88/175

Université de Nancy I

Centre de Recherche en
Informatique de Nancy
Inria Lorraine

Sc N 88/461 A

DIAL,
la composante dialogue d'un système de communication orale homme-machine finalisée en langage naturel

THÈSE

BIBLIOTHEQUE-SCIENCES * UNIVERSITE NANCY *

présentée et soutenue publiquement le 1er **Avril 1988**

pour l'obtention du **doctorat de l'Université de Nancy I**
(Mention Informatique)

par

Azim Roussanaly

Composition du jury :

Président :	Jean-Paul HATON
Rapporteurs :	Gérard SABAH
	Daniel COULON
Examinateurs :	Mario BORILLO
	Jean Marie PIERREL

Université de Nancy I

Centre de Recherche en
Informatique de Nancy
Inria Lorraine

DIAL,
la composante dialogue d'un système de communication orale homme-machine finalisée en langage naturel

THÈSE

présentée et soutenue publiquement le 1er **Avril 1988**

pour l'obtention du **doctorat de l'Université de Nancy I**
(Mention Informatique)

par

Azim Roussanaly

Composition du jury :

Président : Jean-Paul HATON

Rapporteurs : Gérard SABAH
Daniel COULON

Examinateurs : Mario BORILLO
Jean Marie PIERREL

à mes parents,
à Claudine et à Jules.

Université de Nancy I

Centre de Recherche en Informatique de Nancy Inria Lorraine

Thèse de l'Université de Nancy 1
soutenue le 1er Avril 1988 par Azim Roussanaly [1]

DIAL, la composante dialogue d'un système de communication orale homme-machine finalisée en langage naturel.

Résumé

L'objet de cette étude est l'élaboration de la composante dialogue d'un système de communication orale homme–machine. Ce travail s'inscrit dans le cadre d'un projet, actuellement en cours de développement au CRIN, dont l'objectif est de réaliser une interface vocale qui soit à la fois efficace et confortable pour un usager peu familiarisé aux systèmes informatiques. Les dialogues que nous envisageons de traiter sont caractérisés par :

- **le langage** autorisé qui correspond à un *sous–langage* limité par l'univers de l'application,
- **la tâche** à réaliser qui appartient à une famille d'applications : *les demandes de renseignements*. L'application test retenue est la demande de renseignements administratives sur la base des *pages roses de l'annuaire téléphonique*,
- **la population** visée qui est le grand public, c'est-à-dire une population a priori non–entrainée.

La variété des sources de connaissances intervenant dans le processus de compréhension d'un dialogue nous ont conduit à la définition d'une architecture formée de cinq composantes qui reposent sur des bases de connaissances spécifiques et qui interagissent par des échanges d'informations entre elles.

DIAL est l'une de ces cinq composantes. Son rôle consiste à fournir une interprétation contextuelle de l'énoncé de l'usager, à effectuer un raisonnement sur la tâche, à gérer le dialogue, à engendrer une réponse et à prédire les interventions futures. Cette composante s'appuie sur un ensemble de connaissances pragmatiques qui incluent des connaissances sur l'univers de l'application, sur la tâche et sur la structure du dialogue.

Nous proposons un formalisme de représentation de ces connaissances qui permet d'appréhender efficacement les problèmes de la compréhension des dialogues que nous envisageons de traiter et, grâce à la distinction qui apparaît entre les connaissances et les traitements, nous parvenons à un système paramétré et incrémental.

Mots-clefs : Dialogue, raisonnement, compréhension automatique de la parole, langage naturel

[1] CRIN - BP 239 - 54506 Vandoeuvre-lès-Nancy cedex (tel.83 91 20 00)

REMERCIEMENTS

Je tiens à exprimer mes vifs remerciements :

à Jean-Paul HATON, professeur à l'Université de Nancy 1, qui m'a acceuilli dans son équipe et qui me fait l'honneur de présider ce jury,

à Jean-Marie PIERREL, professeur à l'Université de Nancy 1, qui a su me faire découvrir ce domaine passionnant de l'informatique et qui m'a guidé et conseillé tout au long de ce travail,

à Gérard SABAH, directeur de recherche au LIMSI, et à Daniel COULON, professeur à l'INPL, qui ont bien voulu consacrer de leur précieux temps pour juger ce travail,

à Mario BORILLO, directeur de recherche au LSI, qui a aimablement accepter de siéger à ce jury,

à tous ceux qui ont participé à ce projet, particulièrement Noëlle CARBONELL, Guy DEVILLE, Emannuel LORANT, Bernard MANGEOL, Pierre MOUSEL, Hans PAULUSSEN, Laurent ROMARY et Serge SIMONET,

et enfin à tous les autres membres du CRIN qui m'ont aidé, en m'excusant de ne pouvoir tous les nommer ici.

Table des matières

1 LA COMMUNICATION HOMME-MACHINE **3**

1 LA COMMUNICATION 3

2 LES DIFFÉRENTES APPROCHES 4

2.1 Les travaux liés au génie logiciel 4

2.2 Les travaux liés à l'intelligence artificielle 5

3 LES SPÉCIFICITÉS DE LA COMMUNICATION HOMME-MACHINE . . 6

3.1 L'influence de la situation de communication 7

3.2 L'influence de l'univers 7

3.3 L'influence de la population 9

3.4 L'influence du mode de communication 10

4 GÉNÉRALITÉS SUR LE TRAITEMENT AUTOMATIQUE DE LA PAROLE 12

4.1 La production de la parole 12

4.2 La perception de la parole 13

5 LA RECONNAISSANCE 16

6 LES SOURCES DE CONNAISSANCES 19

7 L'ARCHITECTURE 21

2 VERS UNE COMPRÉHENSION DU DIALOGUE **25**

1 LES APPLICATIONS 25

2 NOS MOTIVATIONS ET NOS OBJECTIFS 26

2.1 Le langage naturel pour des sujets non-entraînés 26

2.2 La prise en compte du dialogue 28

3 LES PREMIERS SYSTÈMES DE DIALOGUE 30

4 LE PROJET "DIALOGUE" ACTUEL 34

4.1 Les acquis des projets précédents 34

4.2 Le cahier des charges . . . 35
4.3 L'application retenue . . . 36
4.4 Le dialogue visé . . . 37
4.5 Les sources de connaissances . . . 38

3 UNE PROPOSITION D'ARCHITECTURE 41
1 LES PRINCIPES GÉNÉRAUX . . . 41
2 LES COMPOSANTES . . . 42
2.1 Le décodage acoustico-phonétique : APHON . . . 42
2.2 Les traitements prosodiques : PROSO . . . 43
2.3 Le traitement lexical : LEX . . . 44
2.4 Les analyses syntaxico-sémantiques : SYN-SEM . . . 46
3 LE FLUX D'INFORMATIONS . . . 48
4 LE CONTRÔLE ET LA STRATÉGIE . . . 51
5 LES OBJECTIFS ET LA STRUCTURATION DE LA COMPOSANTE DIALOGUE . . . 53
5.1 Hypothèses de travail . . . 53
5.2 Analyse de la conversation . . . 54
5.3 Rôle de la composante . . . 61
5.4 Structure interne de la composante . . . 62

4 LES CONNAISSANCES ET LES TRAITEMENTS 65
1 STRUCTURES DE BASE . . . 65
1.1 Le formalisme de la présentation . . . 65
1.2 Les concepts . . . 66
1.3 Les formules . . . 69
2 L'INTERPRÉTEUR . . . 71
2.1 L'interprétation contextuelle . . . 71
2.2 Les informations en entrée . . . 72
2.3 Les informations en sortie . . . 74
2.4 Principes de fonctionnement et connaissances . . . 75
3 LE RAISONNEUR . . . 81
3.1 Les fonctions . . . 81
3.2 Les connaissances dynamiques . . . 82
3.3 Les connaissances statiques . . . 84
3.4 Les traitements sur BF . . . 89
3.5 Les traitements sur EB . . . 102
3.6 Les traitements sur EQ . . . 106
3.7 Le contrôle dans le raisonneur . . . 112
4 LE GESTIONNAIRE DE DIALOGUE . . . 113
4.1 Les fonctions . . . 113
4.2 Les connaissances dynamiques . . . 114
4.3 Le modèle de dialogue . . . 116
4.4 Le contrôle dans le gestionnaire . . . 126

