entre la théorie et l'application dans le monde réel. Pour une meilleure performance la tendance actuelle est vers l'intégration des S.E. afin d'impliquer au maximum l'expert et d'éliminer le rôle intermédiaire de l'ingénieur de la connaissance [Boose, 1986] [Gaines, 1988].

I.1. Préoccupations générales:

Après plus de trente (30) ans d'existence, de l'I.A. et par suite des S.E., des questions importantes restent encore sans réponse convaincante. La terminologie et encore faible et manque de clarté et de concision. Les techniques de résolution utilisées ne sont pas toujours originales. Les déclarations des spécialistes sont très souvent exagérées. Et enfin, la recherche est encore, en grande partie, plus philosophique que pratique [Parnas, 1988].

Au niveau de la terminologie, il n'y a toujours aucun standard officiel. Plusieurs termes sont utilisés pour dire la même chose et inversement un même terme est utilisé pour désigner différentes choses [Pavel, 1987]. Plus important encore, plusieurs mots utilisés n'ont pas de définition fixe et standard. L'intelligence par exemple est un mot qui désigne quelque chose qui n'est pas représenté par un système fermé et qu'on ne peut pas évaluer ou quantifier

avec précision. Dire que quelqu'un est à 50% intelligent n'a aucun sens!

Au niveau des techniques de résolution, il n'a pas été prouvé jusqu'à présent, que les langages conventionnels tels que Pascal, C et Basic sont incapables d'égaliser la performance des langages d'I.A.. Sur ce point, plusieurs débats philosophiques se basent sur des affirmations souvent sans aucun fondement pratique tangible. Ce qu'on ne trouve pas c'est une confrontation concrète des deux approches de résolution.

Aujourd'hui la justification et l'explication du raisonnement consistent à un simple renvoie des faits et règles utilisés et l'apprentissage réfère souvent à un stockage mécanique d'informations. Il reste donc encore un champs très vaste de recherche afin de construire des bases théoriques solides pour l'acquisition, la représentation, la justification, la validation, la conversation et l'inférence.

Face à ces critiques, chez les utilisateurs de S.E. règne toujours un climat d'incertitude et d'ambiguïté. Pour quelqu'un qui décide de développer un S.E. il n'est pas évident s'il doit utiliser des machines LISP ou non, s'il doit former ses cadres avec LISP ou PROLOG, si un outil de S.E. est comparable à un langage de programmation, s'il y a une différence entre l'ingénierie de la connaissance et

l'analyse des systèmes.

L'I.A. est une science encore jeune. Il est donc tout à fait compréhensible que de telles questions continuent aujourd'hui à se poser. Les limitations actuelles du point de vue support technique ne doivent cependant pas nous pousser à abandonner cette science [Goebel, 1988].

Plusieurs signes positifs laissent espérer que des applications pratiques performantes viendront bientôt enrichir ce domaine et lui donner plus de crédibilité. En effet, des efforts de standardisation au niveau de la terminologie sont déjà en cours [Pavel, 1987]. En plus, plusieurs chercheurs s'intéressent aux techniques de S.E. d'inférence à un niveau de détail assez élevé. On étudie par exemple le problème de l'auto-gestion de la base de connaissances, le problème de l'acquisition automatique interactive du savoir etc... Ces études devraient contribuer à l'élaboration de bases théoriques solides pour soutenir le processus global de la gestion des connaissances.

Les principaux problèmes qui préoccupent encore les spécialistes de S.E. lors du processus de constructions sont regroupés dans les trois catégories suivantes: - problèmes de professionnalisme,

- problèmes d'acquisition du savoir et,
- problèmes de performance.

1.1. Problèmes de professionnalisme:

Au niveau de la ***théorie*** il n'y a pas de fondements scientifiques solides pour la compréhension de la nature de l'expertise, la représentation, l'acquisition et le traitement de la connaissance. On ne comprend pas encore assez bien **comment** dans les pensées de l'homme, la connaissance est-elle représentée utilisée pour la résolution de problèmes, et transmise d'une personne à l'autre.

Au niveau des ***techniques*** de l'ingénierie des connaissances, il existe un certain folklore de règles-d'arts communément utilisé par les différentes équipes de recherche. Cependant, il n'y a pas eu d'études approfondies jusqu'à présent pour montrer ce qui a été fait et évaluer sa performance. Une telle étude permettrait de converger vers certains standards de base pour l'ingénierie de la connaissance et vers l'identification de techniques bien établies pour ce domaine.

Au niveau des ***outils*** d'aide à la construction des S.E. il manque des systèmes puissants pour l'acquisition des connaissances et leur transfert aux outils de S.E. ("shell").

Au niveau de la ***formation*** des cadres, il n'existe pas de méthodologies avancées et suffisantes pour former des

professionnels bien expérimentés. Les seuls programmes disponibles se donnent au niveau universitaire ou chez certaines organisations commerciales mais la qualité des cours ne répond pas entièrement aux exigences des travaux en cours car ils sont plutôt généraux et non spécialisés. Toutefois on note une amélioration de cette situation puisque la formation est l'atout principal qui supporte l'expansion de la technique des S.E..

1.2. Problèmes d'acquisition du savoir:

On note des problèmes d'accès à l'expertise, de représentation et d'acquisition de la connaissance et de spécification de systèmes (logiciels).

Les experts sont normalement peu disponibles. Il ne faut donc pas sous-estimer le temps qui leur est demandé pour participer au processus de construction d'un S.E.. L'importance de leurs rôles, la faiblesse de leur disponibilité et le coût de leurs services rendent l'accès à leur expertise très difficile. La performance de tous les systèmes réalisés jusqu'à présent est directement reliée à l'expert du domaine de l'étude. Donc il n'y a pas de moyen, pour le moment, de contourner ce problème.

Il existe aussi d'autres problèmes d'ordre technique concernant l'exercice d'acquisition de l'expertise:

* L'expertise peut être aléatoire et les résultats peuvent dépendre de paramètres non contrôlés par l'expert.

* Il est parfois difficile d'exprimer l'expertise par un langage particulier.

* Il est possible que l'expertise exprimée par un langage soit mal ou non comprise.

* La mise en application d'une expertise exprimée par un langage peut être difficile ou impossible.

* L'expertise exprimée peut être inappropriée, incomplète ou incorrecte.

Après l'accès à l'expertise, il s'agit de l'acquérir. La communication des bases de la compétence et du savoir-faire que ce soit entre les hommes ou entre l'homme et la machine est très difficile. Selon le moyen avec lequel l'échange se fait, l'acquisition du savoir nécessite un seul ou deux intervenants. Les méthodes par essai-et-erreur, par analogie ou par application de règles généraux à des situations particulières n'impliquent que la personne ou le système qui veut acquérir un talent ou un savoir-faire quelconque. Cependant, si on procède par un environnement d'apprentissage, par un système d'évaluation du comportement, ou par des exemples, il faut avoir un deuxième intervenant pour guider la personne ou le système qui apprend.

Certains éléments de connaissance ne peuvent pas être **représentés** correctement par les S.E. actuels. Par exemples les règles de production, largement utilisées de nos jours, ne permettent pas de bien représenter des relations non-causales. En général, les formes de représentation connues ne sont pas commodes pour représenter les connaissances procédurales temporelles, qui représentent une composante clé de l'expertise. Finalement, il manque toujours des standards concernant la spécifications des S.E.. Ces spécifications standards sont de nature à garantir une bonne qualité des systèmes à l'intérieur d'un budget et d'une échéance limitée. Ce changement devrait se faire avec l'arrivée sur le marché d'un plus grand nombre de S.E.

1.3. Problèmes de performance.

La construction rapide de prototypes est très importante. Elle représente un objectif assez populaire chez les équipes de traitement de données. Les S.E. ont à cet égard certains problèmes de validation , d'utilisation, de maintenance, d'amélioration et d'expansion.

Au niveau de la ***validation***, il est difficile de définir et de quantifier avec objectivité le degré d'expertise du système. A cause de tous les problèmes

énumérés ci-haut, la validation devient une tâche très difficile. Quand on parle de validation, s'agit-il de valider subjectivement par rapport aux opinions de l'expert ou objectivement par rapport aux performances standards? Comment peut-on vérifier avec certitude si la base de connaissance contient l'expertise appropriée, exacte, consistente, complète et à jour?

L'***utilisation*** d'un S.E. n'est pas toujours facile pour un client non spécialiste. Souvent l'explication donnée par le système, ses recommandations et ses questions exigent de l'usager un degré d'expertise presque identique à celui de l'expert. Ce défaut devrait être éliminé par la construction d'interfaces en langues naturelles pour que le système s'adapte au style de conversation adopté par le client qu'il soit expert, technicien ou ignorant.

Au niveau de la ***maintenance***, la nature superficielle et instable de certaines bases de connaissances peut occasionner des efforts énormes et continus pour les maintenir à jour. La taille généralement très importantes des bases de connaissances augmente considérablement les efforts à déployer pour cet exercice. Dans un contexte où les informations varieraient de façon continue, la mise à jour peut exiger un effort comparable à celui de la construction de la base de connaissances.

L'***intégration*** des S.E. dans des systèmes plus généraux afin d'améliorer leur rendement n'est en général pas possible. En effet, la majorité des S.E. réalisés et les outils qui les supportent ont été développés de façon indépendante sans considérer l'éventualité d'une intégration de ce genre. De ce fait, il sera difficile de les intégrer à d'autres systèmes et d'améliorer leurs performances [Gaines, 1988].

I.2. Processus de développement des S.E.:

Le processus de développement des S.E. implique nécessairement les cinq (5) entités suivantes:

- l'expert,
- l'ingénieur de la connaissance,
- la base de connaissances,
- l'outils de S.E. (shell), et
- le client.

Ce processus définit ainsi le cheminement de la connaissance depuis l'expert jusqu'au client tel qu'il existait pendant les premières années d'existence des S.E. (figure 3.1.).

Quelques années plus tard, on a constaté que le rôle intermédiaire de l'ingénieur de la connaissance était une source de perte ou de déformation de l'information. On a

alors pensé à automatiser le processus d'acquisition du savoir de sorte que l'expert communique avec un système d'acquisition directement. Dès lors, le rôle de l'ingénieur de la connaissance est devenu de superviser l'opération d'acquisition en aidant l'expert à bien communiquer avec ce système. La tâche de ce dernier est devenu plus facile mais il doit connaître à fond le système d'acquisition pour pouvoir apporter l'assistance nécessaire à l'expert, surtout au début de l'opération. Ce deuxième processus (figure 3.2.) montre une séparation entre les deux opérations d'acquisition et d'application. Cette séparation constitue un inconvénient lors de certaines validations ultérieures. En effet, l'expert ayant participé à la construction de la base de connaissance devient, une fois que le S.E. est complété, un client. Il peut alors vouloir effectuer des modifications pour adapter le système à certaines réalités nouvelles. Pour le faire il est souhaitable que la rupture entre l'acquisition et l'application disparaisse et qu'il y ait une intégration du système d'acquisition et de l'outils de S.E. (figure 3.3.). Cette troisième configuration décrit ce qu'on appelle un ***Système d'Aide à la Connaissance*** (S.A.C). La base de connaissances constitue un noyau autours duquel gravitent les différents outils et systèmes que les usagers peuvent utiliser pour l'exploiter (figure3.4.). Les S.A.C

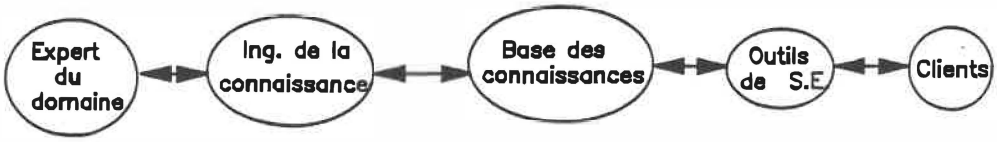


Figure 3.1. : Approche initiale de développement des S.E.

Ref. [Gaines,1988] , (adaptation) .

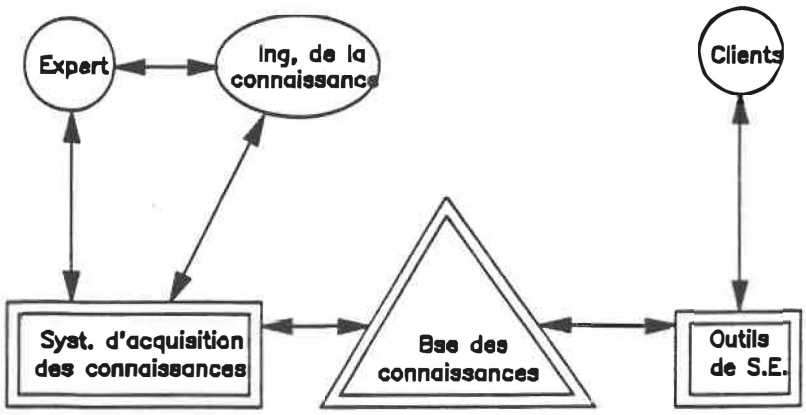


Figure 3.2. : Acquisition automatique des connaissances .

Ref. [Gaines , 1988] , (adaptation) .

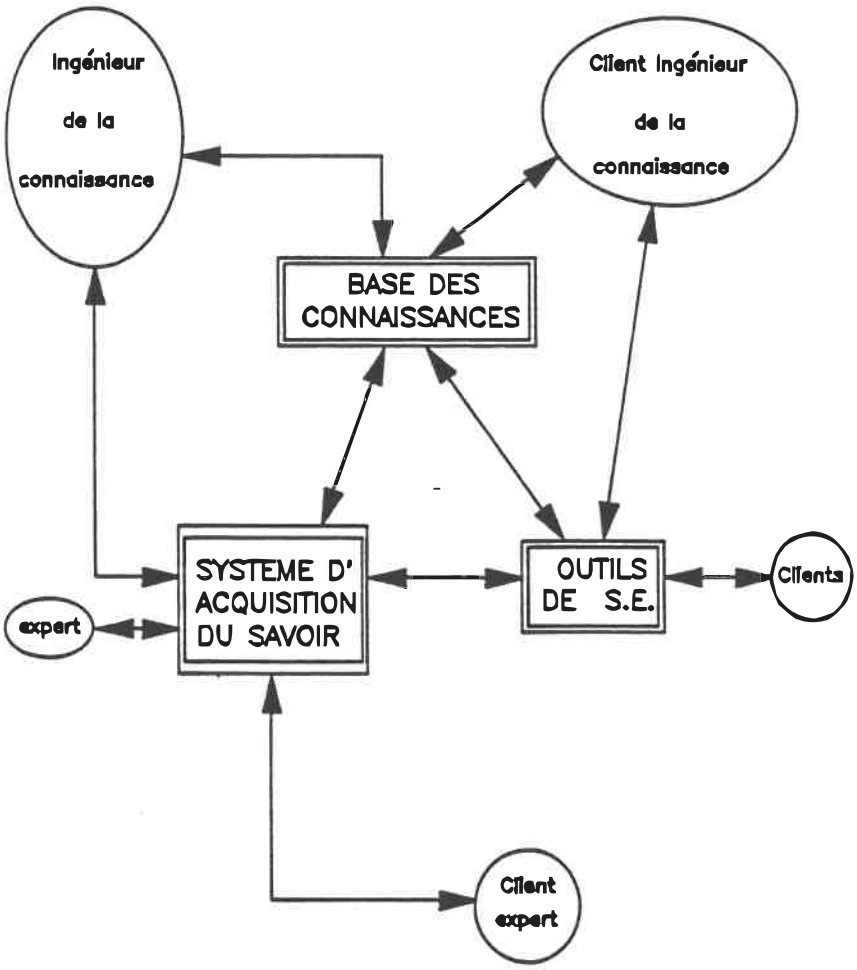


Figure 3.3 : Intégration de l'acquisition des connaissances et de l'outils de S.E.
Réf. [Gaines,1988] , (adaptation) .

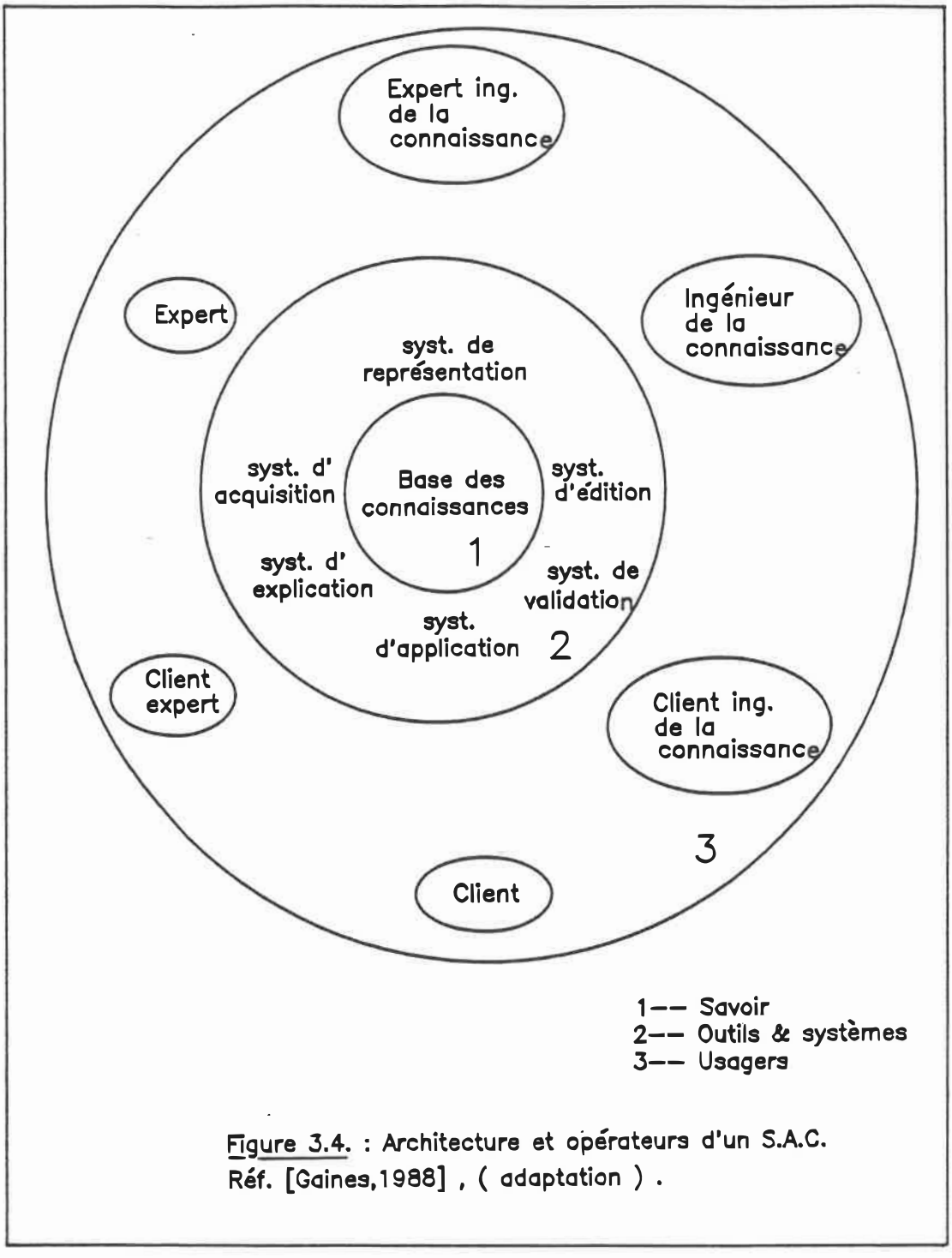


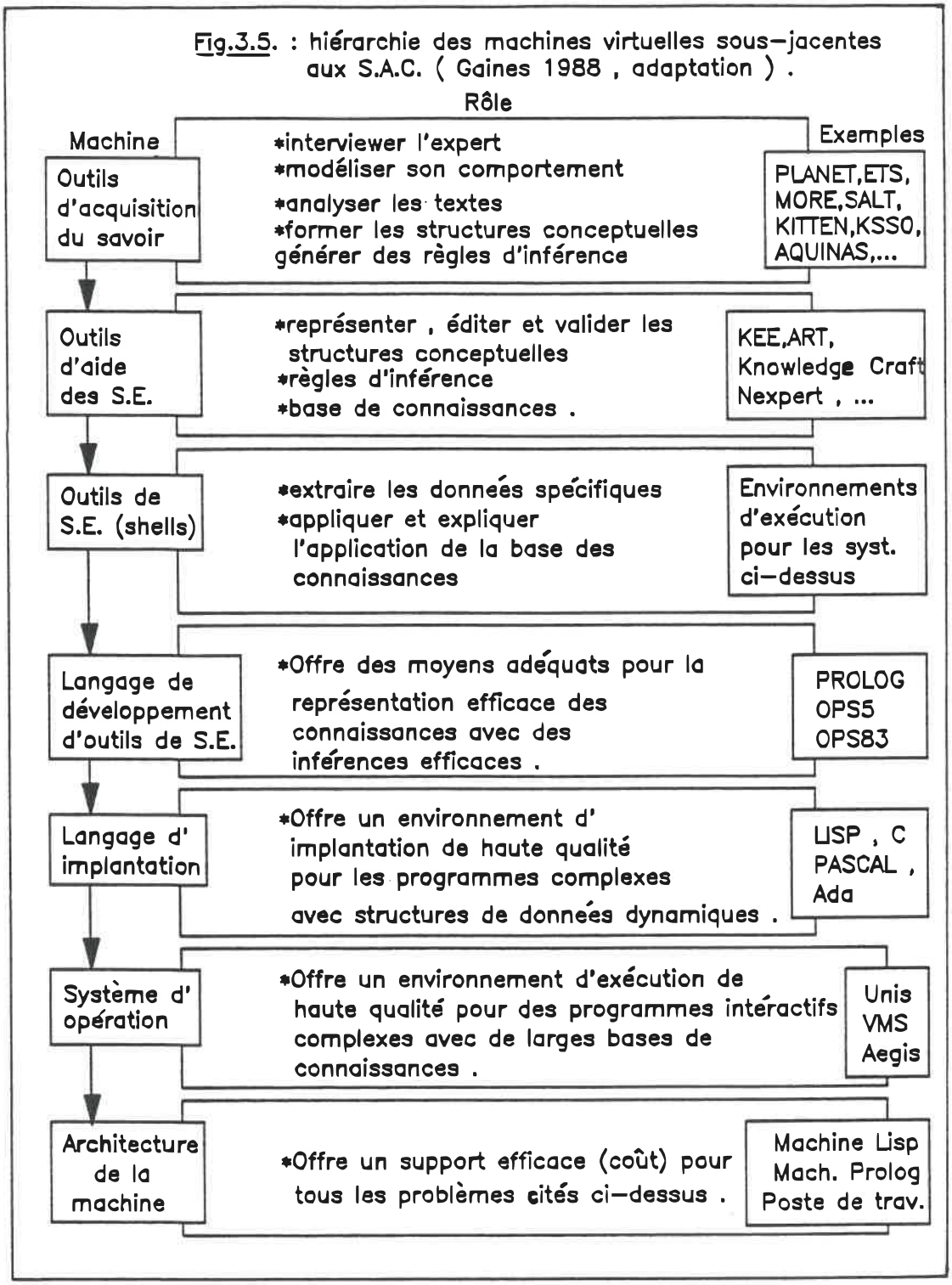
Figure 3.4. : Architecture et opérateurs d'un S.A.C.
Réf. [Gaines,1988] , (adaptation) .

représentent la tendance actuelle au niveau du développement des S.E.. La plupart des équipes de recherche préfèrent intégrer les différents modules du S.E. dans un système global d'aide à la connaissance. Un tel système est supporté par une hiérarchie de machines virtuelles dont la fiabilité a des incidences directes sur l'efficacité d'exécution. A la figure 3.5. on illustre cette hiérarchie dans un ordre de niveau décroissant.

