NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

ISBN 0-315-56347-8

Canada

SCRIPTS COMME MÉCANISME DE BASE D'UN SYSTEME EXPERT: UNE EXPÉRIENCE APPROFONDIE

par

Daniel Charlebois

Thèse déposée à
l'École des études supérieures et de la recherche
en vue de l'obtention de la maîtrise ès sciences appliquées
en informatique

Université d'Ottawa

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Je déclare être le seul auteur de cette thèse.

J'autorise l'Université d'Ottawa à prêter cette thèse à d'autres institutions ou individus dans le but de recherches académiques.

Daniel Charlebois

De plus, j'autorise l'Université d'Ottawa de reproduire cette thèse par photocopies ou autrement, en sa totalité ou en parties, à la demande d'autres institutions ou individus dans le but de recherches académiques.

Cette thèse est dédiée à mes amis et à mes parents. Que je puisse un jour leur offrir autant qu'ils m'ont donné.

REMERCIEMENTS

Cette thèse est le produit de l'amitié et du support des personnes qui ont su m'endurer et m'encourager tout au long de son écriture. J'aimerais d'abord remercier mon directeur de thèse, Stanislaw Matwin, qui m'a toujours offert ses conseils, sa sagesse et sa patience malgré mes caprices. Aussi, je remercie le **Centre Canadien de Recherche sur l'Informatisation du Travail** qui m'ont témoigné leur confiance. Finalement, mes parents, qui ont toujours su m'aider quand j'en ressentais le besoin.

RÉSUMÉ

Les systèmes experts deviennent de plus en plus populaires comme outils de travail. Ils réussissent à produire des résultats souvent égaux à ceux des experts humains et parfois mêmes supérieurs.

Aujourd'hui, les obstacles les plus importants au développement de systèmes experts sont la saisie des connaissances ainsi que leurs différentes représentations possibles.

Le but de cette thèse est d'introduire une nouvelle méthodologie pour aider au développement de systèmes experts. En se basant sur des scénarios, des réseaux sémantiques et des censeurs, nous contrôlons les étapes du développement de systèmes experts de façon à réduire l'incertitude associée à la saisie et à la représentation des connaissances.

1. INTRODUCTION

1.1 Objectifs

1.2 Rapport général

1.2.1 Traductions et définitions
1.2.2 Les outils
1.2.3 Le système
1.2.4 La coquille
1.2.5 Les connaissances

1.3 Un exemple de voyage

1.4 Contenu de la thèse

2. LA LITTÉRATURE

2.1 Les systèmes experts

2.1.1 Pourquoi
2.1.2 Un rappel
2.1.3 L'expert
2.1.4 Une taxonomie
2.1.5 Les connaissances
2.1.6 Moteur d'inférence

2.2 Système Université d'Ottawa

2.2.1 Structure des connaissances
2.2.2 La transformation

2.3 Réseaux sémantiques

2.4 Les scénarios
2.4.1 Définition

2.5 Censeurs

3. SYSTEME EXPERT - VOYAGE

3.1 Les politiques

3.2 Système expert
3.2.1 Exemple

3.3 Programmation du système expert

3.3.1 La taxonomie
3.3.1.1 Structures
3.3.1.2 Scénarios et réseaux sémantiques
3.3.1.3 Les types
3.3.1.4 Les concepts

3.3.2 Options de contrôle

3.3.2.1 Défauts
3.3.2.2 L'ordre d'évaluation
3.3.2.3 Questions
3.3.2.4 Chaînes

3.3.3 Les règles

3.3.3.1 Déclenchement
3.3.3.2 L'ordre et le groupement des règles
3.3.3.3 Les censeurs
3.3.3.4 La négation

3.3.4 Prolog

3.3.4.1 Déclenchement
3.3.4.2 Explications

3.3.6 Forces et faiblesses du système

4. CONCLUSIONS

4.1 La méthode

4.1.1 Réseaux sémantiques
4.1.2 Scénarios
4.1.3 Censeurs

4.2 Application de la méthode

4.2.1 Type des connaissances
4.2.2 Accès à l'expertise
4.2.3 Concepts définissables
4.2.4 Scénarios
4.2.5 Contre exemple

4.3 Faiblesses de la méthode

4.4 Avantages de la méthode

4.5 Recherches futures

APPENDICES
BIBLIOGRAPHIES

1. INTRODUCTION

1.1 Objectifs

Dans la vie quotidienne, il est souvent nécessaire de faire appel à des experts pour résoudre des problèmes. Lorsqu'un expert est disponible, les problèmes peuvent être réglés rapidement et efficacement. Si un expert n'est pas disponible, nous devons attendre qu'il y en ait un qui le devienne ou avoir recours à d'autres moyens pour solutionner l'impasse. Une option qui s'offre aujourd'hui est d'imiter le rôle d'un expert à l'aide d'un ordinateur.

Les systèmes experts peuvent offrir des solutions acceptables à certains de ces problèmes. En effet, des systèmes tels que **RHINOS** [Kimura et al. 85] (diagnostique de maux de tête et douleurs faciales), **LOX** [Scarl et al. 85] (détection de défauts pour un système de chargement d'oxygène sur la navette spatiale), et **CHEF** [Hammond 86] (élaboration de plans) démontrent le potentiel de cette technologie. Par contre, jusqu'aux dernières années, les coûts de développement et d'implantation de ce type de système étaient prohibitifs. Aussi, les systèmes experts font l'objet de beaucoup de recherche dans des domaines d'application qui sont d'un intérêt pratique, mais leur usage sur le marché des affaires est encore très restreint.

Nous avons cherché à produire un système qui encouragerait son usage par sa simplicité d'utilisation et sa puissance comme outil de travail. Pour développer un tel système, nous avons tenter de répondre aux questions suivantes:

- quelles combinaisons de matériel/logiciel permettent de développer de tels systèmes,

- comment peut-on faire l'acquisition des connaissances,

- comment représenter l'expertise nécessaire à la solution de ces problèmes,

- peut-on intégrer un système de ce genre aux autres logiciels déjà existant.

De plus, nous voulions développer notre système sur un ordinateur populaire sur le marché à l'aide d'une coquille pour système expert dit "off the shelf". L'usage d'un tel produit nous semblait particulièrement intéressant parce qu'en plus des méthodes de représentation de connaissances, ils fournissent généralement un mécanisme d'explication.

Pour répondre à ces questions nous avons écrit un système expert qui offre de l'aide aux personnes voyageant en service commandé par le gouvernement fédéral. Lorsque quelqu'un part en voyage, il y a certaines politiques à appliquer quant au choix du mode de transport utilisé, le logement à occuper, les dépenses pour les repas, etc. Les usagers d'un tel système ont principalement accès à des micro-ordinateurs alors le système a été développé sur un IBM-AT. Étant donnée la popularité de cet ordinateur et de ses semblables, ce choix semblait justifié. De plus, nous avons trouvé plusieurs coquilles pour systèmes experts disponibles pour ce type d'ordinateur.

En considérant aussi l'usage répandu de systèmes de gestion de bases de données, de chiffriers électroniques et d'autres logiciels existants, ce serait intéressant d'intégrer un système expert à d'autres types d'application. Entre autres, le système développé pour ce projet pourrait chercher dans une base de données l'information dont il a besoin plutôt que de toujours questionner

l'usager.

Les points soulevés ne seront pas tous explorés en profondeur étant donné le temps alloué pour l'écriture d'une thèse de maîtrise. Malgré ceci, ce document pourra servir de point de départ pour débattre ces sujets.

Dans cette thèse, nous donnons un compte rendu des difficultés que nous avons eues à surmonter pendant le transfert de notre conception du système au niveau du développement à l'aide de la coquille qui a été choisie au début du projet (1987). Nous croyons que ces expériences pratiques peuvent être intéressantes pour établir des critères additionnels au choix d'une coquille lors d'un autre exercice de développement de système expert.

1.2 Rapport général

Dans ce document, nous répondons aux trois premières questions que nous nous sommes posés. La dernière, l'intégration de systèmes experts à d'autres types d'applications est, à notre avis, réalisable mais nous ne nous y sommes pas attardés. Nous avons produit un système expert répondant aux questions les plus importantes du domaine d'application. Malgré qu'il ne puisse répondre à tous les cas, son comportement ainsi que sa structure offrent aux développeurs une nouvelle approche au développement de systèmes experts.

1.2.1 Traductions et définitions

La technologie des systèmes experts et la terminologie qui lui est associé existent depuis quelques années déjà. Par contre, en pratique nous nous servons surtout de la terminologie anglaise.

Aussi, **Arity/Expert** a introduit une terminologie anglaise propre à son environnement. Voici quelques traductions qui aideront à situer le lecteur:

Anglais	Français	Arity/Expert
script	scénario/script	---
frame	cadre	concept
slot	tirroir	rôle
slot-filler	valeur	value

1.2.2 Les outils

Dans le cadre de notre application, notre méthode repose sur l'intégration des approches suivantes:

- les scénarios,
- les réseaux sémantiques,
- les censeurs,
- un module d'explication.

Ce qui rend un système informatique alléchant comme outil à des utilisateurs est sa facilité d'utilisation et son potentiel comme outil de travail. Il faut, entre autre, rendre le comportement de l'interface homme-machine le plus semblable possible à la méthode couramment utilisé. C'est pourquoi nous avons créé un système qui repose sur un scénario bien établi dans le système manuel. Un scénario est une suite d'évènements stéréotypés détaillant une situation donnée. Aussi, dans un scénario, nous nous servons de valeurs de défaut intelligent. Par exemple, quand un voyageur prépare un voyage d'affaire et qu'il ne connaît pas les modes de transport qui sont permis en fonction de

la destination, il peut laisser le système lui suggérer le mode de transport approprié.

La qualité des conclusions ou des réponses qu'un système informatique offre à ses utilisateurs est fortement dépendante de la quantité et de la qualité de l'information à la disposition du système. Notre système enregistre l'information dont il dispose dans un réseau sémantique. Nous avons choisi cette structure principalement parce qu'elle permet, non seulement de conserver l'information, mais aussi d'exprimer les relations qui existent entre les entités et les objets décrits.

L'étude des politiques de voyage nous a révélé que détecter une situation anormale était plus facile que de prouver qu'une situation respectait toutes les politiques. Il s'agit qu'une politique ne soit pas respectée pour rejeter une réclamation, tandis qu'il faut que toutes les politiques soient respectées pour approuver une réclamation. [Winston 84] nous a introduit le censeur comme mécanisme de détection d'anomalies. Notre système profite pleinement du potentiel de cet outil.

La coquille pour système expert qui a servi pour développer notre système offre, entre autres utilitaires, un module d'explications. Celui-ci permet à un utilisateur du système de s'informer pourquoi le système demande une question ou encore comment le système est arrivé à une conclusion. Pendant le développement de notre système, nous avons réalisé que ce module ne répondait pas parfaitement à nos besoins. Il présentait à l'utilisateur les texte des règles du système expert en guise d'explication. Pour résoudre ce problème, nous avons créé notre propre module d'explications. En puisant dans le texte du document des politiques de voyage, le système affiche une explication que l'utilisateur est plus apte à comprendre.

Pendant cette recherche, nous avons produit un système expert de type consultant et diagnostique. Il aide aux utilisateurs qui préparent un voyage en service commandé par le gouvernement, à choisir leurs modes de transport, leurs hébergements, etc. Aussi, il assiste un commis financier pour déterminer si toutes les dépenses réclamées par le voyageur sont acceptables.

1.2.3 Le système

Le gouvernement fédéral a publié des politiques de voyage qui indiquent aux personnes voyageant en service commandé quelles dépenses sont acceptables ou non. Entre autre, on y retrouve les démarches à suivre pour obtenir des billets pour se déplacer en avion, la classe de voyage acceptable, etc. Nous leur avons proposé de développer un système expert qui assisterait à un voyageur à préparer son voyage. Aussi, ce système pourrait être un outil important aux personnes qui traitent les réclamations de dépenses.

Pour produire ce système, il fallait d'abord modéliser son fonctionnement. Puisqu'il devra être un assistant à un employé qui doit planifier un voyage, il semblait naturel d'imiter le comportement de l'employé lorsqu'il préparait son voyage. [Schank Abelson 77] a introduit la notion de "script" ou scénario qui est un mécanisme de description de situation. Par exemple, la phrase "dîner au McDonald", nous fait imaginer le déroulement d'évènements suivant:

- attendre en ligne,
- commander,
- payer la commande,
- s'asseoir,
- manger,

- partir.

C'est le déroulement typique des activités d'une personne qui mange dans un restaurant à service rapide. Nous avons défini un scénario pour les personnes voyageant en service commandé par le gouvernement:

- aller,
- séjour,
  - répéter pour chaque journée:
    - repas,
    - déplacements,
    - hébergement,
- retour.

Pour contrôler le comportement de notre système et garantir son adhérence au scénario que nous avons défini, un mécanisme de détection de situations anormales ou exceptionelles était nécessaire. Dans notre système nous avons tenté de contrôler le déclenchement des règles par l'utilisation de censeurs. Un censeur est un mécanisme par lequel le déclenchement d'un groupe de règles est permis ou non par la détection d'une situation anormale ou exceptionelle.

Pour compléter la représentation des connaissances du système, nous devons pouvoir conserver toutes les informations nécessaires aux traitements. Il faut aussi rendre explicite les relations existantes entre celles-ci. Par exemple, il existe un lien entre le mode de transport et la destination d'un voyage (i.e.: ce n'est pas acceptable de prendre une voiture pour faire un voyage outre mer). Les réseaux sémantiques permettent justement cette puissance de représentation. Sur un plan plus pratique, la structure d'un tel réseau est réalisé à l'aide de cadres.

Quand le système calcule une valeur à placer dans un tirroir d'un cadre, il faut souvent être capable d'expliquer comment cette valeur a été trouvée. Toutes les coquilles pour systèmes experts génèrent des explications. Celles-ci sont tirées directement des règles écrites dans le système. Par exemple, pour expliquer pourquoi en utilisant un mode de transport commercial, le voyageur doit occuper une place en classe économique, le système fournira la même explication pour le train et l'avion. Ce comportement n'est pas adéquat pour les besoins de notre système puisque l'explication pour le train est différente de l'explication pour l'avion. Pour résoudre ce problème, nous avons écrit un module d'explication qui tire ses réponses directement du texte source de la règlementation.

Aussi, ce système a un caractère interactif, donc ce n'est pas suffisant d'écrire un système expert sur micro-ordinateur pour affirmer qu'il est faisable, il faut en plus qu'il ait un temps réponse acceptable.

1.2.4 La coquille

Ce comportement ainsi que les quelques critères mentionnés ont guidé notre choix vers le **Arity/Expert Shell** comme logiciel de développement. Il semblait offrir plusieurs avantages:

- la coquille fonctionne à base de cadre et de règles,

- c'est un Prolog augmenté, ce qui nous permet d'inclure des appels à des clauses Prologs à partir des règles du système expert,

- ses performances comparées à d'autres coquilles sont meilleures (temps réponse plus rapide),

- peut être compilé et ainsi intégré à d'autres applications,

- permet d'appeler les commandes disponibles au système d'exploitation.

Outre ces avantages, la coquille a aussi imposé certaines contraintes qui seront détaillées sous peu.

1.2.5 Les connaissances

En plus du matériel et du logiciel, pour développer un système expert, il faut avoir accès à l'expertise du domaine d'application. Les règles dans notre système sont principalement le résultat de notre interprétation des sections 3, 4, 5 et 6 du chapitre 370 du Manuel de la Politique Administrative du Conseil du Trésor Canada et des personnes travaillant au Centre Canadien de Recherche sur l'Informatisation du Travail (C.C.R.I.T.). Ces politiques indiquent quelles dépenses sont acceptables pour un employé voyageant en service commandé par le gouvernement. Elles couvrent, entre autres, les dépenses pour les modes de transport, les hébergements, les repas et les faux frais encourues.

Nous avons écrit un système expert qui est le produit du mariage de plusieurs technologies. Nous nous servons de cadres, de règles, de scénario et de censeurs. Puisqu'il peut être compilé, le système expert pourra éventuellement être inclu dans d'autres types d'applications.

1.3 Un exemple de voyage

Dans cette section, on retrouve un extrait d'une consultation avec notre système. Cet extrait illustre ce que fait le système quand il traite le logement pour le voyageur. Il demande d'abord si un logement est nécessaire. Ici, la réponse est oui. Le système réplique en demandant si le type de logement choisi est commercial ou non. Ne sachant pas pourquoi cette information est exigée, l'utilisateur demande des explications et le système répond en affichant le texte de la politique appropriée. L'utilisateur indique au système qu'il n'a pas de reçu pour la dépense. Puisque le système lui répond qu'il n'y aura pas de remboursement, l'utilisateur demande une explication et l'obtient.

Il faut aussi noter que le système fournit pour chaque réponse et conclusion un facteur de confiance. Celui-ci indique le degré de certitude de la validité de la réponse, la valeur "1" indiquant la certitude, la valeur "0" indiquant l'incertitude et la valeur "-1" indiquant la certitude d'invalidité. Quand le système affiche une réponse, elle est toujours accompagnée de son facteur de confiance. Aussi, si la réponse qui est obtenue est une valeur de défaut, ce facteur ne dépassera jamais 0.9 (ceci est une restriction imposée par la coquille).

```
1. Est-ce qu'un logement est requis? y

2. Est-ce un logement?

        1) commercial
        2) non commercial
        3) why ?
        4) unknown

? 3

.5.1.1.0

    Type du logement.
```

L'employé est remboursé des frais réels et raisonnables de logement commercial autorisé par l'employeur. Lorsque ces frais excèdent $13.50 par nuit, il doit fournir des pièces justificatives (voir le sous-article .10.5.4 c)). Il appartient à l'employeur de choisir le logement destiné aux voyageurs. Toutefois, à moins que des raisons de service n'imposent l'utilisation d'un logement spécial, l'employeur doit tenir compte de la demande d'un employé qui veut occuper un logement commercial ou non commercial. Les chambres à un lit doivent être autorisées par l'employeur s'il y en a de libres dans des établissements bien situés et confortables.

RETOUR pour continuer

3. Est-ce un logement?

1) commercial
2) non commercial
3) why ?
4) unknown

? 1

4. Est-ce le M.A.S. qui a fait les réservation?

1) oui
2) non
3) why ?
4) unknown

? 2

5. Quel est le montant du frais de logement-1 30

6. Avez-vous un reçu pour les frais de logement-1

1) oui
2) non
3) why ?
4) unknown

? 2

Le mode de remboursement sera

1) aucun / 0.9

Le(s) règlement(s) suivant(s) n'ont pas été respecté(s):

.5.1.1.0

Reçu pour le logement.

L'employé est remboursé des frais réels et raisonnables de logement commercial autorisé par l'employeur. Lorsque ces frais excèdent $13.50 par nuit, il doit fournir des pièces justificatives (voir le sous-article .10.5.4 c)). Il appartient à l'employeur de choisir le logement destiné aux voyageurs. Toutefois, à moins que des raisons de service n'imposent l'utilisation d'un logement spécial, l'employeur doit tenir compte de la demande d'un employé qui veut occuper un logement commercial ou non commercial. Les chambres à un lit doivent être autorisées par l'employeur s'il y en a de libres dans des établissements bien situés et confortables.

RETOUR pour continuer

Le mode de remboursement sera

1) aucun / 0.9

1.4 Contenu de la thèse

Cette thèse présente une nouvelle méthodologie pour assister au développement de systèmes experts et un système expert qui a été développé en s'en servant.

Dans le chapitre 2, un bref historique des systèmes experts est présentée. Une définition de ce qu'est un système expert est introduite. Il y a une revue d'une taxonomie qui classifie les systèmes experts en plusieurs grandes familles. Aussi, une revue de quelques publications pertinentes à cette recherche.

Dans le chapitre 3, les détails du développement du système expert sont présentés. Pour écrire un système expert, il faut connaître le domaine d'application et les techniques nécessaires à son automatisation. Un survol des politiques de voyage est

présenté ainsi qu'un bref exemple de leur application. Ensuite, nous définissons quel rôle un système expert peut jouer, et finalement le développement lui-même est détaillé.

Dans le dernier chapitre, il y a une revue des différentes technologies qui ont été appliquées; le comment et le pourquoi de leurs utilisations; quels impacts elles ont eu sur le système. Nous tentons d'introduire une critique constructive de notre système et de notre méthodologie. Aussi, nous discutons des recherches futures possibles.

2. LA LITTÉRATURE

2.1 Les systèmes experts

Les systèmes experts suscitent depuis quelques années un intérêt toujours croissant. Dans les sections qui suivent, on retrouve une description de cette technologie ainsi que les différents outils qui ont servi au développement de notre méthodologie et de notre système expert.

2.1.1 Pourquoi

Avant de décrire la technologie présentée dans ce document, une justification de l'utilisation des systèmes experts serait importante. [Waterman 85] la justifie par cinq points.

Le premier point est la permanence des connaissances. Un expert humain doit continuellement se servir de sont expertise sinon la qualité de ses solutions diminue. Un système expert automatisé ne souffre pas de ce handicap. Du moment qu'un système expert a été écrit, son expertise est permanente.

Le deuxième point est la communication de l'expertise. Pour transmettre l'expertise d'un humain à un autre, le processus est long et difficile. Pour communiquer l'expertise d'un système expert, il suffit d'interpréter une simple copie des fichiers de connaissances.

Le troisième point est la documentation. L'expertise humaine est difficile à extraire et à exprimer. Par conséquant, elle est aussi très difficile à documenter et à expliquer. L'expertise codée dans un système expert l'est de façon explicite. Les règles

du système sont exprimées sous une forme permettant une documentation facile. Les règles mêmes accompagnées de quelques explications simples servent à documenter les systèmes.

La consistence est aussi un point important. Un système expert produira toujours les mêmes résultats étant données les mêmes circonstances. L'humeur de l'expert humain peut influencer son jugement et changer les résultats.

Finalement, le coût est un facteur important. L'expert humain est toujours dispendieux. Le système expert a un coût de développement élevé mais les coûts de son opération sont minimes.

2.1.2 Un rappel

L'intelligence artificielle est une science qui veut donner à l'ordinateur un comportement qui serait considéré intelligent si c'était celui d'un être humain.

Au début, pour en arriver à cette fin, les chercheurs ont tenté d'inculquer le processus complexe de la pensée à l'ordinateur. Ils croyaient qu'une méthode générale de solution de problèmes pouvait être programmée. Mais, malgré certains progrès, aucune avance significative n'a été réalisée.