5 LES RESULTATS ET LES PERSPECTIVES 127
1 IMPLANTATION . . . 127
2 L'INTERPRÉTEUR . . . 128
2.1 L'organisation . . . 128
2.2 Un exemple . . . 130
2.3 Conclusion . . . 140
3 LE RAISONNEUR . . . 140
3.1 L'organisation . . . 140
3.2 Exemples de déduction . . . 144
3.3 Exemples d'incohérences . . . 147
3.4 Evolution de l'état de la tâche . . . 149
3.5 Conclusion . . . 151
4 LE GESTIONNAIRE DE DIALOGUE . . . 152
4.1 L'organisation . . . 152
4.2 Exemple 1 . . . 154
4.3 Exemple 2 . . . 159
4.4 Conclusion . . . 161

A L'INTERPRÉTEUR 173
1 La base des faits . . . 173
2 La base des règles . . . 174
3 Quelques exemples . . . 179
3.1 Exemple 1 . . . 179

3.2 Exemple 2 184
3.3 Exemple 3 187
3.4 Exemple 4 194
3.5 Exemple 5 198
3.6 Exemple 6 200

B LE RAISONNEUR **203**

1 Les connaissances statiques 203
1.1 Les concepts 203
1.2 La base de règles 209
1.3 La base de données 211
1.4 La base de renseignements 213
1.5 La compilation des connaissances 214
2 Quelques exemples 222
2.1 Notations 223
2.2 Exemple 1 : déduction 223
2.3 Exemple 2 : planification 224
2.4 Exemple 3 : incohérence 226
2.5 Exemple 4 : valeurs par défaut 228
2.6 Exemple 5 : scores 229
2.7 Exemple 6 : mise à jour de l'état de la tâche 231

C LE GESTIONNAIRE DE DIALOGUE **235**

1 Les schémas de dialogue 235
2 Les énoncés figés en langage naturel 241
3 Les procédures de réponses 243
4 Deux exemples 244
4.1 Exemple 1 : dialogue dirigé 244
4.2 Exemple 2 : dialogue à initiative mixte 245

INTRODUCTION

A notre époque où l'ordinateur touche un nombre sans cesse croissant d'individus, le mode de communication entre l'usager et la machine revêt une importante considérable.

Lors de ces dernières années, les moyens de communication ont énormément évolués sur le plan matériel et sont devenus de plus en plus fiables. Ecrans graphiques, synthétiseurs de parole, tablettes graphiques, souris, caméras, scanners, dispositifs d'entrée vocale ...sont autant de media qui simplifient la communication avec une machine et viennent petit à petit se substituer aux traditionnels écrans et claviers.

Dans l'étude présente, nous nous penchons sur la communication orale, moyen encore peu répandu mais qui devrait être l'interface de demain en raison de multiples avantages qu'elle offre. En effet, la parole, élément important de spécificité de l'être humain, constitue un des moyens de communication les plus efficaces qui n'exige aucun effort et qui ne mobilise pas les autres parties du corps.

A l'heure actuelle, les obstacles majeurs à l'utilisation de ces interfaces proviennent, d'une part, des limites des systèmes de reconnaissance de la parole et, d'autre part, de la rigidité des dialogues mis en oeuvre tant par leur structure -menus, questions/réponses...- que par le langage autorisé -langage artificiel-.

L'étude que nous rapportons dans ce manuscrit est effectuée dans le cadre du projet "Dialogue" du CRIN dont l'objectif est de concevoir et de réaliser une interface qui permet de dialoguer oralement en langage naturel avec une machine dans le but d'accomplir une tâche.

A la suite de la définition de l'architecture générale du système, nous nous sommes attachés à la réalisation de l'un des cinq processeurs qui le composent ; il s'agit du processeur pragmatique DIAL. Nous proposons une organisation formée principalement de trois modules spécifiquement dédiés à l'interprétation contextuelle, au raisonnement et à la gestion du dialogue.

Ce manuscrit s'articule en cinq chapitres organisés de la façon suivante :

- Le premier chapitre est une présentation générale du problème de la communication homme-machine : nous y abordons les spécificités impliquées par cette situation par-

ticulière de communication et les problèmes engendrés par l'utilisation de la parole.

- Dans le deuxième chapitre, nous nous intéressons aux principales expériences d'automatisation du dialogue menées dans les laboratoires français et étrangers et nous y présentons notre propre projet "Dialogue".
- Le troisième chapitre est consacré à une proposition d'architecture pour un système de compréhension de dialogue oral finalisé en langage naturel. Nous y examinons chacune des composantes qui apparaissent dans cette architecture.
- Dans le quatrième chapitre, nous présentons de manière plus détaillée, la composante pragmatique DIAL et ses différents modules.
- Et enfin, le cinquième et dernier chapitre contient les résultats que nous obtenons actuellement et les perspectives d'évolution de ce système.

Chapitre 1

LA COMMUNICATION HOMME-MACHINE

1 LA COMMUNICATION

La communication telle que nous l'entendons dans le cadre de notre étude repose sur le modèle de SHANNON [HORMANN 72] illustré par la figure 1.1.

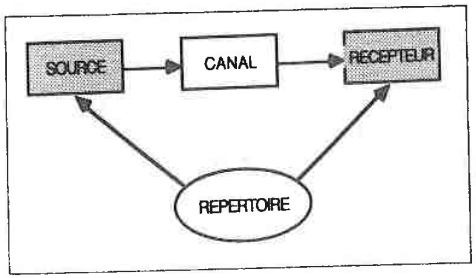


Figure 1.1: Modèle de la communication d'après SHANNON (tiré de [HORMANN 72]).

Ce modèle définit la communication comme étant un processus de transmission d'un message d'une *source* vers un *recepteur* via un *canal*.

Le message est selectionné par la source dans un *répertoire* commun aux deux intervenants. Il subit ensuite une opération d'encodage afin de le rendre apte à être transmis. Inversement, le récepteur décode le message reçu en le reconvertissant en une forme qui pourra être confrontée à son propre répertoire (voir figure 1.2 [HORMANN 72]).

Ces modèles restent cependant très généraux. Les messages peuvent en effet revêtir des formes diverses (signes, images ...). Pour notre part, nous nous intéressons plus particulièrement à la communication orale, c'est-à-dire que le message se présente sous forme de parole.

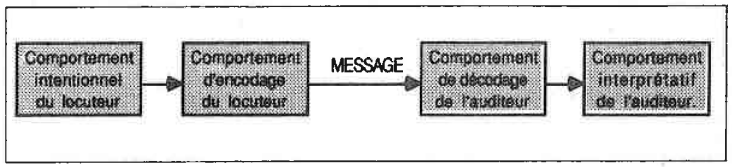


Figure 1.2: Modèle de la communication d'après CARROLL (tiré de [HORMANN 72]).

Dès lors, l'étude de la communication peut être envisagée sur plusieurs plans : la linguistique, la psycholinguistique, la théorie de l'information, les sciences cognitives ...

Chacune de ces approches apporte un éclairage particulier à ce problème. Tenter ici de les aborder en détail nous ferait déborder du cadre de notre étude. Notons seulement que les résultats de certains travaux menés dans ces différents domaines contribuent dans une large mesure à la conception d'un système de communication homme-machine.

2 LES DIFFÉRENTES APPROCHES

On parle de communication homme-machine lorsque l'un des deux intervenants est remplacé par un ordinateur. D'une façon générale, ce dernier joue alternativement le rôle du locuteur et de l'auditeur.

Afin de mieux situer nos objectifs, nous différencions les nombreux travaux qui s'apparentent à la communication homme-machine en deux grandes familles :

- les travaux liés au génie logiciel ;
- les travaux liés à l'intelligence artificielle.