Cette nouvelle tendance est l'aboutissement de trois décennies de recherche dans le domaine des S.E.. Les nombreuses limitations et défaillances observées ont obligé les spécialistes de ce domaine à accepter le fait que les bases théoriques de la science des S.E. ne sont pas assez solides et qu'il faut travailler à la construction de ces bases avant de pouvoir parler de systèmes réellement intelligents . Le fait est qu'on a besoin de mieux comprendre la nature de l'expertise elle même pour pouvoir appliquer ce savoir à l'acquisition des connaissances d'un domaine particulier.

Au niveau de la technologie, on espère que les réalisations de la cinquième génération d'ordinateurs permettra grâce à l'architecture parallèle de ces machines d'améliorer la performance des programmes d'I.A. du point de vue de la vitesse d'exécution et de la taille des problèmes traités.

Fig.3.5. : hiérarchie des machines virtuelles sous-jacentes aux S.A.C. (Gaines 1988 , adaptation) .



II. ENSEIGNEMENTS GENERAUX:

Pour évaluer le prototype de S.E. de renseignement que nous avons construit par rapport aux préoccupations et tendances actuelles dans ce domaine, nous allons revenir au début de l'application pour décrire les principales hypothèses et conditions initiales. La situation se résumait ainsi:

1. nombre de personnes participant de façon directe au processus de développement du S.E. = une personne.

2. Outils et systèmes disponibles = langage TURBO-PROLOG version 1.0 (initiale).

3. Equipement (support matériel) = micro-ordinateur IBM-AT avec 640K de mémoire vive.

4.Contraintes = l'exercice devait se faire à l'intérieur de la période maximale permise pour un mémoire de M.Sc.A. (environ 2 ans).

Etant donnée ces conditions de départ, l'approche initiale du processus de construction d'un S.E. a été utilisée (voir figure 3.1). Notons cependant que nous avons dû jouer le rôle de toutes les personnes impliquées par cette approche, soit l'expert, l'ingénieur de la connaissance et le client. Cette fusion des rôles a occasionné certaines difficultés surtout au niveau de la distinction entre les différentes étapes.

N'étant pas spécialiste dans le domaine de l'ingénierie de la connaissance, le temps d'expérimentation et d'étude de la littérature a été relativement long réduisant ainsi le temps alloué au processus de développement proprement dit.

S'agissant d'une première expérimentation dans ce domaine, il était astucieux, étant donné les limites de temps imposées au projet de poser des hypothèses simplificatrices telles que l'utilisation d'un langage bien déterminé (TURBO-PROLOG) sans passer par une recherche de l'outils de S.E. qui est le plus approprié à l'application envisagée. Cependant, ce choix a constitué une contrainte très restreignante à cause de certaines faiblesses du langage choisi. Nous considérons que pour une application de plus grande envergure, l'étape de choix de l'outil de S.E. est très importante à cause des incidences directes qu'elle peut avoir aussi bien sur la représentation des connaissances que sur l'exécution.

Lors de la construction d'un S.E. nous avons en réalité besoin de deux types différents d'experts: un expert dans le domaine de l'application dont l'expertise sera transférée à la base de connaissances du S.E., et un expert dans le domaine de l'ingénierie de la connaissance dont l'expertise servira à superviser toute l'opération de développement du système. Au cours de l'exercice conduit

dans le présent travail, la dernière forme d'expertise a été acquise en parallèle avec l'opération de développement du système.

Le système construit avait comme but d'illustrer certaines techniques de la programmation logique utilisée dans les S.E.. Il était donc acceptable de choisir une application fictive. Il en a résulté un S.E. d'information dont l'expertise est très modeste.

Ces quelques remarques concernant le processus de développement de notre système de renseignement, les outils utilisés et les résultats obtenus, nous ont permit de formuler quelques enseignements utiles pour la définition de nouvelles applications de S.E. en transport.

III. RECOMMANDATIONS:

En tant que spécialistes du domaine des transports, la technique des S.E. est pour nous un moyen informatique pour atteindre certains objectifs. Nous sommes donc consommateurs d'un produit donné. Il nous appartient alors de nous assurer que ce produit répond bien à nos besoins. Pour cela, deux alternatives nous sont offertes: faire une étude des outils de S.E. disponibles pour choisir celui qui nous satisfait le plus, ou choisir un langage donné pour construire notre outil nous même. La première

alternative demande un investissement d'argent assez important et nous impose toutes les limitations de l'outil, mais permet de gagner du temps. La seconde alternative ne nous coûte pas très cher mais nous oblige à investir un temps supplémentaire pour acquérir les notions d'ingénierie de la connaissance et les appliquer au développement de l'outil recherché. A notre avis, il est plus logique de définir les spécifications de l'outil voulu et de laisser le soins aux spécialistes de programmation logique de nous le construire s'il n'en existe pas un équivalent sur le marché. Mieux encore il serait plus intéressant de réunir toutes les disciplines nécessaires au processus de construction du système, soient:

- l'expert du domaine de transport,
- l'expert de l'ingénierie de la connaissance, et
- l'expert du domaine informatique.

Ensemble, ces trois spécialistes peuvent construire un S.E. très performant tout en économisant du temps. Le spécialiste en transport contribue par l'identification du domaine, du problème à résoudre et de l'expertise attachée à sa résolution. Le spécialiste de l'ingénierie de la connaissance contribue au niveau de l'acquisition du savoir, la gestion de la base des connaissances, et l'évaluation du système. Enfin, le spécialiste de

l'informatique contribue au niveau de l'outil de S.E., de la machinerie et de l'amélioration du rendement du système quant à la vitesse d'exécution et la taille des problèmes traités.

Avant de définir des applications directes de S.E. dans le domaine des transports, nous pensons que certains travaux de recherche théoriques sont nécessaires.

III.1. Travaux préliminaires suggérés:

1.1. Interface en langue naturelle:

L'une des faiblesses de la majorité des prototypes de S.E. en transport est la rigidité du mode de conversation. Ceci dégage un besoin urgent de construire un S.E. qui comprend le dialogue technique relatif au domaine général des transports. Ce système constituerait une interface utile à toute la communauté travaillant dans le domaine. Les différentes étapes seraient:

- La structuration du domaine des transports en différentes disciplines. Par exemple: transport des marchandises, transport des personnes, circulation, constructions routières, sécurité dans les systèmes de transports, économie des transports etc...

- La structuration de chaque discipline en différentes orientations. Par exemple pour le transport des personnes on a les orientations de planification, d'opération des systèmes, de gestion etc...

- La caractérisation de chaque orientation par un ensemble d'attributs. Par exemple l'orientation de la planification peut être caractérisé par l'ensemble des opérations qu'elle implique, la structure professionnelle qui l'appui etc...

- La définition de tous les mots techniques utilisés par chaque discipline avec toutes les significations éventuelles de chaque mot. Ceci constituera un genre de dictionnaire pratique aidant à standardiser le dialogue entre tous les utilisateurs du système.

- La onstruction d'une base grammaticale pour la compréhension des verbes très utilisés dans le langage naturel tels que le verbe avoir (possession) et le verbe être (état). Cette même base doit familiariser le système avec les différents modes de conversation tels que l'interrogation (?) et les commentaires.

L'approche décrite ci-dessus est présentée à titre d'exemple et n'est pas nécessairement complète. Il appartiendra à celui qui procèderait à la réalisation d'une telle application de la détailler et de l'adapter aux différentes conditions qui prévalent.

1.2. Identification et caractérisation de l'expertise:

Avant de construire un S.E. on doit s'assurer qu'il y a vraiment une expertise sous-jacente au problème posé. Autrement dit, le S.E. doit réellement contenir une expertise avancée et complète. Ensuite l'expertise identifiée doit être caractérisée pour que le mode de représentation choisi soit adéquat. Par exemple, une expertise du type causes et effets s'exprime bien par des règles de production. A cet effet, certaines études ont énuméré quelques applications potentielles sans toutefois préciser l'importance et la nature de l'expertise qu'on peut exploiter [Bonsall,86],[Abrahamsohm,1987] et [Yeh,1987]. Ces applications ont été définies par rapport à un ensemble de critères très généraux (chapitre I). Ces critères pourraient être plus précis en introduisant une certaine quantification pour mesurer le potentiel d'application avec des paramètres explicites et plus significatifs.

III.2 Applications suggérées:

Certaines applications de S.E. expert dans le domaine des transports nous paraissent assez prometteuses pour que des prototypes soient développés. Il s'agit essentiellement

de S.E. de diagnostic avec des bases de faits relativement petites.

2.1 S.E. pour la redistribution de la flotte d'autobus en situation de crise:

Suite à des situations d'urgence tel que la variation brusque et importante imprévue de la demande à un point donné du réseau, une décision importante doit être prise pour remettre le système à son état normal. Un répartiteur ne peut pas connaître rapidement et au moment opportun la situation générale du réseau. Comme il faut prendre des autobus sur les parcours sous achalandés et les plus proches du point où la crise a lieu, beaucoup de calculs et d'informations peuvent être requis. Un S.E. peut nous rendre un tel service grâce à une banque d'information à jour et grâce à un raisonnement logique et intelligent.

L'intérêt de ce système vient du fait qu'il possède une expertise spécifique propre aux répartiteurs. Donc si ce spécialiste n'est pas disponible au moment de la crise seul un autre spécialiste ou un S.E. peut accomplir son rôle. En plus la nature de cette expertise s'exprime bien sous forme de règles de production car à chaque type de crise il existe une façon de procéder. Chaque action s'énonce alors facilement sous la forme **si... alors...**

2.2. S.E. pour l'optimisation d'un réseau de T.C.:

Très souvent, le réseau de T.C. doit subir des modifications pour s'adapter aux changements urbains. La transformation d'une zone résidentielle en zone administrative par exemple a des incidences directes sur le comportement des personnes qui la fréquentent donc sur la demande locale de T.C.. Pour tenir compte des différents aménagements urbains prévus on doit disposer d'un système très puissant qui tient compte de toutes les éventualités lors du design d'un réseau de T.C..

La base de connaissance de ce système doit posséder:

1. toutes les données de territoire et d'aménagements urbains à venir,

2. toutes les données sur la demande,

3. l'expertise d'un planificateur de réseaux de T.C., et

4. l'expertise d'un spécialiste en économie des transports.

L'intérêt principal de ce système réside dans les coûts qui peuvent être sauvés avec un design optimum. En plus il permet d'établir des critères objectifs d'attribution du service pour une meilleure équité envers tous les usagers du système.

Le développement de ce système peut être simplifié grâce aux nombreux systèmes existant pour le design des réseaux de transport tels que: MADITUC, EMME/2, HASTUS etc... L'intégration d'un ou plusieurs de ces systèmes peut être particulièrement intéressante.

Une des difficultés à laquelle on devra faire face est la grande taille des réseaux par rapport aux capacités des systèmes existants.

2.3. S.E. d'aide à la réalisation d'enquêtes:

La réalisation des enquêtes pour le transport en commun est un processus très complexe et coûteux. Pour profiter des investissements des années précédentes il est très intéressant d'avoir un système d'aide pour nous y assister. Ce système peut avoir les fonctions suivantes:

1. formation automatique et rapide des enquêteurs,
2. élaboration du processus d'enquête,
3. contrôle de l'opération d'interviews,
4. évaluation de la productivité en cours de l'enquête,
5. validation des données recueillies,
6. analyse des données, et
7. production des rapports.

D'autres modules peuvent être ajoutés pour des fins de

planification ultérieure.

Ce système a son intérêt dans les coût qu'on peut sauver pour la réalisation de l'enquête ainsi que dans la garantie d'une meilleure qualité de résultats. Les nombreuses expériences d'enquêtes sont de nature à enrichir l'expertise du S.E..

IV CONCLUSION:

Dans ce chapitre, nous avons présenté une synthèse des différents enseignements et recommandations découlant de l'application réalisée. Nous espérons avoir énuméré les plus importants problèmes et difficultés dans ce domaine ainsi que les quelques erreurs méthodologiques que nous avons inévitablement commises. Nous estimons que ces remarques sont utiles pour la continuité de la recherche sur les S.E. dans le domaine particulier des transports. Nous rappelons finalement que le domaine des S.E. est encore en effervescence continue et qu'il est donc essentiel de suivre de près toutes les nouvelles études qui abordent ce sujet.

CONCLUSION

Ce travail nous a permi de nous familiariser avec la technique des S.E., d'étudier leurs applications potentielles en transport d'examiner les prototypes existants, de construire un prototype pour l'information des usagers du T.C., et d'élaborer une synthèse des récentes tendances dans ce domaine. Il constitue ainsi une base assez complète pour bien définir et développer un S.E. appliqué au domaine du transport.

La revue des applications potentielles suggérées par les différents chercheurs, et la liste des quelques prototypes existants donnent une idée assez complète sur la façon avec laquelle le domaine des transports peut bénéficier de la technique des S.E..

Les critères d'évaluation du potentiel d'application des S.E. en transport paraissent très généraux. Il faut donc les adapter au domaine précis du S.E. à construire et essayer de les quantifier.

Le système d'information proposé dans ce mémoire s'inscrit dans un cadre d'illustration des différents aspects de la programmation logique supportée par le langage TURBO-PROLOG. Il hérite ainsi de toutes les

limitations de ce langage (chap II). Il serait préférable, pour des applications ultérieures, de faire une étude complète pour choisir l'outil de S.E. ou le langage de programmation le plus intéressant pour la construction du prototype.

Plusieurs hypothèses simplificatrices ont été utilisées comme l'utilisation d'un seul mode de transport 9autobus), l'utilisation d'une seule période de service 9facteur temps ignoré) l'utilisation de la fréquence de service de l'heure de pointe dans tous les calculs, etc... Il est essentiel de reconsidérer toutes ces hypothèses pour construire un système d'information plus réaliste.

Bien que le mode de conversation proposé dans le programme soit simple et facile à adopter, il serait plus intéressant d'avoir une interface en langue naturelle spécialisée pour garantir un dialogue plus flexible entre l'usager et le système.

La limite de mémoire vive 9RAM) disponible fait que le prototype ne peut traiter que des réseaux de petites tailles. Cette insuffisance de mémoire se produit lors de la détermination des itinéraires. TURBO-PROLOG ne permet pas une meilleure performance à ce sujet.

L'intégration du système d'information dans un système global est d'un grand intérêt. On pourrait ainsi informer les usagers du T.C. pour qu'ils aient une meilleure

perception du service et en fassent une meilleure utilisation, et les automobilistes pour remédier au problème de la saturation du réseau routier.

Les S.E. en transport doivent être considérés comme des outils complémentaires aux techniques déjà existantes. Notre but ne doit pas être de construire des systèmes indépendants mais d'utiliser les techniques de S.E. là où elles sont bonnes et les procédures algorithmiques et modèles existants là où ils sont performants.

Les divers outils de S.E. ("shell") qui deviennent de plus en plus disponibles sur le marché peuvent être d'un grand intérêt pour l'acquisition automatique de l'expertise. En les jumelant aux procédures analytiques existantes et à une interface en langue naturelle spécialisée on peut obtenir un système d'aide à la connaissance 9S.A.C.) très intéressant.

L'approche S.E. ne cesse pas d'évoluer aujourd'hui et les spécialistes sont encore à la recherche d'une meilleure spécification du problème et d'un meilleur agencement des différentes composantes de leurs produits. Des applications de plus en plus intéressantes sont actuellement à l'étude et nous croyons que le potentiel est élevé pour que les S.E. deviennent des outils utiles aux planificateurs et ingénieurs de transport dans un avenir assez proche.

BIBLIOGRAPHIE

1. A. BRYSON jr D. et STONE, J. R.; (1987); **"Intersection advisor: an expert system for intersection design."** TRR #1145, pp48-53, TRB.

2. ABRAHAMSOHM, G. A.; IRWIN, N. A. et McNIGHT, P.; (1987) **"Expert systems"**; tribune des transports, vol. 3.3, pp41-45.

3. BARR, A. et FEIGENBAUM, E. A.; (1986); **"Le manuel de l'intelligence artificielle."**, tome 1, Edition Eyrolles Paris.

4. BARRY, R.; (1987); **"Expert systems in PROLOG."**; PC AI, pp23-26, été 1987.

5. BIHAN, P. et PARIZOT, P. E.; (1987); **"Exercices en TURBO-PROLOG."**; Ed. Eyrolles, Paris.

6. BONNET, A.; (1984); **"L'intelligence artificielles: promesses et réalités."**; Inter Editions, Paris.

7. BONSALL, P. et KIRBY, H.; (1986); **"The role of expert systems in transport."**; Information technology applications in transport; chapitre 15, pp 353-382; Edition VNU Science Press.

8. BOOSE, J. H.; (1986); **"Expertise transfer for expert system design."**; Boeing A.I. Center, USA, Edition Elsevier.

9. BORLAND International Inc.; (1986); **"TURBO-PROLOG: owner's handbook"**; 1ère édition.

10. BORLAND International Inc.; (1987); **"TURBO-PROLOG TOOLBOX."** 1ère édition.

11. BOUBKEUR, A.; (1986); **"Diagnostic des systèmes hydrauliques assisté par PROLOG"**; E.P.M.; mémoire de maitrise (M.Sc.A).

12. BOYCE, D.E.; (1985); **"Transportation research: the state-of-the-art and research opportunities."**; Transp. Res. Part.A 19A(5/6), pp349-550.

13. BOYCE, D.E.; (1988); **"Route guidance systems for improving urbain travel and location choices."**; Transp. Res.-A Vol.22a No.4, pp.275-281.

14. CHAPLEAU, R.; ALLARD, B. et LEBEAU, L.; **"Embryon d'un "système expert" pour le transport collectif urbain."** Exposé des communications, 21ième congrès de l'A.Q.T.R., Mars 1986, Québec, pp.172-197.

15. CHEBAYEB, F. et CONNOR, J.; (1986); **"GEPSE-A computer environment for engineering problem solving."; Research Report R86-11; Intelligent Engineering systems laboratory; Dept. of Civil Eng.; M.I.T.**

16. CHING-PING CHANG, E.; (1987); **"Using expert systems to select traffic analysis software"**; TRR #1145, pp9-19; TRB.

17. CHING-PING CHANG, E.; (1987); **"Application of expert systems to left-turn signal treatment."**; TRR #1145, pp28-36; TRB.

18. COHN, L. F.; HARRIS, R. A. et BOWLBY, L.; (1988); **"Knowledge acquisition for domain experts."**; ASCE, The journal of computing in civil engineering; vol.2, No.2; pp107-119.

19. DAVIS, D. B.; (1986); **"Artificial intelligence enters the mainstream"**; HighTechnology, pp16-22; Juillet 1986.

20. DAVIS, R.; (1982); **"Expert systems: Where are we and where are we going from here?"**; A.I. Memo # 665;M.I.T; A.I. laboratory.

21. DESROCHERS, M.; (1984); **"Les systèmes experts et la recherche opérationnelle."**; CRT # 386; U.M.

22. ELGHARBI, B.; (1986); **"Une évaluation de MICRO-PROLOG pour le diagnostic des circuits hydrauliques"**; E.P.M.; projet de fin d'études; département de Génie électrique.

23. EXSYS; (1985); **"EXSYS Expert system development package User's Manual."; EXSYS Inc.; Albuquerque; N.Mex.**

24. FAGHRI, A.; JOSHUA, S. C. et DEMETSKY, M. J.; (1987) **" Expert systems for the evaluation of Rail/Highway crossings"** ASCE; The journal of computing in civil engineering; vol.2 No.1; pp21-37.

25. FARRENY, H.; (1985); **"Les systèmes experts: Principes et exemples"**; Cepadues-Editions.

26. FEIGENBAUM, E.; (1984); **"The fifth generation: Japan's computer challenge to the world."**; Creative Computing; Aout 1984; pp103-114.

27. FORASTE, B. et SCEMAMA, G.; (1986); **"Une approche «système expert» du traitement de la saturation."**; INRETS; Recherche Transport Sécurité, Septembre 1986; pp17-22.

28. FORGY, C.L.;(1981); **"The OPS5 User's Manual"**; Technical report CMU-CS-81-135; Computer Science department; Carneige-Mellon University; Pittsburgh; Pa.

29. FRUIN, J.J.; (1985); **"Passenger Information Systems for Transit Transfer Facilities."**; NCTRDP; Synthesis of Transit Practice no7.; TRB.

30. GAINES, B.R.; (1988); **"Knowledge acquisition systems for rapid prototyping of expert systems."**; INFOR vol. 26 no.4; pp256-258.

31. GALLAIRE, H.;(1985); **"La représentation des connaissances"** La recherche no170, oct. 1985.

32. GANASCIA, J. G.; (1985); **"La conception des systèmes experts"**; La recherche # 170, volume 16.

33. GEVARTER, W.B.; (1983); **"An overview of artificial intelligence and robotics."**; Volume I-Artificial Intelligence PartB-Applications; Technical Memorandum

85838; U.S. department of commerce, NTIS.

34. GOEBEL, R.; (1988); **"Distinguishing science and technology in artificial intelligence: a reply to D. PARNAS."**; INFOR vol. 26, no. 4.

35. GOSLING, G. D.; (1987 O; **"Application of expert systems in air traffic control"**; ASCE, The journal of transportation engineering.

36. GOSLING, G. D.; (1987); **"Identification of artificial intellgence applications in air traffic control."**Transp. Res. A, vol 21A, No 1; G. B.

37. HAJEK, J. J.; CHONG, G. J.; HAAS, R. C. ET PHANG, W. A. (1987); **"Knowledge-Based Expert System technology can benefit pavement maintenance."**; TRR # 1145, TRB.

38. HARRIS, R. A.; COHN, L. F. et BOWLBY, W.; (1987) **"Designing noise barriers using expert system CHINA"**; ASCE The journal of transportation engineering.

39. HAYES-ROTH, F.; (1983); **"Building expert systems."**; Addison-Wesley publishing company.

40. HUNT, V.D.; (1986); **"Artificial Intelligence & Expert Systems sourcebook."**; 1ère édition, CHAPMAN & HALL.

41. JACOB, J. P.; (1987); **"L'ordinateur complice de l'intelligence."**; Conférence Augustin Frigon, 14 octobre 1987; Ecole Polytechnique de Montréal.