Puisqu'une solution générale ne semblait pas possible, ils se sont ensuite acharnés au développement de formalismes de représentation. Ils pensaient aussi que la définition de techniques de recherche de solutions serait un apport important. Encore, malgré certains résultats intéressants, aucun nouveau formalisme introduit n'a été remarquable.

Ce n'est que vers la fin des années soixante-dix qu'ils réalisèrent que la qualité des solutions aux problèmes dépendait surtout de la quantité et de la qualité des connaissances à la disposition du système. Depuis, les recherches sont centrées sur des techniques d'acquisition de connaissances et de solution de problèmes.

2.1.3 L'expert

Un système expert est un programme qui résoud un problème d'une façon comparable à un expert humain. [Waterman 85] définit un expert humain de la façon suivante:

> Un expert humain est une personne qui, grâce à sa formation et son expérience, peut traiter une situation que les autres ne peuvent pas. Ses connaissances sont vastes et lui permettent de filtrer l'information de façon à ignorer les données non-pertinentes. Aussi, il peut aisément reconnaître un problème comme étant une instance d'un autre qu'il connaît déjà. Les chercheurs ont baptisé expertise les connaissances sous-jacentes au comportement des experts.

2.1.4 Une taxonomie

Dans [Waterman 85], les systèmes experts sont divisés en plusieurs grandes catégories:

Interprétation Inférer la description d'une situation à l'aide de données mesurées,

Prévision	Prévoir les conséquences probables d'une situation donnée,
Diagnostique	Identifier la source d'un problème à l'aide de données observables,
Configuration	Configurer des objets étant données des contraintes,
Plannification	Établir une suite d'actions,
Observation	Comparer des observations aux hypothèses posées,
Prescription	Identifier des problèmes et offrir des alternatives de solution,
Réparation	Exécution de plans pour administrer des prescriptions,
Instruction	Diagnostiquer, contrôler et corriger le comportement d'un étudiant,
Contrôle	Gérer le comportment global d'un système.

Les systèmes experts qui font de l'interprétation servent à décrire une situation à partir de données mesurées, par exemple, la lecture de cadrants dans l'industrie chimique pour évaluer l'état d'un processus. Ce type de système traite l'information brute plutôt qu'une représentation symbolique des données. Ils sont particulièrement susceptibles aux bruits et aux données incomplètes et erronées.

Les systèmes de prévisions essaient d'identifier les conséquences d'actions étant donnée une situation quelconque. Par exemple, un système pourrait prévoir l'état des récoltes sur une ferme étant donnés des ravages causés par des insectes.

Les systèmes de diagnostique servent à la description de situations ou d'états. Il peuvent servir comme assistant à la détection de défaut dans un système informatique. Généralement, les systèmes de diagnostique servent surtout dans la médecine. Entre autres, le système **MYCIN** [Shortliffe 76] sert à diagnostiquer des infections bactérielles de patients hospitalisés.

Les systèmes de configuration aident à faire la conception d'objets à partir de contraintes données. On se sert de ce type de système principalement dans deux domaines: la biologie moléculaire et la conception micro-électronique. Un des systèmes les plus reconnus est **XCON** [McDermott 82] qui sert à configurer les ordinateurs de la famille VAX.

Les systèmes de plannification sont responsables de l'élaboration de plans. Ils décident de toutes les actions à entreprendre avant de les exécuter. Par exemple, **TATR** [Callero Waterman Kipps 84] est une application militaire pour plannifier une attaque aérienne sur un ennemie quelconque.

Les systèmes d'observation observent des comportements et les comparent aux comportements prévus. Par exemple, un système dans le département de soins intensifs dans un hopital peut surveiller les lectures cardiaques et s'il y a un comportement anormal, le signaler. Un système existant est **REACTOR** [Nelson 82], il surveille un réacteur nucléaire et signale s'il y a un potentiel d'accident.

Les systèmes de prescription offrent des solutions à des problèmes donnés. Il peuvent, par exemple, suggérer le type d'entretient nécessaire pour réparer des cables téléphoniques défectueux. **ONCOCIN** [Tsuji Shortliffe 83] est un système de prescription qui aide au traitement chimiothérapie pour des patients atteints de cancer.

Les systèmes de réparation proposent un plan pour administrer une prescription. Très peu de ces systèmes existent à cause de la complexité réelle des réparations.

Les systèmes d'instruction servent à diagnostiquer et à corriger les comportements d'étudiants. Ces systèmes développent un modèle des connaissances d'un étudiant et déterminent comment elles sont utilisées. Ils détectent les faiblesses des étudiants et plannifient une suite d'actions pour les réduire. Le système **GUIDON** [Clancey 79] enseigne aux étudiants en médecine à choisir des traitements pour des patients pris d'infections bactérielles.

Les systèmes de contrôle gèrent le comportement global d'un système. La gestion de la manufacture et de la distribution de systèmes informatiques en est un exemple. **VM** [Shortliffe Fagan 82] contrôle les soins que reçoivent des patients dans les soins intensifs après avoir subi une intervention chirurgicale.

Notre système peut être classé dans deux groupes. Il est un assistant à la planification quand l'employé prépare son voyage. C'est aussi un système de diagnostique. Il indique à la personne traitant la réclamation de dépenses lesquelles sont non-autorisées et pourquoi.

2.1.5 Les connaissances

La tâche la plus importante pendant le développement d'un système expert est l'acquisition de connaissances. Il s'agit d'extraire l'expertise, de la représenter sous une forme que l'ordinateur puisse traiter et l'organiser de façon à simplifier son utilisation.

Dans un système expert, l'expression des connaissances se fait généralement à l'aide de deux représentations: des faits et des règles (ici, à ne pas confondre la règle en tant que connaissance et la règle en tant que mode de fonctionnement du système expert). Un fait est l'affirmation d'une propriété d'un concept. Une règle infère des nouveaux faits à partir de ceux qui sont connus. En voici un exemple:

* fait -> **l'ordinateur numéro 5 est un IBM.**

* règle -> **si l'ordinateur est un IBM alors il est fiable.**

Dans le type de système que nous avons développé, les faits sont enregistrés dans un réseau sémantique défini à partir de cadres. Les règles sont écrites de façon à manipuler les faits dans le réseau sémantique.

Les règles sont des constructions [si <condition> alors <action>] où la condition est l'antécédent de la règle et l'action est le conséquent. Dans une règle, lorsque la partie <condition> est vérifiée, la partie <action> est exécutée. L'action peut être la déclaration d'un nouveau fait ou même l'impression de certains résultats. Remarquez que ce type de représentation indique ce qui doit être fait et non comment le faire.

Un réseau sémantique est un réseau de noeuds reliés par des arcs. Chaque noeud correspond à un concept et les arcs représentent des relations qui existent entre les concepts. Les concepts sont des entités (e.g.: personne, dieu, etc.), des objets (e.g.: table, chaise, etc.) ou des idées (e.g.: justice). Un ensemble de propriétés distinguent un concept d'un autre par leurs valeurs. Par exemple, nous identifions un être humain par son sexe, sa race, son nom etc. Ici, le sexe est une propriété et ses valeurs possibles sont homme et femme. De plus, une propriété peut être un autre concept. Par exemple, une voiture a un moteur qui lui a des propriétés (cylindré, carburateur ou injection, etc).

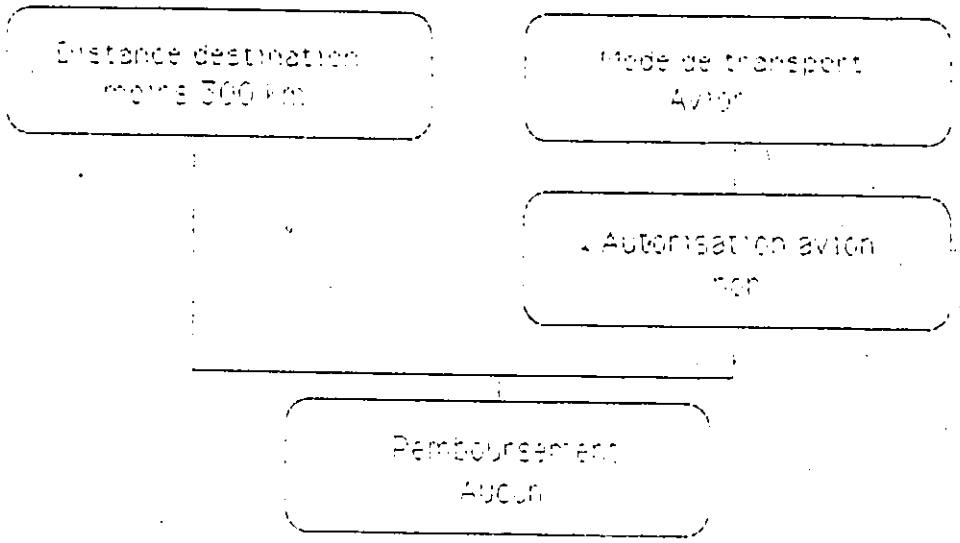
Dans notre système, chaque concept est représenté par un cadre et les propriétés sont conservées dans les tirroirs du cadres.

2.1.6 Moteur d'inférence

Le moteur d'inférence est le mécanisme qui se sert des connaissances. Il décide comment interpréter les connaissances est comment appliquer les règles. Lorsque nous nous servons d'une coquille pour système expert, le moteur y est généralement inclus. Si le système expert est écrit dans un autre langage (e.g.: LISP), il faut écrire le moteur d'inférence nous même.

L'antécédent et le conséquent d'une règle peuvent se référer aux différents tirroirs des cadres. Puisqu'un fait exprime la valeur d'un à placer dans un tirroir, une règle s'unifie avec un fait lorsque les deux affectent le même tirroir. C'est à partir de ce type de mécanisme que des nouvelles connaissances peuvent être inférées.

Le moteur d'inférence doit consulter la base de connaissances pour unifier les faits et les règles. Lorsque l'unification d'une règle et des faits réussit, la partie action est exécutée. On dit alors que la règle a été déclenchée. En ce sens, les nouveaux faits exprimés dans les conséquents des règles peuvent servir d'antécédents aux autres règles. Dans la figure 1, on peut voir comment une suite de nouveaux faits permettent de conclure qu'il n'y a aucun remboursement quand le voyageur n'a pas l'autorisation de prendre un mode de transport qui n'est pas normalement utilisé pour un déplacement de courte distance. Cette représentation s'appelle un réseau d'inférence.



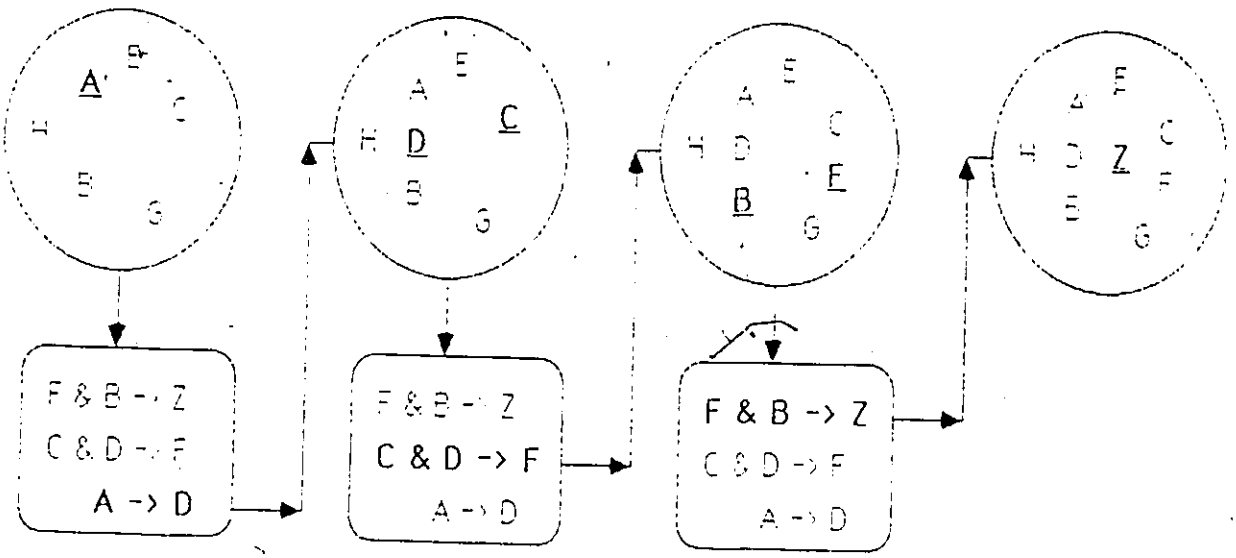
Réseau d'inférence

Figure 1

Il est important de noter que ce genre de structure est nécessaire pour qu'un système puisse justifier une décision quelconque. Les utilisateurs sont souvent à la recherche d'explications pour déterminer pourquoi une requête a été rejetée.

En se servant de la réseau d'inférence créée pendant une consultation, le système peut générer une explication ou en donner une fournie par les développeurs.

Il y a deux façons que le moteur crée des réseaux d'inférence. Dans la figure 2, on illustre le chaînage avant. Le chaînage avant parcourt les règles et exécute la première qui s'unifie aux faits de la base de connaissances. Il répète ce processus jusqu'à ce que plus aucune règle ne peut être déclenchée. Ce type de fonctionnement peut parfois produire beaucoup de résultats intermédiaires qui ne sont pas pertinents à la réponse recherchée. Dans la figure 2, les cercles sont des ensembles de faits. Dans ces ensembles, il y a des faits qui sont soulignés, ceci indique quels faits sont impliqués dans le déclenchement d'une règle.



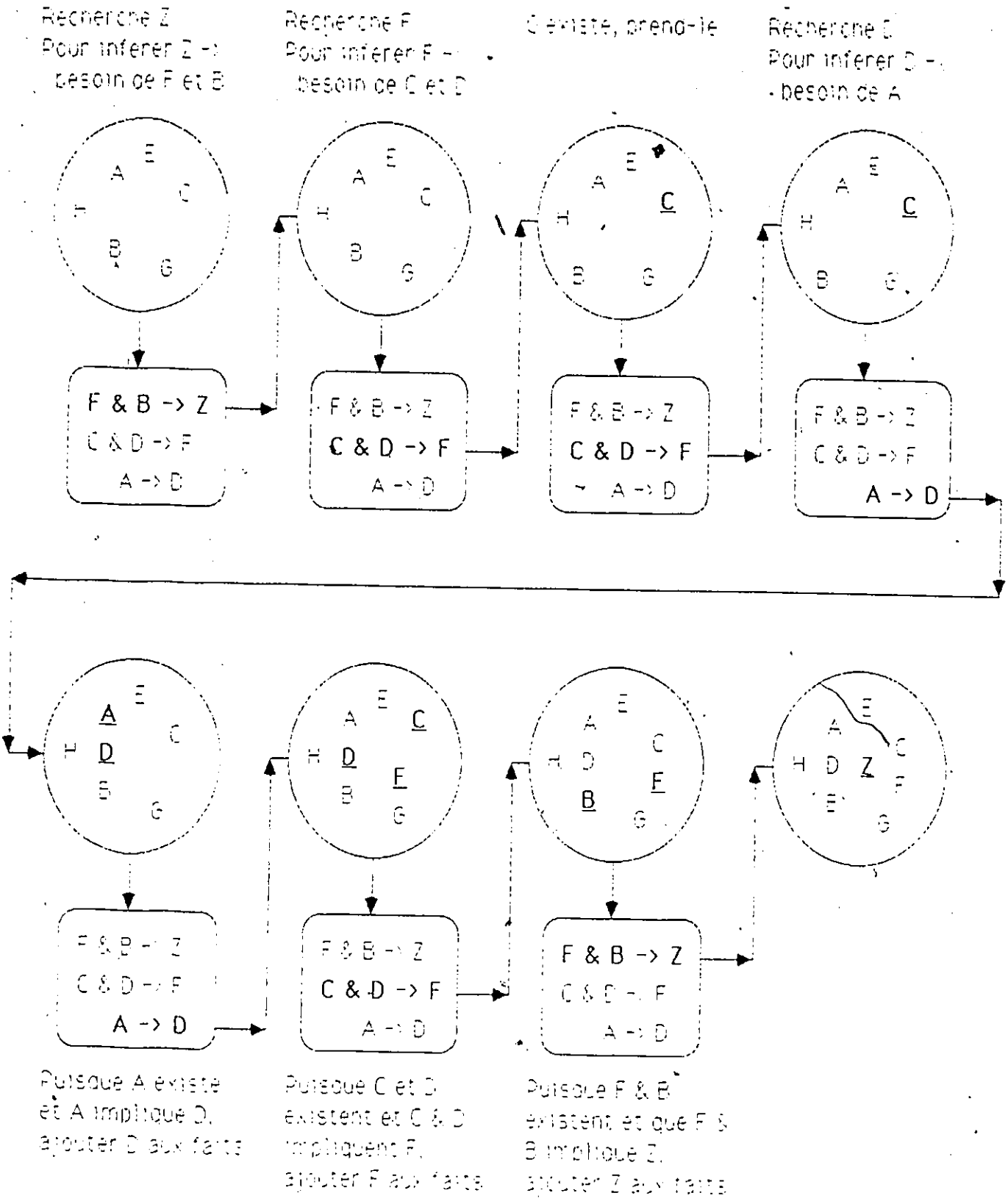
Chaînage avant

Figure #2

Cet exemple est tiré de [Waterman 85]. L'utilisateur cherche à savoir si le fait "Z" peut être inféré. Le système regarde d'abord dans la base de faits pour trouver "Z". Puisqu'il n'y est pas, le système examine les règles une par une dans leur ordre d'entrée et exécute la première dont tous les antécédents sont dans la base de faits. La seule règle qui puisse déclencher est la dernière puisque le fait "A" est dans la base de faits. Il en résulte la déclaration du faits "D" qui est ajouté à la base. Suite à l'inférence de "D", la deuxième règle peut être déclenchée puisque maintenant "C" et "D" sont connus donc "F" est ajouté. Le fait "Z" est ajouté à la base de connaissance puisque "F" et "B" existent. Si nous voulions vérifier l'existance de "Z", ce comportement peut sembler impressionnant, mais en réalité, quand nous travaillons avec une base de connaissances ayant beaucoup de règles, plusieurs résultats peuvent être calculés qui n'ont aucun rapport avec "Z".

Le chaînage arrière fonctionne autrement. Le système pose une première hypothèse et tente de prouver que le fait est vrai en regardant dans la base de faits. S'il s'y trouve, alors l'hypothèse est vérifiée. S'il ne s'y trouve pas, le système cherche dans la base de règles pour trouver une règle dont la conséquence est le fait à vérifier. S'il n'en trouve pas, l'hypothèse est rejetée. S'il en trouve une, il pose autant de nouvelles hypothèses qu'il y a d'antécédents dans la règle et essaie de les prouver à leurs tours. Ce type de moteur d'inférence a l'avantage de ne déclencher que les règles qui sont pertinentes aux hypothèses posées. Ses étapes sont détaillées dans la figure 3 lorsque l'objectif est d'inférer l'existance de "Z".

La première question posée au système est "Z existe-t-il?". Pour répondre, le système regarde d'abord dans la base de faits pour vérifier si "Z" s'y trouve. Puisqu'il n'y est pas, il recherche une règle dont la conséquence est la déclaration du fait



Chaînage arrière

Figure 7

"Z". La première règle est choisie et l'existance de deux nouveaux faits doit être vérifiée. Le premier fait à trouver est "F". Le système regarde dans les faits pour trouver "F". Il n'y est pas, donc il regarde dans les règles et en cherche une dont la conséquence est "F". La deuxième règle a "F" comme conséquence et est déclenchée. Il faut maintenant vérifier l'existance de "C" et "D". "C" est dans la base de faits, il s'agit de trouver "D". Le système regarde à nouveau dans les règles pour une conséquence égale à "D". La dernière règle a "D" comme conséquence et est déclenchée. Il doit vérifier l'existance de "A". Puisque "A" existe, le système déduit que "D" existe. Puisque "C" et "D" existent, "F" existe. Et finalement, puisque "F" et "B" existent, le système conclus que "Z" existe.

La coquille **Arity/Expert Shell** est un système à chaînage arrière. Pour le système voyage, l'hypothèse initiale est que toutes les dépenses d'un voyage sont autorisées. Nous essayons de prouver l'hypothèse en calculant le mode de remboursement de chacune des dépenses.

2.2 Système Université d'Ottawa

[Skuce et al. 85] présentent une nouvelle méthodologie pour le développement de systèmes experts. Ils commencent par introduire une nouvelle taxonomie des systèmes experts qui les divise en trois grandes catégories:

diagnostique	classification d'une situation étant donnée l'information mesurée,
configuration	à partir de document de spécification faire la configuration d'un système

quelconque,

conseiller des système qui donne de l'information d'aide décisionnelle.

Cette taxonomie n'inclus pas autant de catégories que celle présentée dans [Waterman 85]. Il semble aussi évident que tous les types dans [Waterman 85] ne peuvent pas être inclus dans une des ces catégories. Par contre, elle permet de bien situer leur système dans le cadre général des systèmes experts.

[Skuce et al. 85] détaillent le développement d'un système qui est un type de conseiller appelé **KSS** (Knowledge Source System). L'objectif de **KSS** est d'éventuellement remplacer les documents des types suivants:

- politiques,
- spécifications de configuration,
- manuels d'entretient,
- manuels scolaires, médicaux, informatiques,
- textes légaux.

Ces connaissances sont généralement conservées sous forme de textes écrits en langue naturelle. Cette forme n'est pas compréhensible par l'ordinateur dans l'état courant de la science. [Skuce et al. 85] présentent une méthodologie qui permet de transformer ces spécifications sous une forme que l'ordinateur peut comprendre. En plus de l'apport technologique, l'exemple qu'ils ont choisi pour démontrer le potentiel du système est l'encodage des mêmes politiques de voyage qui ont servies pour notre système.

2.2.1 Structure des connaissances

[Skuce et al. 85] présentent une méthode d'acquisition de connaissances. Ils s'attardent aux connaissances déclaratives plutôt que procédurales et les divisent en trois types:

- lexicales,
- faits,
- règles.

Elles sont structurées en unités distinctes compréhensibles par l'humain et la machine. Pour les interpréter, un environnement sémantique est généré pendant la construction du système. Cet environnement contient des déclarations de types et d'autres informations contextuelles nécessaires à l'interprétation des règles.