2.1 Les travaux liés au génie logiciel

La finalité de ces travaux est l'amélioration des interfaces en les adaptant aux processus cognitifs humains afin de minimiser les contraintes d'utilisation d'un système informatique. A cet effet, on a souvent recours aux méthodes et aux résultats de l'ergonomie. Cette approche a permis d'introduire des moyens de communication spécifiques parfois très différents de ceux utilisés par l'homme dans la *communication naturelle*.

La rapidité d'adaptation d'un usager devient alors un facteur de qualité primordial pour les interfaces ainsi obtenues. Or, avec l'évolution du matériel, de nouvelles techniques de communication se sont très vite répandues -écran, souris, crayons optiques...- et leur utilisation quasi-quotidienne -Minitel, guichet automatique...- par le grand public en a facilité l'apprentissage.

A titre d'exemple, citons ici les principales méthodes utilisées dans cette approche :

- les questions/réponses,
- les remplissages de formes,
- les menus,
- les touches fonctions,
- les langages de commandes spécialisés.

Pour une information plus complète sur ce sujet, nous renvoyons le lecteur à [SCAPIN 86].

D'une manière générale, les systèmes utilisant ces méthodes offrent des interfaces d'une grande efficacité et, de ce fait, ont contribué pour une grande part à l'expansion que connaît actuellement l'informatique auprès d'usagers non-spécialistes.

Malgré tous leurs avantages, il n'en demeure pas moins que ces interfaces reposent sur des modes de communication *artificiels* qui demandent un effort d'adaptation à un utilisateur naïf. Ce constat justifie la seconde approche que nous allons maintenant présenter.

2.2 Les travaux liés à l'intelligence artificielle

Cette approche se différencie fondamentalement de la précédente en ce sens que l'objectif est de doter la machine de certaines facultés de communication possédées par l'être humain. Dans cette optique, l'apport de l'intelligence artificielle se situe sur deux plans :

- **sur le plan perceptif** : les résultats fournis au niveau de la reconnaissance des formes, et plus précisément du signal vocal, ouvrent de nouvelles perspectives dans ce domaine ;
- **sur le plan cognitif** : de nombreuses études sont menées sur la modélisation du processus de la compréhension du langage chez l'être humain. Ces modèles font intervenir des mécanismes formels empruntés à des domaines variés tels que la logique [COLMERAUER 78], la linguistique [CHOMSKY 71] [WINOGRAD 83] et la psychologie [SCHANK 77], [SOWA 84].

Notons au passage que, la vision par ordinateur fait également partie de cette approche. Cependant, en raison de nos objectifs, dans ce manuscrit, nous nous en tiendrons strictement à la communication parlée.

Par conséquent, les facultés auxquelles nous nous intéressons sont :

- d'une part, celles de parler, d'entendre,
- et, d'autre part, celles de comprendre (interpréter et raisonner) un message en langage naturel, de le traiter (rechercher une réponse) et de gérer le dialogue.

Cependant l'environnement de communication créé par la présence d'une machine introduit des spécificités qui, comme nous allons le constater maintenant, influent fortement sur la conception d'une interface.

3 LES SPÉCIFICITÉS DE LA COMMUNICATION HOMME-MACHINE

Une des influences importantes engendrées par la communication avec une machine se retrouve sur le *niveau du langage*. C'est la raison pour laquelle nous commençons par expliquer brièvement la notion de *niveau du langage*.

Trois niveaux de langage sont communément distingués :

- **LF (Langages formels) :** il s'agit de langages de programmation ou langages de commandes destinés à des spécialistes et créés de toutes pièces pour répondre à un besoin précis. Le traitement de ces langages repose sur des techniques de compilation actuellement bien maîtrisées [AHO 72].
- **LACN (Langages artificiels à consonance naturelle) :** ce sont les langages ayant une syntaxe formellement définie [WOODS 70]. Dès lors, leur traitement reste comparable à celui des *LF*. La consonance naturelle est alors introduite à différents degrés :
  - par l'utilisation d'un lexique formé de mots de la langue naturelle,
  - par la restriction des structures des messages acceptables à un sous-ensemble de la syntaxe du langage naturel. De plus, ce sous-ensemble peut être plus ou moins étendu ; les syntaxes ainsi définies recouvrent la gamme des langages allant du langage syntaxiquement très contraint au langage quasi-naturel qui possède les propriétés essentielles du langage naturel [DESCLES 82].
- **SL (Sous-langages) :** ce sont les langages où les contraintes sont d'ordre sémantico-pragmatique. Il est cependant à noter que la notion de sous-langage n'est pas clairement définie en linguistique. Il existe effectivement plusieurs points de vue ; nous adopterons celui de DEVILLE ET PAULUSSEN [DEVILLE 86] qui le définissent comme *un ensemble d'énoncés liés par un sujet limité, utilisés pour une fonction particulière et engendrés par une grammaire et un vocabulaire spécifique.*

Les spécificités ayant une influence sur le langage possèdent des origines variées que nous attribuons :

- à la situation de communication,

- au domaine sur lequel porte l'interaction ou l'univers,
- à la population impliquée et enfin
- au mode de communication.

Dans la suite de ce paragraphe, nous allons tenter de montrer dans quelle mesure chacun de ces facteurs intervient dans une conversation.

3.1 L'influence de la situation de communication

Selon les motivations et les objectifs de chacun des participants, la communication revêt divers aspects parmi lesquels nous pouvons distinguer le débat, la narration, l'interview...

Dans notre étude, nous nous intéressons essentiellement aux interfaces entre un utilisateur et une tâche. Cette restriction entraîne une forme de communication spécifique où l'utilisateur se place dans une **situation de requêtes ou de commandes**. Ceci a pour conséquence :

- sur le plan syntaxique, de réduire la variété des structures du message. Ce résultat a été observé sur l'analyse linguistique d'un corpus d'enregistrements de demandes de renseignements dans un service de la *SNCF*[1] [MOREL 85],
- sur le plan sémantique, de restreindre les ambiguïtés dues aux anaphores et aux ellipses [MOREL 85] ,
- sur le plan du dialogue, de nous mettre en présence de ce que FALZON [FALZON 84] définit comme *un dialogue opératif* principalement caractérisé par l'existence d'une structure du dialogue liée à celle de la tâche.

Remarque : Il est à noter, d'un point de vue méthodologique, que les différents auteurs s'appuient sur une analyse statistique de corpus pour émettre ces conclusions.

3.2 L'influence de l'univers

Une communication dans une situation de requête ou de commande se déroule dans un univers restreint délimité par l'application choisie. Nous observons alors une influence principalement au niveau du vocabulaire et plus précisément sur les facteurs suivants :

- **la taille du lexique :** le tableau ci-dessous montre à travers l'étude de trois différents corpus sur des demandes de renseignements, la variation de la taille du

[1] Société National des Chemins de Fers Français

vocabulaire :

Relation entre taille du vocabulaire et application		
APPLICATION	REFERENCE	NB. MOTS
Renseignements aériens	[AMALBERTI 86]	env. 400
Renseignements administratifs (formalités pour les pièces administratives)	[ROUSSANALY 86]	env. 1000
Renseignements administratifs (ministère de l'emploi et du travail)	[DEVILLE 87]	env. 1400

- **la fréquence de l'utilisation des mots :** une expérience portant sur un corpus de conversations entre des pilotes et des contrôleurs aériens [FALZON 84] montre que pour couvrir 90% des mots prononcés, 24, 7% du lexique est suffisant. En raison de la prédominance du jargon professionnel dans de telles conversations, ce résultat met en évidence le rôle de l'application dans la fréquence des mots.

- **la réduction des polysémies :** l'étude de la polysémie dans un recueil de textes ou de conversations consiste à sélectionner un sous-ensemble de mots possédant plusieurs significations dans un dictionnaire et à comparer leur usage effectif par rapport à toutes significations possibles.

  Il apparaît que les phénomèmes de polysémie se trouvent réduites dans les conversations dans un univers d'application précis [KITTERIGDE 84]. Pour illustrer cette influence, reprenons l'étude rapportée par FALZON [FALZON 84] effectuée dans un contexte de commande de matériel où les auteurs constatent notamment que 48 occurrences du mot *order* sur 50 renvoient à un même sens alors qu'il existe 14 significations distinctes de ce mot dans le dictionnaire. Ce résultat nous semble particulièrement significatif de l'influence de l'univers d'application sur la communication.