42. KINNUCAN, P.; (1985); **"Software tools speed expert system development."**; High Technology, mars 1985.

43. KODRATOFF, Y.; (1985); **"Quand l'ordinateur apprend"** La recherche, # 170, vol 16.

44. KRUCHTEN, Anne et Philippe; (1985); **"Programmer en PROLOG."** Ed. Eyrolles, Paris.

45. LAURIERE, J. L.; (1987); **"Intelligence Artificielle: Résolution de problèmes par l'homme et la machine"**; Edition Eyrolles, 3ième édition, Paris.

46. LEWIS, R. T. et JOHNSON, W. F.; (1987); **"Application of expert systems to transportation: A strategy for safety and productivity gains."**; TRR # 1145, TRB.

47. LFR, Inc.; (1986); **"INSIHGT2+ Reference Manual"**; Level Five Research (LFR) Inc.; Indialantic; Fla.

48. MAGWOOD, P.; (1987); **"Ces nouveaux adjoints intelligents"** Transpo/87, vol 10/3.

49. MARCHE, C. et BENOIT, R.; (1987); **"Les systèmes experts dans les décisions de stratégies en CAO."**; Texte de la communica-tion présentée au 3ième colloque canadien sur la théorie des systèmes dans le génie civil, Montréal.

50. McNEIL, S. et FINN, A. M.; (1987); **"Expert system to cost feasible bridge-painting stratégies."**; TRR # 1145 TRB.

51. MOULIN, B.; (1986);**"Les systèmes expert à base de connaissances dans les organisations."**; L'ingénieur, Septembre-Octobre 1986.

52. MOULIN, B.; (1987); **"Sytèmes experts: une technologie à la portée des entreprises."**; L'ingénieur # 379, 73ième année.

53. MOULIN, B.; (1987); **"Une démarche pour la conception des systèmes experts."**; l'ingénieur # 379, 73 ième année.

54. MOULIN, B.; (1985); **"L'ingénierie du savoir."**; l'ingénieur, Mars-Avril 1985.

55. PARENT, R.; (1985); **"Point de vue québecois sur l'intelligence artificielle."**; Notes de cours de Mr. J. BONHOMME; Ecole Polytechnique de Montréal.

56. PARNAS, D.L.; (1988); **"Why engineers should not use artificial intelligence."**; INFOR vol. 26, no. 4.

57. PASCHE, C.; (1987); **"Une approche de l'analyse multicritère par les systèmes experts."**; Cahiers du C.E.R.O. VOL.29, No 1-2, E.P.F. de Lausanne.

58. PAVEL, S.; (1987); **"Intelligence Artificielle"**; Les cahiers de terminologie, édition provisoire, secrétariat d'état du Canada, direction générale de la terminologie et des services linguistiques.

59. PEHLIVANIDIS, M.; (1987); **"L'application des systèmes experts dans la gestion des chaussées."**; Routes et Transport vol.XVII, No.4 / Hiver 1987, AQTR.

60. RITCHIE, S. G.; (1987); **"A Knowledge-Based approach to pavement overlay design."**; TRR # 1145, TRB.

61. RITCHIE, S. G.; YEH, C.; MOHONEY, J. P. et JACKSON, N. C.; (1987); **"Surface condition expert system for pavement rehabilitation planning."**; ASCE, The journal of transportation engineering.

62. RITCHIE, S. G.; (1987); **"Expert systems in pavement management."**; Transpostation Research-A, vol 21A, No2.

63. SANSONNET, J. P.; (1985); **"Les machines de l'intelligence Artificielle."**; La recherche, No 170, vol. 16.

64. SINHA, K. C.; COHN, L. F.; HENDRICKSON, C. T. et STEPHANEDES Y.; (1988); **"Role of advanced technologies in transportation engineering."**; ASCE; The journal transportation enginnering.

65. SHIENG-I TUNG, R. et SCHNEIDER, J. B.; (1987); **"Designing optimal transportation networks: an expert systems approach."** TRR #1145, TRB.

66. SWAINE, M.; (1986); **"TURBO-PROLOG: The langage."** Dr. Dobb's Journal; septembre 1986.

67. TEKNOWLWDGE; (1985); **"M.1 Reference Manual."; Teknowledge; Palo Alto; Californie.**

68. TOWNSEND, C.; (1986); **"Mastering expert systems with TURBO-PROLOG."**; 1ière édition, Howard W. Sams & Company.

69. WALKER, A.; McCORD, M.; SOWA, J. F. et WILSON, W. G. (1987); **"Kowledge systems and PROLOG."**; Addison-Wesley Publishing Company Inc.

70. WALTZ, D.; (1982); **"L'intelligence artificielle."** Pour la science; Décembre 1982.

71. WONG, W. G.; (1986); **"PROLOG, a langage for artificial intelligence."**; Pc magazine, 14 Octobre 1986.

72. YEH, C.; RITCHIE, S. G. et SCHNEIDER, J. B.; (1987) **"Potential applications of knowledge-based expert systems in transporta-tion planning and engineering."**; TRR 1076, TRB.

73. ZOZAYA-GOROSTIZA, C. et HENDRICKSON, C.; (1987); **"Expert system for traffic signal setting assistance."**; ASCE, The journal of transportation engineering.

LISTE DES ANNEXES:

ANNEXE: A.

INTELLIGENCE ARTIFICIELLE: PROMESSES ET REALITES

I. INTRODUCTION:

Depuis les années 50 les chercheurs ont commencé à parler d'un nouveau concept: l'intelligence artificielle (I.A.). Les recherches visaient alors à donner à l'ordinateur des fonctions plus évoluées. On voulait qu'il soit capable de suggérer des plans d'action, de prendre des décisions, de justifier un choix etc...

De telles actions exigent deux facultés de base: la connaissance et la compréhension, qui toutes deux permettent alors de raisonner. Ayant de telles facultés, l'ordinateur pourra alors être considéré intelligent.

Différentes définitions ont été données à l'intelligence Voici quelques critères qui rentrent dans sa composition:

- Capacité d'abstraction ou de généralisation.
- Capacité de faire des analogies entre différentes situations.
- Faculté de s'adapter à de nouvelles situations.
- Faculté de corriger ses erreurs afin d'améliorer ses performances futures, etc... [A.Bonnet, 1984].

En dépit des quelques trente à quarante années d'existence,

l'I.A. reste aujourd'hui un thème abordé avec un peu d'hésitation. Pour plusieurs c'est encore un champ d'aventure! On parle encore en terme de mythes versus réalités, promesses versus réalisations etc...

Toutefois, depuis les années 80, l'I.A. connaît une popularité croissante. Mais son public reste encore petit. [A.Barr, E.A.Feigenbaum,86].

II. ORIGINES DE L'I.A.:

Les deux forces de l'environnement intellectuel qui sont à la base de l'évolution de l'I.A. vers 1940 sont:

- La logique intellectuelle.

- Les nouvelles idées sur l'informatique, [A.Barr, 1986] La logique mathématique a continué d'être une partie active de la recherche sur l' I.A., en partie parce que des sytèmes de déduction logique ont été appliqués avec succès sur des ordinateurs. Mais même avant qu'il n y ' eût d'ordinateurs, la formalisation mathématique du raisonnement logique a orienté la conception vers la relation entre le calcul et l'intelligence. Plusieurs courants intellectuels ont participé à la fondation des principes de base de l'I.A.:

La complexité croissante des ordinateurs (idées sur les mémoires et les processeurs, sur les systèmes et les contrôles, les niveaux de langage et de programmes) a été l'attribut qui a réellement permit l'émergence d'une nouvelle science: l'I.A..

III. DOMAINES DE L'I.A.:

Les systèmes d'I.A. expérimentaux ont déjà généré intérêt et enthousiasme dans l'industrie et sont développés commercialement. Ces systèmes comprennent des programmes qui:

1- Résolvent des problèmes difficiles en chimie, biologie géologie, ingénierie et médecine au niveau des spécialistes.

2- Manipulent des systèmes robotisés pour réaliser des tâches utiles, répétitives avec des moteurs et des systèmes sensitifs.

3- Répondent à des questions posées de façon simple en anglais en français, en japonais, en arabe, ou en toute autre langue naturelle [A.Barr, 1986].

Ainsi il est facile de dégager trois grands domaines de l'I.A.:

1. Les langues naturelles.
2. La robotique.
3. Les systèmes experts (S.E.).

3.1. Les langues naturelles:

Les premiers travaux sur la compréhension du langage naturel par des programmes d'ordinateur débutèrent vers 1950 soit au début de l'histoire de l'I.A. [D. Waltz, 1982].

On distingue dans ce domaine trois secteurs principaux:

. La traduction.

.. La compréhension d'un document.

... Le contrôle des systèmes.

3.2. La robotique:

Un robot intelligent peut être défini comme: un système "fléxible" qui peut jumeler perception et action. Dans ce sens, les humains sont des robots intelligents, ils peuvent voir et sentir les forces qui les entourent et générer des actions en conséquence. La plupart des robots industriels actuels sont "stupides", ils sont destinés à des tâches répétitives et hardues. Ils n'ont pas de facultés de perception comme la vue, l'ouie, le toucher etc...

Les recherches dans ce domaine ont pour but de munir les robots de ces facultés de perception dans l'espoir de les rendre plus intelligents. On distingue trois grands secteurs de recherche:

. La vision des robots.

.. Les senseurs chez les robots.

... La mobilité des robots.

3.3. Les systèmes experts (S.E.):

Ce domaine faisant l'objet du prochain annexe, nous n'en ferons ici qu'une brève introduction. Certains auteurs identifient ce domaine

comme l'ingénierie de la connaissance. Elle consiste à utiliser la connaissance de l'homme (information) et son intelligence pour résoudre des problèmes complexes. Cette science vise à trouver à de tels problèmes des solutions performantes, c.a.d. qui se rapprochent des solutions d'un expert en la matière ou qui les dépassent.

IV. MACHINES ET LANGAGES DE L'I.A.:

Pourquoi parler de machines et de langages spécifiques pour l'I.A. alors qu'on dispose d'ordinateurs et de langages très puissants?

Il est vrai que les ordinateurs conventionnels sont puissants pour toute sorte de traîtement numérique, cependant ils ont deux handicaps majeurs pour servir de machines d'I.A.:

1- ils ne traîtent pas efficacement les symboles.

2- leur mémoire vive directe (RAM) est limitée. [J.P. Sansonnet, 1985].

La question d'une machine adéquate pour l'I.A. se pose donc de façon sérieuse. Deux alternatives s'offrent:

1. Créer de nouvelles machines conçues en fonction des besoins de l'I.A.

2. Doter les ordinateurs conventionnels des atouts et moyens requis pour manipuler efficacement des symboles.

Les deux alternatives ont été explorées et ont aboutit à des résultats différents.

4.1 Le langage LISP:

LISP tire son nom de l'expression "list programming". Ce langage a été inventé en 1959 par John McCarthy au MIT introduisant ainsi le traîtement symbolique de l'information représentée par des listes. Les multiples variantes de ce langage sont encore les outils les plus utilisés en I.A. et surtout en Amérique du nord [J.P. Sansonnet 1985]. Les ordinateurs conventionnels n'étant pas capables d'exécuter efficacement les programmes d'I.A., on a commencé à concevoir et à réaliser des machines spéciales vers les années 70 [J.P. Sansonnet, 1985]. C'est ainsi que les recherches au MIT ont aboutit à une première machine appelée CADR. Cette machine est la base de celle présentée par SYMBOLIC au début des années 80. Texas Instruments et Xerox ont ensuite fabriqué leurs propres machines Lisp.

Ces machines constituent certes un avantage pour le langage Lisp et pour l'I.A., mais leur coût très élevé fait que leur utilisation est encore limitée à la recherche. C'est pourquoi les vendeurs commencent à concevoir des machines moins coûteuses pour le marché commercial [D.B. Davis, 1986].

4.2. Le langage PROLOG:

PROLOG tire son nom de l'expression PROgrammation en LOGique. Il est le fruit de recherches dirigées par Alain Colmérauer à Marseille (France) dans les années 70 [Patrice Bihan, 1987].

Ce langage se distingue des langages habituels comme le basic et le pascal par sa nature déclarative et non impérative Il suffit en prolog d'indiquer au système les données du problème à résoudre et c'est au moteur d'inférence, qui permet de mettre en relation ces données, de rendre toutes les solutions. Le programmeur n'a pas à imaginer toutes les solutions possibles, et ceci grâce à la capacité de déduction dont dispose prolog.

Trois éléments font la force de prolog:

. Le principe d'unification ("pattern matching": mise en correspondance, appariement...).

.. Le retour arrière ("backtracking").

... La base de données ("data base").

Plusieurs versions de prolog sur PC existent actuellement: Prolog-86 plus, Micro-Prolog, Turbo-Prolog, A.D.A Prolog, MProlog Arity/Prolog, et PrologII.

Contrairement à Lisp, il n'y a pas encore de machines dédiées à Prolog. IL faudrait attendre les résultats de la cinquième génération d'ordinateurs annoncée par les Japonais.

En 1982, ces derniers ont annoncé leur projet étalé sur dix ans. Dirigé par le " Institute for new generation computer technology (ICOT) ", il vise à développer des logiciels d'I.A. ainsi que leur machines relatives. Parmi ses objectifs la création de bases de connaissances qui acceptent jusqu'à cent (100) billions bytes d'information et 20,000 règles de S.E. [D.B. Davis, 1986]. Alors que l'ICOT a adopté le langage Prolog dans ses recherches, d'autres

organismes qui travaillent sur le même projet utilisent Lisp. C'est le cas notemment du N.E.C. (manufacturier d'ordinateurs). Les deux grands axes de recherches sur lesquels les Japonais travaillent actuellement sont:

1. L'amélioration de la vitesse d'opération de leurs logiciels (Software) d'I.A.

2. La création de machines pour le traîtement et le stockage d'immenses bases de faits.

Dans les deux cas ils adoptent l'architecture parallèle pour leurs machines. Laquelle architecture utilise plusieurs processeurs simultanément pour résoudre un problème. « Il a été démontré que l'architecture parallèle est très bonne pour des langages de programmation logique comme prolog », déclare M. Yoshihisa Ogawa directeur de la section de la recherche de l'I.C.O.T. [D.B. Davis 1986].

Jusqu'à récemment peu de produits de ce projet ont été présentés. Par exemple, la compagnie Mitsubishi Electric a présenté une machine à inférence séquentielle qui ressemble, dans ces concepts, à la machine Lisp standard. Cette machine a été créée en collaboration avec l'ICOT pour executer Prolog. Il faudra donc attendre 1992 pour voir les résultats de ce fameux projet.

4.3. Concurrence des micro-ordinateurs et des langages conventionnels:

Bien que peu populaire, ce discours n'est aujourd'hui pas encore terminé. En effet, vu le coût très élevé des machines d'I.A. existantes, plusieurs préfèrent utiliser les ordinateurs et micro-ordinateurs conventionnels malgré l'exécution inéfficace, parce que lente, des programmes d'I.A. par les langages conventionnels sur ces machines. Certains vont même jusqu'à proposer d'écrire des programmes en Lisp et de les porter au langage C pour exécution. C'est le cas de L.K. Geisel qui croît que ce serait la solution optimale pour les programmes d'I.A. [D.B. Davis, 1986].

Le compromis qu'il faut satisfaire met donc le coût de la machine face à l'efficacité de l'exécution des programmes d'I.A.

V. LA RECHERCHE EN I.A.:

Cet apperçu général du domaine de l'I.A. montre que cette science jeune, connaît aujourd'hui une évolution très remarquable. Nous devons nous attendre dans les prochaines années à une révolution avec la cinquième génération d'ordinateurs. L'évolution en recherche touchera les sujets suivants:

5.1. La résolution de problèmes:

Résoudre des " puzzles ", jouer aux échecs, au dames etc... font intervenir les techniques fondamentales d'I.A. que sont la recherche et la réduction de problèmes. Aujourd'hui quelques programmes peuvent même améliorer leur performance avec l'expérience. Les questions posées ici dans ce secteur incluent la capacité qu'ont les joueurs humains mais qu'ils ne peuvent expliquer, comme celle d'un maître de voir la configuration d'un échiquier en termes de schémas significatifs. Une autre question de base comprend la conceptualisation originale d'un problème, appelée en I.A. le choix de la représentation du problème.

5.2. Le raisonnement logique:

Après les travaux sur la déduction logique en liaison étroite avec la résolution de problèmes, et la démonstration de théorèmes, parmi les sujets d'intérêt particulier on veut:

. Trouver un moyen de focaliser sur les faits importants dans une base de données importante.

. Garder la trace des justifications d'hypothèses.

. Les mettre à jour quand une nouvelle information arrive.

5.3. Le langage:

Ici on travaille à améliorer les systèmes pour:

. Répondre aux questions posées dans une langue naturelle à partir d'une banque de données interne.

. Traduire un texte d'une langue à une autre.

. Suivre des instructions données en langage naturel.

. Acquérir la connaissance par la lecture de textes et construire une base de données interne.

5.4. La programmation:

Il s'agit de la programmation automatique où des systèmes doivent écrire des programmes d'ordinateur à partir d'une variété de descriptions de leurs besoins. Exemples:

1. Un système semi-automatisé de développement de logiciels.

2. Des programmes d'I.A. qui apprennent en modifiant leur propre programmation (modifient leur comportement lors de l'exécution).

5.5. L'apprentissage:

Certainement l'un des aspects les plus significatifs de l'intelligence humaine est la capacité d'apprentissage . C'est un bon exemple de comportement cognitif qui a été si mal compris que peu de progrès a été fait pour sa réalisation en I.A. Il y eut des essais

incluant des programmes qui apprennent à partir d'exemples, de leur propre réalisation, et de ce qu'on leur dit. Mais en général l'apprentissage reste encore dans un état primitif.

5.6. L'expertise et les S.E.:

L'expertise fait intervenir trois parties:

- l'expert humain.
- l'utilisateur humain.
- le S.E.

La figure A.1 illustre la relation entre ces différentes parties.

Le processus d'acquisition de la connaissance est le goulot d'étranglement dans la construction des S.E..

Une attention particulière est aussi accordée à la capacité du S.E. d'expliquer et de justifier son raisonnement.

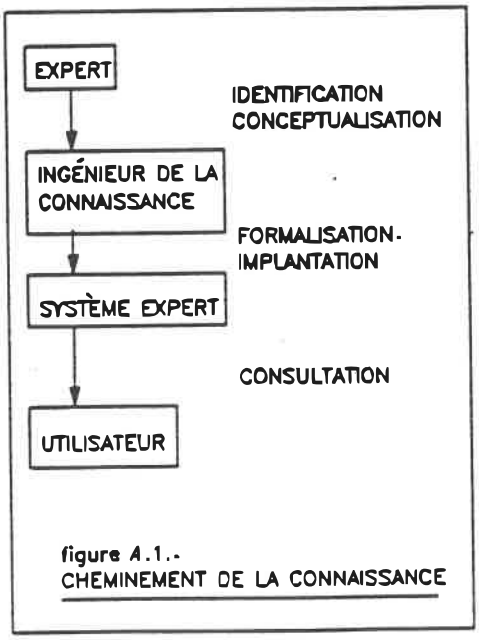


figure A.1.-
CHEMINEMENT DE LA CONNAISSANCE

5.7. La robotique et la vision:

La plupart des robots industriels actuels sont " aveugles " et peu d'entre eux " voient " grâce à une caméra. L'évolution de la robotique est passée de l'optimisation du mouvement d'un bras, aux programmes de commandes pour réaliser des tâches répétitives, et aujourd'hui à l'utilisation de l'information visuelle pour la reconnaissance des objets, des ombres et des changements dans une image. L'utilisation de l'information visuelle est déjà en vigueur pour la reconnaissance aérienne par exemple.

5.8. Les systèmes et les langages d'I.A.:

Il s'agit d'un sujet d'importance capitale pour cette jeune science qu'est l'I.A. et pour laquelle on considère encore que la technologie doit donner beaucoup plus pour seconder la théorie. Parmi les idées informatiques actuellement investiguées on trouve:

- le temps partagé.
- le traîtement de listes.
- le débogage interactif etc...

VI. CONCLUSION:

Les chercheurs à plein temps en I.A. ne sont que quelques centaines au monde, mais leurs résultats concernent tous et chacun. Il faut bien comprendre que cette recherche est lente et difficile car loin de chercher des recettes miracles (des algorithmes), les chercheurs s'éfforcent de reconstituer l'expérience et la connaissance des spécialistes dans tous les domaines et de les mettre en machine. Cette information est en général difficilement accessible, car il faut interroger l'expert et retrouver ce qu'il a mis inconsciemment des années à apprendre. La tâche n'est pas impossible avec les outils et les méthodes déjà acquis. Bien que difficile, elle est cependant passionnante puisqu'elle nous permet d'apprendre beaucoup sur l'homme lui-même et son intelligence [J.L. Laurière, 1987].

ANNEXE B

LES SYSTEMES EXPERTS

I. INTRODUCTION:

Issus du concept et du développement de l'I.A., les S.E. représentent des approches vers des outils informatiques permettant l'emmagasinage et le traitement efficaces des connaissances. Bien que les possibilités offertes par les S.E. soient actuellement limitées la recherche dans les domaines de leur application est encouragée dans divers secteurs dont celui des Transports.

Plusieurs définitions, toutes semblables, ont été données aux S.E.. Nous avons retenu la suivante: « Un S.E. est un programme d'ordinateur dont le but est de répondre, dans des domaines bien délimités de connaissances, à des problèmes généralement considérés comme complexes et difficiles, et dont la performance se compare avec celle d'un expert du domaine ou la dépasse [Pehlivanidis, 1987].

Historique des S.E.:

La révolution des S.E. a connu trois phases principales: une thèse, une anti-thèse, et une synthèse [Walker, 1987].

La thèse (1950s) déclare qu'une méthode générale de résolution experte de problèmes peut être trouvée, et que cette méthode peut être informatisée et appliquée à différents problèmes. L'idée est que

en déclarant la tâche uniquement, le S.E. soit capable de l'accomplir. Cette thèse à connu des efforts inefficients sur le plan pratique. Ces difficultés ont conduit E. Feigenbaum de l'université de Stanford à énoncer l'anti-thèse.

L'anti-thèse déclare qu'au lieu de voir à la généralité, on devrait procéder empiriquement pour rassembler la connaissance humaine et les procédures pour des tâches spécifiques. Cette anti-thèse appelle essentiellement à écrire un nouveau programme pour chaque nouveau problème ou domaine selon le cas. Cette technique a conduit à d'importantes réalisations sur le plan pratique. Par exemple, le S.E. DENDRAL a vu le jour en 1958.