La représentation interne des connaissances (i.e.: immédiatement compréhensible par l'ordinateur) est sous forme de clauses Prolog. Dans ce type de clause, on retrouve des variables logiques qui tiennent la place de termes Prolog. Le format d'une clause est:

antécédent -> conséquent

qui se lit: "une instance des variables qui rend l'antécédent vrai rend le conséquent vrai". Puisque la forme interne est difficile à lire, une représentation intermédiaire est utilisée. La traduction entre cette représentation, LESK (Language for Exactly Stating Knowledge) et la forme interne est directe. Par exemple:

interne:

```
        reclame( Ee, Dep ) & detient( Ee, Recu )
    -> reclame_avec( Ee, Dep, Recu ).
```

LESK:

Si	Employe reclame Depense
	Employe detient un Recu
alors	Employe reclame la Depense avec Recu

L'usage de majuscule ici indique des variables logiques.

2.2.2 La transformation

La première étape pour transformer le document original en sa représentation interne est la classification de tous les mots du texte dans les catégories appropriées par un opérateur humain. Il affecte une ou plusieurs classes syntaxiques à chaque mot. Ce processus garantit que toutes les constituentes du texte seront considérées.

Il transforme ensuite chaque mot selon sa catégorie syntaxique en une forme interne comme suit:

Noms propres:

Ils sont transformés en une constante Prolog i.e. un atom qui représente une entité fixe et qui peut servir d'argument aux prédicats.

Noms communs:

Ils sont transformés en prédicats à un paramètre s'il n'ont aucun modificateur et en prédicats à **n** paramètres s'ils ont **n-1** modificateurs. Les modificateurs sont des phrases

prépositionelles.

Exemples:

homme	-->	homme(X)
pere du garçon	-->	pere_de(Y, Z), garçon(Z)

Adjectifs:

Il y a deux types d'adjectifs, les premiers sont attributifs et les deuxièmes "syncatégorématiques"(1).

Attributif:

balle rouge --> balle(X) & rouge(X)

Syncatégorématique:

maison de repos --> maison_de_repos(X)

Verbes:

Le système a été écrit pour traiter les documents anglais. Le traitement des suites de verbes est donc beaucoup plus complexe qu'un exemple français puisse illustrer. Par contre, les paramètres qui indiquent la forme et les modifications apportées aux verbes peuvent clarifier comment ils sont traités. Ces paramètres sont:

Par	Nom	Valeurs
1	type	inf, 3ps, ppart, ger, is_adj, is_prep, is_rel
2	typicality	nil, normally, exceptionally
3	modality	nil, may, shall
4	sign	yes, no
5	adv	nil, ou un adverbe
6 etc.	paramètres LESK	

(1) syncatégorématique: ce qualificatif a été défini dans [Skuce et al. 85] et sert à une catégorisation sémantique d'un nom.

1- ce paramètre indique le rôle syntaxique du verbe dans l'environnement **LESK.** Il indique si le contexte introduit une qualité (is_adj) ou autre,

2- ce paramètre indique un genre particulier de modalité par les adverbes "**normalement**" ou "**exceptionnellement**",

3- la modalité d'un verbe en anglais indique l'importance de l'activité,

4- le signe est le marqueur de la présence ou de l'absence d'une négation,

5- un endroit pour conserver un adverbe modifiant le verbe,

6- les paramètres dont **LESK** a besoin.

Quand un texte est traduit en **LESK**, un environnement est généré contenant des restrictions de type pour les variables et d'autres informations nécessaires au fonctionnement du système. Voici un exemple d'une transformation du **LESK** à la forme interne:

Langue naturelle:

"An employee shall be reimbursed the actual and reasonable expenses incurred for commercial accommodation authorized by the employer"

LESK:

Environnement: Exp is an expense
Ee is an employee

Cacc is a commercial accomodation

règle:

if Exp is actual
Exp is reasonable
Ee incurs Exp for Cacc
Cacc is authorized
then Ee shall be reimbursed for Exp

Forme interne:

```
   is_actual( is_adj, nil, nil, yes, nil, Exp )
   is_reasonable( is_adj, nil, nil, yes, nil, Exp )
   incurs_for( 3ps, nil, nil, yes, nil, Ee, Exp, Cacc )
   is_authorized( ppart, nil, nil, yes, nil, Cacc )
-> is_reimbursed_for( ppart, nil, shall, yes, nil, Ee, Exp )
```

Il y a quelques points à soulever en considérant cette méthode, ses résultats et notre système. D'abord, [Skuce et al. 85] ont développé une méthode centrée sur l'acquisition de connaissances. Les auteurs se soucient de détailler la saisie et la représentation des connaissances plutôt que décrire comment le système produit doit se comporter. Pendant le développement de notre système, nous avons surtout cherché à produire un système dont le comportement rendrait son utilisation plus facile par l'usage de scénarios.

Aussi, les systèmes experts créés par cette méthode sont à base de règles dépendant d'un environnement sémantique. Sans cet environnement, les règles ne peuvent pas être interprétées. Notre système est à base de concepts, de réseaux sémantiques et de règles. Le principal avantage de cette représentation est que les relations existantes entre les différentes données sont plus

faciles à exprimer et à exploiter.

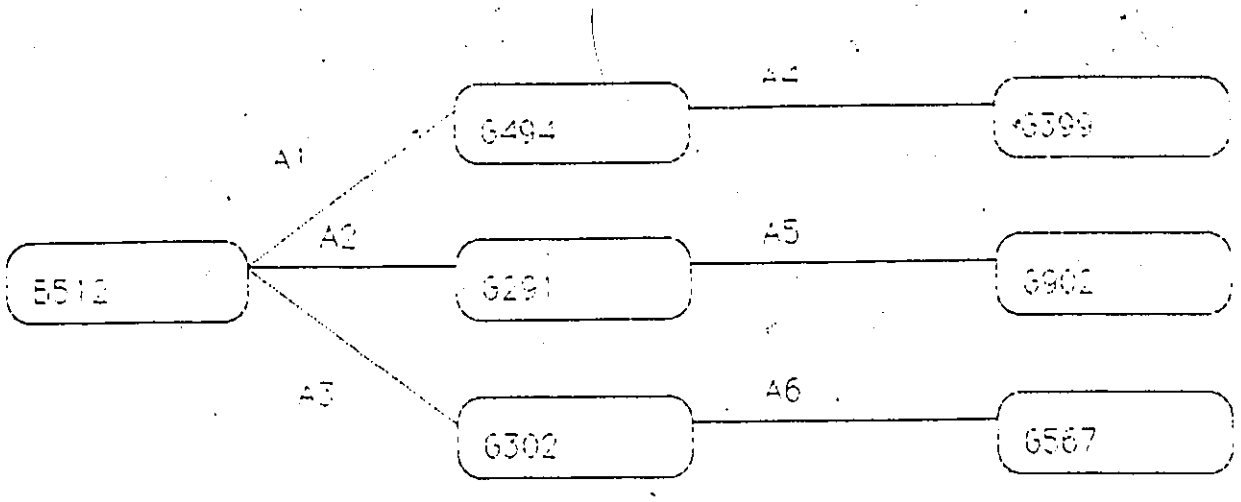
Finalement, un système produit par cette méthode sera volumineux en matière de règles. Chaque phrase du document qui traite une situation particulière a le potentiel de produire une nouvelle règle. Par contre, aucune provision n'a été mentionnée pour réduire la redondance dans ces règles. Pour développer notre système, nous avons fait une étude approfondie de toutes les politiques. Nous avons donc éliminé les redondances introduites par les politiques de voyage qui traitaient des situations semblables.

2.3 Réseaux sémantiques

Selon Winston, le sujet le plus important à considérer pendant le développement d'un système informatique est sans aucun doute la représentation de l'information. En effet, l'expérience a démontré qu'une bonne représentation permet souvent de transformer un problème complexe en un problème beaucoup plus simple.

Beaucoup de représentations définissent des objets et les relations qui existent entre eux. Il est donc normal de prétendre que plusieurs de ces représentations sont exprimées sous une forme quelconque de réseau sémantique puisque ceux-ci représentent des objets et les relations qui existent entre eux.

Dans un réseau sémantique, il y a des objets et des relations définies entre des paires d'objets. Dans les illustrations, il est devenu coutume de représenter un objet par un cercle étiquetté et les relations entre les paires d'objets par un arc étiquetté [Winston 84].



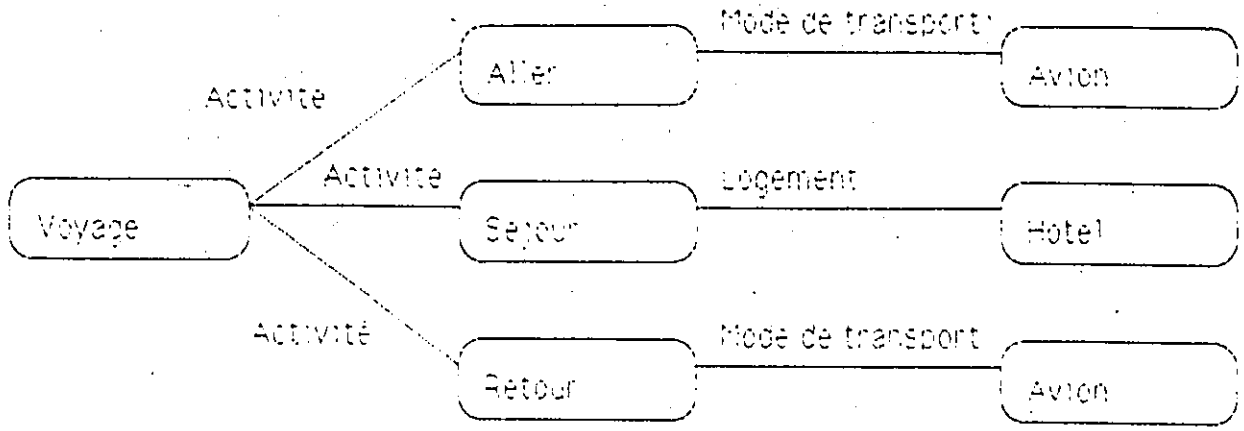
Reseau

Figure #4

Il faut remarquer que tous les réseaux détaillés de cette façon ne sont pas nécessairement des réseaux sémantiques. Par exemple, la figure 4 ne l'est pas.

Pour être un réseau sémantique, nous devons pouvoir attribuer une interprétation sémantique au réseau. De la figure 5, nous pouvons, sans effort considérable, conclure qu'un voyage consiste d'un aller, d'un séjour et d'un retour. Dans un aller, on doit retrouver un mode de transport, dans le séjour on doit retrouver un logement et dans le retour on doit aussi retrouver un mode de transport.

L'interprétation que nous donnons à ce type de diagramme est plutôt intuitive. En conséquence, il est souvent raisonnable et plus facile de remplacer ce diagramme par une représentation équivalente. Pour la programmation de notre système, nous nous sommes servis de cadres. Les détails sont décrits dans la section 3.3.1.1.



Réseau sémantique

Figure #5

2.4 Les scénarios

Tous les systèmes informatisés ont un comportement visible aux usagers qui ne rend pas toujours sès objectifs très clairs. Un des plus gros problèmes de l'informaticien est de définir un comportement pour son programme qui puisse rendre son utilisation plus facile. Dans les systèmes d'information de gestion, on tente de rendre l'usager à l'aise en lui présentant des images à l'écran qui sont des copies de ses propres formulaires. Il comprend ainsi ce que le système lui demande et pourquoi.

En considérant que les actions d'un voyageur qui prépare un voyage sont toujours semblables, cela devient intéressant d'essayer de les identifier. Au point de vue d'un système expert, ces actions permettent de définir un comportement pour le système qui est familier pour l'usager.

2.4.1 Définition

Selon [Schank Abelson 77], il y a deux mécanismes nécessaires pour comprendre des situations. D'abord il faut pouvoir se référer à une suite d'évènements d'une façon floue. Par exemple, il faudrait pouvoir identifier une situation qui se divise en dix étapes par son premier et son dernier évènement. Ainsi, nous considérons que toutes les étapes intermédiares se sont déroulées normalement. Donc le premier mécanisme doit reconnaître quand un scénario a été invoqué (i.e.: reconnaître l'occurence d'évènements faisant partie d'un scénario).

Le deuxième mécanisme doit être capable de récupérer les étapes qui n'ont pas été instantiées par le mécanisme de reconnaissance. Ces étapes sont souvent nécessaires à la compréhension d'un scénario donné. Ce mécanisme s'appelle l'interpréteur de scénario. Il permet de créer un lien de causalité entre les évènements reconnus par le premier mécanisme.

Soit les situation suivantes:

1- Pierre entre dans un restaurant. Il demande une brochette de poulet à la serveuse. Il paie la note et quitte le restaurant.

2- Martine prend l'autobus. Elle s'endort. Elle se réveille à Québec.

3- Lucie est allé à la fête de Jean-Luc. Jean-Luc a déballé ses cadeaux. Lucie a mangé du gateau et est partie.

Ces scènes sont compréhensibles parce qu'elles se réfèrent à des situations qui nous sont familières. Nous comprenons beaucoup plus que les quelques lignes de chaques histoires. Un système qui

comprend des histoires doit être en mesure d'inférer toutes les parties qui ont été omises.

Il existe un lien de causalité entre les parties de situation qui ont été décrites. C'est se qui s'appelle la connectivité d'un scénario. Par exemple:

4- Luc est allé au restaurant. Il a vu une serveuse. Il s'est rendu chez lui.

Les évènements décrits dans la situation 4 peuvent être reliés, mais ils n'arrivent pas à invoquer une situation particulière où nous serions capable de déduire avec confiance les évènements manquants.

Malgré qu'il soit possible de comprendre des histoires sans l'usage de scénarios, ils jouent un rôle important dans la compréhension de situations. Ils nous permettent d'omettre les détails qui n'influencent pas le reste des évènements.

Formellement, un scénario est une structure qui décrit une suite d'évènements dans un contexte donné. Les évènements sont faits de propriétés qui ont des exigeances quant à leurs valeurs. Les scénarios nous permettent de comprendre les situations quotidiennes stéréotypées. Ainsi, un scénario est une suite définie d'évènements qui décrivent une situation bien connue. Ils ne changent presque jamais et n'ont aucune provision pour traiter les situations nouvelles. Les histoires 1 à 3 se réfèrent à des scénarios.

Les scénarios sont définis en fonction de certains rôles. Un scénario dans un restaurant peut être défini du point de vue du client, du serveur ou du cuisinier.

[Schank Abelson 77] ont construit un interpréteur de scénario appelé **SAM** (**Script Applier Mechanism**) et s'en sont servi pour interpréter le scénario du restaurant du point de vue du client. Les scènes sont décrites par un ensemble de primitives qui représentent les activités de bases nécessaires dans un scénarios. Dans la figure 6, quelques détails du scénario d'un voyage sont exprimés. La notation ressemble à celle introduite par Schank et indique l'identité du scénario et les informations nécessaires à son traitement. Il faut remarquer que les repas consommé pendant les déplacements, sont généralement la responsabilité du transporteur. Nous avons donc omis le traitement de ce type de dépense dans cette version du système. Le traitement de ces dépenses peut être ajouté au système aisément.

La figure 6 est divisée en cinq sections et leurs numéros apparaissent au début de chacune. On a indiqué dans la **SECTION 1** l'identité du scénario et les objets nécessaires à son déroulement. Dans la **SECTION 2**, on trouve la liste des conditions qui permettent l'instanciation du scénario. La **SECTION 3** montre les étapes à suivre pour le départ. On y retrouve le mode de transport principal et, si c'est nécessaire, des déplacements locaux pour se rendre à la gare et de la gare àla destination. Dans la **SECTION 4**, il y a les activités quotidiennes qui engendre des dépenses remboursables. Et finalement, dans la **SECTION 5**, il y a les mêmes informations que dans la section pour le départ puisque le retour implique des activités presque identiques.

Quand l'interpréteur rencontre une situation de ce genre, il n'a pas besoin de créer des instances de chaque évènement, il peut sans crainte prétendre leur occurence. Dans l'histoire 1, on indique que le client a commandé un plat et a payé la note, mais pour un interlocuteur humain, il est entendu que tout s'est déroulé normalement et que le client a mangé.

SECTION 1
SCRIPT: voyage
TYPE: voyage d'affaire

SECTION 2
Condition d'activation:

Destination connue
Distance >= 16 kilomètres
Durée >= 1 journée

SECTION 3
Scène 1: Aller

transport principal
- avion, ----------
- train, ----------
- autobus, ----------
- bateau, ----------
- taxi,
- voiture,
- motocyclette

transport local (vers le point de départ et vers l'hébergement) taxi, autobus

---

SECTION 4
Scène 2: Séjour

RÉPÉTER POUR CHAQUE JOUR:

Activités quotidiennes:
Hébergement - commercial
- privé

Repas - montants forfétaires
- frais supplémentaires

Faux frais

Déplacements locaux

SECTION 5
Scène 3: Retour

transport principal
- avion, ----------
- train, ----------
- autobus, ----------
- bateau, ----------
- taxi,
- voiture,
- motocyclette

transport local (de l'hébergement vers le point de départ, etc) taxi, autobus

---

figure #6

Chaque évènement dans un scénario a une activité principale qui doit avoir lieu de façon explicite ou implicite. Dans la situation du restaurant, nous ne nous préoccupons pas de comment le client a obtenu le menu si ce n'est pas mentionné. Mais chose certaine, nous devons assumer qu'il a commandé. Chaque scène a une activité principale qui doit arriver. C'est ce que [Schank Abelson 77] appellent **"main conceptualization"** ou **"MAINCON"**. Ainsi, dans un système comme **"Voyage"**, nous supposons que tout s'est déroulé normalement si aucune situation exceptionnèlle n'a été mentionnée.

[Schank Abelson 77] nous introduit deux outils qui offrent des avantages aux systèmes que nous pouvons développer. Le premier est les scénarios. Ils rendent le comportement du système plus familier aux usagers. Le deuxième est la définition de valeurs de défaut intelligent. Elles permettent aux systèmes de rendre des décisions même si les utilisateurs omettent certains faits.

2.5 Censeurs

L'étude des politiques de voyage nous a révélé que, prise une à une, les politiques sont des règles qu'il ne faut pas transgresser. Il nous est apparu que ce serait plus simple de détecter une situation anormale en vérifiant une politique à la fois plutôt que de s'assurer que toutes les politiques étaient respectées.

Puisque nous présentons les détails de notre système dans la section 3.3.3.3 où nous y avons inclus la discussion de l'application des censeurs, ici nous nous limitons à la présentation de ce qu'est un censeur.

Winston a introduit les censeurs en développant le système Macbeth. Macbeth est un système qui, étant donnée une histoire, essaie de découvrir ce que sera le comportement de certains acteurs dans une histoire semblable. C'est donc un système qui applique un précédent à une nouvelle situation. Soit l'histoire suivante:

> C'est l'histoire de Macbeth, Madame Macbeth, Duncan et Macduff. Macbeth est un méchant noble. Madame Macbeth est une femme avare et ambitieuse. Duncan est un roi. Macduff est un noble. Madame Macbeth réussi à convaincre Macbeth de vouloir devenir roi parce qu'elle est avare. Elle réussi à le convaincre parce qu'ils sont mariés et il est faible. Macbeth tue Duncan avec un poignard. Macbeth tue Duncan parce que Macbeth veut être roi et parce que Macbeth est méchant. Madame Macbeth se tue. Macduff est faché. Macduff tue Macbeth parce que Macbeth a tué Duncan et parce que Macduff est fidel à Duncan.

et l'exercice:

> Ceci est l'histoire d'un faible noble et de sa femme avare. Montrez que le noble peut vouloir devenir roi.

Puisque l'histoire doit être considérée comme un précédent, le programme construit une règle à l'effet que la faiblesse d'un noble et l'avarice de sa femme peuvent lui faire désirer être roi. Par exemple, la règle suivante:

Si	un faible noble est marié à une femme avare
alors	ce noble peut vouloir être le roi

Ce système tente d'améliorer ses règles par l'assimilation de censeurs. Considérez l'exercice suivant:

> Cet exercice est l'histoire d'un faible noble et de sa femme avare. **Le noble n'aime pas sa femme.** Montrez que le noble peut vouloir devenir roi.

Cette situation est différente puisque si le noble n'aime pas sa femme, elle aura peut-être de la difficulté à convaincre le noble de vouloir devenir roi. Le rôle du censeur est de détecter la nouvelle situation et de bloquer le déclenchement de la règle ci-haut. Sa nouvelle forme sera:

Si	un faible noble est marié à une femme avare	(1)
alors	ce noble peut vouloir être roi	
sauf si	la femme ne peut pas influencer le noble elle ne peut pas le convaincre de devenir roi	

La partie "**sauf si**" dans cette règle indique la condition anormale qui pourrait prévenir le succès de la règle. La condition est vérifiée à l'aide d'une autre règle. Celle-ci est appelée un censeur. Dans Macbeth, le censeur qui vérifie si le noble peut être influencé par sa femme par est la règle suivante:

Si	la personne **X** n'aime pas la personne **Y**
alors	**Y** ne pourra peut-être pas influencer **X**

Puisque le censeur est lui-même une règle, il peut à son tour être bloqué. Par exemple:

Si	la personne **X** n'aime pas la personne **Y**	(2)
alors	**Y** ne pourra peut-être pas influencer **X**	
sauf si	**X** fait confiance à **Y**	

L'exercice suivant démontre l'application de ces règles.

Cet exercice est l'histoire d'un faible noble et de sa femme avare. Il n'aime pas sa femme. Il fait confiance à sa femme. Montrez que le noble peut vouloir devenir roi.

Les règles seront déclenchée de la façon suivante:

1- le problème est posé et la règle (1) est déclenchée. Sa partie condition est vérifiée.
2- la partie "sauf si" est examinée. Le résultat est le déclenchement de la règle (2). La condition de la règle (2) est vérifiée.
3- avant de bloquer la règle (1), Macbeth vérifie la partie "sauf si" de la règle (2).
4- Puisque cette condition est vrai, la règle (2) est bloquée et elle ne peut pas bloquer la règle (1).

Dans notre système, la construction "sauf si" n'existe pas. Mais l'idée de blocage en détectant des conditions qui ne sont pas acceptables est réalisée en déclenchant les règles censeurs avant d'exécuter la règle principale.

Un avantage dans notre système est que toutes les situations anormales sont détectées beaucoup plus rapidement. Par contre, toutes les règles vérifiant ces situations sont toujours déclenchées donc certaines conditions sont vérifiées inutilement.