D'autre part, cette influence se manifeste également dans la structure du dialogue. Ce point a déjà été mentionné dans le paragraphe précédent lorsque nous évoquions le lien existant entre la tâche et le dialogue.

Pour une application donnée -ou pour une famille d'applications-, une récente expérimentation dans un centre médical [AMALBERTI 84] tend à montrer qu'un dialogue qui s'appuie sur un scénario conditionné par l'application, reste très efficace[2]. Nous reproduisons, à figure 1.3 le scénario issu cette étude.

[2] L'efficacité est ici appréciée dans le sens où la tâche est pleinement, partiellement ou pas du tout satisfaite.

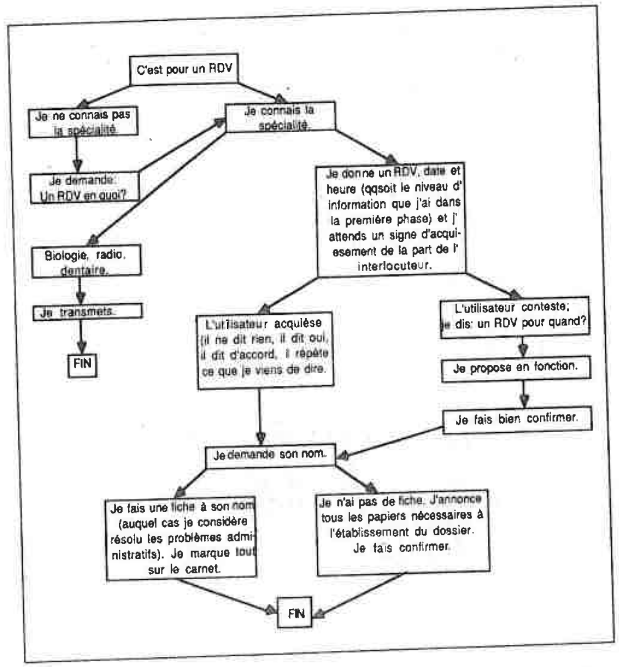


Figure 1.3: Scénario de stratégie d'une opératrice dans un centre médical.

3.3 L'influence de la population

Des observations sur la communication naturelle montrent une augmentation sensible de la production verbale dans le cas d'un usager qui possède peu de connaissances sur le domaine de sa requête. Cette variation s'explique par un effort d'explications supplémentaires face à un interlocuteur peu expérimenté.

Dans le cadre de la communication avec une machine, l'ensemble des utilisateurs peut être regroupé en trois classes de populations, en fonction de leur connaissance de l'application :

- **les spécialistes :** ils constituent une population qui communique fréquemment avec un ordinateur, mais les applications élaborées à leur intention sont traditionnellement du ressort du génie logiciel et de ce fait, ils nous intéressent moins que les deux classes suivantes.

- **les professionnels ou les locuteurs entrainés :** cette classe est composée de

sujets amenés à communiquer de façon relativement fréquente dans un jargon préalablement appris.

- **les utilisateurs naïfs ou le grand public :** nous regroupons, sous ces dénominations, l'ensemble des utilisateurs qui désirent occasionnellement réaliser une tâche et qui utilisent à cet effet, la langue naturelle.

Ces trois classes d'usagers correspondent approximativement aux trois classes de langages que nous avons présentées au début de cette section (voir figure 1.4).

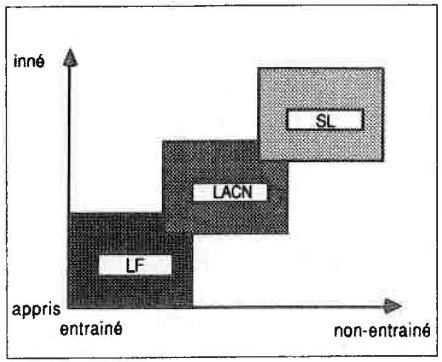


Figure 1.4: Relation entre la population et le langage.

Sur l'abscisse, nous plaçons, dans l'ordre croissant de population non-entrainée, d'abord les spécialistes, puis les locuteurs entrainés et enfin le grand public, et sur l'ordonné, nous trouvons un degré d'apprentissage des langages.

3.4 L'influence du mode de communication

Dans ce paragraphe consacré à l'influence du mode de communication sur le langage, nous allons aborder le problème sous deux angles : l'aspect psychologique et l'aspect matériel.

- **L'aspect psychologique :** deux expériences, récemment effectuées en France, tendent à prouver que la présence d'une machine constitue un facteur psychologique ayant une influence sur le comportement langagier d'un utilisateur.

  La première des ces deux expériences a été menée par le groupe *GRECO*[3] *Communication parlée* [GRECO 85] dans un centre de renseignement de la SNCF, et la seconde, dans un centre de renseignements téléphoniques aériens [AMALBERTI 86].

[3] Groupe de REcherches COordonnées

L'expérience consiste à recueillir un corpus de conversations en deux phases au moins. Dans un premier temps, les conversations sont enregistrées dans une situation normale, c'est-à-dire entre un utilisateur et un opérateur humain. Puis, dans un deuxième temps, on effectue un enregistrement avec la méthode du *Wizard of Oz* qui consiste à faire simuler une machine par un compère à l'insu de l'utilisateur.

L'analyse des corpus ainsi obtenus par les différents expérimentateurs révèle que :

- Les sujets, croyant parler à une machine, restreignent leur vocabulaire.
- Le nombre d'interventions est plus élevé dans la situation homme-machine.
- En présence d'une machine, les expressions du locuteur sont plus stabilisées ; les demandes initiales sont plus complètes et possèdent des structures moins variées.

Il semble donc que l'impact psychologique créé par la présence d'une machine provoque, chez l'utilisateur, une modification du comportement langagier.

- **L'aspect matériel :** bien souvent, un utilisateur communique avec un ordinateur par l'intermédiaire d'un clavier et d'un écran. Ce dispositif permet d'envoyer et de recevoir des messages sous une forme textuelle de manière très fiable. Mais la parole constitue le moyen de communication le plus courant chez l'homme. Par conséquent, il est important, dans une approche comme la nôtre, de disposer d'une interface vocale avec la machine. Il existe actuellement, sur le marché, un ensemble de systèmes permettant de réaliser une telle interface avec une fiabilité encore limitée. Il est certain que cela pose des problèmes, mais on peut envisager raisonnablement une nette amélioration de ces systèmes dans un avenir proche.

  La question qui se pose alors naturellement est de savoir si le message oral se résume à une transcription sonore du message écrit.

  Traditionnellement, les linguistes distinguent les deux aspects du français parlé : l'écrit et l'oral. NOUHEN-BELLEC ET SIROUX [AN 81] rapportent cette citation de SAUVAGEOT qui note qu'*il est extrêmement rare de trouver des personnes chez qui la langue parlée et la langue écrite se superposent, et cet écart est plus ou moins grand selon les personnes ou, pour les mêmes personnes, selon les circonstances où elles s'expriment.*

  Nous avons effectivement pu vérifier ces remarques sur les corpus de l'oral et de l'écrit que nous disposons [ANGELETTI 84] [ROUSSANALY 86].

  Nous constatons notamment des phénomènes tels que les hésitations, les reprises ou les messages de maintien du canal de communication, qui sont très fréquents à l'oral et quasiment inexistants à l'écrit.

Nous notons également, à l'oral, la présence de messages spécifiques tels que des demandes de répétition ou de reformulation occasionnées par une ambiguïté ou une défaillance du canal de communication.

Ces phénomènes sont encore plus accentués dans un système de communication oral homme-machine car les techniques utilisées ne fournissent pas une reconnaissance idéale de 100% du signal acoustique émis par le locuteur - actuellement, elle est de l'ordre de 70%-.