Cette anti-thèse a donc eu du succès et a même alimenté l'intérêt commercial. Cependant, quand on examine l'effort requis chaque fois qu'il est question de l'acquisition des connaissances, cette approche paraît très lourde. C'est alors qu'entre la thèse appelant à la généralité absolue,et l'anti-thèse soutenant la spécificité de chaque S.E. pour un rôle unique, a surgit la synthèse (1970s) qui est une sorte de compromis entre les deux.

En effet la synthèse repose sur l'idée que plusieurs tâches ou problèmes ont des exigences communes et que ces exigences peuvent être rassemblées dans un shell de S.E. (ou S.E. squelettique, ou kit de S.E.). Alors lors de la construction du S.E. on n'aura qu'à rentrer les connaissances et ajouter des règles complémentaires pour que le système soit complet. Comme exemples de shells de S.E. on peut siter EMYCIN (1984) et OPS5 (1981).

Exemples de S.E.:

Table 3.: Description de quelques S.E.

S.E.	Description.
DENDRAL	1958, Université de Stanford. Donne la formule développée d'un corps organique à partir de sa formule brute et de son spectrogramme de masse.
META-DENDRAL	1968. Infère automatiquement des règles de fragmentation de molécules qui servent à DENDRAL.
MYCIN	1974, Université de Stanford. Aide aux médecins dans le diagnostic et le traitement des infections bactérienne du sang.
TEIRESIAS	1976. Assiste l'expert dans l'élaboration de systèmes utilisant une grande base de connaissances (jumelé à MYCIN).
EMYCIN	1979. Shell qui sert à l'acquisition des règles pour un S.E. quelconque. Il a été utilisé avec MYCIN.
NEOMYCIN	1981. Découpe les règles à fin d'isoler la hiérarchie des entités chimiques et d'exprimer par ailleurs leurs liens.
BAOBAB	1980. Constitution automatique de la base des faits relatifs à un patient à partir de son dossier médical.
RITA	1976. modèle d'aide à la conception.
PROSPECTOR	1977, SRI. Recherches minières.

SACON	1980. ingénieur conseil en mécanique. On y a utilisé EMYCIN.
DRILLING ADVISOR	1982. Analyse des accidents de forages pétroliers.
SOPHIE	1979. Aide à l'enseignement par ordinateur de la détection des pannes dans les circuits électriques.
NUDGE	1977. Elaboration d'emplois de temps.
HEARSAY	1981. Compréhension de la parole continue.
CESSOL	1984. Choix des essais géotechniques pour l'étude des caractéristiques d'un terrain.
DART	1981 (IBM). Consultant pour pannes de matériels et de systèmes.
APE	S.E. pour la programmation automatique (écrit des programmes en Lisp).
CAMELIA	1984. Aide à la construction de théorèmes en calcul intégral.
PONTIUS-O	Simule le comportement d'un homme apprenant à pilôter un avion.
MAXYMA	1968 (MIT). Résolution de problèmes mathématiques.

(Table 3. suite)

XCON	1979. Déscendant de R1. Configuration des micro-ordinateurs VAX pour les consommateurs.
DELTA	S.E. pour la maintenance des véhicules
DIRECTOR	S.E. éducationnel de modèles de simulation pour les décisions en T.U.
HERCULES	S.E. pour les stratégies de recouvrement du trafic après les situations de crise (saturation etc...).

(Table 3. suite) Réf.[Yeh,87],[Laurière,87],[Townsend,86]

Une liste plus longue des S.E. existants peut être consultée par les lecteurs intéressés [Walker, 1987]. Au chapitre suivant nous siterons d'autres applications de S.E. relativement au domaine des transports.

II. POURQUOI LES S.E.?:

En général, lors d'un premier contact avec une nouvelle technologie, l'ingénieur se pose une question classique: pourquoi adopter cette technique? Il est donc tout à fait normal d'avoir cette même attitude face à la technique des S.E..

Parmi les raisons invoquées par les différents chercheurs pour justifier ce choix de technologie on retrouve les suivantes [Boubkeur,1986], [Boose, 1986], [Desrochers, 1984]:

1. Les experts partent un jour en retraite et leurs connaissances disparaissent avec eux.

2. Il y a des façons plus intéressantes d'occuper un expert que de le laisser répondre aux questions redondantes des utilisateurs du système en question.

3. L'expertise peut être rare ou absente en temps et lieux voulus.

4. L'expertise peut être chère à reproduire (formation) et à exploiter.

5. Un S.E. peut assurer qu'un produit ou service soit délivré dans les meilleurs délais, alors que l'expert humain peut tomber malade ou s'absenter pour une raison ou une autre, entrainant un retard par rapport au délai établi.

6. Les experts ne sont pas nécessairement toujours consistents.

7. Le problème à résoudre peut être difficile et non résolvable par les procédures algorithmiques disponibles.

8. Un S.E. peut accumuler le savoir de plusieurs experts dans sa base de connaissances.

III. DIFFERENTS TYPES DE S.E.:

L'expertise humaine est généralement classifiée selon des domaines de spécialisation. C'est ainsi qu'un expert de design ne l'est pas forcement en diagnostic par exemple.

Un S.E. touche, en général, un domaine bien précis de cette expertise. Pour celà, on les classe dans diverses catégories, soient:

1. S.E. de diagnostic.
2. S.E. de prévision.
3. S.E. de consultation.
4. S.E. de contrôle et de surveillance.
5. S.E. de planification.
6. S.E. d'instruction.
7. S.E. d'interprétation.

IV. LES S.E. ET LES PROGRAMMES CONVENTIONNELS D'ORDINATEUR:

L'arrivée des S.E. avec leur procédure déclarative de programmation a soulevé un débat inévitable concernant la controverse déclaratif/procédural. Plusieurs comparaisons ont été faites pour montrer les avantages des techniques de S.E. comparées aux procédures impératives de programmation habituelles. Avant d'examiner ces comparaisons, définissons tout d'abord ce qui est une approche procédurale et ce qui est une approche déclarative.

Dans l'approche procédurale impérative il faut prévoir toutes les questions à priori et écrire une procédure pour chacune d'entre elles. Elle traduit syntaxiquement comment transite la connaissance.

Dans l'approche déclarative on exprime des connaissances factuelles de manière purement énonciative indépendante de leur

utilisation, et des données en vrac de façon totalement modulaire Elle traduit une relation, une information sémantique et répond à la question: "quoi?" [Laurière, 1987].

La différence fondamentale entre un programme usuel et un S.E. à règles n'est pas le caractère plutôt "procédural" du premier et le caractère plus ou moins "déclaratif " du second, mais bien le fait que, dans le second, la structure de contrôle soit entièrement séparée des connaissances entrées sous forme de règles et donc qu'elles puissent être données en vrac. C'est la modularité absolue qui fait l'intérêt des S.E. . Les unités de connaissances sont indépendantes et la communication entre elles est assurée par la seule base de faits. L'ordre, aux deux sens du mot (instruction et séquence précise), est supprimé et on a la possibilité du retour en arrière sur une même règle.

En résumé, la différenciation entre S.E. et P.P. (programmes procéduraux) se situe au niveau des trois points suivants:

- les connaissances.
- le contrôle.
- le raisonnement.

Cette différenciation se résume ainsi (table 4.):

Table 4.

Comparaison entre S.E. et P.P.

Composante	P.P.	S.E.
Connaissances.	Codées. (instructions)	données: règles & faits.
Contrôle.	Codés. (Instructions)	Donné sous forme de méta-règles.
Raisonnement.	Boîtes noires.	Accessible et explicite. boîtes en verre.

V. ARCHITECTURE D'UN S.E.:

Un S.E. comprend essentiellement trois composantes:

1. Le moteur d'inférence.

2. La base de connaissances.

3. Le langage d'expression.

La base de connaissances contient deux éléments de base qui sont les faits (connaissances assertionnelles) généralement fournis par l'utilisateur pour définir le problème, et les règles (connaissances opératoires) fournies par l'expert du domaine concerné pour la résolution du problème [Farreny, 1985].

Le moteur d'inférence, ou machine déductive, accomplit essentiellement les deux rôles suivant:

1. Combiner et/ou chaîner des groupes de règles pour inférer ou dériver, des connaissances telles que les jugements les plans les preuves, les décisions, les prédictions, les nouvelles règles etc...

2. Souvent, rendre compte de la manière dont les nouvelles connaissances ont été inférées.

Le langage d'expression permet de faciliter l'expression la plus directe possible des règles par rapport à leur forme d'émergence chez les experts. Il doit faciliter la communication que ce soit avec l'utilisateur (consultation) ou avec l'expert (instruction).

L'anatomie d'un S.E. standard est illustrée à la figure B.1., et en général, la communication entre ses différentes composantes se présente comme illustrée à la figure B.2..

VI. ETAPES DE CONSTRUCTION D'UN S.E.:

La construction d'un S.E. nécessite normalement les étapes suivantes [Hayes-Roth,1985], [Parent,1985]:

Etape no: 1: *Identification des exigences.*

Cette étape correspond à une définition plus précise du problème à résoudre. On doit caractériser les aspects importants du domaine du problème, identifier ses caractéristiques, l'expertise additionnelle requise, les buts du S.E. et les ressources disponibles.

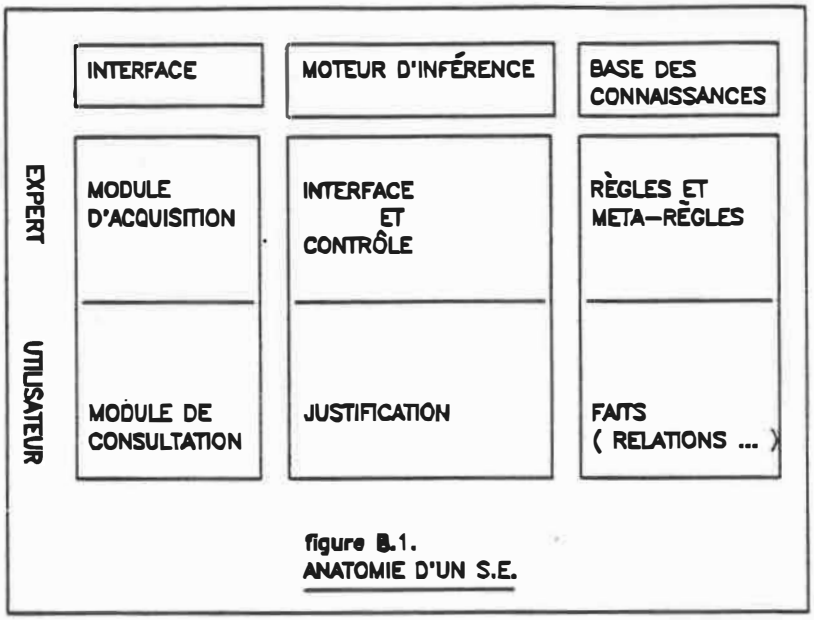


figure B.1.
ANATOMIE D'UN S.E.

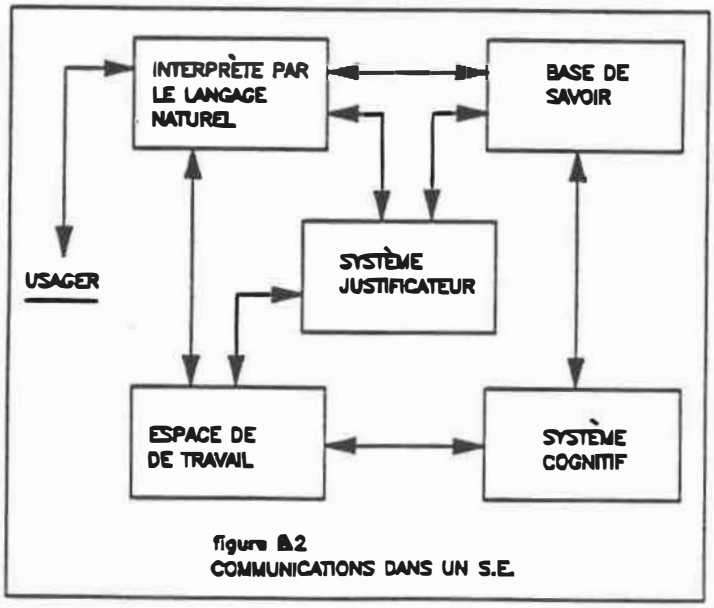


figure B.2
COMMUNICATIONS DANS UN S.E.

Etape no: 2: *Conceptualisation.*

Une fois que le problème est bien identifié, on doit décrire le schéma fonctionnel du processus de résolution. On détermine les concepts clés, les relations de la base de savoir, les types de données disponibles, les sous-tâches, les stratégies, les hypothèses de départ, les procédures, les contraintes etc...

Etape no: 3: *Formalisation.*

On établi une correspondance formalisée entre les éléments déterminés à l'étape précédente. L'ingénieur du savoir entame une conversation avec l'expert pour établir une structure et une organisation formalisée de ses connaissances. Pour chaque élément ou type de connaissances, il propose une technique de base d'I.A. ou un S.E. existant comme outils d'exploitation.

Etape no: 4: *Implantation.*

Cette étape concerne l'ingénieur du savoir qui va développer un programme complet fonctionnel à l'aide du savoir déjà formalisé. Ce programme comprendra les relations assertionnelles, les règles opératoires, la stratégie de contrôle à l'intérieur du programme (méta-règles), et éventuellement un module de justification.

Etape no: 5: *Expérimentation.*

Pour évaluer le programme, on utilise des exemples variés avec différents degrés de difficulté. On détecte les faiblesses ou erreurs qui concernent la base de savoir ou la structure d'inférence. Parmi les problèmes qu'on peut rencontrer il y a: les problèmes dans l'entrée des données, la forme des solutions, les règles incorrectes incomplètes, incohérentes, ou manquantes, et la qualité de la structure de contrôle. On apporte alors les changements nécessaires et on recommence l'évaluation jusqu'à ce que le résultat soit satisfaisant.

Cependant, ce processus peut être simplifié en ayant recours à un shell de S.E.. Cette approche nous permet de gagner du temps aux étapes 2 et 4 décrites ci-dessus. Toutefois, il faut que le shell soit conçu pour des applications identiques à celle qu'on veut résoudre.

Pour construire un S.E. performant, il faut que les conditions suivantes soient satisfaites:

1. Au moins un expert du domaine doit participer à sa construction.

2. La performance de l'expert doit être basée sur son savoir spécial, son jugement, et son expérience.

3. L'expert doit pouvoir expliquer son savoir et son expérience ainsi que les méthodes de leurs applications à des problèmes particuliers.

4. La tâche du S.E. doit avoir une application dans un domaine bien délimité.

Avec les moyens (langages d'I.A.) actuels, la construction d'un S.E. semble converger vers un effort équivalent à cinq (5) personne-années. Dans la majorité des cas, deux (2) à cinq (5) personnes participent à la construction [Gevarter,1983].

VII. LA REPRESENTATION DES CONNAISSANCES:

Pour résoudre un problème l'expert ne fait pas seulement appel à sa capacité de raisonnement, mais aussi, d' une manière plus ou moins consciente, à une foule de connaissances [Gallaire 1985].

La question de la représentation des connaissances est au coeur de l'I.A., «représenter quelque chose signifie d'abord savoir l'isoler, avoir saisi son importance et son intérêt pratique. Celà implique ensuite que ses propriétés aient été repérée et traduite sous une forme manipulable. En un mot, représenter c'est comprendre.» [Laurière,1987]. En effet un livre par exemple contient beaucoup d'information, sauf qu'il n'est pas possible de dire qu'il les connait ; parce que connaître implique la capacité de manipuler l'information pour répondre à des questions spécifiques. Il faut donc distinguer les données ou informations, des connaissances.

VII.2. Différentes techniques de représentation:

Plusieurs façons de représenter les connaissances sont utilisées par les spécialistes de S.E., leurs avantages reciproques sont plutôt techniques et la différence fondamentale concerne la facilité de modification des connaissances.

Voici une description rapide des techniques les plus utilisées actuellement:

2.1. Les réseaux sémantiques:

Vers la fin des années 1960, des chercheurs dont M. QUILLIAN et S. SHAPIRO, ont amorcé des discussions qui fûrent à l'origine de l'élaboration de la théorie des réseaux sémantiques [Elgharbi, 1986].

Dans cet arrangement, le domaine est représenté par une collection de noeuds et de liens tel qu'illustré à la figure B.3.

Bien qu'il n'y ait pas de conventions pour nommer les noeuds et les liens, les deux règles suivantes sont en général valables [Townsend,1986]:

1. Les objets et les descriptions d'objets sont représentés par des noeuds.

2. Les relations entre deux objets ou entre un objet et ses descriptions sont représentées par des liens.

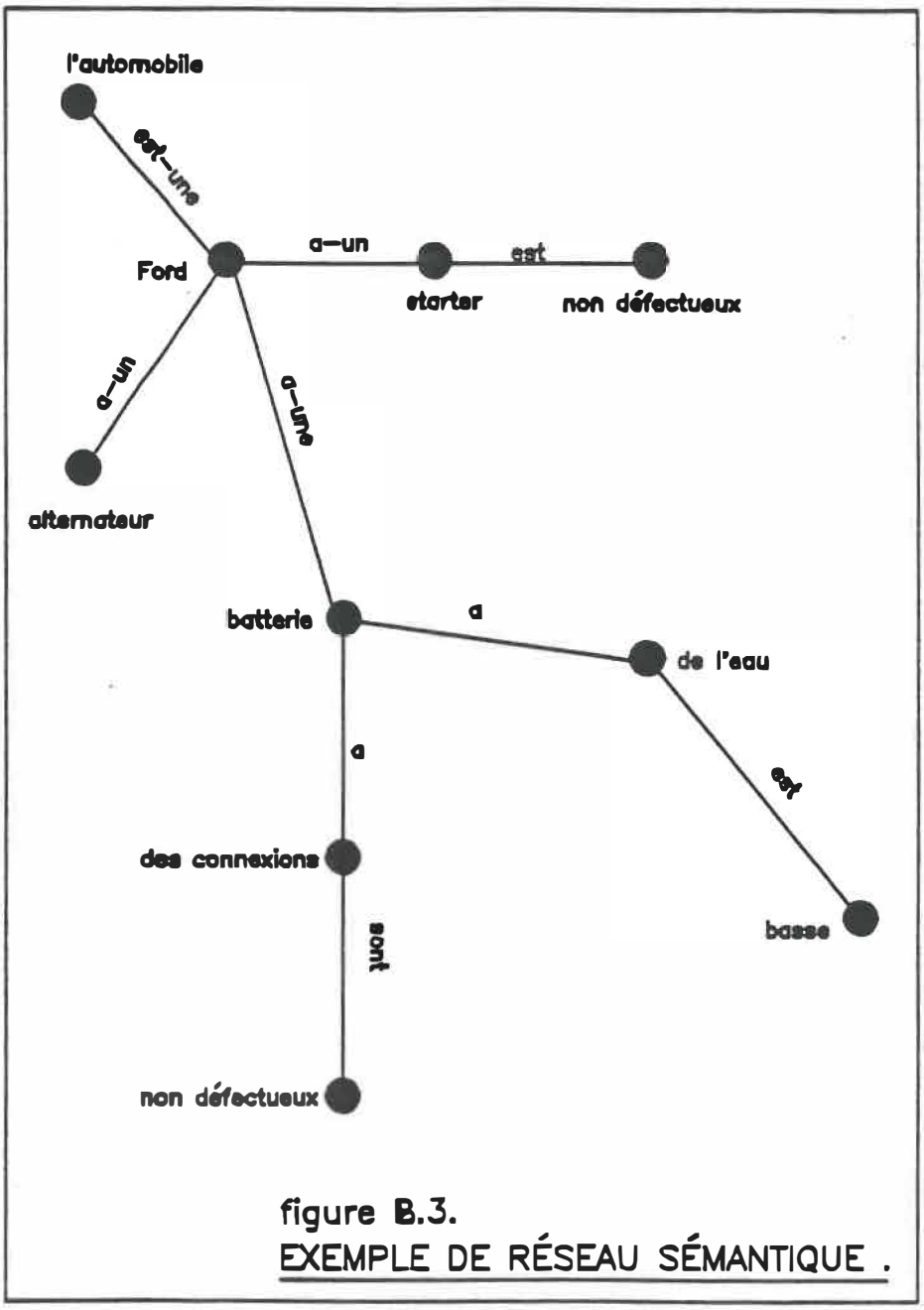


figure B.3.
EXEMPLE DE RÉSEAU SÉMANTIQUE .

En général, deux types de liens sont très utilisés. Ce sont le lien "A-UN(E)", et le lien "EST-UN(E)". "EST-UN(E)" exprime l'appartenance d'un objet à une classe d'objets plus générale ; alors que "A-UN(E)" exprime les propriétés ou attributs des objets. Evidemment, d'autres liens doivent être définis pour compléter le programme.

L'inférence dans les réseaux sémantiques est basée sur le principe de l'héritage de l'information des niveaux supérieurs de la hiérarchie jusqu'aux niveaux les plus bas. On exploite ainsi la diffusion de l'information à travers plusieurs noeuds du réseau pour économiser des déclarations répétitives inutiles.

Les réseaux sémantiques ont cependant certaines lacunes dès qu'il s'agit de représenter des relations complexes. Des situations de conflits peuvent se produire quand l'application devient assez grande [Elgharbi, 1986].

2.2. Les triplets O-A-V (Objet-Attribut-Valeur):

La connaissance des faits est représentée par une table de triplets objets, attributs et valeurs. Par exemple ont peut avoir les triplets suivants:

(autobus, capacité, 75).

(métro, vitesse moyenne, 50 km/h).

Les triplets O-A-V sont un cas particulier des réseaux sémantiques où la relation entre objet et attribut (O-A) correspond au

lien de propriété "A-UN(E)", et la relation entre attribut et valeur (A-V) correspond au lien de définition "EST" ou "EST-UN(E)". Les objets, attributs, et valeurs sont alors des noeuds du réseau.

Pour des applications simples, on peut utiliser des paires A-V à la place des triplets ; cependant, le principe d'héritage ne peut alors pas être représenté.

2.3. Les règles de production:

Les règles de production ont été proposées par A.Newell pour modéliser le comportement humain. Elles ont la forme "Si... Alors" et signifient que si la situation A a lieu alors exécuter l'action B (Si A alors B). Elles sont aussi une façon de représenter les relations entre les triplets O-A-V ou les paires A-V ; en voici un evemple:

Si le lien a-b est saturé
ou le lien a-b est interdit
ou la limite de vitesse est très basse
Alors passer par un autre lien.

Chaque règle a une conclusion et un antécédent. Si toutes les conditions de l'antécédent (les prémisses) sont vraies, alors la conclusion devient aussi vraie et si elle consiste en une action elle est déclanchée. La règle est donc un opérateur qui offre une technique pour faire passer le système d'un état à un autre.

Les règles reliant des faits sont appelées des règles de production, alors que celles définissant des heuristiques ou des stratégies sont appelées des méta-règles.

Les S.E. utilisant les règles comme mode de représentation des connaissances sont appelés systèmes de production.

Les règles permettent d'utiliser des variables pour représenter un objet de façon abstraite et offrent la possibilité de considérer la probabilité de certitude de leurs conclusions.