3. SYSTEME EXPERT - VOYAGE

Pour développer notre système expert, nous avons choisi les politiques de voyage du gouvernement fédéral du Canada parce qu'elles nous permettaient d'exploiter l'application de plusieurs technologies. Malgré qu'une solution algorithmique soit possible, il nous semblait que, puisqu'il y avait un potentiel d'inférence (voir section 2.1.6) marqué, un système expert pourrait résoudre l'évaluation des réclamations de dépenses de voyage plus facilement. Beaucoup d'information était inutilement fournie par le réclamant. Aussi, les démarches faites par les voyageurs étaient presque toujours identiques.

3.1 Les politiques

Le coût toujours croissant des déplacements et des séjours à l'extérieur de leurs zones d'affectation a incité le gouvernement fédéral à émettre des politiques que tous les employés voyageant en service commandé doivent respecter. Elles assurent un confort raisonnable pour tous les voyageurs ainsi que des frais acceptables pour les contribuables.

Les politiques de voyages tentent de couvrir toutes les dépenses prévisibles pendant un voyage d'affaire. Généralement, les dépenses peuvent se rapporter aux trois activités principales suivantes:

- transport,
- hébergement,
- repas et faux frais.

La section des transports indique quels modes de transport sont acceptables en fonction de la distance de la destination et

le but du voyage. Dans la section des hébergements, les procédures à suivre pour se trouver un logement sont détaillées.

Avant de partir en voyage, l'employé doit remplir le formulaire 1 dans l'appendice A pour en obtenir l'autorisation. Sur ce formulaire il indique le but du voyage, le mode de transport principal et la destination. Ensuite, à la fin du formulaire, il y a une section qui lui permet de faire une demande d'avance de fonds.

Les politiques du conseil du trésor stipulent que lorsqu'un employé prend l'avion pour se rendre à sa destination, les billets d'avion seront fournis par le **Ministère des Approvisionnements et Services (M.A.S.)** ou que le voyageur sera remboursé sur présentation d'une preuve de paiement. En se servant de ce formulaire, l'employé n'indique pas qui est responsable de faire ces réservations, ni comment la facture sera réglée.

Les politiques indiquent aussi que, si l'employé réserve sa chambre, il doit s'assurer que l'hôtel qu'il occupera est dans le répertoire du **M.A.S.** Si l'hôtel ne s'y trouve pas, les tarifs du logement choisi ne doivent pas dépasser ceux d'un hôtel se trouvant dans le répertoire.

Le **M.A.S.** émet régulièrement une liste de montants forfétaires couvrant les dépenses de repas et les faux frais. Lorsque l'employé ne prévoit aucune dépense extraordinaire, il devrait calculer son avance de fond en fonction des montants indiqués.

Au retour, l'employé doit remplir deux autres formulaires (appendice A). Ce sont des formulaires de réclamations de dépenses. Le premier est un formulaire général et le deuxième est un formulaire où chaque dépense réclamée doit être détaillée.
sans crainte prétendre leur occurence. Dans l'histoire 1, on indique que le client a commandé un plat et a payé la note, mais

pour un interlocuteur humain, il est entendu que tout s'est déroulé normalement et que le client a mangé.

3.2 Système expert

Nous avons écrit un système expert pour essayer de réduire la complexité de la tâche d'un employé qui prépare un voyage. Les règles que nous y avons inclus sont notre interprétation des sections 3, 4, 5 et 6 du chapitre 370 du Manuel de la Politique Administrative du Conseil du Trésor Canada. Son rôle principal en tant qu'assistant à la préparation d'un voyage est d'offrir un système d'information intelligent.

L'employé qui prépare son voyage n'est pas toujours familier avec les politiques et ainsi doit avoir recours à un conseiller qui lui indique les démarches à suivre pour organiser son séjour. Le système offre, entre autre, de l'information concernant les modes de transport. Un voyageur qui se déplace vers une destination à plus de 300 kilomètres de son lieu de travail peut prendre l'avion mais il doit respecter certaines contraintes. L'avion doit être d'immatriculation canadienne et l'employé doit occuper un siège en classe économique.

Le système permet à son usager de demander des explications quant aux restrictions décrites ci-haut. Aussi, il a des provisions pour certaines situations exceptionnelles. Par exemple, voyager en première classe dans un avion est permis quand les circonstances l'exigent. Ainsi, lorque l'employé prépare son voyage, il saura à l'avance quelles dépenses seront autorisées et quel montant demander comme avance de fonds.

Il permettra aussi à un évaluateur de traiter les réclamations d'une façon plus efficace. Il aura à sa disposition un système qui pourra répondre à toutes ses questions pour une réclamation des

dépenses d'un voyage normal.

En plus de répondre aux questions du domaine d'application, nous avons cherché à produire un système plus accessible aux usagers. Il devait être facile d'utilisation, facile à modifier, efficace et performant.

La facilité d'utilisation est un facteur important puisque la plupart des usagers sont des novices en informatique. Il s'agit de réduire le nombre d'interventions que doit faire l'utilisateur au stricte minimum.

La considération la plus importante que doivent avoir les développeurs d'un système est l'entretient et la mise à jour des programmes qu'ils produisent. Le code doit être clair, facile à lire et facile à modifier. Là où le programme ne répond pas à ces critères, des commentaires explicatifs sont de mise.

L'efficacité du système est aussi très important. Malgré que le système ne couvre pas toutes les politiques du chapitre 370, il doit bien traiter les politiques stipulées.

Aussi, son caractère interactif dicte un comportement qui, en temps réel, soit acceptable à la majorité des usagers. S'ils doivent attendre au delà de 8 à 10 secondes pour chaque réponse, le système a un grand besoin d'amélioration.

La possibilité d'intégrer ce système à d'autres types d'application est aussi un atout important. Il pourrait puiser dans une base de données pour trouver certaines informations. Il pourrait aussi créer des fichiers qui serviraient comme intrants à des chiffriers électroniques ou des traitements de texte.

3.2.1 Exemple

Dans cette section, on retrouve un exemple d'une consultation avec le système expert. L'impression en caractères normaux indique ce que le système affiche, les caractères gras indiquent la réplique de l'usager et le texte commençant avec un "%" précède des commentaires explicatifs ajoutés après la consultation.

Il faut aussi noter que les choix de réponse offerts quand le système demande une question à l'usager, sont présentés dans un ordre que le système génère.

```
?- [voyage].                % charger les utilitaires Prolog
yes
?- rc.                      % charger le système expert
Chargement des règles...
Consultation du texte source des règlements...
fin

yes
?- voyage.


1. De quelle province/territoire part le voyageur?

          1) Labrador
          2) Terre Neuve
          3) Ile du Prince Édouard
          4) Nouvelle Écosse
          5) Nouveau Brunswick
          6) Québec
          7) Ontario
          8) Manitoba
          9) Saskatchewan
          10) Alberta
          11) Colombie Britannique
          12) Yukon
          13) Territoire du Nord-Ouest
          14) autre
          15) why ?
          16) unknown

? 6
% certains tarifs varient selon la province de départ

2. Quelle est la destination du voyage?
```

1) Canada
2) États Unis
3) autre
4) why ?
5) unknown

? 1

3. Quelle en est sa distance? 350

4. Combien de jours le voyageur sera-t-il parti (minimum 1 jour) ? 1

5. Quel mode de transport sera utilisé?

1) autobus
2) autre
3) avion
4) bateau
5) motocyclette
6) taxi
7) train
8) voiture
9) why ?
10) unknown

? 8

6. Avez-vous l'autorisation d'emprunter un mode autre que celui stipulé dans les politiques?

1) oui
2) non
3) why ?
4) unknown

? 3
% puisque la destination du voyage est à plus de 300 km et que le
% voyageur s'y rend en voiture, le système doit savoir s'il a eu
% l'autorisation de prendre une voiture. Ici, l'usager veut savoir
% pourquoi une autorisation est nécessaire.

.4.1.1.0

Autorisation du mode de transport.

S'il y a plus de 300 kilomètres à parcourir, l'utilisation d'un véhicule conduit par l'employé ne devrait pas normalement être autorisée à moins que le voyage par transport commercial ne présente des inconvénients majeurs ou que l'employeur ne le juge pas pratique sous le rapport du coût global, y compris le salaire,

les dépenses personnelles et les frais de voyage. Voir l'appendice B, qui indique les distances approximatives entre les agglomérations canadiennes. Pour les voyages sur de courtes distances, il faut faire preuve de discernement pour déterminer s'il y a lieu d'autoriser l'utilisation des véhicules automobiles particuliers, surtout s'il est possible de recourir à des moyens de transport commerciaux commodes et plus économiques, à moins que l'utilisation d'un tel véhicule ne s'impose en raison de certaines circonstances, dont la nécessité de transporter plusieurs passagers.

RETOUR pour continuer

7. Avez-vous l'autorisation d'emprunter un mode autre que celui stipulé dans les politiques?

1) oui
2) non
3) why ?
4) unknown

? 1

8. Qui est le chauffeur de la voiture?

1) voyageur
2) autre
3) why ?
4) unknown

? 1
% certaines dépenses peuvent être remboursées quand le voyageur est
% un passger dans une voiture

.4.1.2.0

Dans l'intérêt de la sécurité au volant, l'employé autorisé à conduire lui-même une voiture ne devrait pas normalement franchir une distance supérieure:

a) à 250 kilomètres après avoir travaillé toute une journée,
b) à 350 kilomètres après avoir travaillé une demi-journée, ou
c) à 500 kilomètres le jour où l'employé n'a pas travaillé.

RETOUR pour continuer

% lorsque le mode de transport autorisé est une voiture, cette
% explication est automatiquement affichée

9. Est-ce une voiture?

1) privée
2) louée
3) état
4) why ?
5) unknown

? 3
% le mode de paiement de la dépense varie selon le le type de voiture

10. Est-ce qu'une carte de crédit gouvernementale sera fournie au voyageur?

1) oui
2) non
3) why ?
4) unknown

? 1
% normalement, pour une voiture appartenant au gouvernement, les
% dépenses sont imputées au gouvernement à l'aide d'une carte
% de crédit

Le mode de remboursement sera

1) billet / 1.0

11. Prenez-vous le même mode de transport pour le voyage de retour?

1) oui
2) non
3) why ?
4) unknown

? 1

Le mode de remboursement sera

1) billet / 1.0

12. Est-ce qu'un logement est requis? n.

Les dépenses du voyage sont

1) retour / 0.9
2) aller / 0.9

Le système affecte un facteur de confiance à chaque réponse. Ici, la valeur **0.9** est une conséquence des valeurs de défauts du système.

3.3 Programmation du système expert

Arity/Expert Shell est une coquille à base de concepts et de règles. Pour programmer le système, il y a quatre types d'information à coder. Le premier est la taxonomie. Cette étape consiste en la déclaration de tous les concepts utilisés par le système. Le deuxième est la définition des options de contrôle. Le troisième est la liste de toutes les règles du système expert. Et finalement, le quatrième est la déclaration de toutes les clauses utilitaires Prolog.

La dimension totale du système peut être exprimée de deux façon:

1> Dimension des fichiers:

- fichiers **Arity/Expert:** 58 K-octets
- fichiers **Arity/Prolog:** 62 K-octets

2> Règles et clauses:

- règles **Arity/Expert:** 64
- cadres **Arity/Expert:** 23
- options de contôle **Arity/Expert:** 260
- clauses **Arity/Prolog:**
  - prédicat ex/3: 118

- prédicat explication/3: 36
- contrôle et utilitaires: 31
clauses **Arity/Prolog** total: 185

3.3.1 La taxonomie

Dans la taxonomie du système, il y a deux types d'informations. Le premier est la définition des types et le deuxième est la déclaration des cadres nécessaires au système. La définition des cadres dépend de la structure logique des données et du comportement que nous voulons donner à notre système. Le réseau sémantique construit en déclarant les cadres représente notre structure de données et les scénarios nous ont permit de définir le contôle pour notre système.

3.3.1.1 Structures

La représentation de connaissances que nous avons choisie d'utiliser est un réseau sémantique. Il a été réalisé en se servant de cadres définis en **Arity/Expert**. Cahque cadre **Arity/Expert** représente un concept et chaque tirroir représente une propriété. Dans les figures 7,8 et 9, la structure que nous avions originalement définie est détaillée. C'est un réseau sémantique en forme d'arbre ayant comme racine le concept principal du système, un voyage. Il y a dans chaque concept une liste de propriétés. Certaines des propriétés prennent des valeurs simples. Les autres sont définis par des concepts.

La coquille ne permettait pas de définir une structure semblable à celle-ci. Nous avons dû aplatir notre arbre pour pouvoir se servir de tous les concepts. Par exemple, nous avons défini le concept **"mode de transport"** où nous avons inclu toutes

les propriétés communes à tous les modes de transport. Ensuite, nous avons défini le concept "**voiture**" comme étant un "**mode de transport**" (voiture hérite ainsi toutes les propriétés d'un mode de transport) et nous lui avons ajouté quelques propriétés supplémentaires propres aux voitures. Ensuite, nous avons défini le concept "**aller**" avec la propriété "**mode de transport**". Nous croyions pouvoir créer de cette façon une instance du cadre approprié selon les circonstances (i.e.: distance plus courte que 300 kilomètres, donc le système doit créer une instance du concept voiture). Mais comme les définitions ci-dessous l'illustrent, le concept "**aller**" était défini avec le concept "**mode de transport**" comme propriété, et non avec le concept "voiture". La notation qui suit est celle de **Arity/Expert**.

```
define primitive c_aller with
  r_aller_local  = c_deplacement_local and
       r_transport    = c_transport and
       r_retour_local = c_deplacement_local.


define primitive c_transport with
       r_frais_transport = c_frais_transport and
       p_duree = (0.0, 24.0) and
       any p_reservation_M.A.S. and
       any p_type.


define primitive c_voiture as a c_transport with
       any p_cargaison and
       any p_conducteur and
       any p_demande_de_employeur and
       any p_dimension_voiture and
       any p_type_voiture.
```

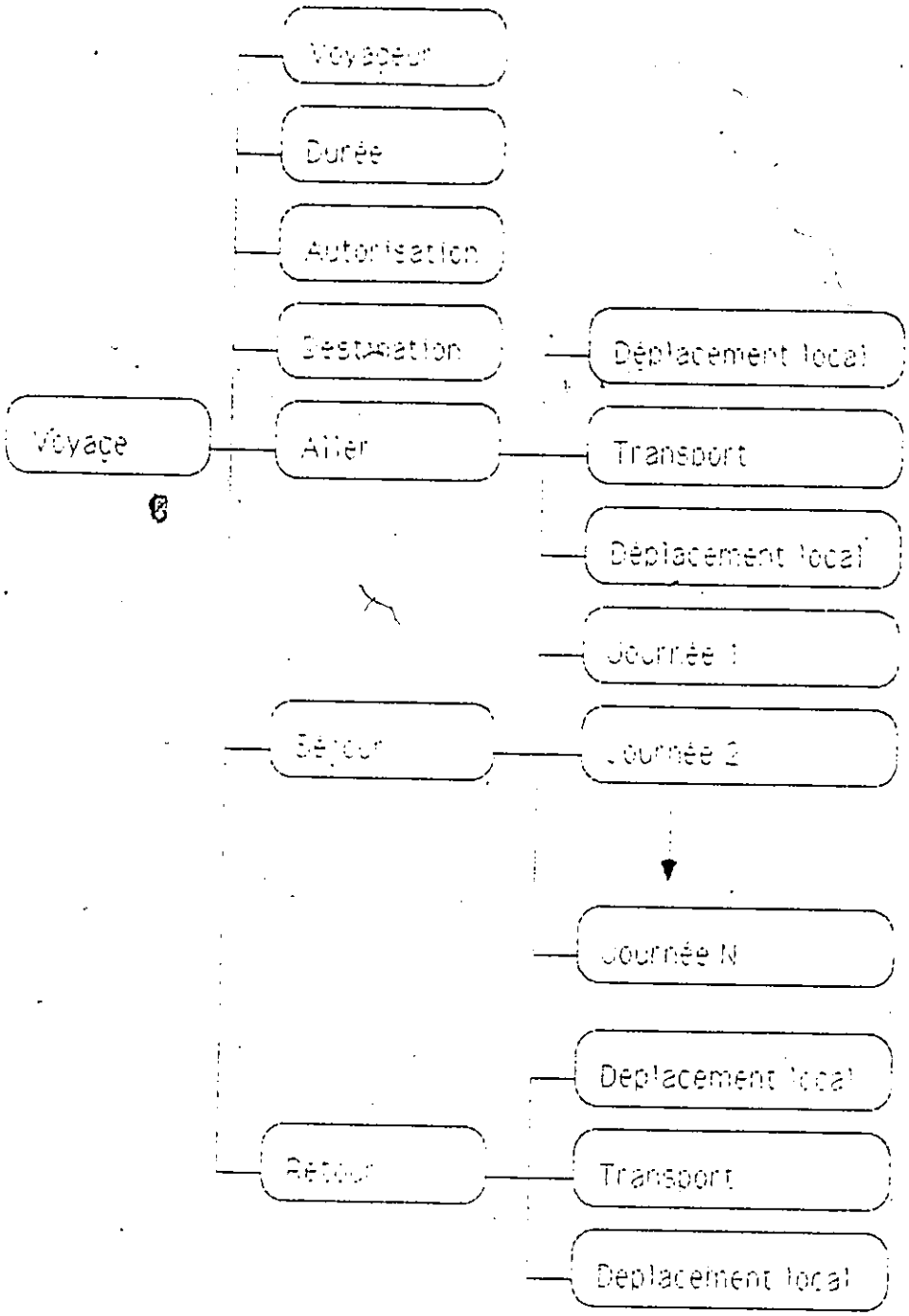

Voyageur
Durée
Autorisation
Destination
Déplacement local
Voyage
Aller
Transport
Déplacement local
Journée 1
Séjour
Journée 2
Journée N
Deplacement local
Retour
Transport
Deplacement local

Figure #7

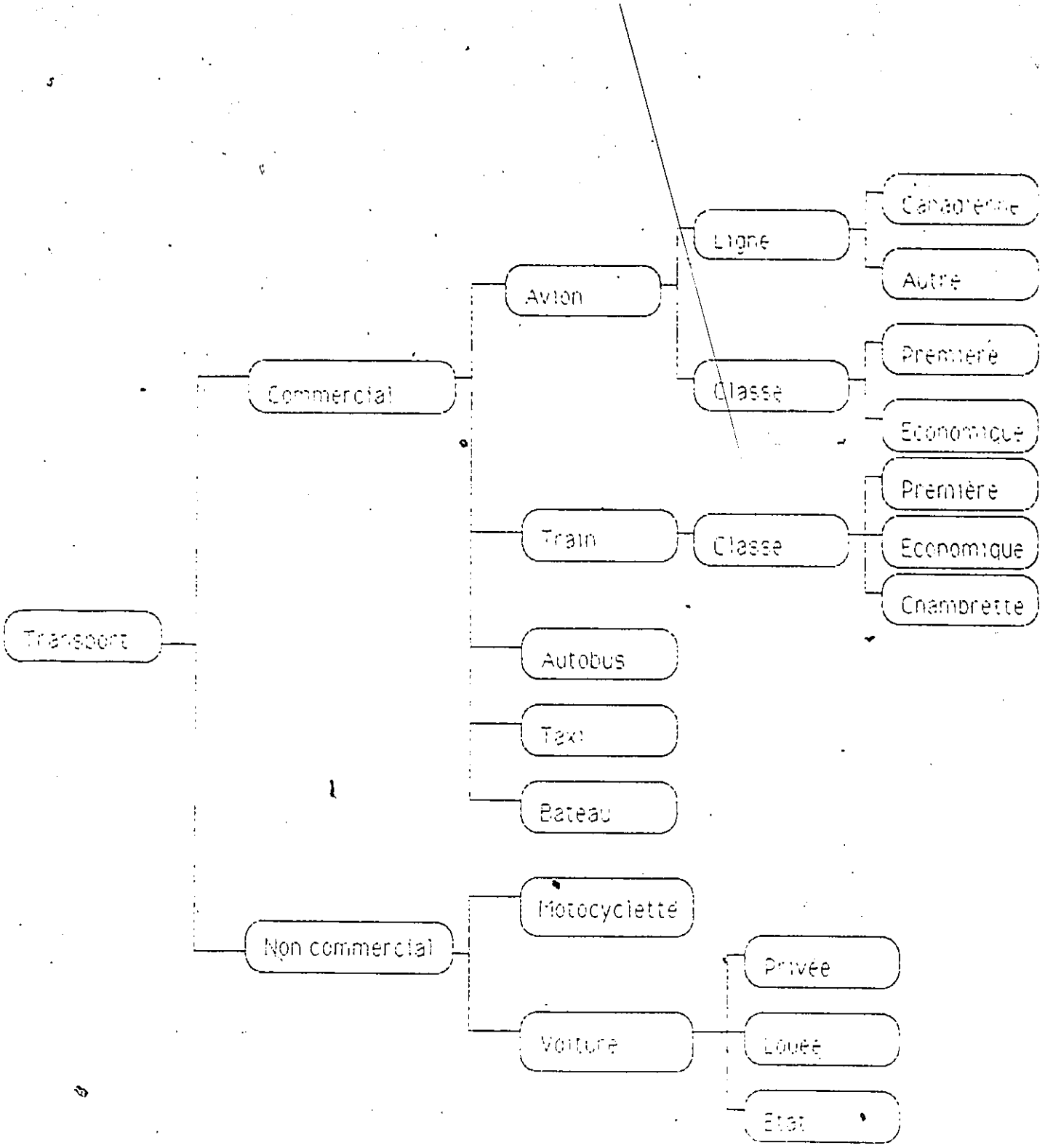

Transport
Commercial
Avion
Ligne
Classe
Train
Classe
Autobus
Taxi
Bateau
Non commercial
Motocyclette
Voiture
Privée
Louée
Etat
Première
Economique
Première
Economique
Chambrette