En conclusion, de ce rapide survol de la communication homme-machine, il ressort :

- d'une part, qu'il ne s'agit pas de reproduire sur une machine toutes les situations de communication que nous pouvons rencontrer dans la communication homme-homme. Pour partir sur des bases réalistes, il faut, au contraire, se limiter à une classe de communication caractérisée par la situation de requête dans le cadre de tâches déterminées. C'est ce type de communication que nous qualifions de *dialogue finalisé*,
- d'autre part, qu'il faut tenir compte de nouvelles contraintes introduites par l'utilisation d'une machine avec son propre moyen -oral ou écrit- d'acquisition et d'émission des messages.

C'est sur ce deuxième point que nous allons poursuivre notre présentation.

4 GÉNÉRALITÉS SUR LE TRAITEMENT AUTOMATIQUE DE LA PAROLE

Nous allons nous situer d'emblée dans le domaine de l'oral, moyen de communication privilégié chez l'homme. Notre souci de mettre en oeuvre une interface se rapprochant de la communication naturelle justifie pleinement ce choix. Signalons par ailleurs que dans des situations telles que la communication avec un individu atteint de certains handicaps (cécité, infirmité des membres supérieurs...), la parole devient indispensable.

Ce choix induit un système possédant à la fois les facultés de production et de perception de la parole

4.1 La production de la parole

Historiquement, l'apparition des premières machines parlantes remonte au XVIIIème siècle. A cette époque, on était déjà en mesure de reproduire certains sons du langage à l'aide de dispositifs mécaniques.

Ces machines ont beaucoup évolué depuis ce temps, car on trouve aujourd'hui sur le marché des synthétiseurs très performants, capables d'émettre tous les sons du système phonatoire humain. En outre, interfacés par des logiciels adaptés, ces synthétiseurs permettent actuellement la génération de phrases entières avec une intelligibilité très acceptable.

Comme en témoignent les travaux de DANLOS [DANLOS 85], les problèmes de la génération de la parole ne sont pas totalement résolus pour autant. De sa représentation conceptuelle jusqu'à sa forme acoustique, un message passe par différents stades qui comportent des choix complexes dus à sa nature subjective. Parmi ces problèmes, les deux suivants nous semblent assez représentatifs de leur complexité :

- **le choix de formulation :** il est souvent possible d'exprimer une idée sous de nombreuses formes, mais, selon le contexte, certaines formulations sont plus appropriées que d'autres. Il faut donc mettre en oeuvre des solutions permettant de prendre des décisions optimales sur les mots à utiliser ainsi que sur la construction des phrases.
- **la pertinence du message :** il arrive parfois qu'un message véhicule plusieurs informations à la fois. Les questions que nous pouvons nous poser sont alors les suivantes :
  - Dans quel ordre faut-il les énoncer?
  - Lesquelles faut-il expliciter et lesquelles peuvent rester implicites?

En ce qui concerne la production de la parole, nous nous bornerons à évoquer ces problèmes encore ouverts aujourd'hui, car notre étude s'intègre, pour une grande part, à la perception.

4.2 La perception de la parole

Les premières machines aptes à reconnaître la parole datent seulement du milieu de notre siècle. Elles se présentent sous une forme entièrement câblée et atteignent des performances de reconnaissance qui s'élèvent à une dizaine de mots. En 1958, DULEY ET BALASHEK [DULEY 58] présentent une machine qui reconnaît les dix chiffres prononcés par plusieurs locuteurs.

Dans la décennie suivante, grâce à l'ordinateur et aux techniques de traitements numériques, ces expériences prennent un nouvel essor. Il en résulte une plus grande souplesse et de meilleures performances des systèmes. A cette époque, l'objectif étant de reconnaître des mots isolés, leur fonctionnement repose sur un modèle de comparaison du mot à identifier avec un ensemble de formes de référence mémorisé par un apprentissage préalable.

Puis, l'utilisation de contraintes linguistiques a permis de franchir une étape supplémentaire dans ce domaine. En 1969, VINCENS [VINCENS 69] applique cette technique pour un langage de commandes du robot *Hand-Eye-Ear* et en 1970 TUBACH [TUBACH 70] réalise un système permettant de saisir un programme en ALGOL prononcé mot à mot.

Actuellement, les recherches sur la reconnaissance de mots isolés continuent essentiellement sur :

- **l'aspect multilocuteur :** les chercheurs dans ce domaine projettent d'élaborer des systèmes indépendants des variabilités inter-locuteurs du signal vocal.
- **la taille du vocabulaire :** les recherches portent, dans ce cas, sur la mise en oeuvre de techniques de traitement efficaces et rapides afin d'obtenir des systèmes utilisant un vocabulaire étendu en temps réel.

Sur le plan industriel, grâce à l'intégration d'algorithmes complexes de reconnaissance dans les circuits VLSI, des firmes américaines, japonaises et européennes proposent actuellement, sur le marché, une centaine de systèmes [HATON 85a].

Parallèlement, en 1971, l'ARPA [4] consacre des moyens considérables au financement d'un projet de réalisation de systèmes automatiques de compréhension de la parole continue. La figure 1.5 regroupe les principales performances attendues de ces systèmes. Ce projet marque véritablement le début des recherches dans ce domaine.

A l'encontre de la reconnaissance, l'objectif de la compréhension n'est plus seulement d'identifier la forme phonétique ou orthographique d'un message, mais d'aller au-delà afin d'obtenir une représentation conceptuelle globale malgré les imperfections de la phase de reconnaissance. Ces lacunes sont compensées par l' introduction de connaissances linguistiques, telles que la syntaxe, la sémantique...

Seuls quatre systèmes sont retenus dans le cadre du projet de l'ARPA ; ils sont l'oeuvre des laboratoires suivants :

- **SRI-SDC :** conçu conjointement par *Stanford Research Institute* et par *System Development Corporation*, ce système n'a jamais été entièrement implanté mais il renferme des solutions originales, surtout en ce qui concerne les traitements sémantiques et pragmatiques [WALKER 78].
- **BBN :** dans le cadre de ce projet, la firme *Bolt Beranek and Newman* a réalisé le système HWIM [BBN 76].

[4] Advanced Research Project Agency : Agence du département de la défense américaine ayant financé un projet de recherches sur la compréhension de la parole de 1971 à 1976, avec un budget de 15 millions de Dollars.

	Le sytème devrait :
(1)	accepter la parole continue,
(2)	de plusieurs locuteurs
(3)	coopératifs parlant l'Américain commun,
(4)	dans une pièce calme,
(5)	avec un microphone de bonne qualité,
(6)	autorisant une légère adaptation du locuteur,
(7)	mais en restant toutefois naturel,
(8)	avec un vocabulaire d'environ 1000 mots,
(9)	avec une syntaxe artificielle,
(10)	et une tâche fortement contrainte,
(11)	avec également un modèle psychologique simple de l'usager,
(12)	procurant une intéraction agréable,
(13)	tolérant moins de 10% d'erreurs sémantiques,
(14)	fonctionnant approximativement en temps réel,
(15)	sur une machine de 100 Mips,
(16)	pouvant être démontré en 1976.

Figure 1.5: Cahier des charges du projet de l'ARPA (1971).

- **CMU :** la participation de *Carnegie-Mellon University* se traduit par deux réalisations : le système HEARSAY [ERMAN 77] et le système HARPY [LOWERRE 76].

Ces réalisations ont eu un effet catalyseur. Ainsi, on a vu, aussi bien aux USA [JELINEK 76] qu'au Japon [WAKITA 80] ou en Europe [HATON 80], naître des projets similaires.

En France, l'essentiel des activités de recherche dans ce domaine s'effectue dans les laboratoires suivants : le CERFIA[5] à Toulouse [CAELEN 79], le CNET [6] à Lannion [MERCIER 77], le CRIN[7] [HATON 77]. l'ICP[8] à Grenoble, et le LIMSI[9] à Orsay [MARIANI 82]

Actuellement, ces systèmes tendent à évoluer vers la diversification des sources de connaissances et vers une plus grande coopération entre celles-ci.

Dans la suite de ce chapitre, nous allons essayer de donner une vue générale de la compréhension de la parole en abordant quelques aspects spécifiques tels que la reconnaissance, les sources de connaissances et l'architecture, et en tentant de faire leur synthèse à travers les différentes études menées dans ce domaine.