Cet arrangement des connaissances est considéré, jusqu'à présent, le plus naturel [Elgharbi, 1986].

2.4. Les cadres (ou prototypes):

Un cadre a pour unité de base le concept qui est formé d'une collection d'entités physiques ou conceptuelles d'une même classe. Sa représentation ressemble à un arbre renversé (voir figure B.4.).

La propriété d'héritage y est analogue aux réseaux sémantiques. Par exemple, la ligne 51 appartient à la S.T.C.U.M qui offre un service en autobus de T.C. comme mode de transport.

A l'intérieur d'un même cadre, les attributs et les rôles pour un concept sont représentés par l'information contenue dans des fentes ("slots"). Par exemple pour le cadre "modes de transport" de la figure précédente, on peut retrouver dans l'une des fentes la part respective de demande (en %) de chacun des modes: (T.C.,25%), (Auto, 50 %) (Taxi, 0.5 %) etc...

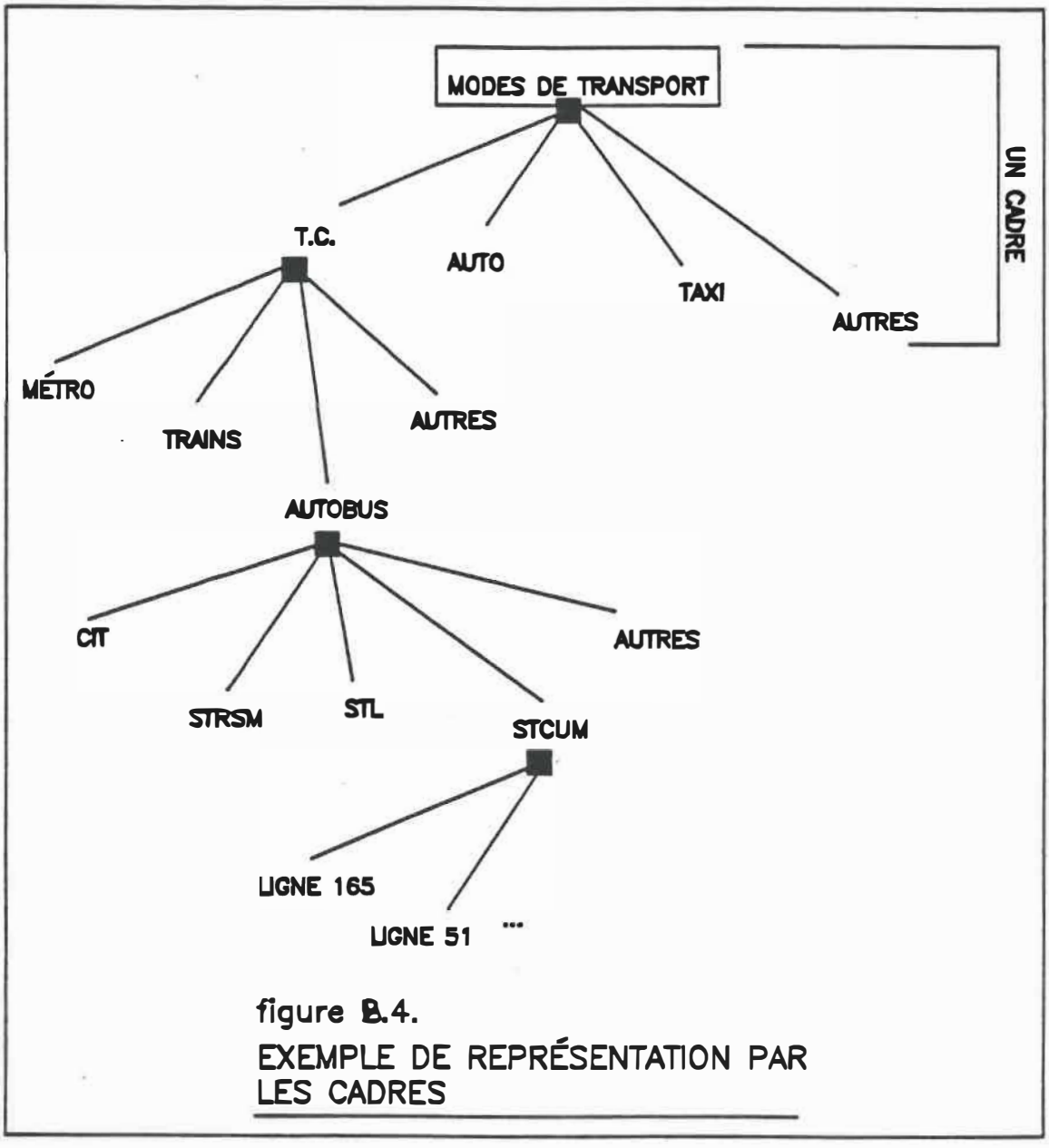


figure B.4.
EXEMPLE DE REPRÉSENTATION PAR LES CADRES

Les cadres permettent de bien visualiser les relations et la hiérarchie du domaine, sauf que, d'une part ils sont difficiles à traiter pour des grandes applications, et d'autre part ils ne permettent pas d'exprimer des exceptions dans un héritage particulier [Townsend, 1986]. Il sont utilisés surtout pour les problèmes de compréhension du langage [Laurière, 1987].

Parmi ces quatre formes de représentation, les règles sont les plus décalaratives et demeurent pour celà les plus populaires pour les S.E..

L'ingénieur des connaissances utilisera la technique de représentation qui convient à son application. L'expert fait une abstraction du monde réel et l'ingénieur modélise cette abstraction à l'aide de la technique choisie. Durant cet exercice les niveaux d'abstraction se présentent comme suit:

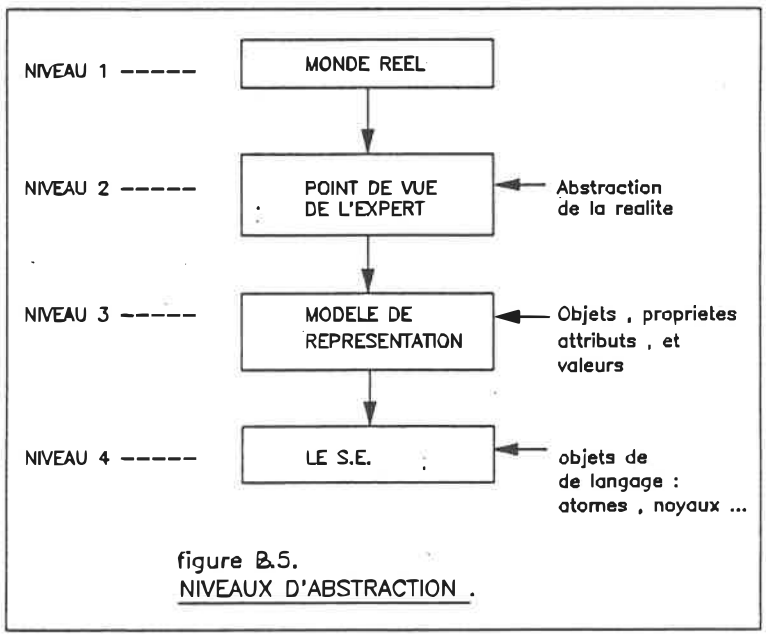


figure B.5.
NIVEAUX D'ABSTRACTION .

En plus des modèles de représentation des connaissances d'autres modèles moins populaires existent. C'est le cas par exemple du modèle de la logique de premier ordre conçu au départ pour les preuves mathématiques. Ce modèle est critiqué quand il s'agit de traiter des cas d'incertitude. Les recherches actuelles portent sur la logique non monotonique et la logique floue pour remédier à cette faiblesse.

VIII. L'UTILISATION DES CONNAISSANCES:

Soit l'implication P ===> Q qui serait par exemple l'abstraction de l'affirmation "Tout homme est mortel". Le modus ponens permet de faire la déduction suivante:

Tout homme est mortel.

Socrates est un homme.

alors Socrates est mortel.

Une première utilisation de l'implication P ===> Q consiste à appliquer le modus ponens pour les deux cas suivants :

1. Une antité est mortelle dès que je peux prouver que c'est un homme.

2. Si je veux prouver que quelque chose est mortelle, un sous-but possible est de montrer que c'est un homme.

Une deuxième utilisation consiste à utiliser le modus tollens (non Q et P ===> Q alors non P) de la même implication pour les deux cas suivant:

1. Si une antité est non mortelle, elle ne peut être un homme (utilisation directe).

2. Si je désire prouver que quelque chose n'est pas un homme, je doit prouver qu'elle n'est pas mortelle (utilisation de la méta-connaissance).

Pour évaluer la prémisse P ou la conclusion Q d'une règle une stratégie de recherche doit être utilisée.

VIII.1. Les stratégies de recherches:

Quatre (4) stratégies fondamentales définissent la procédure de recherche:

1. Le chaînage avant.

2. Le chaînage arrière.

3. Le raisonnement par heuristiques.

4. L'arbre de décision.

Ces stratégies sont appliquées à un espace de recherche formé par l'ensemble des buts et des sous-buts du S.E.. Une stratégie de recherche s'impose dans un S.E. car l'espace de recherche est en général très grand. Le nombre d'essais à faire est considérable d'où la nécessité de hiérarchiser cet espace.

1.1. Le chaînage avant:

Ici les prémisses d'une règle sont comparées aux éléments de la base de connaissances pour vérifier leur véracité. Si elles sont vraies, la conclusion le devient aussi et devient un nouveau élément de la base de connaissances qu'on peut utiliser pour les vérifications subséquentes.

Cette stratégie est plus appropriée quand le nombre de buts est élevé et que les informations à entrer ne sont pas très nombreuses.

Des S.E. utilisant la stratégie de chaînage avant sont dits orienté-données (data-driven).

1.2. Le chaînage arrière:

La recherche part du but de la règle et tente de le prouver en satisfaisant les prémisses à partir de la base des connaissances.

Cette stratégie est très valable quand le nombre de buts finaux est limité et que la base de connaissances est très considérable.

Des S.E. utilisant la stratégie de chaînage arrière sont dits orienté-buts (goal-oriented).

Pour ces deux stratégies la recherche peut se faire en profondeur ou en largeur.

En profondeur, un but est analysé jusqu'à ce qu'il soit prouvé faut. On recommence alors avec un autre but du même niveau et ainsi de suite jusqu'à ce qu'un but soit prouvé vrai. Cette stratégie est plus

proche du raisonnement de l'homme, donc plus naturelle, et c'est pourquoi elle est souvent préférée à la recherche en largeur [Townsend, 1986].

En largeur, tous les but d'un même niveau sont considérés en même temps avant de passer au niveau suivant. Cette stratégie est efficace si le nombre de noeuds par niveau est faible. Notons cependant qu'en réalité le nombre de noeuds par niveau augmente de façon imporatante à mesure qu'on descend en profondeur.

Chacune de ces stratégies peut être avantageuse dépendamment de l'application considérée. Voici une comparaison qui pourrait aider à faire un choix entre les deux:

Table 5

Comparaison entre le chaînage avant et arrière.

Chaînage arrière.	Chaînage avant.
orienté-buts.	orienté-données.
commence à partir de solutions possibles et travaille en reculant pour voir si l'une est correcte.	Commence à partir des données et travaille en avançant pour voir si une solution est consistente avec les données.
Commence en posant une question à propos de l'état d'un but.	Commence à partir d'une situation et essaie de répondre.
peut expliquer son raisonnement.	n'explique pas son raisonnement.

table 5. (suite).

Raisonnement du haut vers le bas.	raisonnement du bas vers le haut.
réduit l'espace de recherche en minimisant et en structurant les questions.	réduit l'espace de recherche en trouvant des contraintes pour limiter la recherche.

réf: [Townsend,1986].

1.3. L'arbre de décision:

L'arbre de décision impose une structure de recherche au système en élaborant une topologie de l'espace de recherche. En imposant cette structure, elle entraine un raisonnement procédural dans le système.

1.4. La recherche par heuristiques:

Le système examine les connaissances disponibles et utilise ensuite ses propres règles internes pour déterminer si le chaînage avant ou le chaînage arrière est le plus approprié et l'ordre dans lequel la recherche se fera parmi les règles.

IX. L'APPRENTISSAGE DANS UN S.E.

L'apprentissage c'est l'amélioration des performances par l'expérience [Laurière, 1987]. Ce sujet est actuellement au coeur de la recherche en I.A..

L'homme est préprogrammé pour apprendre ; cependant il se heurte au problème de l'oubli. L'ordinateur, lui, a besoin d'être programmé pour apprendre mais il n'oublie pas. Mais devant certaines situations, il y a tellement de possibilités qu'il faut savoir en oublier (voir ignorer) quelques unes. Des études sont actuellement en cours pour tenter d'apprendre aux programmes d'ordinateurs d'apprendre. Deux questions ont été soulevées:

1. Comment un programme peut-il apprendre?
2. Qu'est ce qu'un programme peut apprendre?

IX.1. Comment apprendre?

Les études sur l'apprentissage l'ont classifié en cinq niveaux différents [Walker,1987].

1. l'apprentissage par les programmes.
2. l'apprentissage par coeur.
3. l'apprentissage par les statistiques.
4. l'apprentissage par les exemples.
5. l'apprentissage par l'observation et la découverte.

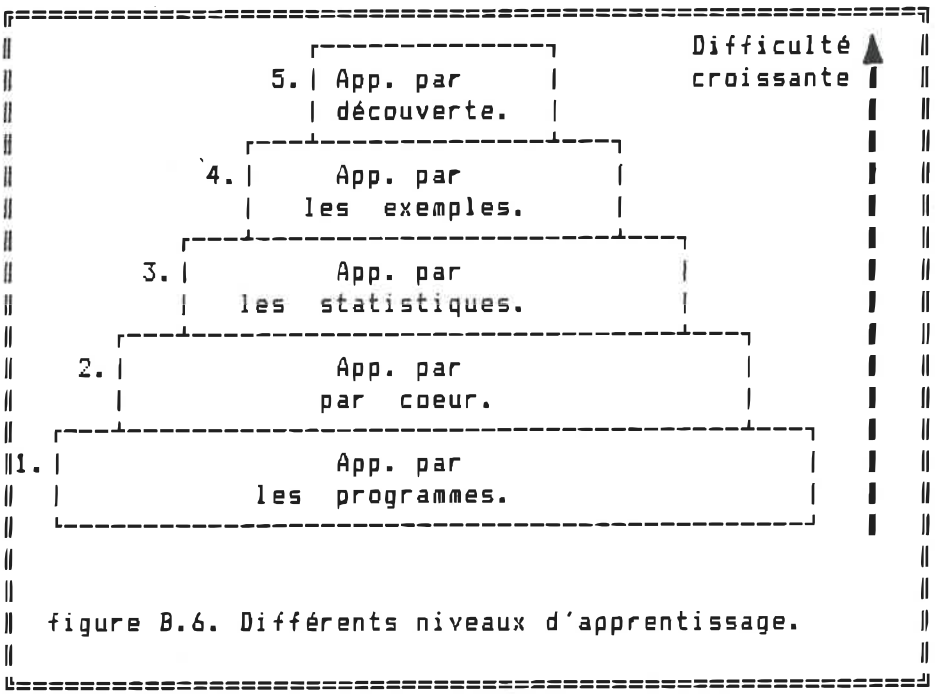


figure B.6. Différents niveaux d'apprentissage.

IX.2. Quoi apprendre?

Un S.E. peut apprendre:

- des règles,
- des plans (stratégies),
- des caractéristiques,
- ou des concepts.

X. LE TRAITEMENT DE L'INCERTITUDE:

L'ingénieur rencontre continuellement des situations d'incertitude causées par des comportements plus ou moins aléatoires.

Pour celà il a établi quelques méthodes pour résoudre ce problème. Parmi ces méthodes, les plus utilisées dans les S.E. sont:

1. La logique floue (fuzzy logic).
2. Les facteurs de certitude (certainty factors).
3. Les probabilités.

X.1. La logique floue:

Inventée par L. Zadeh en 1965, elle vise à étendre les concepts de la logique booléenne au raisonnement formel (formalisé). La logique booléenne permet deux valeurs de certitude: 1 (vrai) et 0 (faux). En logique floue on a une marge de valeurs de certitude qui varient de 0 à 1. Elle permet ainsi de représenter une vérité partielle. Les règles de base de la logique floue sont [Townsend, 1986].

1. p1 et p2 et p3 = min (p1, p2, p3).
2. p1 ou p2 ou p3 = max (p1, p2, p3).
3. non (p1) = 1 - p1.

où pi est la probabilité pour que l'évènement "i" se produit.

X.2. Les facteurs de certitude:

La logique floue est bonne dans le cas de vérité partielle connue, mais elle ne peut pas être utilisée quand l'information est manquante ou connue sans certitude.

Le concept de facteur de certitude, appelé aussi facteur de confiance a été introduit par Shortliffe dans le S.E. MYCIN. Il consiste à exprimer le degré de confiance attaché à la conclusion d'une règle. Sa formule de base s'écrit [Townsend,86].

soit FC[f:e]: le facteur de certitude de h étant donné e

MC[h:e]: la mesure de croyance en h étant donné e.

MN[h:e]: la mesure de non croyance en h étant donné e

on a:

FC[h:e] = MC[h:e] - MN[h:e].

N.B. MC et MN varient de 0 à 1 alors que FC varie de -1 à 1

Plusieurs théories existent pour le calcul de MC en tenant compte des prémisses de h. On veut exprimer que la mesure de croyance en une conclusion h augmente quand un plus grand nombre de ses prémisses sont vérifiées. Une des formules rencontrées dans la littérature dit:

MC (h) = MC (h1) + MC (h2) [1 - MC (h1)].

où h1 et h2 sont les prémisses de h.

X.3. La probabilité:

La probabilité bayesienne est souvent utilisée pour calculer la vraissemblance d'une hypothèse. La formule de base utilisée est: LR [H:E] = p [E:H] / p [E:H'].

où H' = non (H).

Cette formule dit explicitement que: la vraissemblance d'une hypothèse est égale à la probabilité d'évidence étant donné une hypothèse particulière divisée par la probabilité d'évidence étant donnée la "fausseté" de cette hypothèse.

XI. LANGAGES ET OUTILS DE S.E.:

Les langages de S.E. doivent répondre à certaines exigences dont les suivantes [Gaines,88] et [Townsend,86].

1. le langage doit supporter les structures orientée-données (data-driven).

2. le langage doit supporter le traitement de symboles, c.a.d. qu'on doit pouvoir représenter un objet par des symboles et non seulement que par des valeurs numériques. On doit aussi pouvoir appliquer un raisonnment formalisé aux variables symboliques.

3. le langage doit supporter le traitement des listes.

4. le langage doit supporter la récursion.

Actuellement, la tendance est vers l'utilisation d'outils de S.E. et non de simples langages. Ces outils qu'on appelle shells de S.E. contiennent des stratégies spécifiques de représentation des connaissances, d'inférence, et de contrôle. Souvent, des structures pour modéliser des types spécifiques de domaines sont inclus dans ces outils en plus d'une interface-usager et d'un éditeur.

Ces shells peuvent être construits avec des langages conventionnels comme le langage "C". Ils sont plus chers qu'un langage mais permettent de gagner beaucoup de temps lors du développement d'un S.E..

Un bon shell de S.E. doit offrir plusieurs possibilités telles que:

1. de multiples stratégies de représentation des connaissances.
2. de multiples stratégies de recherche.
3. le support de certain facteurs.
4. le retraçage (tracing) ou la justification.
5. l'acceptation de données en temps réel.

Un micro-ordinateur ne peut pas supporter toutes ces fonctions simultanement à cause de la limitation de la mémoire de travail. Pour celà, on doit étudier le problème posé et travailler à rebours pour déterminer les fonctions les plus utiles pour notre application. Le choix du shell se fera ensuite en fonction des fonctions retenues.

Les langages d'expression sont classés selon un niveau décroissant tel qu'illustré à la figure B.7.

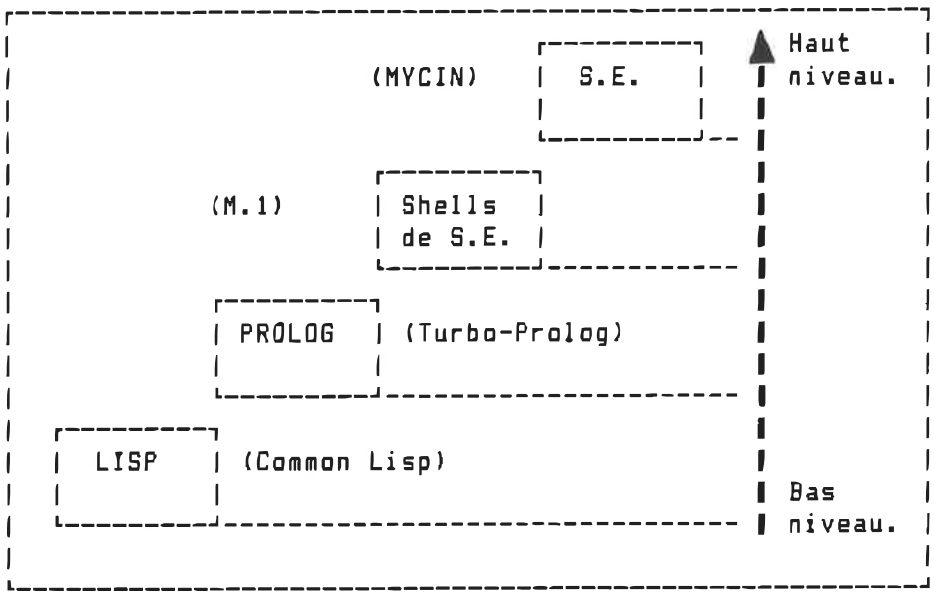


figure B.7.: Outils et langages de S.E.

Parmi les exemples de shells de S.E. fonctionnant sur micro-ordinateurs, on retrouve: KEE, M.1, Texas Instrument Personnal Consultant, Expert-Ease, etc...

XII. LES S.E. ET LA RECHERCHE OPERATIONNELLE:

En transport, plusieurs problèmes sont résolus à l'aide des techniques de recherche opérationnelle (R.O) telles que les algorithmes de calcul de chemin, de distribution, d'affectation et d'évaluation multi-critères.

L'objectif en R.O est en général de trouver la solution optimale lorsqu'elle existe. Ce travail d'optimisation n'est pas toujours des plus simples, et parfois on est limité à des applications

pour des problèmes de petites et moyennes tailles.

La technique S.E. présente une nouvelle approche en tentant de trouver une solution réalisable au moins aussi performante que celle qu'un expert du domaine pourrait trouver. Elle permet de représenter les problèmes réels de façon plus réaliste et explicite à l'aide d'un ensemble de relations. Ce mode de représentation permet d'éliminer l'intervention de l'homme pour modéliser ou formuler mathématiquement les différentes composantes du problème à résoudre. Ainsi on peut par exemple exprimer la priorité d'une commposante sur une autre par des relations de priorité flexibles au lieu de recourir à un poids affecté à chacune d'elles.