Figure 6

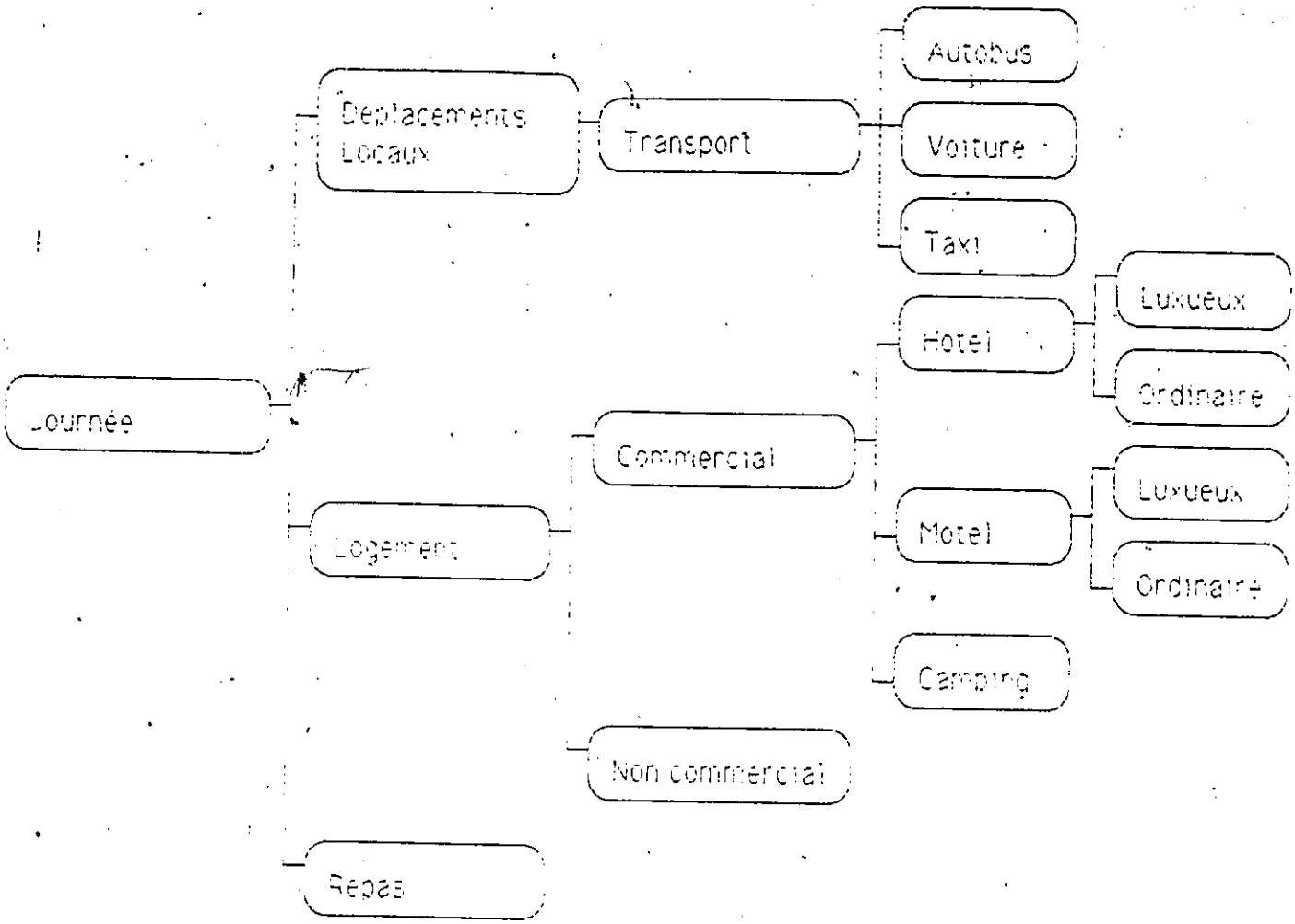

Journée
Déplacements Locaux
Transport
Autobus
Voiture
Taxi
Logement
Commercial
Hotel
Luxueux
Ordinaire
Motel
Luxueux
Ordinaire
Camping
Non commercial
Repas

Figure #9

L'instruction "**define primitive <cadre> with <liste de tirroirs>**" permet de définir un cadre nommé **<cadre>** avec une série de tirroir. Pour chacun des tirroirs, on doit définir leurs types respectifs. Ainsi, le cadre "**c_aller**" est défini avec trois tirroirs. Le tirroir contenant le cadre "**r_aller_local**" est de type "**c_deplacement_local**". Le reste de la structure se lit de façon semblable.

Pour régler ce problème, nous avons défini un cadre représentant le concept "**mode de transport**" avec toutes les propriétés de tous les modes de transport qui peuvent servir dans cette application:

```
define primitive c_transport with
        r_frais_transport = c_frais_transport and
        any p_autorisation_classe and
        any p_autorisation_duree and
        any p_autorisation_ligne and
        any p_autorisation_mode and
        any p_autorisation_taxi and
        any p_cargaison and
        any p_classe and
        any p_conducteur and
        any p_demande_de_employeur and
        any p_dimension_voiture and
            p_duree = (0.0, 24.0) and
        any p_ligne and
        any p_mode_de_transport and
        any p_reservation_M.A.S. and
        any p_type and
        any p_type_voiture.
```

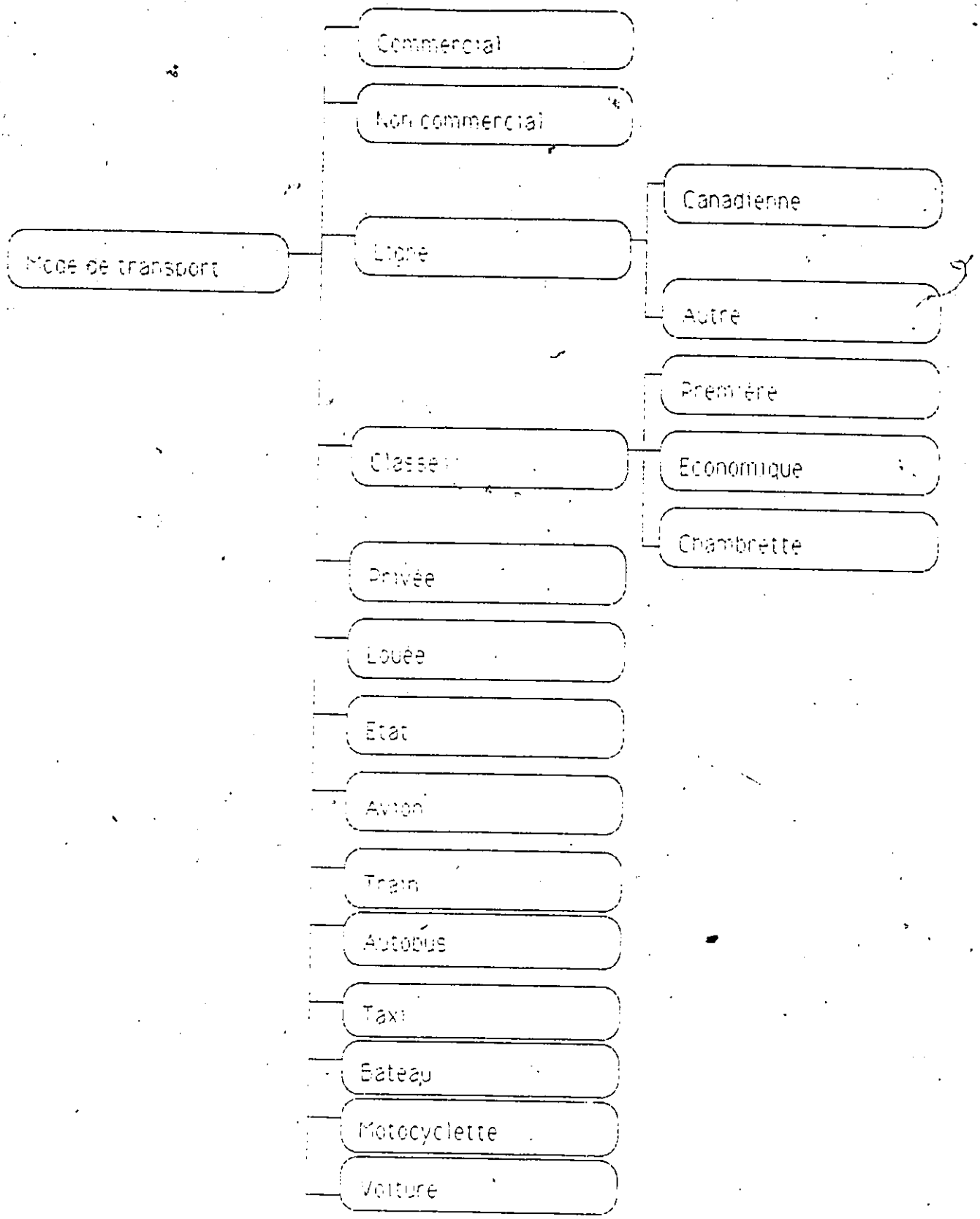

Commercial
Non commercial
Mode de transport
Ligne
Canadienne
Autre
Classe
Première
Economique
Chambrette
Privée
Louée
Etat
Avion
Train
Autobus
Taxi
Bateau
Motocyclette
Voiture

Figure #10

En conséquent, nous avons défini notre structure d'une façon différente. La partie principale illustrée dans figure 7 ne change pas. Les changements importants se trouvent dans la figure 10.

3.3.1.2 Scénarios et réseaux sémantiques

Nous avons construit cette structure pour pouvoir exprimer les relations existantes entre les différents concepts et aussi parce qu'elle nous permettait d'inclure dans notre système un comportement inhérent aux scénarios.

Un voyage a typiquement trois étapes, l'aller, le séjour et le retour. Dans la taxonomie du système, ces trois cadres ont été inclus dans le cadre "voyage" dans les tirroirs appropriés. L'aller peut être décrit par un déplacement local (ex: transport vers l'aéroport), un transport principal (ex: l'avion) et un deuxième déplacement local (ex: de l'aéroport vers la destination). Le retour peut être décrit de la même façon. Le séjour se divise en étapes qui, pour les fins des réclamations de dépenses, exigent un déboursé quelconque. Il y a l'hébergement quotidien (ex: un hotel ou un motel), les repas et les déplacements locaux (ex: taxi ou autobus) entre les différents cites que doit visiter le voyageur.

Le comportement du système (i.e.: tel que prescrit par notre scénario) est possible parce que cette structure contient tous les éléments d'information qui nous permettent de le contrôler. Le comportement lui-même est géré par l'ordre du déclenchement des règles et les options de contrôle.

Nous avons tenté de définir le scénario du voyage pour qu'il soit totalement indépendant des instances de cadres créés pendant

l'interprétation des règles. Ainsi, lorsqu'un employé prépare un voyage, si des informations sont absentes, le système peut aisément prétendre que tout s'est déroulé normalement.

3.3.1.3 Les types

Pour chaque tirroir, il faut déclarer un type. Si c'est un tirroir qui peut contenir des valeurs numériques, nous devons l'indiquer au début du fichier de taxonomie. Si c'est un tirroir prenant l'équivalent d'un atom Prolog comme valeur, nous devons déclarer la liste complète de tous les atomes acceptables. Quand le tirroir peut contenir un autre cadre, dans **Arity/Expert** on en parle en terme de rôle dans un autre cadre et il doit être déclaré ainsi. Voici un exemple des trois déclarations de type possibles:

```
p_duree = numeric

p_type = [ v_commercial, v_non_commercial ]

r_mode_de_transport = role
```

Dans notre système, nous avons préfixé toutes les propriétés ordinaires d'un "p_", tous les rôles avec un "r_", toutes les valeurs par un "v_" et tous les concepts par un "c_".

3.3.1.4 Les concepts

Pour exprimer les relations entre les différents concepts, nous nous sommes servis des cadres **Arity/Expert**. Un cadre est une structure compréhensible par l'ordinateur qui représente une

entité, un objet ou une idée. Dans voyage, nous avons déclaré le cadre principal, "c_voyage", de la façon suivante:

```
define primitive c_voyage with
     r_commanditaire = c_commanditaire and
     r_commandite    = c_commandite and
     r_destination   = c_destination and
     r_temps_depart  = c_instant and
     r_temps_retour  = c_instant and
     p_duree         = (0.0, 5000.0) and
     any p_province and
     any p_autorisation and

     r_aller  = c_aller and
     r_sejour = c_sejour and
     r_retour = c_retour.
```

Pour signaler la définition d'un cadre au système, nous devons nous servir de la commande **"define primitive"** et ensuite indiquer la liste de touts les tirroirs du cadre. La convention que nous avons établie nous permet d'identifier les tirroirs en regardant le préfixe des identificateurs. Lorsqu'elles sont préfixées d'un "p_", les tirroirs prennent des valeurs simples (numérique ou un atome Prolog). Ici, nous devons définir toutes les valeurs acceptables pour chaque tirroir du cadre. Le tirroir "p_duree" peut prendre des valeurs réelles entre 0.0 et 5000.0 (dans notre système, nous n'employons que les valeurs entières). Le tirroir "p_province" peut prendre n'importe quelle valeur qui a été définie dans la section de déclaration de types. Les tirroirs qui peuvent contenir un cadre sont identifiées par le préfixe "r_", pour "role" et sont suivies du signe "=" et du cadre qu'elles représentent.

Dans le cadre "c_voyage", presque touts les tirroirs sont eux-mêmes des cadres. L'existance des trois dernièrs peut être justifié par l'usage des scénarios puisque le scénario d'un voyage consiste en un aller, un séjour et un retour. La définition des autres cadres peut être trouvée dans l'appendice B.

Aussi, plusieurs cadres ont des tirroirs semblables. Le mode de transport, les hôtels ainsi que le repas engendrent tous des frais. Nous avons donc défini le cadre "c_frais" qui devait servir à plusieurs endroits différents. Mais quand le système déclenche une règle, il ne différencie pas les cadres qui ont le même nom. Il applique les règles qui ont été définies pour le tirroir "r_frais" du cadre "[illegible]aller", au tirroir "r_frais" du cadre "c_logement". C'est la raison pour laquelle on retrouve dans le fichier de taxonomie la déclaration du cadre "c_frais" et des cadres:

```
define c_frais_transport as a c_frais
define c_frais_logement as a c_frais
define c_frais_repas as a c_frais
define c_frais_sejour as a c_frais
```

Ces déclarations assurent que les cadres représentant les différents frais ont les mêmes tirroirs mais pas les mêmes noms.

Pendant l'exécution du système, l'interpréteur crée une instance de tous les cadres qui servent pour la consultation. Lorsqu'il nous donne une réponse, celles-ci ne sont pas détruites, elles existents toujours. Cette situation est semblable à une assertion d'un terme Prolog. La figure 11 illustre l'état du système après que l'utilisateur ait demandé une deuxième consultation sans quitter l'interpréteur.

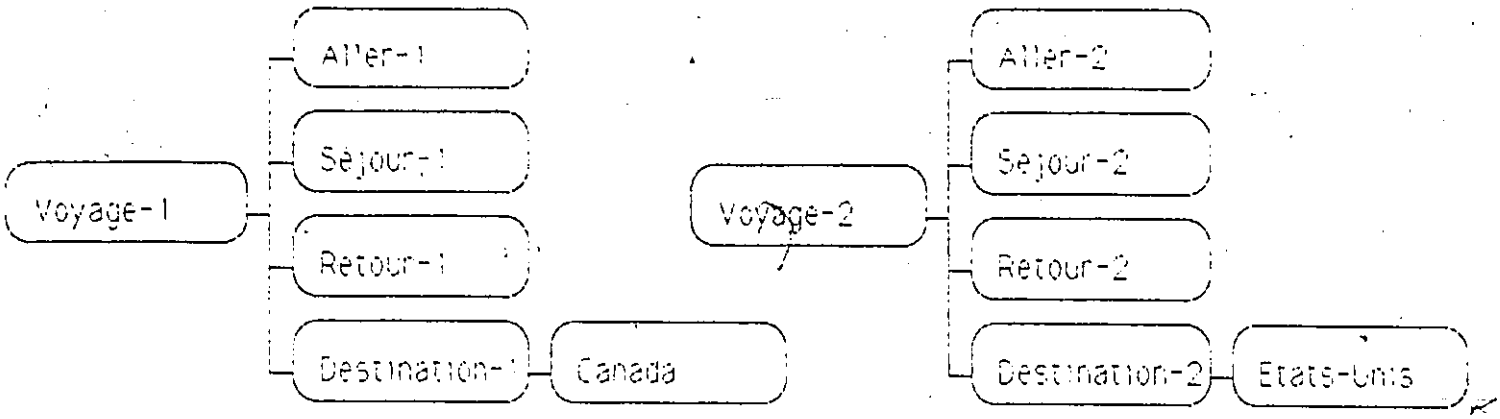


Figure # 1

3.3.2 Options de contrôle

Arity/Expert est une coquille de système expert. Le travail minimum qu'un informaticien doit fournir avec un tel système est de définir les cadres et les règles pour son application. La coquille s'occupe d'interroger l'usager, de fournir les explications et de déterminer l'ordre du déclenchement des règles. Aussi, il fournit des valeurs de défaut aux tirroirs dont l'usager ne connaît pas les valeurs. Généralement, le comportement que la coquille nous donne n'est pas acceptable. Si, par exemple, le système tente d'évaluer la valeur du tirroir "p_autorisation" du cadre "c_frais", le système vérifie d'abord s'il existe une règle qui calcule sa valeur, ensuite, s'il n'en existe pas, il pose la question:

"what is the p_autorisation of the c_frais"

et si l'usager ne connaît pas la réponse, il dépose dans le tirroir une valeur de défaut. Ce comportement n'est pas acceptable, parce que quand l'usager se sert du système pour évaluer l'autorisation, il ne faut pas que le système le lui demande.

Il existe dans la coquille **Arity/Expert** une série d'options de contrôle qui permettent aux programmeurs de guider le comportement de leurs systèmes. Les détails qui suivent couvrent les options de contrôle qui ont un impact important sur le système. Beaucoup d'autres options existent qui n'ont pas servi ou qui ne sont pas dignes de mention dans ce document.

Le système expert est déclenché par une demande d'évaluation du contenu du tirroir **"p_autorisation"** du cadre **"c_voyage"**. Il cherche dans sa base de règles, la première qui fait l'affectation d'une valeur à ce tirroir.

3.3.2.1 Défauts

Quand un utilisateur ne connaît pas la réponse à une question (i.e.: la valeur d'un tirroir) ou qu'il n'y a pas de règle qui réussit à affecter une valeur, le système dépose dans le tirroir sa valeur de défaut. Normalement, la valeur de défaut est **"unknown"**. Une des options de contrôle nous permet de redéfinir cette valeur et de lui affecter une valeur appropriée. Entre autre, dans voyage si le système ne réussi pas à calculer la valeur de l'autorisation des dépenses de voyage, il dépose la valeur **"v_non_autorise"** dans le tirroir grâce à l'énoncé suivant:

```
default( p_autorisation of c_voyage ) = v_non_autorise / 0.9
```

Le chiffre décimal qui se trouve à la fin de l'expression est un facteur de confiance. La coquille nous permet d'associer à

chaque valeur affecté à un tirroir un facteur de confiance entre -1.0 et 1.0 (1.0 indiquant la certitude, 0.0 l'incertitude et -1.0 la certitude que la réponse n'est pas acceptable). Il permet ainsi d'inclure dans les systèmes experts des raisonnements flous. Avec l'option **"default"**, nous sommes obligés de spécifier un facteur de confiance ayant une valeur maximale acceptable de 0.9 (ceci est une restriction imposée par la coquille). Étant donnée la nature de notre système, nous n'avions pas besoin de manipuler les facteurs de confiance.

En général, une seule valeur de défaut est acceptable. Mais plusieurs situations dans le système voyage impliquent des valeurs différentes selon certaines informations. Par exemple, nous ne pouvons pas définir une valeur de défaut, à l'aide de cette option de contrôle, pour le mode de transport puisqu'elle dépend de la distance de la destination. Ceci rend le mécanisme de défaut du système incomplet pour nos besoins. Nous avons donc inclus des règles additionnelles pour calculer les valeurs appropriées.

3.3.2.2 L'ordre d'évaluation

Il y a trois façon dont la coquille dépose des valeurs dans un tirroir:

- le succès du déclenchement d'une règle,
- une réponse de l'utilisateur,
- une valeur de défaut.

Normalement, le système vérifie d'abord dans sa base de règles en cherchant un conséquent qui affecte une valeur au tirroir qu'il cherche à évaluer. S'il trouve une règle, il évalue l'antécédent. S'il n'en trouve pas, il demande à l'usager de lui fournir une valeur. Si l'usager ne connaît pas la valeur, le système affecte

à la propriété la valeur de défaut définie. Dans plusieurs cas ce comportement est bon, mais souvent c'est inutile de demander à l'usager de fournir la réponse puisqu'il se sert du système justement pour évaluer ce résultat. Par exemple, si l'usager cherche à savoir si une dépense est autorisée et qu'il n'y a aucune règle qui réussit à le calculer, le système demandera à l'usager si la dépense est autorisée. L'option "**order**" nous permet de prévenir ce comportement. Dans l'énoncé

```
order( p_autorisation of c_voyage ) = [ r, i ]
```

le "**r**" indique au système de regarder d'abord dans la base de règles pour affecter une valeur à "**p_autorisation**" par le déclenchement d'une règle. Si aucune règle ne réussit à lui affecter une valeur, le "**i**" dicte au système de lui affecter une valeur de défaut, en l'occurence "**v_non_autorise**".

Les valeurs possibles pour cette option de contrôle sont:

```
[ q, r, i ]
```

Le "**q**" indique au système qu'il peut demander une question à l'usager pour trouver la valeur d'un tirroir. Le "**r**" indique que le système peut se servir d'une de ses règles pour évaluer la valeur d'un tirroir. Le "**i**" indique de prendre sa valeur de défaut. N'importe quelle combinaison de ces trois options est valides. Aussi, l'ordre a une importance. Puisque dans cet exemple le "**q**" est le premier choix, le système demandera à l'utilisateur de fournir une valeur à un tirroir avant de se servir d'une règle ou de sa valeur de défaut.

En forçant le système à d'abord demander une question à l'usager plutôt que de déclencher une règle, nous pouvons ajouter au système un mécanisme de défaut plus intelligent. Par exemple,

quand l'usager ne connaît pas quel mode de transport sera utilisé, nous pouvons écrire une règle qui vérifie la distance pour affecter une valeur au tirroir, plutôt que de laisser la coquille générer une valeur de défaut.

3.3.2.3 Questions

Arity/Expert Shell génère ses propres questions quand il a besoin de demander à un usager la valeur d'un tirroir. Par exemple, lorsque le système doit demander la valeur du tirroir **"p_destination"** du cadre **"c_destination"** il génère la question suivante:

what is the p_destination of the c_destination?