[5] CERFIA : Cybernétique des Entreprises, Reconnaissance des Formes et Intelligence Artificielle.
[6] CNET : Centre National d'Etude des Télécommunications
[7] CRIN : Centre de Recherche en Informatique de Nancy.
[8] ICP : Institut de Communication Parlée.
[9] LIMSI : Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur.

5 LA RECONNAISSANCE

Dans le domaine de la reconnaissance de la parole, on a coutume de distinguer deux méthodes de traitement : la méthode globale et la méthode analytique.

Avant d'effectuer une comparaison entre ces deux méthodes, il nous semble nécessaire d'exposer rapidement les premières étapes du traitement de la parole : l'analyse et la paramétrisation.

A la sortie d'un microphone, le signal acoustique numérisé présente des caractéristiques qui rendent son interprétation très complexe. D'autre part, il est extrêmement redondant. Afin de conserver les paramètres les plus pertinents, on effectue, sur ce signal, une analyse acoustique par le calcul d'indices temporels (nombre de passages par zéro, maximum d'énergie, coefficients LPC) ou fréquentiels (FFT, banc de filtre). On trouvera dans [PISTER 84] et [dM 84] une description plus détaillée de ces différents techniques.

A la suite de ce premier traitement, on peut représenter la parole par une suite de vecteurs que l'on peut visualiser, par exemple, sous la forme d'un spectrogramme (voir figure 1.6) dans le cas d'une analyse fréquentielle.

Un spectrogramme est une représentation du signal vocal sur trois axes : le temps, la fréquence et l'intensité (schématisée par un noircissement variable).

La méthode globale consiste à considérer le mot prononcé comme une forme globale à comparer à un ensemble de formes de référence.

Les techniques de programmation dynamique, introduites, du reste, dans un cadre plus général de recherche d'une solution optimale, sont largement utilisées dans cette approche. Elles permettent, dans certains cas, d'obtenir une reconnaissance allant jusqu'à 99% des mots prononcés [BOYER 87].

Cette méthode est bien adaptée à la reconnaissance de mots isolés avec un vocabulaire restreint, car le temps de traitement pour un mot varie en fonction de la taille du lexique.

Elle est également utilisée dans les deux cas suivants :

- **la reconnaissance de mots enchaînés :** le principe consiste à segmenter un énoncé en mots que l'on reconnaît ensuite à l'aide de la méthode globale. En introduisant la segmentation, cette approche rajoute une difficulté supplémentaire due à la complexité des critères de segmentation.
- **la reconnaissance globale de phrases :** une autre manière de traiter une suite de mots est d'appliquer le principe de reconnaissance de mots isolés directement sur l'image acoustique d'une phrase. La forme de référence est alors construite par une combinaison de mots du vocabulaire.

Ces techniques sont également utilisées pour des applications qui comportent un vocabulaire restreint.

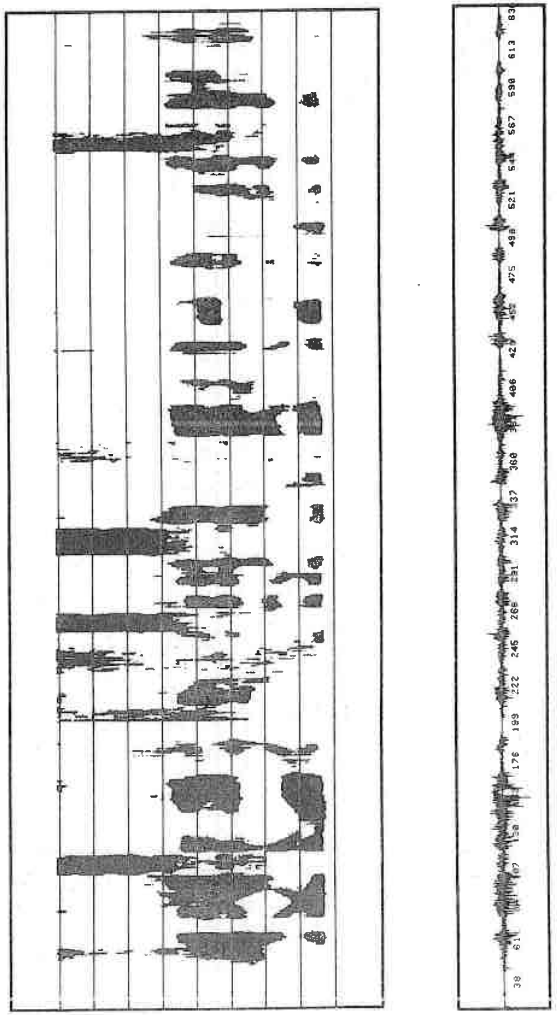


Figure 1.6: Spectrogramme vocal de la phrase :*Je voudrais savoir à qui il faut s'adresser pour faire une demande de double-nationalité?*.

La méthode analytique, quant à elle, consiste à considérer un message comme une suite d'unités phonétiques. Les unités utilisées dans les systèmes développés au CRIN sont *les phonèmes*. La figure 1.7 est une reproduction de l'ensemble des phonèmes du français, tirée de [PIERREL 87] où nous pouvons trouver une étude plus détaillée sur les autres unités phonétiques telles que les diphones et les syllabes.

VOYELLES			CONSONNES		
Phon.	exemples	classe	Phon.	exemples	classe
a	patte, bas, pâte	Orales	p	pas	Plosives sourdes
i	il		t	tas	
y	nu		k	cas	
ɔ	bol		b	bon	Plosives sonores
o	eau		d	dans	
ə	le, peu, peur		g	gars	
e	blé		v	vie	Fricatives sonores
ɛ	merci		z	zéro	
u	ou		ʒ	je	
ã	an	Nasales	f	feu	Fricatives sourdes
õ	on		s	sous	
ɛ̃	lin		ʃ	chat	
œ̃	un		n	nous	Nasales
SEMI-VOYELLES			m	ma	
			ɲ	agneau	
ɥ	huit		l	la	Liquides
w	oui		r	rue	
j	yeux, baille				

Figure 1.7: Les phonèmes du français utilisés pour la reconnaissance.

Par conséquent, reconnaître une phrase revient à déterminer la suite d'unités phonétiques qui la compose. C'est le décodage acoustico-phonétique qui fournit ce résultat à partir du signal paramétré. Cette opération s'effectue en deux phases :

- **la segmentation :** en s'appuyant sur les variations locales dans le signal, les zones contenant les unités phonétiques sont repérées ; on obtient ainsi un signal segmenté [FOHR 86]. Dans la pratique, il arrive qu'un segment contienne deux phonèmes ou, à l'inverse, qu'un seul phonème soit découpé en deux segments. Ces phénomènes d'élision et d'insertion sont nécessairement pris en compte lors des traitements ultérieurs.

- **L'identification :** Chaque segment est ensuite identifié en lui attribuant l'étiquette du phonème qu'il est censé contenir. Cette opération repose sur la détection de traits acoustiques (voir figure 1.7) à l'aide d'indices tels que le voisement, l'énergie, les formants...

En réalité, le décodage acoustico-phonétique propose un ensemble de phonèmes candidats pour chaque segment, formant ainsi un treillis phonétique.

En raison de la stabilité des formes de références et de leur indépendance vis-à-vis de l'importance du vocabulaire, il est préférable d'utiliser la méthode analytique pour les applications nécessitant un vocabulaire étendu.

Par ailleurs, cette méthode devient indispensable lorsqu'on traite la parole continue, car elle permet de faire appel à d'autres sources de connaissances de niveau supérieur.

6 LES SOURCES DE CONNAISSANCES

Si nous nous référons au modèle de la perception humaine d'après LIBERMAN à la figure 1.8 [LIBERMAN 70], nous constatons que le processus de compréhension met en jeu plusieurs niveaux de connaissances composés hiérarchiquement de l'acoustique, de la phonétique, de la phonologie, de la prosodie, du lexique, de la syntaxe, de la sémantique et de la pragmatique.

Cette classification trouve son origine principalement dans la linguistique où chaque niveau fait l'objet d'études fondamentales. Pour notre part, nous nous attacherons surtout à leur rôle dans le cadre plus restreint de la compréhension automatique de la parole.