Pour résoudre les mêmes problèmes habituellement traités par la R.O, les S.E. mettent l'anphase plutôt sur l'utilisation judicieuse des techniques de base de l'algorithmique telles que la recherche en largeur, la recherche en profondeur, et diviser-pour-régner. Les algorithmes ne sont pas nécessairement utilisés à cause de la grande taille des problèmes à résoudre. Ceci s'explique par le caratère heuristique des S.E. [DESROCHERS 1984].

Quelques S.E. existants permettent de résoudre des problèmes de R.O. Cependant, l'utilisation des techniques classiques de la R.O reste limitée aux critères d'optimisation locaux dans les heuristiques. Par exemple, différentes techniques de probabilité sont utilisées pour le calcul des coefficients de vraissemblance des faits et des prémisses (conclusions) des règles.

Parmi les applications de S.E. aux problèmes de R.O on rencontre dans la littérature les suivants:

1. S.E. pour l'analyse multicritère:

Le problème de choix multicritère se définit comme la sélection de la ou les meilleures actions parmi un ensemble d'actions potentielles évaluées à l'aide d'un ensemble de critères donné. La méthode du surclassement est l'une des approches qui permettent cette analyse. L'approche S.E. a été utilisée pour construire le surclassement [PASCHE, 1987]. L'auteur suggère dans cette approche le remplacement des poids par les informations suivantes:

- un ordre partiel sur les critères correspondant à leur importance.

- des connaissances du décideur sur l'importance de certains sous-ensembles de critères.

A une représentation quantitative de l'importance (poids) des critères se substitue ainsi une représentation purement formelle (relations).

Le prototype a été construit avec le langage C-PROLOG et fonctionne sur un ordinateur VAX8600. Le plus grand problème traité comporte 15 actions évaluées sur la base de 18 critères. Voir chapitre I.

2. S.E. pour le calcul du chemin le plus court:

Un mini-prototype a été construit pour le calcul du chemin le plus court [BIHAN, 1987], et a été amélioré dans le présent travail (voir chap II). Dans ce mini S.E., aucun algorithme n'a été utilisé. Seule une petite heuristique a été définie. La performance est très bonne pour des petits réseaux. Cependant, le temps d'exécution et la quantité de mémoire nécessaire augmentent de façon très rapide. Le système écrit en TURBO-PROLOG fonctionne sur micro-ordinateur. La limite des 640k est très vite atteinte pour des réseaux moyens de quelques dizaines de noeuds, et aucune solution n'est alors obtenue.

3. S.E. pour le "Routing":

Ce problème consiste à desservir un réseau par une flotte de véhicules. Pour chaque véhicule, on doit déterminer une route réalisable tout en minimisant le coût total. Concrètement, il peut s'agir d'un problème de distribution de marchandises à des clients.

Deux approches principales sont habituellement utilisées pour résoudre ce genre de problèmes:

- la programmation mathématique et les algorithmes d'optimisation.
- les heuristiques.

La première a l'avantage de permettre de trouver une solution optimisant la fonction des coûts, cependant, elle n'est pas flexible

et ne permet pas facilement des modifications du problème. En plus, le temps de résolution augmente généralement très vite à mesure que la taille du problème augmente.

La seconde approche a l'avantage de permettre facilement certaines modifications telles que l'ajout de nouvelles contraintes. Cependant, il n'existe pas de logiciel général ; pour chaque nouveau problème on doit créer un nouveau assemblage d'heuristiques.

Les S.E. constituent une bonne manière d'implanter la seconde approche. En effet, les connaissances de l'expert du domaine (expertise) nous permettent d'expliquer la façon de choisir la meilleure heuristique pour résoudre un problème particulier. En plus l'interaction avec l'usager permet de considérer de nouvelles contraintes répondant aux conditions qui prévalent au cours de l'exercice [DESROCHERS, 1984].

D'autres applications comme l'algorithme de PERT [BIHAN,1987] existent dans la littérature. Dans toutes ces applications on remarque en général que le lien entre la R.O et les S.E. reste assez fragile, probablement à cause du degré d'expertise très élevé impliqué dans la résolution des problèmes de R.O. Toutefois deux axes pourraient permettre un meilleur couplage entre ces deux techniques.

Le premier axe concerne les mécanismes internes des S.E. tels que la stratégie de recherche. En effet, presque la totalité des S.E. existant font usage soit de la recherche en largeur, soit de la recherche en profondeur. L'utilisation de la programmation dynamique

par exemple, pourrait aboutir à l'optimisation de la stratégie de recherche et à la simplification de la structure de contrôle sur les règles du S.E..

L'autre axe est celui qui a jusqu'à présent été expérimenté par certain chercheurs. Il consiste à développer des S.E. pour résoudre des problèmes de R.O tel que le problème du "routing" [DESROCHERS, 1984].

XIII. CONCLUSION:

La science des S.E. est encore nouvelle, les premiers systèmes productifs datent du début des années 70.

Contrairement à d'autres domaines de l'informatique, plusieurs termes n'ont pas encore de définition standard et les théories sont encore entrain de changer très rapidement. Il y a encore beaucoup de place pour la créativité et la découverte par les ingénieurs de la connaissance [Townsend,86].

Le plus grand problème lors de la construction d'un S.E. n'est pas le développement du programme, mais plutôt la distillation des connaissances des experts dans une forme codifiée de façon à pouvoir la représenter objectivement dans le S.E..

Lors de la résolution d'un problème, l'expert humain utilise le raisonnement formalisé et l'analyse procédurale, mais fait aussi appel à l'analogie, à l'intuition, et au bons sens. Il ne faut pas s'attendre, en tout cas avec les moyens actuels, à ce qu'un S.E.

puisse faire usage de l'analogie, de l'intuition ou du bon sens [Townsend,1986].

Les S.E. ont aussi une importante faiblesse dûe au fait qu'ils n'ont pas de connaissances en dehors de leur domaine d'expertise. En effet un S.E. n'a aucune idée de ce qu'il ignore concernant son domaine d'expertise.

Mais malgré toutes ces limitations, les S.E. ont beaucoup d'avantages. Un S.E. est économiquement intéressant quand l'expertise humaine de son domaine est très chère. Il est objectif, consistent et non influençable dans sa façon de procéder. En plus il est capable de gérer une base de connaissances beaucoup plus large que celle d'un expert humain dans un domaine spécifique.

Pour cette jeune science, il est essentiel de sortir les recherches des laboratoires universitaires au monde réel. Il reste beaucoup à faire et la construction d'interpréteurs généraux efficaces documentés et dotés de mécanismes d'inférence assez riches n'est pas encore chose acquise [Laurière, 1987].

De même, la construction de prototypes de S.E. avec les moyens disponibles comme les micro-ordinateurs doit continuer pour sensibiliser les investisseurs à ce nouveau domaine.