Cette question, quoique compréhensible par l'auteur du système, n'est peut-être pas compréhensible par les autres utilisateurs. L'option de contrôle qui nous permet de changer cette formulation est **"question"**. Voici sa syntaxe:

```
question(p_destination of c_destination) =
        $Quelle est la destination du voyage?$
```

Maintenant, lorsque le système fonctionne, la phrase entre les signes **"$"** sera affichée quand le système cherchera à donner une valeur au tirroir **"p_destination"**.

3.3.2.4 Chaînes

Pour calculer l'autorisation des dépenses de voyage, nous devions évaluer le remboursement des frais d'aller, des frais de séjour et des frais de retour. Le système exige qu'on se réfère à l'instance appropriée d'un cadre pour permettre le déclenchement

des règles. En regardant la structure des cadres, figure 12, nous pouvons voir que, pour manipuler les frais d'aller, nous devons nous servir du **"role chain"** suivant:

P_remboursement of r_frais_aller of r_aller of c_voyage

Toutefois, **Arity/Expert** n'accepte pas cette formule dans les règles du système. Il faut la remplacer par un synonyme qui a la forme suivante:

set(r_frais_aller of r_aller of c_voyage) = frais_aller

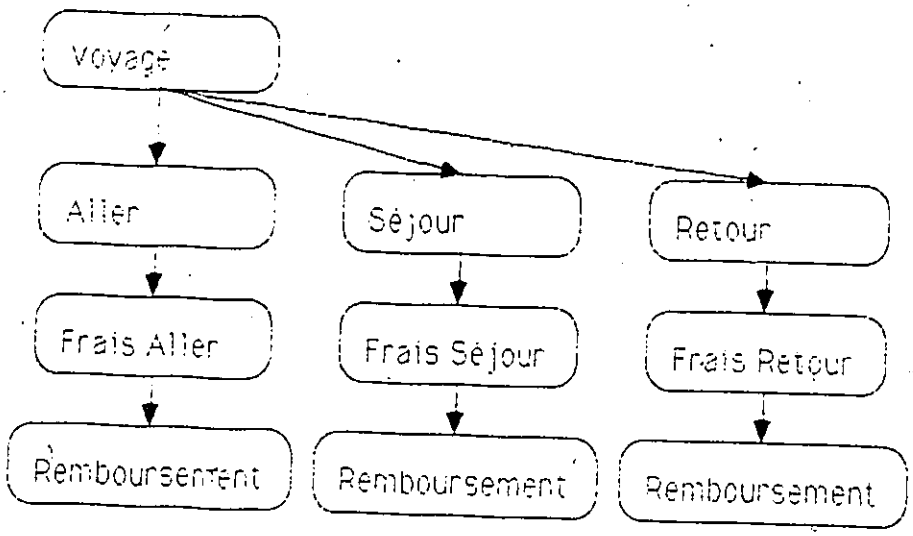


Figure 12

Maintenant pour se référer au tirroir "p_remboursement", il suffit d'écrire l'énoncé de la façon suivante:

P_remboursement of frais_aller

3.3.3 Les règles

Arity/Expert est un interpréteur Prolog augmenté de certaines fonctions pour permettre le développement de systèmes experts. Il permet la définition de cadres. Il génère les questions quand il a besoin de la valeur des tirroirs. Il gère la création des réseaux d'inférence. Il fournit un module d'explication qui indique aux utilisateurs pourquoi une question a été demandée et comment une réponse a été determinée.

Pour fonctionner, l'interpréteur dépend d'un ensemble de règles qui sont écrites dans un format semblable à celui du Prolog:

conséquent <- antécédent

qui se lit approximativement: **"une instance des variables qui rend l'antécédent vrai, rend le conséquent vrai"**. Aussi, elles sont déclenchées selon leur ordre. La première règle entrée est la première à être déclenchée.

Le rôle d'une règle dans **Arity/Expert** est d'affecter une valeur à un tirroir. Le conséquent de la règle indique quelle valeur sera affectée. L'antécédent contient les conditions qui doivent être vérifiées pour que la valeur soit affectée.

La syntaxe exacte de la composante principale d'une règle est:

[the | a] <tirroir> of [the | a] <cadre> is <valeur>

La coquille a été écrite pour développer des systèmes anglais d'où la présence de mots clefs anglais. Les parties entre crochets sont optionnelles (i.e.: les omettre ne change aucunement la règle) et ne sont présentes que pour rendre les règles plus lisibles. Ici

nous avons le choix d'utiliser un des deux mots (i.e.: "the" ou "a") mais pas les deux. Les mots entre les signes < et > sont une description de ce qui doit apparaître à cet endroit. Les mots "of" et "is" sont obligatoires.

Cette composante peut faire partie de l'antécédent d'une règle aussi bien que du conséquent. Dans le conséquent, il ne peut y avoir qu'une seule occurence de cette construction et elle se lit:

"le tirroir T du cadre C prend la valeur V"

Dans l'antécédent d'une règle, il peut y avoir une composante ou une conjonction de plusieurs composantes séparées par le mot "and". Une règle complète a la forme:

the <tirroir> of the <cadre> is <valeur>
if
the <tirroir 1> of the <cadre 1> is <valeur 1> and

the <tirroir n> of the <cadre n> is <valeur n>

Il faut noter aussi que dans l'antécédent d'une règle nous pouvons inclure des appels à des prédicats Prolog.

3.3.3.1 Déclenchement

Le système déclenche une règle quand il cherche à calculer la valeur d'un tirroir. Il trouve une règle dont le conséquent est l'affectation d'une valeur au tirroir désirée et vérifie l'antécedent. Soit l'antécédent:

"le tirroir T du cadre C a la valeur V"

Le système vérifie si le tirroir T a la valeur V. Si le système n'a pas encore affecté une valeur au tirroir, il cherche à déclencher une règle ayant ce tirroir dans sa partie conséquent. Si aucune règle ne correspond, il présente une liste des valeurs possibles à l'utilisateur et lui demande de choisir la réponse appropriée.

Voici la première règle déclenchée quand le système démarre:

```
the p_autorisation of c_voyage is v_non_autorise
    if
  the p_distance of destination is Distance and
  Distance =< 16
```

En déclenchant cette règle, le système tente d'affecter la valeur "v_non_autorise" au tirroir "p_autorisation". Dans l'antécédent, la première ligne sert à instancier la variable Prolog "Distance". Les politiques de voyages stipulent que pour être considéré un voyage, un déplacement doit être plus de 16 kilomêtres. Puisqu'aucune règle n'affecte de valeur au tirroir "p_distance", le système demandera à l'utilisateur de lui fournir une valeur. La deuxième ligne est un appel au prédicat Prolog "=<" pour vérifier la valeur de "Distance".

3.3.3.2 L'ordre et le groupement des règles

Puisque la coquille est un Prolog augmenté (voir section 3.3.4), l'ordre dans lequel les questions sont posées à un usager par ce système est une conséquence directe de l'ordre que nous avons choisi pour écrire les règles. En s'inspirant du scénario

que nous avons défini, nous avons ordonné les règles du système pour qu'il se comporte comme un employé du gouvernement qui prépare un voyage d'affaires. Il est alors important de réaliser qu'il n'existe aucune définition explicite du scénario dans le code du système expert. On le retrouve de façon implicite dans l'ordre du déclenchement des règles.

De plus, une considération importante lors de l'écriture de ce système est le groupement des règles. Les politiques couvrent principalement trois grandes catégories énumérées au début de ce chapitre. Le système a été développé en fonction de cette classification et il en résulte quatres groupes de règles:

- modes de transport commerciaux,
- modes de transport non commerciaux,
- l'hébergement,
- repas et faux frais.

Ainsi, toutes les règles représentant les politiques pour les transports commerciaux ont été regroupées.

3.3.3.3 Les censeurs

Un censeur est une règle qui sert à bloquer le déclenchement d'une autre règle. Son rôle est de détecter une condition exceptionnelle qui ferait que la règle principale ne doit pas réussir. Dans la section 2.4, les règles ont la forme:

Si	un faible noble est marié à une femme avare
alors	ce noble peut vouloir être roi
sauf si	la femme ne peut pas influencer le noble
	elle ne peut pas le convaincre de devenir roi

Lors d'un voyage, ce n'est pas raisonnable qu'un employé occupe la première classe dans un avion à moins d'avoir obtenu une autorisation spéciale. Une règle vérifiant ces conditions pourrait être écrite de la façon suivante:

Si	un employé prend l'avion et	(1)
	il occupe la première classe	
alors	la dépense n'est pas raisonnable	
sauf si	il a obtenu une autorisation spéciale	

Dans **Arity/Expert**, ce n'est pas possible d'écrire des règles ayant la partie "**sauf si**". Nous avons donc transformer cette règle sous une forme compréhensible par notre interpréteur. La nouvelle règle est:

```
the p_raisonnable of frais_transport is v_non                    (2)
    if
  the p_mode_de_transport of c_transport is v_avion and
  the p_classe of c_transport is v_premiere and
  the p_autorisation_classe of c_transport is v_non.
```

Nous avons inclus la partie "**sauf si**" dans l'antécédent de la règle en inversant sa condition. Dans (1), la partie "sauf si" s'assure que le voyageur a obtenu une autorisation spéciale. Dans (2), la règle vérifie que le voyageur ne l'a pas obtenu.

Une réalisation importante est que la majorité des documents détaillant des politiques, ne sont qu'une liste de descriptions de situations ou de conditions qui doivent être vérifiées. En somme, dans voyage, ce sont des censeurs qui servent à bloquer l'approbation des réclamations. Si aucun censeur ne réussit, la dépense est approuvée.

3.3.3.4 La négation

La première version de notre système était une tentative de traduction presque directe des politiques. Chaque paragraphe pouvait définir une règle. Une des règles que nous avons obtenues avait la forme suivante:

```
the p_raisonnable of frais_transport is v_oui                    (1)
    if
  the p_distance of destination is Distance and
  Distance >= 300 and
  the p_mode_de_transport of c_transport is v_avion and
  the p_type of c_transport is v_commercial and
  the p_ligne of c_transport is v_canadienne and
  the p_classe of c_transport is v_economique.
```

Les paragraphes suivants dans le document traitaient des situations semblables. Entre, dans la section des modes de transport, les conditions qui permettaient à un voyageur de prendre l'avion devaient être vérifiées à plusieurs reprises. Par exemple, quand l'avion est utilisé, un voyageur doit occuper un siège dans la classe économique d'une ligne canadienne. C'est ce que la règle (1) ci-haut indique.

Si le voyageur prend un avion d'immatriculation étrangère, il doit normalement en avoir obtenu l'autorisation avant son départ. Dans cette situation, il doit aussi normalement occuper un siège en classe économique. Une règle, comme (2), doit être introduite pour traiter la nouvelle situation. Remarquer, que la classe de voyage est à nouveau vérifiée.

```
the p_raisonnable of frais_transport is v_oui                    (2)
    if
  the p_distance of destination is Distance and
  Distance >= 300 and
  the p_mode_de_transport of c_transport is v_avion and
  the p_type of c_transport is v_commercial and
  the p_ligne of c_transport is v_autre and
  the p_autorisation_ligne of c_transport is v_oui and
  the p_classe of c_transport is v_economique.
```

En transformant ces règles en censeurs qui détectent une situation négative, nous pouvons réduire la complexité de certaines règles. Les règles (1) et (2) peuvent être réécrites de la façon suivante:

```
the p_raisonnable of frais_transport is v_non                    (3)
    if
  the p_mode_de_transport of c_transport is v_avion and
  the p_type of c_transport is v_commercial and
  the p_distance of destination is Distance and
  Distance < 300 and
  the p_autorisation_distance of c_transport is v_non.
```

```
the p_raisonnable of frais_transport is v_non                    (4)
    if
  the p_mode_de_transport of c_transport is v_avion and
  the p_type of c_transport is v_commercial and
  the p_ligne of c_transport is not v_canadienne and
  the p_autorisation_ligne of c_transport is v_non.
```

```
the p_raisonnable of frais_transport is v_non                    (5)
    if
  the p_mode_de_transport of c_transport is v_avion and
```

```
the p_type of c_transport is v_commercial and
the p_classe of c_transport is not v_economique and
the p_autorisation_classe of c_transport is v_non.

the p_raisonnable of frais_transport is v_oui                (6)
   if
the p_mode_de_transport of c_transport is v_avion and
the p_type of c_transport is v_commercial.
```

Ici, nous avons effectivement doublé le nombre de règles, mais nous avons aussi prévu plus que les deux situations mentionnées au début de la section. Ces règles couvrent toutes les combinaisons possibles de classes de voyage, de lignes aériennes et de distance.

3.3.4 Prolog

Arity/expert Shell est un interpréteur Prolog augmenté de provision pour l'écriture de systèmes experts. Ce lien étroit avec Prolog offre aux programmeurs tous les avantages d'une coquille pour système expert et la puissance d'un environnement Prolog.

3.3.4.1 Déclenchement

Pour utiliser notre système expert, il faut d'abord démarrer l'interpréteur **Arity/Expert**, ensuite il faut charger le code du système expert et finalement, il faut lancer son exécution. Il y a deux étapes au déclenchement du système expert:

- créer une instance du cadre "c_voyage" par un appel au prédicat "root_instance",

- un appel au prédicat "**eval**" démarre le système expert.

```
root_instance( c_voyage, I, N ),
eval( p_autorisation, c_voyage, I, Val, true, CF )
```

Nous pouvons inclure des appels à des clauses Prolog dans les règles du système expert.

```
the p_autorisation of c_voyage is v_non_autorise
     if
  the p_distance of destination is Distance and
  Distance < 16
```

Aussi, lorsque le comportement du système expert n'est pas adéquat, nous pouvons provoquer le déclenchement de clauses Prolog à l'aide des options de contrôl.

3.3.4.2 Explications

Puisque les explications générées par **Arity/Expert** ne répondaient pas à nos besoins, nous avons écrit un module d'explications en Prolog. Celui-ci comprend deux prédicats. Le premier, "**ex**", a deux paramètres dont le premier est un numéro de paragraphe et le deuxième est le texte retourné de la section correspondante. Le deuxième, "**explication**", sert à choisir et afficher le paragraphe approprié selon les paramètres qui lui sont passés.

Pour écrire le prédicat "**ex**", nous avons divisé le texte des politiques de voyage en blocs. Aucun critère formel n'a servi à la création de ces blocs. Nous les avons extrait du texte de façon à ce que chacun puisse expliquer le pourquoi d'une question ou les raisons d'une décision.

Le prédicat "**explication/3**" explique à l'utilisateur du système pourquoi une question a été posée. Le prédicat "**explication/4**" explique comment le système est arrivé à une conclusion. Une des raisons principales pour l'écriture de ces prédicats était que les explications dépendaient de l'état courant de la réseau d'inférence. Cependant, nous avons réalisé qu'un recul d'un noeud dans le réseau était suffisant pour déterminer quelle explication devait être fournie étant données les valeurs des propriétés impliquées. Par exemple, si le système évalue si le voyageur a l'autorisation de prendre une voiture pour se rendre à une destination à plus de 300 kilomètres, il est inutile, pour le module d'explication, de vérifier si le mode de transport est une voiture.

Nous avons choisi de n'offrir des explications que si une dépense n'était pas approuvée. Puisqu'un utilisateur cherche à vérifier si toutes ses dépenses sont acceptables, il ne sera pas intéressé à avoir des explications si elles sont approuvées. Par contre, si une dépense ne l'est pas, il cherchera à savoir pourquoi.

3.3.6 Forces et faiblesses du système

Dans la section 3.2, nous avons fixé quelques objectifs pour le système expert. Nous voulions qu'il soit facile à utiliser, facile à modifier, efficace et performant. Aussi, le module d'explications offre aux usagers une aide importante. Malgré ses

points forts, le système a une faiblesse importante: lorsqu'il a besoin d'information, il n'a pas a sa disposition une base de connaissances exhaustives et il doit demander beaucoup d'information à l'utilisateur.

La réalisation d'un système expert facile à utiliser dépendait surtout de la familiarité de l'information et du comportement du système aux usagers. L'information fournie au système expert par l'usager est tirée des formulaires qu'il doit remplir pour chaque voyage. Le comportement du système est un scénario qui vise à rendre l'utilisateur à l'aise pendant la consultation. Aussi, l'usager du système peut en tout temps demander des explications qui lui indiquent pourquoi une question lui est demandée et comment le système est arrivé à une réponse.

La facilité d'entretient de tous les systèmes informatiques est une des considérations les plus importantes lors de leurs développements. Pour en arriver à cette fin, nous avons divisé les règles du système expert en modules. Chaque module traite un sujet particulier de l'application. Entre autre, il existe un module traitant les politiques pour les transports commerciaux. Aussi, dans chaque module les règles ont été divisées selon des critères semblables, par exemple, pour les transports non-commerciaux, les règles pour les voitures privées ont été regroupées.

L'efficacité peut être estimée en comparant les résultats que le système produit aux résultats du système manuel. Les situations que nous avons essayées donnent des résultats qui semblent acceptables, mais la vrai mesure de l'efficacité du système sera produite par l'utilisation extensive du système.

Malgré le grand nombre de clauses Prolog, le système est beaucoup plus rapide lorsqu'il les déclenche qu'il ne l'est avec

les règles du système expert (185 claudes Prolog vs 64 règles **Arity/Expert**). Ceci dépend surtout de la création et la gestion du réseau d'inférence et de la manipulation des cadres en plus de l'exécution des fonctions propres au Prolog. Le résultat est un temps d'attente maximal entre chaque intervention par l'utilisateur de 2 à 3 secondes sur un IBM-AT.

Pendant une consultation, la source principale d'information pour le système est l'usager. Il fournit des données brutes telles que la destination, sa distance, la durée du voyage, etc. Beaucoup d'information est inutilement fournie par l'usager, par exemple, connaissant la destination d'un voyage, la distance pourrait aisément être automatiquement retrouvée par le système si elle était déjà enregistrée dans une base de données.

Si le répertoire des hôtels du **Ministère des Approvisionnement et Service** était à la disposition du système expert, il pourrait recommender un logement à un usager ainsi que vérifier si les tarifs imputés n'excèdent pas les montants permis.

Aussi, les politiques pour les frais de repas et les faux frais se rapportent tous à des montants forfétaires. Ces montants varient selon la destination du voyage. Si ces montants se trouvaient dans une base de données, le système expert pourrait facilement calculer l'autorisation de ces dépenses.

Il devient évident qu'une base de données ou une base de connaissances élargie contenant certaines informations serait un grand avantage pour ce type de système. La quantité d'interventions faites par l'usager serait réduites. Le système serait alors plus facile à utiliser et les réponses seraient meilleures étant donné une meilleure qualité d'information.

Ce type de système aurait pu être développé complètement à l'aide d'un langage comme Prolog. Plusieurs des faiblesses du système, entre autre l'applatissement de notre structure, n'aurait peut-être pas été soulevées. Mais un des objectifs qui avait été fixé au départ, était de produire un système expert à l'aide d'un produit du type de **Arity/Expert**.

4. CONCLUSIONS

La méthode que nous avons utilisée pour développer notre système est basée sur l'utilisation de scénarios, de réseaux sémantiques et de censeurs. Le résultat est un système expert du type consultant qui assiste des voyageurs à préparer leurs voyages. De même, c'est un système du type diagnostique qui assiste aux commis dans traitement les réclamations de dépenses. Dans les sections qui suivent, nous discuterons de notre méthode, de l'applicabilité de notre méthode, ses forces et ses faiblesses, ainsi que des suggestions de recherches futures.

4.1 La méthode

Il y a trois différentes facettes au développement d'un système expert:

- conceptuel (structure logique de l'information),
- implantation (structure physique de l'information),
- utilisation (comportement du système)

Le niveau conceptuel est la définition logique de toutes les structures d'information sous-jacentes au fonctionnement d'un système. Dans notre système expert, ces structures sont les connaissances sous forme de réseau sémantique, de scénario et de censeurs. L'implantation est la définition physique des structures d'information qui supportent le niveau conceptuel.

Les systèmes expert sont développés à partir de deux structures physiques d'information:

- les règles,
- les concepts

Les règles sont les connaissances qui permettent de déduire des nouveaux faits à partir de données connues. Les cadres sont les supports nécessaires à l'enregistrement et la conservation d'information. Ces deux structures nous ont permis d'exploiter la puissance de représentation des réseaux sémantiques, le comportement inhérent aux scénarios et la détection de dérogations aux politiques à l'aide de censeurs.

Pour utiliser le système, un usager ne doit pas connaître les structures physiques ou conceptuelles qui ont servies au développement du système. Ce qui lui importe est le comportement du système et la facilité de son utilisation.

4.1.1 Réseaux sémantiques

Pour que notre système ait à sa disposition toutes les informations nécessaires au traitement d'une réclamation, nous avons défini plusieurs concepts. Un concept est une idée, un objet ou une entité et peut être reconnu par la valeur de ses propriétés. Une propriété est un attribut ou une qualité d'un concept et sert à distinguer une instance d'un concept d'une autre.

Chaque concept, défini par un cadre dans notre système, est un noeud dans la structure d'information. Chaque noeud contient une série de tirroirs qui qui peuvent contenir des valeurs qui sont ou bien de type simple, ou un autre cadre.

```
define primitive c_voyage with
        ...
    r_aller  = c_aller and
    r_sejour = c_sejour and
```

```
        r_retour = c_retour.


define primitive c_aller with
        r_aller_local   = c_deplacement_local and
        r_transport     = c_transport and
        r_retour_local  = c_deplacement_local.
```

Les deux cadre ci-hauts peuvent être représentés graphiquement de la façon suivante:

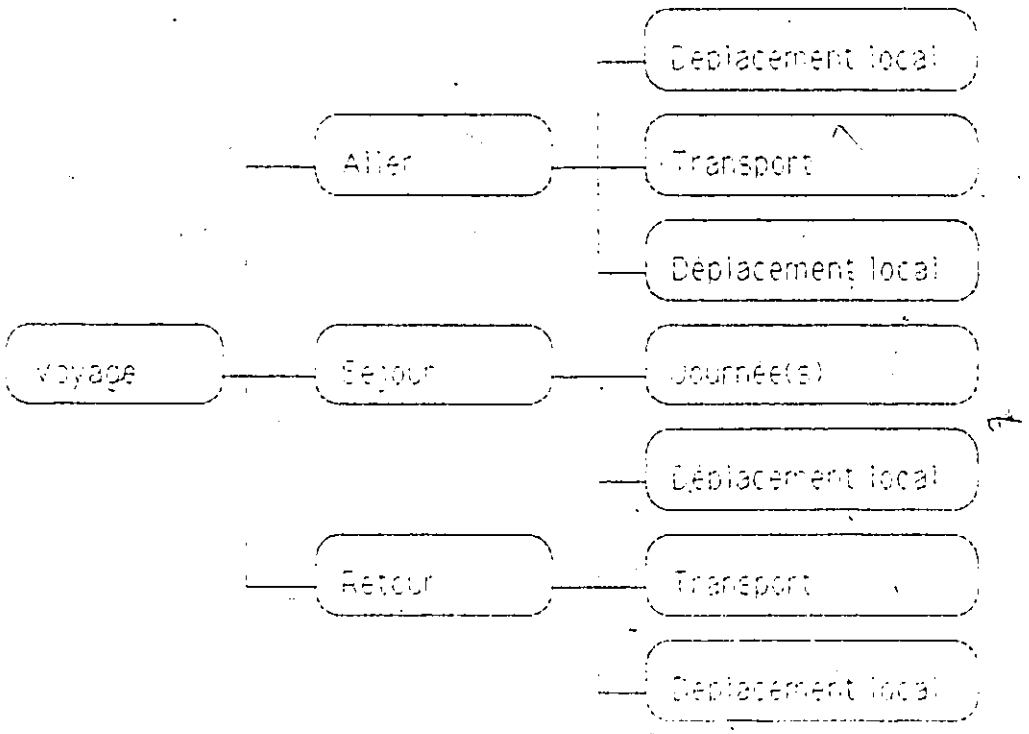


Figure 4.3

d'où l'appellation **"réseau sémantique"** pour la structure globale puisque les liens entre chaque cadre représentent une relation sémantique entre deux idées.