- **Le niveau acoustique :** ce sont les connaissances qui permettent, d'une part, de détecter les débuts et les fins de message et, d'autre part, d'extraire du signal, les paramètres qui vont être interprétés par les niveaux supérieurs.

- **Le niveau phonétique :** ce niveau regroupe les connaissances sur les différentes unités phonétiques qui apparaissent dans la parole, indépendamment de leur contexte d'émission ; par exemple : la classification des phonèmes (voir figure 1.7).

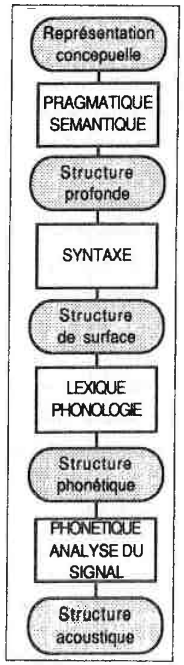


Figure 1.8: Un modèle de la perception de la parole d'après Liberman.

- **Le niveau phonologique :** ces connaissances complètent les précédentes, car elles concernent les variations possibles des sons en fonction de leur contexte. Elles rendent compte :
  - des altérations du signal provoquées par des spécificités individuelles (accent, sexe,...)
  - des phénomènes propres à la langue tels que la coarticulation et la liaison.

  Les trois premiers niveaux sont essentiellement utilisés lors du décodage acoustico-phonétique.

- **Le niveau prosodique :** ces connaissances sont utilisées pour rechercher les frontières de syntagmes -mot ou groupe de mots- repérés à l'aide de marqueurs prosodiques, calculés en fonction des variations temporelles -durée-, de l'intensité -énergie- et de la fréquence fondamentale -intonation- dans le signal [ROSSI 81]. Ces informations seront exploitées à leur tour par les processeurs lexical et syntaxique. En outre, une étude prosodique permet de déterminer si une phrase prononcée est affirmative ou interrogative en fonction de sa mélodie globale.

- **Le niveau lexical :** le lexique contient l'ensemble des mots du vocabulaire, auxquels sont associés leur transcription phonétique respective, et un certain nombre de traits syntaxiques et sémantiques. Un processeur lexical possède un double rôle dans un système de compréhension de la parole :
  - il sélectionne un sous-ensemble du lexique en s'appuyant sur des hypothèses fournies par les niveaux supérieurs,
  - il vérifie leur présence sur le treillis phonétique.

- **Les niveaux syntaxique et sémantique :** il semble difficile de dissocier ces deux niveaux, puisque les connaissances correspondantes constituent, toutes deux, les règles qui régissent l'agencement des mots dans une phrase. Cependant nous distinguons les connaissances syntaxiques, qui déterminent l'**ordre** des mots et qui les décrivent à l'aide de **catégories syntaxiques**, des connaissances sémantiques, qui portent sur la **combinaison** des mots et qui les caractérisent en fonction de leurs **propriétés sémantiques**.

  Cette distinction apparaît nettement à travers l'exemple proposé par PIERREL [PIERREL : qui note, en comparant les sources de connaissances sémantiques et syntaxiques , *c'est elle qui, après "orage de", conduit à retenir "pluie", "neige", "grêle"... et rejeter "soleil", "température"...*

- **Le niveau pragmatique :** ces connaissances rendent compte de l'interprétation d'un message dans le contexte créé par la tâche et le dialogue. Ce niveau comprend :
  - la résolution de références diverses,
  - la prise en compte de présuppositions,
  - la détermination d'informations implicites,
  - l' enchaînement du dialogue.

Dans un système automatique, ces connaissances interviennent dans le processus de compréhension dans un ordre déterminé par une stratégie globale d'activation des processeurs. Cette organisation des processeurs, les uns par rapport aux autres, est définie dans l'architecture logicielle d'un système.

7 L'ARCHITECTURE

Le modèle de LIBERMAN de la figure 1.8 suggère une organisation hiérarchique ascendante du processus de compréhension d'un message oral. Notons que la notion d'ascendance

fait référence à l'ordre d'activation des sources de connaissances, en situant au niveau inférieur, l'acoustique et au niveau supérieur, la pragmatique.

Les systèmes qui ont opté pour cette approche, comme, par exemple, le système KEAL [QUINTON 82], sont utilisés pour un vocabulaire limité, car le nombre d'hypothèses lexicales devient rapidement très élevé, pénalisant de ce fait, le temps de traitement.

Pour y remédier, le système HARPY [LOWERRE 76] effectue, préalablement, une précompilation des connaissances en partant des niveaux supérieurs vers les niveaux inférieurs. Ce pré-traitement fournit les transcriptions phonétiques de tous les messages possibles dans une application, organisées dans un vaste réseau. La phase de reconnaissance, proprement dite, consiste à rechercher un chemin dans le réseau ainsi constitué.

Cette méthode s'est révélée efficace, mais la taille du réseau peut devenir rapidement rédhibitoire.

L'approche inverse est utilisée dans le système guidé par la syntaxe : MYRTILLE I [PIERREL 75]. L'analyseur syntaxique émet des hyphothèses de mots à vérifier, sur le treillis phonétique, par le processeur lexical, de façon à réduire la combinaison des mots pouvant apparaître sur le signal.

Cette méthode convient au traitement d'un langage avec un vocabulaire important et avec une syntaxe bien définie -exemple : un langage artificiel-.

Mais, lorsqu'on désire établir une communication dans un langage acceptant une grande liberté syntaxique, il devient indispensable de faire coopérer les différents niveaux de connaissances. C'est dans cette optique qu'ont été définies, par exemple, les architectures des systèmes HWIM [WOLF 80] et MYRTILLE II [PIERREL 81] (voir figure 1.9 et 1.10).

Ces deux architectures sont comparables dans la mesure où toutes deux possèdent une structure de contrôle -ou superviseur- qui séquence l'activation des processeurs. A l'encontre des systèmes totalement hiérarchiques, l'ordre d'activation est établi de façon dynamique, en fonction de l'état de traitement du message. De plus, nous pouvons remarquer que certaines sources de connaissances -lexique, phonétique ... - sont utilisées de manière à la fois descendante et ascendante.

Le système HEARSAY II [ERMAN 77] va plus loin dans l'élimination de la hiérarchie entre les différents processeurs (voir figure 1.11). Chaque processeur est considéré comme un module indépendant qui communique avec les autres niveaux par l'intermédiaire d'un tableau noir ou *blackboard*.

Le *blackboard* est une structure de communication commune à toutes les sousrces de connaissances ou *KS*[10] ; à un instant donné du traitement d'un message, il contient :

[10]KS : Knowledge Source.

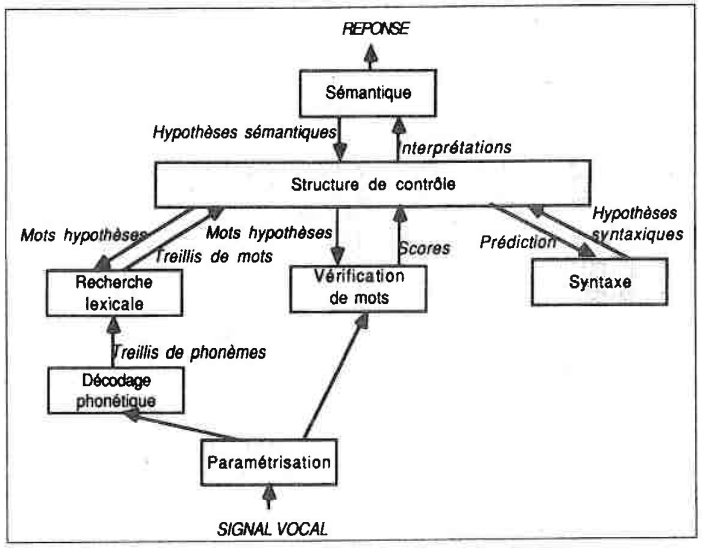


Figure 1.9: L'architecture du système HWIM.

- l'ensemble des hypothèses sur le message courant, provenant de tous les niveaux du système ; ces hypothèses sont théoriquement disponibles pour tous les modules,
- l'état courant du processus de compréhension.