ANNEXE C

PROGRAMME (TURBO-PROLOG)

```
code=4000
/*****************************************************************************
******************************************************************************
**                                                                          **
**            SYSTEME EXPERT DE RENSEIGNEMENT                               **
**                                                                          **
******************************************************************************
******************************************************************************/

domains

    lautobus=string*
    lnoeuds =string*
    lnaccès =string*
    lcritères=string*
    liste_real=real*
    lnuméros=integer*

database

    critère_essentiel(string)
    critère_important(string)
    critère_facultatif(string)
    critère_imp_satisfait(string)
    critère_fac_satisfait(string)
    itinéraire_retenu(string,string,lnoeuds,integer)
    liste_critères_essentiels(lcritères)
    liste_critères_importants(lcritères)
    liste_critères_facultatifs(lcritères)
    noeud_interdit(string)
    noeud_obligatoire(string)
    liste_noeuds_interdits(lnoeuds)
    liste_noeuds_obligatoires(lnoeuds)
    fréquence(string,real)
    tronçon(string,string,string,real,real,lautobus,string)
    artère(string,lnoeuds)
    ligne(string,lnoeuds,real)
    monument(string,lnaccès)
    saturé(string,string,string,real).
    élément(string)
    contrainte(integer)
    paysage(integer)
    liste_contraintes(lnuméros)
    liste_paysages(lnuméros)
    temps_max(real)
    long_max(real)

predicates

    critères(lcritères)
    pondération(lcritères)
    satisfait(string,string,lnoeuds,string)
    recherche_itinéraire(string,string,lnoeuds,lnoeuds,real)
    menu_principal
    procède(integer)
    menu_2
    exécute(integer)
    appartient(string,lnoeuds)
    appartient(string,lautobus)
    appartient(integer,lnuméros)
```

```
échelle
longueur(lcritères,integer)
longueur(lautobus,integer)
longueur(lnaccès,integer)
longueur(lnoeuds,integer)
longueur(lnumèros,integer)
plus_grand(liste_real,real)
plus_petit(liste_real,real)
informer(integer,string,string,lnoeuds,lnoeuds)
plus_court_itinéraire(string,string,lnoeuds,real)
sous_menu(string,string,lnoeuds,lnoeuds)
imprime(lcritères)
interprète(string,string,lnoeuds,integer)
temps_nécessaire(lnoeuds,real)
temps(string,string,real)
attente(lnoeuds,real)
vide(lautobus)
attendre(lautobus)
état_de_saturation(integer)
exécuter(string,lnoeuds)
imprime_6(lnaccès)
imprime_1(string)
imprime_2(string,lnoeuds,string,string)
imprimer(lnoeuds)
appartiennent(lnoeuds,lnoeuds)
appartiennent(lautobus,lautobus)
procéder(integer,string,string)
temps_total(string,string,lnoeuds,lnoeuds,real)
itinéraire_transfert(string,string,lnoeuds,real)
écrire(lcritères,integer)
bon_itinéraire(string,string,lcritères,integer)
itinéraire(string,string,lnoeuds,lcritères)
importance(string,string,lnoeuds,real)
importance_1(string,string,lnoeuds,real)
importance_2(string,string,lnoeuds,real)
évalue_1(string,string,lnoeuds,lcritères,real)
évalue_2(string,string,lnoeuds,lcritères,real)
test_1(string,string,lnoeuds,string,real)
test_2(string,string,lnoeuds,string,real)
itinéraire_temps(string,string,lnoeuds,lnoeuds,real)
indice_importance(string,string,lnoeuds,real)
classe(string,real)
cherche_bus(string,string,string)
unies(lnoeuds,lnoeuds)
vérifie(string,string,lnoeuds,integer)
valide(string,string)
valide_2(string,string)
identification(string,string)
accès_1
accès_2(string,string)
menu1_1
mise_à_jour(integer)
modifier(integer)
ajouter(integer)
annuler(integer)
test
modifie(string,string,string)
modifie(real,real,real)
suite1(string)
suite2(string)
suite3(string)
suite4(string)
suite5(string)
entrée_1
entrée_2
entrée_3(integer,lautobus,lautobus)
```

```
    entrée_3(integer,lnoeuds,lnoeuds)
    entrée_3(integer,lnaccès,lnaccès)
    entrée_4(integer,lnoeuds)
    entrée_4(integer,lautobus)
    entrée_4(integer,lnaccès)
    entrée_5
    entrée_6
    dernier(string)
    spécifier(lcritères)
    remplir(string)
    interpréter(string,string,lnoeuds,lnuméros)
    vérifier(string,string,lnoeuds,lnuméros)
    lecture
    clear(integer)
    nettoyer(integer)
    effacer(integer)
    suite21(string)
    suite22(string)
    suite23(string)
    suite24(string)
    suite25(string)
    suite31(string)
    suite32(string)
    suite33(string)
    suite34(string)
    suite35(string)
    les_contraintes(lnuméros)
    les_paysages(lnuméros)

/*
 goal

    menu_principal.


*/
 clauses

 /*********************   LES TRONCONS DU RESEAU   **********************
  *                                                                      *
  * tronçon( noeud1,noeud2,rue,longueur du tronçon,vitesse moyenne de  *
  *          circulation,[ liste des autobus opérants sur ce tronçon ],*
  *          paysage ) .                                                 *
  *                                                                      *
  ************************************************************************/

 tronçon("sher-deni","dup-deni","st denis",3,25,["10"],naturel).
 tronçon("dup-deni","dor-deni","st denis",2,25,["10"],industriel).
 tronçon("dup-deni","sher-deni","st denis",3,25,["10"],naturel).
 tronçon("dor-deni","dup-deni","st denis",2,25,["10"],industriel).

 tronçon("sher-hub","dec-hub","st hubert",1.5,20,["20"],résidentiel).
 tronçon("dec-hub","sher-hub",",st hubert",1.5,20,["20"],résidentiel).
 tronçon("dec-hub","dup-hub","st hubert",1.5,20,["20","40"],résidentiel).
 tronçon("dup-hub","dec-hub","st hubert",1.5,20,["20","40"],résidentiel).
 tronçon("dup-hub","guy-hub","st hubert",1.5,20,["20"],résidentiel).
 tronçon("guy-hub","dup-hub","st hubert",1.5,20,["20"],résidentiel).
 tronçon("guy-hub","dor-hub","st hubert",0.50,20,["20"],résidentiel).
 tronçon("dor-hub","guy-hub","st hubert",0.50,20,["20"],résidentiel).

 tronçon("sher-laur","sher-hub","sherbrooke",3,30,["50"],résidentiel).
 tronçon("sher-hub","sher-laur","sherbrooke",3,30,["50"],résidentiel).
 tronçon("sher-hub","sher-deni","sherbrooke",3,30,["50"],résidentiel).
 tronçon("sher-deni","sher-hub","sherbrooke",3,30,["50"],résidentiel).

 tronçon("dec-laur","dec-hub","decelles",3,20,["50"],résidentiel).
```

```
tronçon("dec-hub","dec-laur","decelles",3,20,["50"],résidentiel).

tronçon("dup-hub","dup-deni","dupuis",3,20,["40"],naturel).
tronçon("dup-deni","dup-hub","dupuis",3,20,["40"],naturel).

tronçon("guy-laur","guy-hub","guy",3,20,[],résidentiel).
tronçon("guy-hub","guy-laur","guy",3,20,[],résidentiel).

tronçon("dor-laur","dor-hub","dorchester",3,30,["30"],résidentiel).
tronçon("dor-hub","dor-laur","dorchester",3,30,["30"],résidentiel).
tronçon("dor-hub","dor-deni","dorchester",3,30,["30"],industriel).
tronçon("dor-deni","dor-hub","dorchester",3,30,["30"],industriel).

/************************      LES ARTERES DU RESEAU      ***************
 *                                                                      *
 *               artère( nom , [ liste des noeuds de l'artère ] ).      *
 *                                                                      *
 ************************************************************************/

artère("st denis",["sher-deni","dup-deni","dor-deni"]).
artère("st hubert",["sher-hub","dec-hub","dup-hub","guy-hub","dor-hub"]).
artère("sherbrooke",["sher-laur","sher-hub","sher-deni"]).
artère("decelles",["dec-laur","dec-hub"]).
artère("dupuis",["dup-hub","dup-deni"]).
artère("guy",["guy-laur","guy-hub"]).
artère("dorchester",["dor-laur","dor-hub","dor-deni"]).


/***************  LES LIGNES D'AUTOBUS ACTIVES DU RESEAU  ***************
 *                                                                      *
 *  ligne ( numéro de la ligne , liste des noeuds desservis,            *
 *          fréquence en heure de pointe ,                              *
 *          début de la période de service , fin de la période de       *
 *          service ) .                                                 *
 *                                                                      *
 ************************************************************************/

ligne("10",["sher-deni","dup-deni","dor-deni"],5).
ligne("10",["dor-deni","dup-deni","sher-deni"],5).
ligne("20",["sher-hub","dec-hub","dup-hub","guy-hub","dor-hub"],4).
ligne("20",["dor-hub","guy-hub","dup-hub","dec-hub","sher-hub"],4).
ligne("30",["dor-laur","dor-hub","dor-deni"],3).
ligne("30",["dor-deni","dor-hub","dor-laur"],3).
ligne("40",["dec-laur","dec-hub","dup-hub","dup-deni"],7).
ligne("40",["dup-deni","dup-hub","dec-hub","dec-laur"],7).
ligne("50",["sher-laur","sher-hub","sher-deni"],4).
ligne("50",["sher-deni","sher-hub","sher-laur"],4).


/********************    LES MONUMENTS DU QUARTIER    *********************
 *                                                                      *
 *  monument ( nom du monument , [ liste des noeuds d'accès ] ,         *
 *             heure d'ouverture , heure de fermeture ) .               *
 *                                                                      *
 ************************************************************************/

monument("parc1",["sher-deni","dup-deni"]).
monument("marché",["dup-deni","dor-deni"]).
monument("hopital",["dup-deni","dup-hub","guy-hub","dor-hub","dor-deni"]).
monument("université",["sher-deni","sher-hub","dec-hub","dup-hub",
                       "dup-deni"]).


/************************  LES LIENS SATURES  ***************************
 *                                                                      *
 *      saturé ( nom de la rue , noeud1 , noeud2 , pourcentage ) .      *
```

```
*                                                                          *
****************************************************************************/

saturé("dorchester","dor-hub","dor-deni",95).


/********** LISTE DES CRITERES D'EVALUATION D'UN ITINERAIRE ***************
*                                                                          *
*      critères ( liste des critères ) .                                   *
*      les_contraintes(liste des numéros de contraintes ) .                *
*      les_paysages(liste des numéros de type de paysage ) .               *
*                                                                          *
****************************************************************************/

critères(["éliminer les transferts",
          "bonne qualité du paysage",
          "imposer des contraintes à l'itinéraire"]).

les_contraintes([1,2,3,4]).

les_paysages([1,2,3,4]).

/********************* LISTE DES PERSONNES PRIVILEGIEES ********************
*                                                                          *
*      IDENTIFICATION ( NOM DE LA PERSONNE , CODE D'ACCES ).               *
*                                                                          *
****************************************************************************/

identification(souissi,sou100163).
identification(pepin,pep151257).

/************************ LANCEMENT DU PROGRAMME ************************
*                                                                          *
*      Mot-clé pour partir la consultation : menu_principal .              *
*                                                                          *
****************************************************************************/

menu_principal :-
                makewindow(1,5,5,"  MENU PRINCIPAL  ",0,0,25,80),
                write("\n\n\n    1   Déterminer un itinéraire .",
                      "\n\n      2   Demander une information partielle .",
                      "\n\n      3   Mettre à jour la base de faits .",
                      "\n\n      4   Fin de la consultation .",
                      "\n\n\n            TAPEZ LE NUMERO DE VOTRE CHOIX ."),
                nl,nl,
                readint(Choix),
                procède(Choix).

/*********************** RECHERCHE D'ITINERAIRE ************************
*                                                                          *
*      Les étapes de recherche se résument comme suit :                    *
*                                                                          *
*              1.  choix du noeud de départ .                              *
*              2.  choix du noeud de destination .                         *
*              3.  pondération des critères d'évaluation .                 *
*              4.  détermination des itinéraires possibles.                *
*              5.  élimination des itinéraires non conformes .             *
*              6.  détermination du meilleur itinéraire .                  *
*                                                                          *
****************************************************************************/


procède(1) :-  removewindow,
               makewindow(2,6,6,"  MODALITE DE RECHERCHE  ",0,0,25,80),
               write("\n\n   En fonction de quel critère voulez-vous ",
                     "\n   que la recherche soit effectuée ? " ,
```

```
                "\n\n     1  distance minimale de parcours .",
                "\n      2  temps minimum de voyage .",
                "\n      3  aucun transfert .",
                "\n      4  combinaison de critères .",
                "\n\n   TAPEZ LE NUMERO DE VOTRE CHOIX"),nl,
        readint(MOD),
        removewindow,
        makewindow(21,12,12,"  RECHERCHE  ",2,2,21,76),
        write("   Point de départ ?   "),      readln(ORIG),
        valide(ORIG,DEP),
        write("   Point de destination ?   "), readln(ARR),
        valide(ARR,DEST),
        procéder(MOD,DEP,DEST).

procède(2) :-  menu_2 .

procède(3) :-  removewindow,
        makewindow(2,6,6,"  MISE A JOUR DES FAITS  ",0,0,25,80),
        accès_1,
        menu1_1.

procède(4) :-  exit .

procéder(1,O,D) :-
        findall(L,recherche_itinéraire(O,D,[O],_,L),TOUT),
        plus_petit(TOUT,K),
        recherche_itinéraire(O,D,[O],ITIN,K),
        write("\n   Le chemin le plus court pour aller de :",
              "\n   ",O,"  à  ",D,"  est :\n   ",ITIN,
              "\n   La longeur de parcours est de ",K,"  km .",
              "\n\n          PRESSEZ <ENTER>"),readln(_),
        menu_principal.

procéder(2,O,D) :-
        findall(T,temps_total(O,D,[O],_,T),TOUT),
        plus_petit(TOUT,K),
        temps_total(O,D,[O],ITIN,K),
        write("\n   Le chemin le plus rapide pour aller de :",
              "\n   ",O,"  à  ",D,"  est :\n   ",O,ITIN,
              "\n   Le temps de parcours est de ",K,"  min .",
              "\n\n          PRESSEZ <ENTER>"),readln(_),
        menu_principal .

procéder(3,O,D) :-
        findall(TR,itinéraire_transfert(O,D,_,TR),TOUT),
        write(" -----> ",TOUT),readln(_),
        plus_petit(TOUT,K),
        recherche_itinéraire(O,D,[O],ITIN,K),
        write("\n   Le chemin sans transfert , le plus court , ",
              "\n   pour aller de :",
              "\n   ",O,"  à  ",D,"  est :\n   ",O,ITIN,
              "\n                Sa longeur est de : ",K," km .",
              "\n\n          PRESSEZ <ENTER>"),
              readln(_),
              menu_principal,!.

procéder(3,O,D) :-
        tronçon(O,_,_,_,_,_,_),
        tronçon(_,D,_,_,_,_,_),
        write("   Impossible ! ",
                  "\n   Au moins un transfert est nécessaire .",
                  "\n\n          PRESSEZ <ENTER>"),
        readln(_),
        menu_principal .

procéder(4,O,D) :-
```

```
                write(" Les critères de choix d'itinéraires sont :"),nl,
                critères(C),
                longueur(C,LONG),
                écrire(C,LONG),nl,
                write("          PRESSEZ <ENTER>"), readln(_),
                makewindow(22,23,23," ECHELLE DE PONDERATION ",0,40,25,38),
                échelle,
                makewindow(5,28,28," PONDERATION DES CRITERES ",0,0,25,40),
                pondération(C),
                findall(X,critère_essentiel(X),ESS),
                assertz(liste_critères_essentiels(ESS)),
                shiftwindow(21),
                clearwindow,
                write(" Seuls les itinéraires répondant aux critères ",
                    "\n essentiels suivant seront retenus:"),nl,nl,
                imprime(ESS),nl,
                readln(_),
                spécifier(ESS),
                bon_itinéraire(O,D,ESS,1),
                test,
                findall(COEFF,indice_importance(O,D,_,COEFF),IND),
                plus_grand(IND,G),
                indice_importance(O,D,ITIN,G),
                write("\n  l'itinéraire le plus important est :\n  ",ITIN,
                    " .","\n  Son indice d'importance est : ",G," .",
                    "\n\n             PRESSEZ <ENTER>"),
                readln(_),
                sous_menu(O,D,[O],ITIN).

valide(X,Y) :- artère(_,L),
                appartient(X,L),
                Y=X
                ;
                write("   Erreur ! Retapez ce dernier point .\n     "),
                readln(Y),
                valide_2(X,Y).

valide_2(_,Y) :-
                artère(_,L),
                appartient(Y,L)
                ;
                write("   Erreur ! impossible de procéder .",
                    "\n\n        PRESSEZ <ENTER> ."),readln(_) ,
                menu_principal .

/******************** procéder(3) ************************************
 *************************************************************************/

accès_1 :-      write("\n      NOM DE L'USAGER ?      "),readln(NOM),
                identification(NOM,CODE),
                accès_2(NOM,CODE),!.

accès_1 :-      write("\n      DESOLE !  ACCES NON AUTORISE .",
                    "\n        PRESSEZ <ENTER> ."),readln(_),
                menu_principal.

accès_2(N,CODE) :-
                write("\n      CODE D'ACCES ?         "),readln(Y),
                CODE=Y,
                write("\n      ACCES AUTORISE POUR :  ",N,
                    "\n                        CODE :  ",CODE,
                    "\n\n          PRESSEZ <ENTER> "),readln(_),
                menu1_1,
                !.

accès_2(_,_) :-
```

```
                write("\n      DESOLE !  CODE NON AUTORISE .",
                      "\n        PRESSEZ <ENTER> ."),readln(_),
                menu_principal.

menu1_1 :-      clearwindow,
                write("              Voulez-vous : ",
                      "\n\n      1. Ajouter une information ?",
                      "\n\n      2. Annuler une information ?",
                      "\n\n      3. Modifier une information ?",
                      "\n\n      4. Aide ?",
                      "\n\n      5. Quitter ?",
                      "\n\n         TAPEZ LE NUMERO DE VOTRE CHOIX ."),nl,
                readint(CHOIX),
                mise_à_jour(CHOIX),
                write("\n\n      FIN DE LA MISE A JOUR .",
                      "\n\n         PRESSEZ <ENTER> . "),readln(_),
                menu_principal.

mise_à_jour(1) :-
                clearwindow,
                makewindow(111,13,13," AJOUT D'INFORMATION ",0,30,25,50),
                write("      Quel est le prédicat concerné ?",
                      "      \n1.tronçon  2.ligne  3.artère  4.saturation",
                      "      \n5.monument  6.aucun .",
                      "      \n\n ENTREZ LE NUMERO DE VOTRE CHOIX ."),
                readint(NUM),
                ajouter(NUM),
                menu1_1 .


mise_à_jour(2) :-
                clearwindow,
                makewindow(112,13,13," ANNULATION D'INFORMATION ",0,30,25,50),
                write("      Quel est le prédicat concerné ?",
                      "      \n1.tronçon  2.ligne  3.artère  4.saturation",
                      "      \n5.monument  6.aucun .",
                      "      \n\n ENTREZ LE NUMERO DE VOTRE CHOIX ."),
                readint(NUM),
                annuler(NUM),
                menu1_1 .

mise_à_jour(3) :-
                clearwindow,
                makewindow(113,13,13," MODIFICATION D'INFORMATION ",0,30,25,50),
                write("   Quel prédicat voulez-vous modifier ?  1. tronçon ",
                      "\n    2. ligne  3. artère  4. saturation ",
                      "\n    5. monument  6. aucun .",
                      "\n\n          TAPEZ UN NUMERO .    "),readint(NUM),
                modifier(NUM),
                menu1_1.

mise_à_jour(4) :-
                clearwindow,
                makewindow(114,13,13," AIDE ",0,30,25,50),
                write("\n\n       Les informations qu'on peut traiter dans",
                      "\n       cette section sont les suivantes :",
                      "\n         - ajouter des éléments du réseau tels que ",
                      "\n           tronçon , artère , monument , ligne de ",

                      "\n           T.C. etc...",
                      "\n         - éliminer des éléments du réseau .",
                      "\n         - ajouter/éliminer une information sur ",
                      "\n           l'état de la circulation ( vitesse de la",
                      "\n           circulation , état de saturation ... ) .",
                      "\n\n            PRESSEZ <ENTER>"),readln(_),
                menu1_1.
```

```
mise_à_jour(5) :-
                menu_principal .

modifier(1) :- write("============ MODIFICATION D'UN TRONçON ============",
                     "\n\n     Premier noeud ?     "),readln(N),
               valide(N,N1),nl,
               write("     Deuxième noeud ?     "),readln(M),
               valide(M,N2),
               tronçon(N1,N2,RUE,LONG,V,BUS,PAY),
               write("     Pour chacun des arguments , tapez la nouvelle ",
                     "\n     valeur .",
                     "\n\n     Premier noeud ?     "),readln(NO1),
                     modifie(N1,NO1,NOEUD1),nl,
               write("     Deuxième noeud ?     "),readln(NO2),
               modifie(N2,NO2,NOEUD2),nl,
               write("     Rue ?     "),readln(R),
               modifie(RUE,R,RUE2),nl,
               write("     Longueur du tronçon ?     "),readreal(L),

               modifie(LONG,L,LONG2),nl,
               write("     Vitesse moyenne ?     "),readreal(VIT),
               modifie(V,VIT,V2),nl,
               write("     Liste des autobus ?     ",
                     "\n     Entrez 0 pour garder la liste actuelle",
                     "\n     ou     1 pour la modifier ."),readint(I),
               entrée_3(I,BUS,BUS2),nl,nl,
               write("     Paysage ?     "),readln(P),
               modifie(PAY,P,PAY2),nl,nl,
               Write("     Le prédicat < tronçon(",N1,",",N2,",",RUE,",",
                     LONG,",",V,",",BUS,",",PAY,") >",
                     "\n sera remplacé par le suivant : ",
                     "\n       < tronçon(",NOEUD1,",",NOEUD2,",",RUE2,",",
                     LONG2,",",V2,",",BUS2,",",PAY2,") > ."),nl,
               retract(tronçon(N1,N2,RUE,LONG,V,BUS,PAY)),
               assertz(tronçon(NOEUD1,NOEUD2,RUE2,LONG2,V2,BUS2,PAY2)),
               write("     Autre tronçon à modifier (o/n) ?     "),
               readln(S),
               suite1(S) .

modifier(2):-  write("============ MODIFICATION D'UNE LIGNE ============",
                  "\n\n     Nom de la ligne à modifier ?     "),readln(L1),
               ligne(L1,LIST1,F1),
               write("     Pour chacun des arguments entrez la nouvelle ",
                     "\n     valeur .",
                     "\n\n     Nom de la ligne ?     "),readln(LI),
               modifie(L1,LI,L2),nl,
               write("     Liste des arrêts ? ",
                     "\n     Tapez 1 pour modifier ou 0 pour garder la ",
                     "\n     valeur actuelle .     "),readint(I),nl,nl,
               entrée_3(I,LIST1,LIST2),nl,
               write("     Fréquence en heure de pointe ?     "),readreal(F),

               modifie(F1,F,F2),nl,
               write("     Le prédicat : < ligne(",L1,",",LIST1,",",F1,") >",
                     "\n     sera remplacé par le suivant : ",
                     "\n     < ligne(",L2,",",LIST2,",",F2,") > ."),nl,
               retract(ligne(L1,LIST1,F1)),
               assertz(ligne(L1,LIST2,F2)),
               write("     Autre ligne à modifier (o/n) ?     "),readln(S),
               suite2(S).

modifier(3) :- write("============ MODIFICATION D'UNE ARTERE ============",
                     "\n\n     Nom de l'artère à modifier ?     "),
               readln(A1),
               artère(A1,Ln1),
```

```
                write("     Pour chacun des arguments entrez la novelle ",
                      "\n     valeur .",
                      "\n\n     Nouveau nom de l'artère ?     "),
                readln(A2),
                write("     Nouvelle liste de noeuds ?",
                      "\n     Entrez 1 pour modifier ou 0 pour garder ",
                      "\n     la liste actuelle .     "),readint(I),nl,
                entrée_3(I,Ln1,Ln2),
                write("     Le prédicat : < artère(",A1,",",Ln1,") > sera ",
                      "\n     remplacé par le suivant :",
                      "\n     artère(",A2,",",Ln2,") . "),nl,
                retract(artère(A1,Ln1)),
                assertz(artère(A2,Ln2)),
                write("     Autre artère à modifier (o/n) ?     "),readln(S),
                suite3(S).

modifier(4) :- write("===== MODIFICATION DE L'ETAT DE SATURATION =====",
                      "\n\n     Quelle est l'artère concernée ?     "),
                readln(A1),nl,
                write("     1er noeud du tronçon saturé ?     "),readln(N1),
                write("     2ieme noeud du tronçon saturé ?     "),readln(N2),
                saturé(A1,N1,N2,P1),
                write("     Quel est le nouveau pourcentage de saturation ",
                      "\n     sur ce tronçon ?     "),readreal(P),
                modifie(P1,P,P2),
                write("     Le prédicat : < saturé(",A1,",",N1,",",N2,",",P1,
                      ") > ","\n     sera remplacé par le suivant :",
                      "\n     saturé(",A1,",",N1,",",N2,",",P2,") ."),nl,
                retract(saturé(A1,N1,N2,P1)),
                assertz(saturé(A1,N1,N2,P2)),
                write("   Autres modifications de l'état de saturation (o/n)",
                      " ?     "),readln(S),
                suite4(S).

modifier(5) :- write("============  MODIFICATION D'ADRESSE  ============",
                      "\n\n     Nom du monument concerné ?     "),
                readln(MON1),
                monument(MON1,LNA1),
                write("     Pour chacun des arguments entrez la nouvelle ",
                      "\n     valeur .",
                      "\n\n     Nouveau nom du monument ?     "),readln(MON),
                modifie(MON1,MON,MON2),nl,
                write("     Nouvelle liste des noeuds d'accès ? ",
                      "\n     Entrez 1 pour modifier ou 0 pour garder la ",
                      "\n     liste actuelle ."),nl,readint(I),
                entrée_3(I,LNA1,LNA2),nl,nl,
                write("     Le prédicat : < monument(",MON1,",",LNA1,") > ",
                      "\n     sera remplacé par le suivant :",
                      "\n     monument(",MON2,",",LNA2,") ."),nl,
                retract(monument(MON1,LNA1)),
                assertz(monument(MON2,LNA2)),
                write("     Autres modifications d'adresses (o/n) ?     "),
                readln(S),
                suite5(S).


modifier(6) :- ! .

modifie(_,Y,X2) :-
                X2=Y .

suite1(o) :-    modifier(1) .
suite1(n) :-    write("============ FIN DE LA MODIFICATION ============").

suite2(o) :-    modifier(2) .
suite2(n) :-    write("============ FIN DE LA MODIFICATION ============").
```

```
suite3(o) :-   modifier(3) .
suite3(n) :-   write("============ FIN DE LA MODIFICATION ============").

suite4(o) :-   modifier(4) .
suite4(n) :-   write("============ FIN DE LA MODIFICATION ============").

suite5(o) :-   modifier(5) .
suite5(n) :-   write("============ FIN DE LA MODIFICATION ============").

entrée_3(0,BUS,BUS2) :-
               BUS2=BUS ,
               write("     ======== la liste restera la même . ========").

entrée_3(1,_,BUS2) :-
               write("     Entrez les éléments de la liste un par un et ",
                     "\n     à la fin tapez < fin > ."),nl,
               lecture,
               findall(X,élément(X),BUS2),
               longueur(BUS2,L),
               effacer(L).

  /* ==================     AJOUT D'INFORMATION     ====================*/

ajouter(1) :-  write("========== AJOUT D'UN TRONçON ==========",
                     "\n\n     Premier noeuds ?     "),readln(N1),
               write("\n     Deuxième noeud ?     "),readln(N2),
               write("\n     Nom de la rue ?     "),readln(RUE),
               write("\n     Longueur du tronçon ?     "),readreal(LONG),
               write("\n     Vitesse moyenne ?     "),readreal(VIT),
               write("\n     Liste des autobus ?",
                     "\n     Pour rentrer les éléments de la liste ",
                     "     tapez 1 , et ",
                     "\n     si aucun autobus ne dessert ce tronçon , alors",
                     "\n     tapez 0 . "),nl,nl,
               readint(REP),
               entrée_4(REP,BUS),
               write("     Paysage prédominant ?     "),readln(PAY),
               write("\n\n     Le prédicat suivant sera ajouté :",
                     "\n      tronçon(",N1,",",N2,",",RUE,",",LONG,",",
                     VIT,",",BUS,",",PAY,") .") ,nl,nl,
               assertz(tronçon(N1,N2,RUE,LONG,VIT,BUS,PAY)),
               write("     Autres tronçons à ajouter ? (o/n)     "),readln(S),
               suite21(S).

ajouter(2) :-  write("========== AJOUT D'UNE LIGNE ==========",
                     "\n\n     Nom de la ligne ?     "),readln(LI),
               write("\n     Liste des arrêts ? "),nl,
               entrée_4(1,ARRET),
               write("     Fréquence en heure de pointe ?     "),
               readreal(FR),
               write("\n\n     Le prédicat suivant sera ajouté :",
                     "\n      ligne(",LI,",",ARRET,",",FR,") .") ,nl,nl,
               assertz(ligne(LI,ARRET,FR)),
               write("     Autres lignes à ajouter ? (o/n)     "),readln(S),
               suite22(S).

ajouter(3) :-  write("========== AJOUT D'UNE ARTERE ==========",
                     "\n\n     Nom de l'artère ?     "),readln(A),
               write("\n     Liste des noeuds ?     "),
               entrée_4(1,LN),
               write("\n\n     Le prédicat suivant sera ajouté :",
                     "\n      artère(",A,",",LN,") .") ,nl,nl,
               assertz(artère(A,LN)),
               write("     Autres artères à ajouter ? (o/n)     "),readln(S),
               suite23(S).
```

```
ajouter(4) :-  write("========== AJOUT DE LIEN SATURE ==========",
                     "\n\n      Rue concernée ?      "),readln(RUE),
               write("\n      1er noeud du tronçon saturé ?      "),
               readln(N1),
               write("\n      2ieme noeud du tronçon saturé ?      "),
               readln(N2),