4.1.2 Scénarios

Un système informatique doit être aussi facile d'utilisation que possible puisque ses usagers n'ont généralement que très peu de formation. Par exemple, les système de gestion de bases de données tentent de faire afficher à l'écran des images de formulaires qui servent dans les systèmes manuels. Un tel système est plus facile à utiliser puisque les usagers sont familiers avec la présentation et avec le type d'information demandée.

Pour rendre notre système plus accessible aux usagers, nous avons choisi d'essayer de rendre son comportement le plus familier possible. Le système tente de passer par les mêmes étapes qu'une personne préparant un voyage en service commandé. Le scénario ainsi développé donne un comportement au système qui est familier à un voyageur.

Un des avantages du développement d'un scénario pour un tel système est l'existence de valeurs de défaut intelligent. Entre autre, quand l'usager n'est pas certain du mode de transport qu'il doit utiliser, il peut laisser le système lui faire une suggestion. La règle suivante en est une illustration:

```
the p_mode_de_transport of transport_aller is v_avion
    if
  the p_distance of detination is Distance and
  Distance > 300.
```

Étant donnée cette règle, si l'usager répond qu'il ne connaît pas le mode de transport et que la distance de la destination est supérieure à 300 kilomêtre, le système lui suggère l'avion.

4.1.3 Censeurs

Parfois, dans un système informatique qui doit rendre une décision quelconque, il s'avère plus efficace de détecter une situation anormale que d'essayer de vérifier si une situation est acceptable. [Winston 84] a introduit les censeurs comme mécanisme qui empêche le déclenchement d'une règle de réussir. Un censeur sert justement à détecter quand une situation ne se conforme pas aux normes établies.

Dans notre système, il est rapidement devenu évident qu'il était plus facile d'empêcher l'approbation d'une dépense que de prouver qu'elle était acceptable. En effet, pour qu'une dépense soit approuvée, elle doit se conformer à une série de politiques qui doivent toutes être vérifiées. En se servant de l'approche suggérée par [Winston 84], il suffit qu'une seule politique ne soit pas respectée pour rejeter la réclamation. Ainsi les règles que nous avons écrites sont beaucoup plus simples et vérifient beaucoup moins de conditions.

```
the p_raisonnable of frais_transport is v_non
    if
  the p_mode_de_transport of c_transport is v_avion and
  the p_classe of c_transport is v_premiere and
  the p_autorisation_classe of c_transport is v_non.
```

La règle ci-haut vérifie que le voyageur n'a pas obtenu l'autorisation de voyager en première classe. Si elle réussit, la dépense ne sera pas autorisée. Si elle ne réussit pas, la prochaine condition sera vérifiée par le déclenchement d'une autre règle.

4.2 Application de la méthode

Pour déterminer si notre méthode se prête au développement d'un système pour résoudre un problème quelconque, il faut que le domaine de l'application respecte certains critères. Entre autre, le type de politiques ou de règlementation est un des facteurs les plus importants à considérer. Aussi, nous devons avoir accès à l'expertise du domaine d'application.

4.2.1 Type des connaissances

Une politique est l'énoncé d'une règle qui suggère une ligne de conduite étant donnée une situation quelconque. Les politiques qui ont servies au développement de notre système se fondent surtout sur des connaissances identifiables. Chaque règle se rapporte à des faits vérifiables où la conclusion repose sur l'absence ou la présence de données précises. Par exemple, si le mode de transport est un avion, il doit être immatriculé au Canada à moins de conditions exceptionelles. Notre système peut rendre une décision dans cette situation. Par contre, les connaissances nécessaires pour admettre un nouveau produit sur le marché canadien avec ou sans tarifs douaniers sont difficiles à rendre explicites (voir section 4.2.5).

4.2.2 Accès à l'expertise

Il y a deux sources principales qui nous ont fourni l'expertise nécessaire au développement de notre système. La première est les sections 3, 4, 5 et 6 du chapitre 370 du Manuel de la Politique Administrative du Conseil du Trésor Canada. Nous avons extrait de ce document les connaissances brutes qui ont servi à écrire les règles du système expert. Il y a deux types de règles dans le système. Près de 50% des règles du système sont tirées directement du document. Chacune de ces règles a été écrite en s'inspirant d'une section particulière des politiques de voyage. Elles sont responsable de l'approbation ou le refus d'une réclammation. Le reste des règles sont dédiées au contrôle du comportement du système.

Les voyageurs et les personnes responsables de traiter les réclamation de dépenses sont la deuxième source de connaissances. Les employés du gouvernement au C.C.R.I.T. nous ont fourni les connaissances qui ne se trouvaient pas dans le document du Conseil du Trésor. Ces connaissances détaillaient le comportement d'une personne en voyage ainsi que le cheminement des documents de demande d'avance de fonds et de réclamation de dépenses. Elles nous ont permi de définir un scénario pour notre système. Ces connaissances n'ont aucune représentation explicite dans le système. Entre autre, il n'y a aucune règle qui indique que l'aller se fait avant le retour. Par contre, elles nous ont permi de définir un scénario pour le système et ainsi décider de l'ordre du déclenchement des règles.

Il faut noter que dans le système on retrouve deux types de connaissances:

- explicites,
- implicites.

Les connaissances explicites sont simples à retracer puisque ce sont les règles mêmes. Dans le code du système, on ne retrouve aucun détail explicite exprimant les différentes étapes du scénario. Ces connaissances sont inclus dans le système de façon implicite grâce à l'ordre du déclenchement des règles et le choix des valeurs de défauts.

4.2.3 Concepts définissables

Dans le réseau sémantique que nous avons créé pour notre système, chaque noeud est un concept. Chaque concept est une définition complète, pour nos fins, de l'objet qu'il représente. Pour tous les concepts dans notre système, il est nécessaire que chacune de ses propriétés qui puissent influencer les résultats produits par le système doit être connue à l'avance, ainsi que toutes les valeurs possibles qu'elles peuvent avoir. Par exemple, le concept "logement" est défini par le cadre suivant:

```
define primitive c_logement with
        any p_type and
        any p_propriete and
        any p_genre and
        any p_reservation_MAS and
        any p_repertoire and
        r_frais_logement = c_frais_logement
```

Pour notre système, cette définition est complète puisqu'elle comprend toutes les informations nécessaires à une prise de décision. Dans la section **4.2.5** il y a un exemple d'objets essentiels au fonctionnement du système lesquels nous ne pouvons définir que pendant l'utilisation du système.

4.2.4 Scénarios

Lorsqu'une personne part en voyage en service commandé, les activités qui se déroulent sont stéréotypées. Certaines de ces activités sont essentielles à l'accomplissement de ses tâches et ont un ordre chronologique inchangeable à suivre.

Pour qu'un système manuel puisse servir au développement d'un système expert par notre méthode, il doit inclure un comportement ayant une suite d'évènements ordonnés à accomplir, tous essentiels au succès de l'activité. Dans voyage, les étapes principales qui doivent avoir lieu, sont facilement identifiables. Dans la section 4.2.5, le système manuel décrit n'a pas de suite d'évènements définissables.

4.2.5 Contre exemple

Cette section rend compte d'un système gouvernemental manuel qui ne peut pas être automatisé en se servant de notre méthode. Il répond à certains de nos critères d'applicabilités, mais il ne rencontre pas toutes les exigences nécessaires.

Les ministères gouvernementaux sont généralement divisés en plusieurs départements qui ont des responsabilités très spécifiques. Au seins du ministère des **Énergies, Mines et Ressources**, le **Département d'Expansion Industrielle Régionale (D.E.I.R.)** est responsable d'autoriser l'exemption de tarifs douaniers sur des biens étrangers importés au Canada. D.E.I.R. est divisé à son tour en plusieurs branches qui chacune est responsable d'un type de biens. Au début de notre recherche, nous avons contacté la branche d'ingénierie électrique et de machinerie pour essayer de développer un système expert.

Malgré l'enthousiasme qu'il nous ont témoigné quand nous les avons approchés, nous ne pouvions pas produire un système expert pour eux en utilisant notre méthode.

Quand ils reçoivent une demande d'exemption de tarifs douaniers d'un manufacturier, les officiers qui traitent la demande doivent s'assurer que le produit ou un substitut acceptable n'est pas disponible sur le marché canadien. Si le produit n'est pas disponible, l'exemption est accordée, sinon il peut soit utiliser le produit sur le marché canadien ou il doit payer les tarifs douaniers.

Il y a trois raisons pourquoi notre méthode ne s'applique pas pour développer un système expert qui assisterait dans ce processus:

- il n'y a pas de scénario,
- ce type de système traite toujours des concepts qui ne sont pas définissables pendant le développement du système,
- beaucoup de l'expertise est informelle.

Quand l'officier reçoit la demande d'exemption, il n'y a aucune suite d'évènements qui inspire la définition d'un scénario. Son comportement se résume à comparer le produit décrit sur la demande d'exemption à sa liste de produits disponibles au Canada. Construire un scénario pour aider dans cette situation est inutile.

Chaque nouveau produit qui est le sujet d'une réclamation a des propriétés différentes. Donc c'est impossible de définir à l'avance tous les concepts dont aurait besoin ce système.

L'expertise dont se sert l'officier n'a pas été formalisée. Quand il doit évaluer une réclamation, il se fie à une photo du produit pour en retracer d'autres semblables. Il a lui-même de la difficulté à indiquer ce qu'il recherche dans la photo. C'est un procédé basé presqu'uniquement sur des connaissances intuitives difficiles à formaliser.

4.3 Faiblesses de la méthode

Un problème qui doit être adressé pendant le développement de tout système expert est l'accès aux données brutes. La majorité des systèmes rendent des décisions aux utilisateurs en se basant sur les faits à leur disposition. Dans certains cas, il suffit au système d'interroger l'usager. Ce comportement est acceptable lorsque l'information demandée ne peut pas être fournie autrement au système, mais beaucoup de faits nécessaires à une prise de décision sont souvent des données qui pourraient être conservées en permanence. Dans notre système, par exemple, si la distance entre les villes était enregistrée dans une liste de faits ou dans un fichier de base de données, beaucoup d'information pourrait être traitée sans que l'utilisateur n'ait à intervenir.

L'utilisation de censeurs pendant le développement offre des avantages qui seront détaillés dans la section suivante, mais il faut noter leur effet sur les performances du système. Pour accélérer le système, il faut réduire les étapes qu'il exécute pour déterminer la valeur d'un tirroir ou d'une réponse. L'utilisation de censeurs fait que le système doit, dans certains cas, déclencher plus de règles que si nous avions utilisé une règle comme la suivante:

```
the p_raisonnable of frais_transport is v_oui                    (1)
    if
     the p_distance of destination is Distance and
     Distance >= 300 and
     the p_mode_de_transport of c_transport is v_avion and
     the p_type of c_transport is v_commercial and
     the p_ligne of c_transport is v_canadienne and
     the p_classe of c_transport is v_economique.
```

En utilisant la règle ci-haut, le système ne déclenche qu'une règle si le voyage n'a rien d'exceptionel. Mais, pour vérifier les mêmes conditions avec des censeurs, nous devons déclencher une suite de règles semblable aux règles (3) à (6) de la section 3.3.3.4.

Le système est nécessairement plus lent puisqu'il déclenche plus de règles, mais si toutes les règles étaient écrites en s'inspirant de la règle (1), elles seraient beaucoup plus complexes.

Le dernier point à soulever est la fragilité du système. Les coquilles pour systèmes experts offrent peu d'outils pour protéger les applications contre les erreurs commises par les usagers. Dans notre système ce problème est rapidement devenu évident lorsqu'un utilisateur fournissait une réponse du mauvais type. Par exemple, si le système attend un chiffre pour indiquer notre choix et qu'on lui entre un caractère, le comportement est imprévisible et la consultation ne réussira pas. Nous pourrions aisément régler ce type de problème si nous faisions appel à des prédicats Prolog, mais un de nos objectifs était de nous servir d'une coquille pour nous libérer de ce type de tâche de programmation.

4.4 Avantages de la méthode

La saisie des connaissances est l'obstacle le plus difficile à surmonter pendant le développement d'un système expert. Les connaissances sont généralement le fruit de l'expérience des experts et sont difficiles à formaliser. En sachant à l'avance quel type de comportement nous voulons obtenir, nous pouvons mieux définir les connaissances nécessaires à la production d'un système. La méthode que nous avons développée fixe un système cible et encadre ainsi les connaissances à saisir. Un tel encadrement transforme la création de systèmes experts en un processus mieux défini.

Dans la section 2.2 nous avons revu une méthodologie qui produit un système expert dont les règles sont semblables aux nôtres. En se servant de censeurs plutôt que de règles conventionelles, nous avons pu réduire considérablement le nombre de règles dans le système résultant.

Étant donnés les critères que nous avons définis pour déterminer si notre méthode peut servir pour la création d'un système expert, cette étape du développement exige beaucoup moins d'efforts. Un informaticien constate rapidement si un système expert est réalisable.

Le système que nous avons produit a deux qualités qui le rendent plus familier aux usagers. En définissant un scénario pour le système, nous offrons à l'utilisateur un produit qui ressemble au système manuel. En plus, les explications fournies par le système sont extraites directement du texte source des politiques de voyage.

Aussi, quand un voyageur ne peut pas fournir toute l'information au système, celui-ci se sert de valeurs de défaut

pour continuer la consultation. Ces valeurs permettent au système d'offrir des suggestions aux usagers telles que choisir le mode de transport étant donnée la distance de la destination.

4.5 Recherches futures

L'objectif de cette recherche était de développer une méthodologie qui faciliterait la création de systèmes experts. Quoique nous avons introduit une méthode qui atteint cet objectif, il reste encore beaucoup de travail à accomplir.

La première tâche serait d'éliminer les faiblesses du système expert. Le problème le plus important à résoudre serait de développer un interface à une base de données. Cet interface libérerait l'usager de la tâche d'entrer les informations qui pourraient facilement être enregistrées dans une base de données. Il réduirait aussi les possibilités de mal répondre aux questions du système et le rendrait un peu plus robuste.

Étant donnée la fragilité du système au niveau de l'interface homme-machine, ce serait important d'y porter une certaine attention. Les interfaces qui existent pour les systèmes experts sont surtout à base de questions-réponses. Malgré un comportement acceptable pour plusieurs systèmes, certains changements pourraient peut-être rendre les systèmes plus robustes ainsi que plus faciles à utiliser.

Beaucoup de recherche se fait pour trouver des techniques d'apprentissage de l'ordinateur. En appliquant ces nouvelles technologies au développement de systèmes experts, nous pourrions créer des outils qui permettent aux systèmes d'apprendre des nouvelles règles à partir d'exemples lorsque c'est nécessaire. Ce serait un atout important puisqu'ils réduiraient la tâche des

informaticiens et permettraient aux utilisateurs d'ajouter eux-mêmes des nouvelles connaissances à leurs systèmes.

Le traitement des langues naturelles par l'ordinateur est un domaine de recherche en pleine expansion. Cette technologie permet aux ordinateurs de lire du texte écrit en français, en anglais ou en une autre langue, et de le traduire en une représentation interne qu'ils comprennent. Aujourd'hui, ces représentations ne servent presque pas dans les systèmes informatiques. Étant donné le format très particuliers des textes de politiques, les représentations internes des ces documents pourraient peut-être traduites en connaissances utilisables par une coquille pour systèmes experts.

APPENDICE A

Dans les pages suivantes, vous retrouverez les documents qui doivent être remplis par les voyageurs avant et après avoir participé à un voyage en service commandé.

Le premier document, **AUTORISATION DE VOYAGER ET AVANCE**, doit être rempli avant le départ du voyageur. Sur ce formulaire, le voyageur doit fournir ses coordonnées au gouvernement, le but du voyage, la destination, le coût estimatif et la schédule de ses déplacements. Une fois complété, il doit faire autoriser la dépense et faire parvenir le document au service des finances de son département.

Il y a aussi au bas de ce formulaire une section de demande d'avance de fond pour que le voyageur puisse en faire la demande s'il en sent le besoin.

Normalement, si la dépense a été autorisée par le directeur du département, le voyageur devrait pouvoir faire sa réclamation de dépenses sans problème.

A son retour, le voyageur doit remplir deux documents. Le premier est la **DEMANDE D'INDEMNITÉ DE DÉPLACEMENT**. Sur celui-ci, il indique les déboursés globaux pour le déplacement, l'hébergement, les repas et les faux frais. Le deuxième, **RELEVÉ DES FRAIS DE DÉPLACEMENT**, est une formule détaillée des dépenses sur laquelle chaque dépense doit être inscrite et vérifiée par un reçu.

Ces deux formulaires sont envoyés au service des finances qui doit vérifier chaque dépense et ensuite émettre un chèque au voyageur.

Government of Canada — Gouvernement du Canada

1

S.I. No. — N° d'assurance sociale	Travel No. — N° du voyage

Type 1	Name of Traveller — Nom du voyageur

Send cheque to — Envoyer le chèque à l'adresse suivante

TRAVEL AUTHORITY AND ADVANCE
AUTORISATION DE VOYAGER ET AVANCE

Department — Ministère	Branch — Direction
Address — Adresse	Telephone — Téléphone

TRAVEL AUTHORITY — AUTORISATION DE VOYAGER

Purpose of Travel — But du voyage	Duration Days / Durée jours	Estimated Cost — Coût estimatif: Transportation / Transport	Estimated Cost — Coût estimatif: Other Expenses / Autres dépenses
		$	$

DEPARTURE — DÉPART				ARRIVAL — ARRIVÉE			
Date	Time / Heure	Place — Lieu	Mode	Date	Time / Heure	Place — Lieu	May be contacted at / On peut rejoindre la personne à

☐ Coach or Economy / Classe touriste ☐ First Class / Première classe ☐ Car Rental / Voiture louée ☐ Private Vehicle / Véhicule particulier @ ________ ☐ Mile — La mille ☐ km

☐ Composite Allowance / Indemnité détaillée @ $ ______ day / jour ☐ Comprehensive Allowance / Indemnité globale @ $ ______ day / jour

Traveller — Voyageur	Remarks — Remarques
Signature ______ Date ______	
Recommended by — Recommandée par	Approved by — Approuvée par
Signature ______ Date ______	Signature ______ Date ______

REQUEST FOR ADVANCE — DEMANDE D'AVANCE

Type 3	Particulars — Détails	Cheque Amount / Montant du chèque $	Date cheque required / Chèque requis le	Y-A M D-J

Type 4	ACCOUNTING INFORMATION — RENSEIGNEMENTS DE LA COMPTABILITÉ		
Sub Type	Dept. Ref. No. / N° de réf. min.	Coding — Codage	Amount — Montant

Requisitioned for payment pursuant to section 26 of the Financial Administration Act and certified in accordance with subsection 7(1) of the Account Verification and Payment Requisition Regulations. Demandé de paiement aux termes de l'article 26 de la Loi sur l'administration financière. Certifié conformément au paragraphe 7(1) du Règlement sur la vérification des comptes et la demande de paiement. Signature	Verified correct — Vérifié conforme. Signature. Services Officer — Agent des services. Signature	Cheque No. — N° de chèque / Date

GC 12

INSTRUCTIONS

* See accompanying instructions issued by the Treasury Board and your department.
* Voir les instructions du Conseil du Trésor et de votre ministère.

Government of Canada / Gouvernement du Canada

TRAVEL EXPENSE CLAIM
DEMANDE D'INDEMNITÉ DE DÉPLACEMENT

EXPENSES PAID BY DEPARTMENT — DÉPENSES PAYÉES À L'AVANCE PAR LE MINISTÈRE

1. TRANSPORTATION / TRANSPORT — Date / Time of Departure / Heure du départ — Airline / Aérien — $
2. OTHER EXPENSES / AUTRES DÉPENSES — $

AUTHORIZED ALLOWANCES — INDEMNITÉS AUTORISÉES

PRIVATE VEHICLE ALLOWANCE / INDEMNITÉ VÉHICULE PARTICULIER	¢	¢	¢	
PREVIOUSLY CLAIMED THIS FISCAL YEAR / DÉJÀ RÉCLAMÉ PRÉSENT EXERCICE				
THIS CLAIM / PRÉSENTE DEMANDE				$

OR — OU

COMPOSITE ALLOWANCE (Meals and incidental expenses for full calendar days only) / INDEMNITÉ DÉTAILLÉE (Repas et frais divers pour les jours de calendrier complets seulement)

COMPLEMENTARY ALLOWANCE (Accommodation, meals and incidental expenses) / INDEMNITÉ GLOBALE (Logement, repas et frais divers)

RATE / TAUX		____ days / jours @ $ ____	____ days / jours @ $ ____	____ days / jours @ $ ____	$
PARTIAL COMPOSITE ALLOWANCES / INDEMNITÉ DÉTAILLÉE PARTIELLE	MEALS / REPAS	Breakfast / Petit déjeuner ____ @ $ ____	Lunches / Déjeuners ____ @ $ ____	Dinners / Dîners ____ @ $ ____	$
	INCIDENTAL EXPENSES / FRAIS DIVERS	____ days / jours @ $ ____	____ days / jours @ $ ____	____ days / jours @ $ ____	$

OTHER ALLOWANCES (Specify) / AUTRES INDEMNITÉS (Préciser les genres)

EXPENSES NOT INCLUDED ABOVE — DÉPENSES AUTRES QUE CI-DESSUS

3. TRANSPORTATION / TRANSPORT — Veh. No. — P. N° — $
4. ACCOMMODATION / LOGEMENT — $
5. MEALS / REPAS — $
6. ALL OTHER EXPENSES (Specify) / AUTRES DÉPENSES (Préciser) — $

I certify that the amounts included in this claim were incurred on authorized government business travel.
Je certifie que les montants inscrits dans la présente réclamation concernent des dépenses engagées par des employés pour affaires officielles.