               write("\n      Pourcentage de saturation ?      "),
               readreal(P),
               write("\n\n      Le prédicat suivant sera ajouté :",
                     "\n         saturé(",RUE,",",N1,",",N2,",",P,") .") ,nl,nl,
               assertz(saturé(RUE,N1,N2,P)),
               write("      Autres artères à ajouter ? (o/n)      "),readln(S),
               suite24(S).

ajouter(5) :-  write("========== AJOUT DE MONUMENTS ==========",
                     "\n\n      Nom du monument ?      "),readln(MON),
               write("      Liste des noeuds d'accès ? "),nl,
               entrée_4(1,LNA),
               write("\n\n      Le prédicat suivant sera ajouté :",
                     "\n         monument(",MON,",",LNA,") .") ,nl,nl,
               assertz(monument(MON,LNA)),
               write("      Autres monuments à ajouter ? (o/n)      "),readln(S),
               suite25(S).

ajouter(6) :-  !.

suite21(o) :-  ajouter(1) .
suite21(n) :-  write("==========  AJOUT TERMINE ==========").

suite22(o) :-  ajouter(2) .
suite22(n) :-  write("==========  AJOUT TERMINE ==========").

suite23(o) :-  ajouter(3) .
suite23(n) :-  write("==========  AJOUT TERMINE ==========").

suite24(o) :-  ajouter(4) .
suite24(n) :-  write("==========  AJOUT TERMINE ==========").

suite25(o) :-  ajouter(5) .
suite25(n) :-  write("==========  AJOUT TERMINE ==========").

entrée_4(0,[]).
entrée_4(1,BUS) :-
               write("      Donner les éléments de la liste un par un et ",
                     "\n      à la fin tapez < fin > . "),
               lecture,
               findall(X,élément(X),BUS),
               longueur(BUS,L),
               effacer(L).

/*========================  ANNULATION DE FAITS  ========================*/

annuler(1) :-  write("========== ELIMINATION D'UN TRONCON  ==========",
                     "\n\n      Premier noeuds ?      "),readln(N1),
               write("\n      Deuxième noeud ?      "),readln(N2),
               write("\n      Nom de la rue ?      "),readln(RUE),
               tronçon(N1,N2,RUE,LONG,VIT,BUS,PAY),
               write("\n\n      Le prédicat suivant sera éliminé :",
                     "\n         tronçon(",N1,",",N2,",",RUE,",",LONG,",",
                     VIT,",",BUS,",",PAY,") .") ,nl,nl,
               retract(tronçon(N1,N2,RUE,LONG,VIT,BUS,PAY)),
               write("      Autres tronçons à éliminer ? (o/n)      "),readln(S),
               suite31(S).

annuler(2) :-  write("========== ELIMINATION D'UNE LIGNE ==========",
```

```
                    "\n\n     Nom de la ligne ?     "),readln(LI),
                ligne(LI,BUS,F),
                write("\n\n     Le prédicat suivant sera éliminé :",
                    "\n       ligne(",LI,",",BUS,",",F,") .") ,nl,nl,
                retract(ligne(LI,BUS,F)),
                write("     Autres lignes à éliminer ? (o/n)     "),readln(S),
                suite32(S).

annuler(3) :-   write("========== ELIMINATION D'UNE ARTERE ==========",
                    "\n\n     Nom de l'artère ?     "),readln(A),
                artère(A,LN),
                write("\n\n     Le prédicat suivant sera annulé :",
                    "\n       artère(",A,",",LN,") .") ,nl,nl,
                retract(artère(A,LN)),
                write("     Autres artères à éliminer ? (o/n)     "),readln(S),
                suite33(S).

annuler(4) :-   write("========== ELIMINATION DE  SATURATION ==========",
                    "\n\n     Rue concernée ?     "),readln(RUE),
                write("\n     1er noeud du tronçon saturé ?     "),
                readln(N1),
                write("\n     2ieme noeud du tronçon saturé ?     "),
                readln(N2),
                saturé(RUE,N1,N2,P),
                write("\n\n     Le prédicat suivant sera annulé :",
                    "\n       saturé(",RUE,",",N1,",",N2,",",P,") .") ,nl,nl,
                retract(saturé(RUE,N1,N2,P)),nl,nl,
                write("     Autres saturation à éliminer ? (o/n)     "),readln(S
),
                suite34(S).

annuler(5) :-   write("========== ELIMINATION DE MONUMENTS ==========",
                    "\n\n     Nom du monument ?     "),readln(MON),
                monument(MON,LNA),nl,
                write("\n\n     Le prédicat suivant sera éliminé :",
                    "\n       monument(",MON,",",LNA,") .") ,nl,nl,
                retract(monument(MON,LNA)),
                write("     Autres monuments à éliminer ? (o/n)     "),readln(S)
,
                suite35(S).

annuler(6) :-   !.

suite31(o) :-   annuler(1) .
suite31(n) :-   write("==========  ANNULATION TERMINE ==========").

suite32(o) :-   annuler(2) .
suite32(n) :-   write("==========  ANNULATION TERMINE ==========").

suite33(o) :-   annuler(3) .
suite33(n) :-   write("==========  ANNULATION TERMINE ==========").

suite34(o) :-   annuler(4) .
suite34(n) :-   write("==========  ANNULATION TERMINE ==========").

suite35(o) :-   annuler(5) .
suite35(n) :-   write("==========  ANNULATION TERMINE ==========").

lecture :-      write("     ?   "),
                readln(T),
                not(dernier(T)),
                assertz(élément(T)),
                lecture
                ;
                dernier(T),
                write("================ liste complétée ==============="),
```

```
                !.

dernier(X) :-   X="fin" .

effacer(L) :-   L>0,
                retract(élement(_)),
                L1=L-1,
                effacer(L1).

effacer(0) .

/*************************************************************************
**************************************************************************/
écrire([],_).
écrire([T|Q],L) :-
                longueur(Q,L1),
                NUM=L-L1,
                write("\n    ",NUM,"    ",T," ."),
                écrire(Q,L).

bon_itinéraire(O,D,ESS,NUM) :-
                recherche_itinéraire(O,D,[O],I,_),
                not(itinéraire_retenu(O,D,I,_)),
                itinéraire(O,D,I,ESS),
                assertz(itinéraire_retenu(O,D,I,NUM)),
                NUM2=NUM+1,
                bon_itinéraire(O,D,ESS,NUM2),
                !.

bon_itinéraire(_,_,_,NUM) :-
                NUMO=NUM-1 ,nl,nl,
                write("      ",NUMO,"   itinéraires retenus .").

itinéraire(_,_,_,[]).
itinéraire(DEP,DEST,I,[T|Q]) :-
                satisfait(DEP,DEST,[DEP|I],T),
                itinéraire(DEP,DEST,I,Q).

indice_importance(O,D,I,COEFF) :-
                itinéraire_retenu(O,D,I,_),
                importance(O,D,I,COEFF).

importance(O,D,I,COEFF) :-
                importance_1(O,D,I,C1),
                importance_2(O,D,I,C2),
                liste_critères_essentiels(LC),
                longueur(LC,L),
                COEFF=3*L+C1+C2.

importance_1(O,D,ITIN,VAL):-
                findall(T,critère_important(T),IMP),
                évalue_1(O,D,ITIN,IMP,VAL).

test :-         itinéraire_retenu(_,_,_,_),!
                ;
                write("      Aucun itinéraire ne satisfait l'ensemble des ",
                      "\n      critères essentiels .",
                      "\n                    PRESSEZ <ENTER>"),nl,readln(_),
                menu_principal .

évalue_1(_,_,_,[],0) .

évalue_1(O,D,X,[H|Q],V) :-
                test_1(O,D,X,H,V1),
                évalue_1(O,D,X,Q,V2),
                V=V1+V2.
```

```
test_1(O,D,I,C,V) :-
                satisfait(O,D,I,C),
                assertz(critère_imp_satisfait(C)),
                V=2
                ;
                not(satisfait(O,D,I,C)),
                V=0 .

importance_2(O,D,ITIN,VAL):-
                findall(T,critère_facultatif(T),FAC),
                évalue_2(O,D,ITIN,FAC,VAL).

évalue_2(_,_,_,[],0) .

évalue_2(O,D,X,[H|Q],V) :-
                test_2(O,D,X,H,V1),
                évalue_2(O,D,X,Q,V2),
                V=V1+V2.

test_2(O,D,I,C,V) :-
                satisfait(O,D,I,C),
                assertz(critère_fac_satisfait(C)),
                V=1
                ;
                not(satisfait(O,D,I,C)),
                V=0 .

temps_total(O,D,[O],I,TEMPS) :-
                itinéraire_temps(O,D,[O],I,TPARC),
                attente([O|I],TATT),
                TEMPS=TPARC+TATT.

itinéraire_temps(O,O,_,[],0).
itinéraire_temps(O,D,V,[Z|ITIN],TPARC):-
                tronçon(O,Z,_,_,_,_,_),
                not(appartient(Z,V)),
                temps(O,Z,T1),
                itinéraire_temps(Z,D,[Z|V],ITIN,T2),
                TPARC=T1+T2.

itinéraire_transfert(O,D,I,L) :-
                plus_court_itinéraire(O,D,I,L),
                ligne(_,STOP,_),
                appartiennent(I,STOP),
                appartient(O,STOP),!.

/********** DETERMINATION DES TOUS LES ITINERAIRES POSSIBLES **********
*                                                                      *
*   recherche_itinéraire ( origine , destination ,' liste des noeuds  *
*                          visités , liste des noeuds de l'itinéraire  *
*                          possible , longueur de l'itinéraire ) .     *
*                                                                      *
***********************************************************************/

recherche_itinéraire(O,D,V,[Z|ITIN],LONG) :-
                tronçon(O,Z,_,LONG1,_,_,_) ,
                not(appartient(Z,V)) ,
                recherche_itinéraire(Z,D,[Z|V],ITIN,LONG2),
                LONG = LONG1 + LONG2.

recherche_itinéraire(O,O,_,[],0) .

spécifier([]) .
spécifier([T|Q]) :-
                remplir(T),
```

```
            - spécifier(Q) .

remplir("imposer des contraintes à l'itinéraire") :-
       makewindow(51,12,12,"  CONTRAINTES  ",0,0,25,60),nl,
       write(" Quelle contrainte voulez-vous imposer ?",
             "\n\n   1 interdiction de passer par un/des noeud(s) ?",
             "\n   2 obligation de passer par un/des noeud(s) ?",
             "\n   3 distance maximale de parcours ?",
             "\n   4 temps maximum de parcours ?",
             "\n\n      ENTREZ UN PAR UN LES NUMEROS DE VOTRE",
             "\n        CHOIX . A LA FIN TAPEZ 0 ."),nl,
       makewindow(52,18,18,"  CONTRAINTES  ",0,65,25,15),nl,
       entrée_5 .

remplir("bonne qualité du paysage") :-
       makewindow(53,12,12,"  PAYSAGES  ",0,0,25,60),nl,
       write(" Quel paysage désirez-vous ?",
             "\n\n   1 résidentiel ?",
             "\n   2 industriel ?",
             "\n   3 administratif ?",
             "\n   4 naturel ?",
             "\n\n      ENTREZ UN PAR UN LES NUMEROS DE VOTRE",
             "\n        CHOIX . A LA FIN TAPEZ 0 ."),nl,
       makewindow(54,18,18,"  PAYSAGES  ",0,65,25,15),nl,
       entrée_6 .

remplir("éliminer les transferts") .

entrée_5 :-    write("  ?   "),
               readint(E),
               les_contraintes(A),
               appartient(E,A),
               assertz(contrainte(E)),
               entrée_5,!
               ;
               E=0,
               findall(X,contrainte(X),LC),nl,
               write("   Les contraintes imposées sont :  ",LC,
                     "\n\n PRESSEZ <ENTER> "),nl,
               readln(_),
               longueur(LC,LENGTH),
               clear(LENGTH),
               assertz(liste_contraintes(LC)),!
               ;
               write("   ERREUR ! ","\n   Numéro invalide , Recommencez ."),
               nl, entrée_5 .

entrée_6 :-    write("  ?   "),
               readint(E),
               les_paysages(B),
               appartient(E,B),
               assertz(paysage(E)),
               entrée_6,!
               ;
               E=0,
               findall(Y,paysage(Y),LP),nl,
               write("   Liste des paysages exigés :  ",LP,
                     "\n\n PRESSEZ <ENTER>" ),nl,
               readln(_),
               longueur(LP,LENGTH),
               nettoyer(LENGTH),
               assertz(liste_paysages(LP)),!
               ;
               write("   ERREUR ! ","\n   Numéro invalide , Recommencez ."),
               nl, entrée_6 .
```

```
clear(0) .
clear(X) :-     X>0,
                retract(contrainte(_)),
                X1=X-1,
                clear(X1) .

nettoyer(0) .
nettoyer(X) :- X>0,
                retract(paysage(_)),
                X1=X-1,
                nettoyer(X1) .

interpréter(_,_,_,[]) .
interpréter(O,D,ITIN,[T|Q]) :-
                interprète(O,D,ITIN,T),
                interpréter(O,D,ITIN,Q).

vérifier(_,_,_,[]) .
vérifier(O,D,ITIN,[T|Q]) :-
                vérifie(O,D,ITIN,T),
                vérifier(O,D,ITIN,Q) .

satisfait(O,D,ITIN,"imposer des contraintes à l'itinéraire") :-
                liste_contraintes(K),
                interpréter(O,D,ITIN,K).

satisfait(O,D,ITIN,"bonne qualité du paysage") :-
                liste_paysages(K),
                vérifier(O,D,ITIN,K).

satisfait(_,_,ITIN,"éliminer les transferts") :-
                ligne(_,LIST,_),
                appartiennent(ITIN,LIST).

vérifie(_,_,ITIN,1) :-
                tronçon(X,Y,_,_,_,_,"résidentiel"),
                appartiennent([X,Y],ITIN).

vérifie(_,_,ITIN,2) :-
                tronçon(X,Y,_,_,_,_,"industriel"),
                appartiennent([X,Y],ITIN).

vérifie(_,_,ITIN,3) :-
                tronçon(X,Y,_,_,_,_,"administratif"),
                appartiennent([X,Y],ITIN).

vérifie(_,_,ITIN,4) :-
                tronçon(X,Y,_,_,_,_,"naturel"),
                appartiennent([X,Y],ITIN).

interprète(_,_,ITIN,1) :-
                makewindow(11,15,15," NOEUDS INTERDITS ",0,30,25,50),
                liste_noeuds_interdits(INT),
                write("    Liste des noeuds interdits ",
                      "\n      ",INT),nl, readln(_),
                not(appartiennent(INT,ITIN)),!
                ;
                not(liste_noeuds_interdits(_)),
                write(" Nommez les noeuds interdits , un par un .",
                      "\n\n A la fin tapez 0 ."),nl,
                entrée_1,
                findall(X,noeud_interdit(X),INT),
                assertz(liste_noeuds_interdits(INT)),
                write("    Liste des noeuds interdits ",
                      "\n      ",INT),nl, readln(_),
                not(appartiennent(INT,ITIN)).
```

```
interprète(_,_,ITIN,2) :-
                makewindow(12,15,15," NOEUDS OBLIGATOIRES ",0,30,25,50),
                liste_noeuds_obligatoires(OBL),
                write("    Liste des noeuds obligatoires : ",
                      "\n       ",OBL,"    PRESSEZ <ENTER>"),nl, readln(_),
                appartiennent(OBL,ITIN),!
                ;
                not(liste_noeuds_obligatoires(_)),
                write(" Nommez les noeuds obligatoires , un par un .",
                      "\n\n A la fin tapez 0 ."),nl,
                entrée_2,
                findall(X,noeud_obligatoire(X),OBL),
                assertz(liste_noeuds_obligatoires(OBL)),
                appartiennent(OBL,ITIN).

interprète(O,D,ITIN,3) :-
                makewindow(13,15,15," LONGUEUR MAXIMALE ",0,30,25,50),
                long_max(L),
                recherche_itinéraire(O,D,[O],ITIN,Long),
                Long<=L ,!
                ;
                not(long_max(_)),
                write(" Longueur maximale tolérée ( km ) ?  "),
                readreal(L),
                assertz(long_max(L)),
                recherche_itinéraire(O,D,[O],ITIN,Long),
                Long<=L .

interprète(_,_,ITIN,4) :-
                makewindow(14,15,15," TEMPS MAXIMUM ",0,30,25,50),
                temps_max(T),
                temps_nécessaire(ITIN,TN),
                attente(ITIN,ATT),
                Temps=TN+ATT,
                Temps<=T ,
                write(Temps," et ",T),readln(_),!
                ;
                not(temps_max(_)),
                write(" Temps de voyage maximal toléré (minutes) ?   "),
                readreal(T),nl,
                assertz(temps_max(T)),
                temps_nécessaire(ITIN,TN),
                attente(ITIN,ATT),
                Temps=TN+ATT,
                Temps<=T ,
                write(Temps," et ",T),readln(_).

entrée_1 :-     write("   ?  "),
                readln(N),nl,
                artère(_,Liste),
                appartient(N,Liste),
                assertz(noeud_interdit(N)),
                entrée_1
                ;
                N="0" ,!
                ;
                write(" Ce noeud n'appartient pas au réseau .",
                      "\n\n Essayez encore ."),nl,
                entrée_1 .

temps_nécessaire([_],0).
temps_nécessaire([X|[Y|Q]],T) :-
                temps(X,Y,T1),
                temps_nécessaire([Y|Q],T2),
                T=T1+T2,write(T).
```

```
temps(X,Y,T) :-
                tronçon(X,Y,_,D,V,Bus,_),
                not(vide(Bus)),
                T=60*D/V , ! .

temps(X,Y,T) :-
                tronçon(X,Y,_,D,_,Bus,_),
                vide(Bus),
                T=60*D/5 .

vide([]).

attente([X|[Y|Q]],A) :-
                tronçon(X,Y,_,_,_,BUS,_),
                not(vide(BUS)),
                attendre(BUS),
                findall(L,fréquence(_,L),FREQ),
                plus_petit(FREQ,W1),
                A=W1/2 ,
                !
                ;
                attente([Y|Q],A),! .

attente(["_"],0) .

attendre([]).
attendre([X|Q]) :-
                ligne(X,_,F),
                not(fréquence(X,_)),
                assertz(fréquence(X,F)),
                attendre(Q)
                ;
                attendre(Q).

entrée_2 :-     write("   ?  "),
                readln(N),nl,
                artère(_,Liste),
                appartient(N,Liste),
                assertz(noeud_obligatoire(N)),
                entrée_2
                ;
                N="0" ,!
                ;
                write(" Ce noeud n'appartient pas au réseau .",
                      "\n\n Essayez encore ."),nl,
                entrée_2 .

sous_menu(O,D,V,I) :-
                makewindow(7,10,10," AUTRES RENSEIGNEMENTS ",5,5,20,70),
                write("\n Quelles informations additionnelles voulez-vous ",
                "avoir : "),nl,nl,
                write("              1 longueur de parcours . ",
                      "\n                2 temps total de parcours .",
                      "\n                3 liste des critères satisfaits . ",
                      "\n                4 aucune .",
                      "\n\n       ENTREZ LE NUMERO DE VOTRE CHOIX ."),nl,nl,
                readint(ADD),
                informer(ADD,O,D,V,I).

informer(1,O,D,V,ITIN) :-
                recherche_itinéraire(O,D,V,ITIN,L),
                write(" La longueur de parcours est de :",L," km ."),
                nl,nl,
                write("          PRESSEZ <ENTER> "),nl,
                readln(_),
```

```
                sous_menu(O,D,V,ITIN) .

informer(2,O,D,V,ITIN) :-
                temps_nécessaire(ITIN,T),
                write("\n   Temps total de voyage =  ",T," min .",
                      "\n\n        PRESSEZ <ENTER> "),nl,
                readln(_) ,
                sous_menu(O,D,V,ITIN).

informer(3,O,D,V,ITIN) :-
                write(" Tous les critères essentiels sont satisfaits :",
                      "\n Ces critères sont :") , nl,nl,
                findall(E,critère_essentiel(E),ESS),
                imprime(ESS),nl,nl,
                write(" Les critères importants satisfaits sont :")
                ,nl,nl,
                findall(Y,critère_imp_satisfait(Y),IMP),
                imprime(IMP),nl,
                write(" Les critères facultatifs satisfaits sont :")
                ,nl,
                findall(Z,critère_fac_satisfait(Z),FAC),
                imprime(FAC),
                write("     PRESSEZ <ENTER> ") , readln(_),
                sous_menu(O,D,V,ITIN) .

informer(4,_,_,_,_) :- menu_principal .

/******************  PONDERATION DES CRITERES   ***********************
*                                                                       *
************************************************************************/

pondération([]).
pondération([T|Q]) :-
                write("Quel poids attribuez-vous au critère :",
                      "\n",T," ?"),
                readreal(Poids),nl,
                classe(T,Poids),
                pondération(Q).

classe(X,Poids) :-
                Poids=3,
                assertz(critère_essentiel(X)),
                !
                ;
                Poids=2,
                assertz(critère_important(X)),
                !
                ;
                Poids=1,
                assertz(critère_facultatif(X)),
                !
                ;
                !.

échelle :-      write(" Les critères d'évaluation d'un ",
                       "\n itinéraire sont pondérés selon",
                       "\n l'échelle suivante :",
                      "\n\n POIDS       SIGNIFICATION",
                      "\n -----       ------------",
                      "\n\n   3           critère essentiel .",
                      "\n   2          critère important .",
                      "\n   1          critère facultatif .",
                      "\n   0          critère inutile .",
                      "\n\n     PRESSEZ <ENTER>"),readln(_).

/*********************   INFORMATIONS  PARTIELLES   *********************
```

```
*                                                                    *
*      Les informations partielles disponibles sont    :             *
*                                                                    *
*      1 Déterminer la ligne de transport à emprunter .              *
*      2 Déterminer la longueur d'un parcours .                      *
*      3 Déterminer la vitesse moyenne sur une rue .                 *
*      4 Déterminer les zones saturées ( congestionnées ) .          *
*      5 Déterminer le paysage local le long d'un tronçon .          *
*      6 Déterminer l'adresse d'un monument .                        *
*                                                                    *
**********************************************************************/

menu_2 :-       removewindow,
                makewindow(3,15,15,"  INFORMATIONS PARTIELLES  ",2,15,20,55),
                write("     Informations partielles disponibles :",
                      "\n 1  Déterminer la ligne de transport à emprunter .",
                      "\n 2  Déterminer la longueur d'un parcours .",
                      "\n 3  Déterminer la vitesse moyenne sur un tronçon .",
                      "\n 4  Déterminer les zones saturées .",
                      "\n 5  Déterminer le paysage local d'un tronçon .",
                      "\n 6  Déterminer l'adresse d'un monument .",
                      "\n\n                  TAPEZ LE NUMERO DE VOTRE CHOIX ",
                      "\n                        OU 0 POUR QUITTER ."),
                nl,
                readint(INFO),
                exécute(INFO).

exécute(1) :-   write("   Point de départ ?   "),
                readln(DEP),
                write("   Point de destination ?   "),
                readln(DEST),
                cherche_bus(BUS,DEP,DEST),
                write("   Prenez l'autobus  : ",BUS," .",
                      "\n\n     PRESSEZ <ENTER>"),nl,
                readln(_),nl,
                menu_2 .

exécute(2) :-   makewindow(20,5,5," LONGUEUR DE PARCOURS ",0,0,25,80),
                write(" Point de départ ? "),nl,
                readln(ORIG),valide(ORIG,DEP) ,
                write(" Point de destination ?") ,nl,
                readln(ARR),valide(ARR,DEST),
                findall(L,recherche_itinéraire(DEP,DEST,[DEP],_,L),LONG),
                plus_petit(LONG,K),
                recherche_itinéraire(DEP,DEST,[DEP],ITIN,K),
                write(" \n\n La distance minimale de parcours est de : ",
                      K," km ."),nl,
                write("\n selon le chemin : ",ITIN," ."),nl,
                write("\n\n     PRESSEZ  <ENTER>"),nl,
                readln(_),
                menu_2 .

exécute(3) :-   makewindow(30,5,5," VITESSE MOYENNE ",0,0,25,80),
                write(" De quelle rue s'agit-il ? "),nl,
                readln(RUE),nl,nl,
                tronçon(_,_,RUE,_,VIT,_,_) ,
                write(" La vitesse moyenne sur la rue :  ",RUE,
                      "\n                              est de :  ",VIT," km .",
                      "\n\n\n          PRESSEZ <ENTER> "),nl,
                readln(_),
                menu_2 ,!
                ;
                write("\n   Rue introuvable !   Essayez encore .",
                      "\n\n\n               PRESSEZ <ENTER>  "),
                readln(_),
                menu_2.
```

```
exécute(4) :-  makewindow(40,5,5," ETAT DE SATURATION ",0,0,25,80),
               write(" Voulez-vous avoir : ",
                     "\n\n     1 la liste de tous les liens saturés ?",
                     "\n     2 l'état d'un lien particulier ?",
                     "\n\n          ENTREZ LE NUMERO DE VOTRE CHOIX"),nl,nl,
               readint(NUM),
               état_de_saturation(NUM),
               menu_2.

exécute(5) :-  makewindow(50,5,5," PAYSAGE LOCAL ",0,0,25,80),
               write(" Nom de la rue ?"),nl,
               readln(RUE),nl,
               artère(RUE,LIST),
               write(" Un tronçon particulier o/n ?"),nl,
               readln(REP),nl,
               exécuter(REP,LIST),nl,nl,
               write("          PRESSEZ <ENTER>"),nl,
               readln(_),
               menu_2
               ;
               write("\n   Nom de rue incorrect ! Recommencez .",
               "\n\n          PRESSEZ <ENTER>"),nl,
               readln(_),
               menu_2.

exécute(6) :-  makewindow(60,5,5," ADRESSES ",0,0,25,80),
               write(" Nom du monument ?  "),
               readln(MONUM),nl,nl,
               monument(MONUM,Liste),
               imprime_6(Liste),nl,
               write("          PRESSEZ <ENTER>"),nl,
               readln(_),!,
               menu_2
               ;
               write("   Ce monument n'existe pas .",
                     "\n\n       PRESSEZ <ENTER>" ),readln(_),
               menu_2.

exécute(0) :-  menu_principal .

état_de_saturation(1) :-
               write(" De quelle rue s'agit_il ?     "),
               readln(RUE),
               artère(RUE,_),nl,nl,
               write(" Les liens saturés sont :",
                     "\n          LIEN            RUE            % ",
                     "\n     ============     =======         =====",),nl,nl,
               imprime_1(RUE) ,nl,
               write("          PRESSEZ <ENTER>"),nl,
               readln(_)
               ;
               write(" Nom de rue incorrect ! ",
                     "\n\n             PRESSEZ <ENTER>"),readln(_).

état_de_saturation(2) :-
               write(" De quelle rue s'agit-il ?  "),
               readln(RUE),nl,
               artère(RUE,LIST),
               write(" Premier noeud du tronçon ?"),readln(N1),
               appartient(N1,LIST),
               write(" Deuxième noeud du tronçon ?"),readln(N2),
               appartient(N2,LIST),
               imprime_2(RUE,LIST,N1,N2),
               write("          PRESSEZ <ENTER>"),nl,
               readln(_)
               ;
```

```
                write(" Nom de rue incorrect ou tronçon mal défini ! ",
                      "\n\n                PRESSEZ <ENTER>"),readln(_).

imprime_1(RUE) :-
                saturé(RUE,N1,N2,POURCENT),
                writef("%10",N1),
                writef("%10",N2),
                writef("%12",RUE),
                writef("%12",POURCENT),
                nl,!.

imprime_1(RUE) :-
                write("Pas de saturation sur la rue : ",RUE," .").

imprime_2(RUE,LIST,N1,N2) :-
                saturé(RUE,N1,N2,POURCENT),
                write(" Rue : ",RUE,"\n Le lien : ",N1," ",N2,
                      "\n est saturé à : ",POURCENT," % ."),nl,
                !
                ;
                write(" Ce lien n'est pas saturé .").

exécuter(o,_) :-
                write(" Premier noeud ?  "),readln(N1),nl,
                write(" Deuxième noeud ?  "),readln(N2),nl,nl,
                tronçon(N1,N2,RUE,_,_,_,PAYSAGE),
                write(" Le paysage entre ",N1,"  et  ",N2,
                      "\n est de caractère :  ",PAYSAGE," ."),nl
                ;
                tronçon(N2,N1,RUE,_,_,_,PAYSAGE),
                write(" Le paysage entre ",N1,"  et  ",N2,
                      "\n est de caractère :  ",PAYSAGE," .")
                ;
                write("   Tronçon introuvable ! Essayez encore .").

exécuter(n,LIST) :-
                imprimer(LIST).

imprimer([_]).
imprimer([N1,N2|Q]) :-
                tronçon(N1,N2,_,_,_,_,PAYSAGE),
                write(" Le paysage entre ",N1,"  et  ",N2,
                      "\n est de caractère :  ",PAYSAGE," ."),nl,
                imprimer([N2|Q]) .

imprime_6([]).
imprime_6([T|Q]) :-
                artère(RUE,Ln),
                appartient(T,Ln),
                write(" Noeud d'accès : ",T," . Sur la rue :",RUE," .")
                ,nl,
                imprime_6(Q).

cherche_bus(B,O,D) :-
                ligne(B,LISTE,_),
                appartient(O,LISTE),
                appartient(D,LISTE),
                !
                ;
                ligne(B,LISTE1,_),
                appartient(O,LISTE1),
                ligne(_,LISTE2,_),
                appartient(D,LISTE2),
                unies(LISTE1,LISTE2).

unies([H|Q],P) :-
```

```
                appartient(H,P),
                !
                ;
                unies(Q,P).

plus_court_itinéraire(O,D,I,LONG) :-
                findall(L,recherche_itinéraire(O,D,[O],_,L),K),
                plus_petit(K,LONG),
                recherche_itinéraire(O,D,[O],I,LONG).

/************************  REGLES GENERALES  ************************
 *                                                                  *
 *      1. Test d'appartenance d'un élément à une liste .           *
 *      2. Test d'appartenance des éléments d'une liste à une       *
 *         autre liste .                                            *
 *      3. nombre d'éléments ( longueur ) d'une liste .             *
 *      4. Plus grand élément d'une liste .                         *
 *      5. Plus petit élément d'une liste .                         *
 *      6. Impression des éléments d'une liste .                    *
 *                                                                  *
 ********************************************************************/

appartient(X,[X|_]) .
appartient(X,[_|Y]) :-
                appartient(X,Y) .

appartiennent([],_) .
appartiennent([T|Q],H) :-
                appartient(T,H),
                appartiennent(Q,H).

longueur([],0).
longueur([_|Y],Z) :-
                longueur(Y,L),
                Z=L+1 .

plus_grand([X],X).
plus_grand([X,Y|Z],R) :-
                X>=Y,
                plus_grand([X|Z],R)
                ;
                X<Y,
                plus_grand([Y|Z],R).

plus_petit([X],X).
plus_petit([X,Y|Z],R) :-
                X<=Y,
                plus_petit([X|Z],R)
                ;
                X>Y,
                plus_petit([Y|Z],R) .

imprime([]) .
imprime([T|Q]) :-
                write("          ",T," ."),nl,
                imprime(Q).
```