Signature — Date

Certified pursuant to section 34 of the Financial Administration Act.
Certifié en vertu de l'article 34 de la Loi sur l'administration financière.

Signature — Date

Approved by — Approuvé par

Signature — Date

TOTAL PAYABLE / TOTAL À PAYER — $
TOTAL ADVANCE / MOINS L'AVANCE — $
AMOUNT DUE EMPLOYEE / MONTANT DÛ À L'EMPLOYÉ — $
AMOUNT DUE GOVERNMENT / MONTANT DÛ AU GOUVERNEMENT — $

Type — Particulars — Details

ACCOUNTING INFORMATION — RENSEIGNEMENTS DE LA COMPTABILITÉ

Sub Type	Doc. Ref. No. / N° de réf. doc.	Coding — Codage	Amount — Montant	GST/QST
		Description	Fin. Encumbrance No. / N° de la charge financière	

Pre-audited for payment pursuant to section 34 of the Financial Administration Act and certified in accordance with subsection 5(1) of the Treasury Verification and Payment Regulations.
Examen préalable aux fins de paiement en vertu de l'article 34 de la Loi sur l'administration financière. Certifié conformément au paragraphe 5(1) du Règlement sur la vérification et le paiement du Trésor.

Signature

Verified correct — Vérifié conforme

Signature

Certified (Officer — Agent autorisé)

Signature

Cheque No. / N° du chèque — Date

INSTRUCTIONS

Government of Canada — Gouvernement du Canada

RECORD OF TRAVEL EXPENSES — RELEVÉ DES FRAIS DE DÉPLACEMENT

Page ___ of / de ___

SI No. — N° d'assurance sociale	Travel No. — N° de voyage

Name — Nom

DATE	Time of arrival - departure / Heure d'arrivée et de départ	DETAILS (Including mode of transportation, names of places and number of meals) / DÉTAILS (Mode de transport, destinations, nombre de repas)	DISTANCE: Miles / Milles	DISTANCE: Kilometres	Expenses prepaid by department / Dépenses payées à l'avance par le ministère: ① Transportation / Transport	② Other expenses / Autres dépenses	Expenses not prepaid by department and not included in authorized allowances / Dépenses pour lesquelles le ministère n'a fait aucune avance et qui ne sont pas couvertes par une indemnité: ③ Transportation / Transport	④ Accommodation / Logement	⑤ Meals / Repas	⑥ Other expenses / Autres dépenses	Vr. No. / P. N°
		SUB-TOTAL / SOUS-TOTAL									
		SUB-TOTAL FROM PREVIOUS PAGE / SOUS-TOTAL DE LA PAGE PRÉCÉDENTE									
		CUMULATIVE TOTAL / TOTAL CUMULATIF									

APPENDICE B

Le système expert (les différentes sections sont des extraits)

1. La taxonomie

```
/*****************************************************************************
* Conventions établies pour les structures de la taxonomie                   *
* -------------------------------------------------------------------------- *
*                                                                            *
* Tous les concepts, roles, propriétés et valeurs définis par des identifi-  *
* cateurs sont préfixes de la façon suivante:                                *
*                                                                            *
*    -> concept          c_                                                  *
*    -> role             r_                                                  *
*    -> propriété        p_                                                  *
*    -> valeur           v_                                                  *
*                                                                            *
*****************************************************************************/
/*
  Déclarations des types
  ----------------------
*/

/* Types numériques */

type p_distance    = numeric.
type p_duree       = numeric.
type p_montant     = numeric.

/* Types liste */

type p_autorisation = [v_autorise,
                       v_non_autorise,
                       v_aller,
                       v_sejour,
                       v_retour].

type p_classe = [v_economique,
                 v_troisieme,
                 v_deuxieme,
                 v_premiere,
                 v_chambrette].

type p_mode_de_transport = [v_autobus,
                            v_autre,
                            v_avion,
                            v_bateau,
                            v_motocyclette,
                            v_taxi,
                            v_train,
                            v_voiture].

type r_aller               = role.
type r_frais_jour          = role.
```

```
type r_frais_logement    = role.
type r_frais_repas       = role.
type r_frais_sejour      = role.
type r_frais_transport   = role.
type r_logement          = role.
type r_retour            = role.
type r_sejour            = role.
type r_transport         = role.

/*
  Définition des concepts
  -----------------------
*/

define primitive c_aller with
       r_aller_local  = c_deplacement_local and
       r_transport    = c_transport and
       r_retour_local = c_deplacement_local.

define c_frais with
       p_montant = (0.0, 2000.0) and
       any p_remboursement and
       any p_recu and
       any p_raisonnable and
       p_autorisation = [v_autorise, v_non_autorise].

define c_frais_sejour as a c_frais.
define c_frais_logement as a c_frais.
define c_frais_transport as a c_frais.

define primitive c_retour with
       any p_meme_mode and
       r_aller_local  = c_deplacement_local and
       r_transport    = c_transport and
       r_retour_local = c_deplacement_local.

define primitive c_sejour with
       r_frais_sejour   = c_frais_sejour and
       r_jour           = c_jour and
       number_of r_jour = (1.0, 365.25).

define primitive c_transport with
       r_frais_transport = c_frais_transport and
       any p_autorisation_classe and
       any p_autorisation_duree and
       any p_autorisation_ligne and
       any p_autorisation_mode and
       any p_autorisation_taxi and
       any p_cargaison and
       any p_classe and
       any p_conducteur and
       any p_demande_de_employeur and
       any p_dimension_voiture and
           p_duree = (0.0, 24.0) and
```

```
        any p_ligne and
        any p_mode_de_transport and
        any p_reservation_MAS and
        any p_type and
        any p_type_voiture.

define primitive c_voyage with
        r_commanditaire = c_commanditaire and
        r_commandite    = c_commandite and
        r_destination   = c_destination and
        r_temps_depart  = c_instant and
        r_temps_retour  = c_instant and
        p_duree         = (0.0, 5000.0) and
        any p_province and
        any p_autorisation and
        r_aller  = c_aller and
        r_sejour = c_sejour and
        r_retour = c_retour.
```

2. Les options de contrôle

```
default(p_raisonnable of c_frais_logement)  = v_oui/0.9.
default(p_raisonnable of c_frais_transport) = v_oui/0.9.
default(p_raisonnable of c_frais_sejour)    = v_oui/0.9.
default(p_raisonnable of c_frais_jour)      = v_oui/0.9.
default(p_raisonnable of c_frais_repas)     = v_oui/0.9.
default(p_raisonnable of c_faux_frais)      = v_oui/0.9.


default_option(why)  = pourquoi.
default_option(how)  = comment.
default_option(name) = ignore.


order( p_mode_de_transport of c_transport ) = [ q, r, i ].
order( p_type of c_transport )              = [ r, q, i ].
order( p_classe of c_transport )            = [ r, q, i ].


question(p_destination of c_destination) =
        $Quelle est la destination du voyage?$.

question(p_distance of c_destination) =
        $Quelle en est sa distance?$.


set(r_frais_transport of c_transport) = frais_transport.

set(r_aller of       c_voyage) = aller.
set(r_commandite of  c_voyage) = voyageur.
set(r_destination of c_voyage) = destination.
set(r_retour of      c_voyage) = retour.
set(r_sejour of      c_voyage) = sejour.

set(r_transport of   r_aller of       c_voyage) = transport_aller.
set(r_transport of   r_retour of      c_voyage) = transport_retour.

set(r_frais_transport of r_transport of r_aller of  c_voyage) = frais_aller.
set(r_frais_transport of r_transport of r_retour of c_voyage) = frais_retour.


synonym(c_aller)       = $aller$.
synonym(c_deplacement) = $déplacement$.
synonym(c_destination) = $destination$.
synonym(c_frais)       = $frais$.
synonym(c_retour)      = $retour$.
synonym(c_transport)   = $transport$.
synonym(c_voyage)      = $voyage$.
```

3. Les règles

La première série de règles sont les premières à être déclenchées quand le système démarre.

```
the p_autorisation of c_voyage is [ v_non_autorise ]
    if
  the p_province of c_voyage is unknown.

the p_autorisation of c_voyage is [ v_non_autorise ]
    if
  the p_destination of destination is unknown.

the p_autorisation of c_voyage is [ v_non_autorise ]
    if
  the p_distance of destination is Distance and
  Distance < 16.

the p_autorisation of c_voyage is [ v_non_autorise ]
    if
  the p_duree of c_voyage is Duree and
  Duree < 1.

the p_autorisation of c_voyage is [ v_aller, v_sejour, v_retour ]
    if
  the p_remboursement of frais_aller is not v_aucun and
  the p_remboursement of frais_retour is not v_aucun and
  the p_remboursement of frais_sejour is not v_aucun.
```

Cette série de règles servent à la vérification des transport commerciaux.

```
% 3.2.1.0
the p_raisonnable of frais_transport is v_non
    if
  the p_mode_de_transport of c_transport is v_avion and
  the p_type of c_transport is v_commercial and
  the p_ligne of c_transport is not v_canadienne and
  the p_autorisation_ligne of c_transport is v_non.

% 3.2.2.0
the p_raisonnable of frais_transport is v_non
    if
  the p_mode_de_transport of c_transport is v_avion and
  the p_type of c_transport is v_commercial and
  the p_classe of c_transport is not v_economique and
  the p_autorisation_classe of c_transport is v_non.

/*-------------------------------------------------------------------------
  Étant donné que la ligne et la classe ont été évaluées dans le cas de
  l'avion commercial, forcer la valeur de raisonnable avec un facteur de
  confiance = 1.
-------------------------------------------------------------------------*/

the p_raisonnable of frais_transport is v_oui
    if
  the p_mode_de_transport of c_transport is v_avion and
  the p_type of c_transport is v_commercial.

the p_raisonnable of frais_transport is v_oui
    if
  the p_mode_de_transport of c_transport is v_autobus.

/*-------------------------------------------------------------------------
  Calcul du type de remboursement (Fournit-on le billet ou l'argent pour le
  payer)
-------------------------------------------------------------------------*/

% 3.1.1.0 - 3:1.1.1
the p_remboursement of frais_transport is v_billet
    if
  the p_type of c_transport is v_commercial and
  the p_raisonnable of frais_transport is v_oui and
  the p_reservation_MAS of c_transport is v_oui.

% 3.1.1.0
the p_remboursement of frais_transport is v_argent
    if
  the p_type of c_transport is v_commercial and
  the p_raisonnable of frais_transport is v_oui and
  the p_recu of frais_transport is v_oui.
```

4. Prolog

La première section contient la clause pour déclencher le système expert ainsi que quelques utilitaires.

```
/*-------------------------------------------------------------------------
  Le déclenchement du système expert
-------------------------------------------------------------------------*/

voyage
   :-   ouvririmprimante,
        repeat, nl, nl,
        [!
           une_fois,
           confirme(Fc, $Désirez-vous une autre consultation$, Reponse)
        !],
        gc(full),
        Reponse = non,
        fermerimprimante.

une_fois
   :-   root_instance(c_voyage,I,N),
        eval(p_autorisation , c_voyage, I, Val, true, CF),
        fail.

une_fois
   :-  nl, nl.

/*-------------------------------------------------------------------------
  Des clauses utilitaires
-------------------------------------------------------------------------*/

egal(X,X).

not_calced(Chaine) :- Chaine = '$not_calced'.
calced(Chaine)     :- Chaine \= '$not_calced'.

/*-------------------------------------------------------------------------
  Retourne le nombre de jours qui doivent être créés
-------------------------------------------------------------------------*/

n_jour(Instance, Role, Concept, Minimum, Maximum, Valeur) :-
       get_parent(Instance, Voyage),
       get_cf(Voyage, p_duree, Valeur, Fc).

pourquoi(Instance, Concept, Propriete, D) :-
                  explication(Instance, Propriete, Concept), nl,
                  pause, nl, nl, nl.

comment(Instance, Concept, Propriete, Valeur) :-
                  explication(Instance, Concept, Propriete, Valeur), nl,
                  pause, nl, nl, nl.
```

L'interface aux explications.

```
% explication/3

explication(Transport, p_autorisation_mode, c_transport) :-
    case([
          get_cf(Transport, p_mode_de_transport, v_avion, FC) ->
              (ex('.3.1.0.1', Explication), nl,
               imprime('.3.1.0.1', Explication),
               ecrireln('.3.1.0.1'),
               ecrireln($
  Autorisation du mode de transport.$),
               ecrireln(Explication)),

          get_cf(Transport, p_mode_de_transport, v_voiture, FC) ->
              (ex('.4.1.1.0', Explication), nl,
               imprime('.4.1.1.0', Explication),
               ecrireln('.4.1.1.0'),
               ecrireln($
  Autorisation du mode de transport.$),
               ecrireln(Explication)),

          get_cf(Transport, p_mode_de_transport, Mode, FC) ->
              (ex('.3.1.0.1', Explication), nl,
               imprime('.3.1.0.1', Explication),
               ecrireln('.3.1.0.1'),
               ecrireln($
  Autorisation du mode de transport.$),
               ecrireln(Explication))

        |. fail]).

explication(Instance, p_mode_de_transport, c_transport) :-
    ex('.3.1.0.1', Explication1), nl,
    ecrireln('.3.1.0.1'),
    imprime('.3.1.0.1', Explication1),
    ecrireln($
  Mode de transport.$),
    ecrireln(Explication1),
    pause,
    ex('.4.1.1.0', Explication2), nl,
    imprime('.4.1.1.0', Explication2),
    ecrireln('.4.1.1.0'),
    ecrireln(Explication2).

explication(_, _, _) :-
    nl,
    ecrireln($        ****************************************$),
    ecrireln($        **** AUCUNE EXPLICATION DISPONNIBLE ****$),
    ecrireln($        ****************************************$).
```

```
% explication/4

explication(Instance, c_frais_transport, p_remboursement, [v_aucun / _]) :-
    assert(explication),
    get_parent( Instance, Transport ),
    imprime(' ', $Le(s) règlement(s) suivant(s) n'ont pas été respecté(s):$),
    ecrireln($Le(s) règlement(s) suivant(s) n'ont pas été respecté(s):$),
    get_cf( Transport, p_mode_de_transport, Mode, _ ),
    case(
        [get_cf(Transport, p_autorisation_mode, v_non, FC) ->
             explication(Transport, p_mode_de_transport, c_transport),

         get_cf(Transport, p_autorisation_ligne, v_non, FC) ->
             (explication(Transport, p_ligne, c_transport),
              explication(Transport, p_autorisation_ligne, c_transport)),

         ((egal(Mode, v_avion) ; egal(Mode, v_train)),
          get_cf(Transport, p_autorisation_classe, v_non, FC) ) ->
             (explication(Transport, p_classe, c_transport), pause,
              explication(Transport, p_autorisation_classe, c_transport)),

         (egal(Mode, v_voiture),
          get_cf(Transport, p_conducteur, v_autre, FC) ) ->
             explication(Transport, p_conducteur, c_transport),

         (egal(Mode, v_voiture),
          get_cf(Transport, p_type_voiture, v_privee, FC) ) ->
             ecrireln($VOITURE PRIVÉE NON-AUTORISÉE$),

         (egal(Mode, v_voiture),
          get_cf(Transport, p_type_voiture, v_louee, FC) ) ->
             (explication(Transport, p_dimension_voiture, c_transport),
              explication(Transport, p_cargaison, c_transport))

         | fail]),
    retract(explication).

explication(Instance, c_frais_logement, p_remboursement, [v_aucun / _]) :-
    assert(explication),
    get_parent( Instance, Logement ),
    imprime(' ', $Le(s) règlement(s) suivant(s) n'ont pas été respecté(s):$),
    ecrireln($Le(s) règlement(s) suivant(s) n'ont pas été respecté(s):$),
    get_cf( Instance, p_recu, v_non, _ ),
    explication( Instance, p_recu, c_frais_logement ),
    retract(explication).

explication(_,_,_,_) :-
    nl,
    ecrireln($          ****************************************$),
    ecrireln($          **** AUCUNE EXPLICATION DISPONNIBLE ****$),
    ecrireln($          ****************************************$).
```

Les explications.

ex('.3.1.0.1', $
Du souci d'économiser l'énergie et de la montée en flèche des frais de transport est né le besoin de se refaire une opinion sur les moyens de transport généralement admis, notamment pour les déplacements entre deux grands centres situés à proximité l'un de l'autre. Bien que le facteur temps soit important, il est douteux qu'il soit avantageux de prendre l'avion pour les trajets courts, et d'autres modes de transport possibles, qui pourraient être considérés comme inadmissibles pour de plus longs trajets, peuvent dans ce cas être jugés préférables. Pour les distances de moins de 300 kilomètres dans un sens, l'autocar et le train se présentent comme des solutions économiques et commodes par rapport à l'avion et au véhicule automobile.$).

ex('.3.1.2.0', $
L'autorisation fait l'objet de l'article .2.1.4 et la preuve de paiement, de l'article .10.5.4. L'employé doit joindre à sa demande de remboursement de frais de voyage les talons des billets de transport commercial, qu'il ait ou non lui-même acheté ces derniers.$).

ex('.4.1.1.0', $
S'il y a plus de 300 kilomètres à parcourir, l'utilisation d'un véhicule conduit par l'employé ne devrait pas normalement être autorisée à moins que le voyage par transport commercial ne présente des inconvénients majeurs ou que l'employeur ne le juge pas pratique sous le rapport du coût global, y compris le salaire, les dépenses personnelles et les frais de voyage. Voir l'appendice B, qui indique les distances approximatives entre les agglomérations canadiennes. Pour les voyages sur de courtes distances, il faut faire preuve de discernement pour déterminer s'il y a lieu d'autoriser l'utilisation des véhicules automobiles particuliers, surtout s'il est possible de recourir à des moyens de transport commerciaux commodes et plus économiques, à moins que l'utilisation d'un tel véhicule ne s'impose en raison de certaines circonstances, dont la nécessité de transporter plusieurs passagers.$).

ex('.5.1.1.0', $
L'employé est remboursé des frais réels et raisonnables de logement commercial autorisé par l'employeur. Lorsque ces frais excèdent $$13.50 par nuit, il doit fournir des pièces justificatives (voir le sous-article .10.5.4 c)). Il appartient à l'employeur de choisir le logement destiné aux voyageurs. Toutefois, à moins que des raisons de service n'imposent l'utilisation d'un logement spécial, l'employeur doit tenir compte de la demande d'un employé qui veut occuper un logement commercial ou non commercial. Les chambres à un lit doivent être autorisées par l'employeur s'il y en a de libres dans des établissements bien situés et confortables.$).

[Callero Waterman Kipps 84]
Callero, M., Waterman, D. A., Kipps, J., TATR: A PROTOTYPE EXPERT SYSTEM FOR TACTICAL AIR TARGETING, Rand report R-3096-ARPA, The Rand Corporation, Santa Monica, CA, 1984.

[Clancey 79]
Clancey, W. J., TUTORING RULES FOR GUIDING A CASE METHOD DIALOGUE, International Journal of Man-Machine Studies, vol. 11, pp. 25-49, 1979.

[Hammond 86]
Hammond, K. J., CHEF: A Model of Case-based Planning, Proceedings of the Fifth National Conference on Artificial Intelligence, AAAI-86, pp. 267-271, 1986.

[Kimura et al. 85]
Kimura, M., Matsumura, Y., Matsunage, T., Hata, R., Matsumura, H., RHINOS: A Consultation System for Diagnoses of Headache and Facial Pain, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, pp. 393-396, 1985.

[McDermott 82]
McDermott, J., R1: A RULE-BASED CONFIGURER OF COMPUTER SYSTEMS, Artificial Intelligence, vol. 19, pp. 39-88, 1982.

[Nelson 82]
Nelson, W. R., REACTOR: AN EXPERT SYSTEM FOR DIAGNOSIS AND TREATMENT OF NUCLEAR REACTOR ACCIDENTS, AAAI Proceedings, 1982.

[Scarl et al. 85]
Scarl, A. E., Jamieson, J. R., Delaune, C. I., A Fault Detection and Isolation Method Applied to Liquid Oxigen Loading For the Space Shuttle, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, pp. 414-416, 1985.

[Schank Abelson 77]
Schank, Roger, Abelson, Robert, SCRIPTS PLANS GOALS AND UNDERSTANDING, Lawrence Erlbaum Associates, Publishers, 1977.

[Shortliffe 76]
Shortliffe, E. H., COMPUTER BASED MEDICAL CONSULTATIONS : MYCIN, New York: Elsevier, 1976.

[Shortliffe Fagan 82]
Shortliffe, E. H., Fagan, L. M., EXPERT SYSTEMS RESEARCH: MODELING THE MEDICAL DECISION MAKING PROCESS. HEURISTIC PROGRAMMING PROJECT REPORT HPP-82-3, Departments of Medicine and Computer Science, Stanford University, Stanford, CA, March 1982.

[Skuce et al. 85]
Skuce, D., Matwin, S., Tauzovich, B., Oppacher, F., Szpakowicz, S., A LOGIC-BASED KNOWLEDGE SOURCE SYSTEM FOR NATURAL LANGUAGE DOCUMENTS, Data & Knowledge Engineering 1, 1985, pp. 201-231, North-Holland.

[Tsuji Shortliffe 83]
Tsuji, S., Shortliffe, E. H., GRAPHICAL ACCES TO THE KNOWLEDGE BASE OF A MEDICAL CONSULTATION SYSTEM. HEURISTIC PROGRAMMING PROJECT REPORT HPP-83-6, Departments of Medicine and Computer Science, Stanford University, Stanford, CA, February 1983.

[Waterman 85]
Waterman, Donald A., A GUIDE TO EXPERT SYSTEMS, Addison-Wesley, 1985.

[Winston 84]
Winston, Patrick H., ARTIFICIAL INTELLIGENCE, second edition, Addison-Wesley, 1984.