L'activation d'un module est déclenchée par une modification du tableau noir. Une architecture avec un *blackboard* définit un modèle valide dans un contexte, plus général, de contrôle de processus indépendants [NII 86a] Ce modèle permet, en outre, l'exploitation du parallélisme entre les différents processeurs qui composent le système. æ

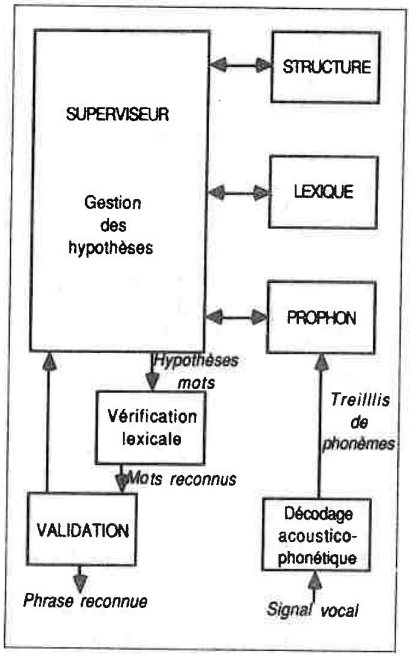


Figure 1.10: L'architecture du système MYRTILLE II.

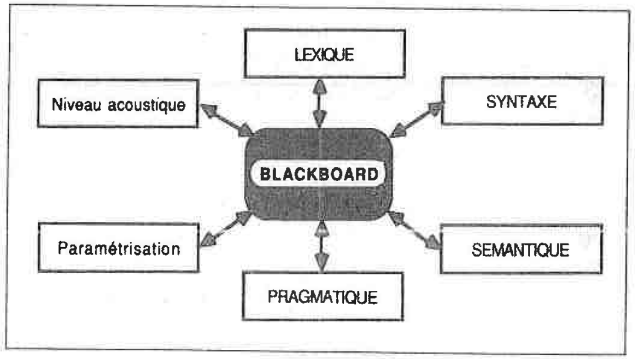


Figure 1.11: L'architecture du système HEARSAY II.

Chapitre 2

VERS UNE COMPRÉHENSION DU DIALOGUE

1 LES APPLICATIONS

Les domaines d'application du traitement automatique de la parole sont nombreux et variés. Parmi eux, nous pouvons citer :

- **la dictée automatique :** il s'agit de construire un système permettant d'acquérir un texte dicté ; la finalité est d'obtenir une représentation orthographique de ce texte [CHARPILLET 85]. Cet outil est destiné à la saisie de documents tels que les lettres, les rapports...
- **l'aide aux handicapés :** grâce à l'utilisation d'une interface vocale, une certaine catégorie d'handicapés acquiert la possibilité d'accéder à des outils informatiques ou de s'en servir pour réduire leur handicap. Dans cette gamme d'application, citons deux exemples de systèmes développés au CRIN :
  - la première réalisation est une application du système MYRTILLE I [LAFONT 83]. Il s'agit d'une interface vocale pour les commandes d'un système informatique à l'intention de sujets non-voyants.
  - La seconde est le système SIRENE[1] [HATON 85b], formé d'un ensemble d'outils d'aide aux malentendants et de rééducation vocale.
- **l'identification d'un sujet :** [GRENIER 78] chaque individu possède une voix qui permet de l'identifier. En calculant les paramètres pertinents à partir d'un signal vocal et en comparant ces paramètres à des références stockées, un système d'identification arrive à retrouver le propriétaire de la voix. Ces applications sont utilisées dans un univers de grande sécurité, où il existe des privilèges d'accès reservés à un nombre limité d'individus.

[1] SIRENE : Système Intéractif de Rééducation d'Enfants Non-Entendants

- **l'enseignement assisté par ordinateur : (EAO)** [BENNANI 83] la parole amène un confort supplémentaire dans les systèmes d'EAO et semble surtout indispensable dans les systèmes de dictée ou d'enseignements des langues.

Toutes ces applications se situent cependant en marge de notre étude actuelle. Il faut, en effet, rappeler que notre objectif est de traiter la communication orale homme-machine en situation de requête ou de commande. En conséquence, nous allons exclusivement nous intéresser aux systèmes servant d'interfaces entre un sujet et une tâche devant être accomplie par une machine.

Dans ce cadre, les applications usuellement rencontrées sont :

- **la communication de commandes à un robot ou à un logiciel,**
- **l'interface avec un système de résolution de problèmes** tel qu'un système expert,
- et **l'interrogation d'une banque de données.**

Le tableau de la figure 2.1 présente quelques réalisations qui ont été validées avec cette catégorie d'applications.

SYSTÈME	APPLICATION
ESOPE	Standard téléphonique
HEARSAY II	Recherche de documents
HWIM	Agence de voyages
KEAL	Renseignements téléphoniques
MYRTILLE I	Standard téléphonique
MYRTILLE II	Renseignements météorologiques
SRI	Renseignements sur la flotte navale

Figure 2.1: Les applications des systèmes de compréhension de la parole.

2 NOS MOTIVATIONS ET NOS OBJECTIFS

2.1 Le langage naturel pour des sujets non–entraînés

L'objectif du projet auquel se rattache notre travail, est la réalisation d'une interface vocale destinée au grand public. Afin d'être suffisamment confortable pour un utilisateur, le système doit posséder des qualités de souplesse et d'efficacité.

La souplesse d'un système de communication se juge par sa capacité d'adaptation à un grand nombre d'utilisateurs. Généralement, l'utilisation d'une interface vocale est

soumise à une phase d'apprentissage visant à adapter le système à l'utilisateur. Cela est essentiellement dû a' deux raisons :

- **l'aspect monolocuteur des systèmes :** avant d'accéder à une application, un utilisateur doit généralement fournir ses références vocales. Cette opération de stockage, qui constitue un apprentissage, est viable pour une application concernant un nombre restreint d'utilisateurs -exemple : les applications professionnelles-, mais elle devient rapidement contraignante lorqu'on s'adresse à un nombre élevé d'individus.

  Pour échapper à cette contrainte, on a recours à des modules de traitements acoustico-phonétiques relativement indépendants du locuteur -multilocuteur-. Il faut cependant noter que cette solution comporte un inconvénient car les techniques actuellement utilisées fournissent des résultats moins satisfaisants.

- **la structure du langage :** les meilleurs résultats de compréhension de la parole sont actuellement obtenus par les systèmes fonctionnant avec des mots–clefs. Leur commercialisation, grâce à l'utilisation de techniques suffisamment dominées, en est la preuve.

  Il existe un large éventail d'applications pouvant fonctionner sur ce principe. Ce type de communication n'est cependant pas tout à fait naturel.

  Certains systèmes, tels que MYRTILLE I appliqué aux commandes vocales d'un système informatique à l'intention de non–voyants [LAFONT 83], opèrent avec un langage artificiel. Dans ce cas, le langage est défini à l'aide d'une grammaire formelle, car ce formalisme reste parfaitement adapté.

  D'autres applications ont été développées de façon similaire avec ce système ; nous pouvons citer à nouveau l'application au standard téléphonique [PIERREL 75] ou encore à la commande d'une console sonar [ALINAT 87]. Ces applications regroupent des tâches suffisamment simples, autorisant l'utilisation d'une grammaire formelle pour décrire les requêtes d'un utilisateur. Mais cette fois–ci, le langage possède une consonance naturelle (LACN), car le vocabulaire est spécifique à l'application et les structures autorisées sont les structures simples de la syntaxe du langage naturel.

  Mais lorsqu'on aborde des tâches un peu plus complexes, où un utilisateur a la possibilité de formuler une requête dans un langage plus varié, il devient indispensable d'aller au-delà d'un langage artificiel. Imposer une syntaxe rigoureuse à une population de non–spécialistes nous semble une entreprise peu réaliste allant à l'encontre de nos objectifs.

  Le langage naturel n'est pas exempt de syntaxe ; certains écarts sont toutefois tolérés sans qu'ils ne nuisent, pour autant, à la compréhension. Notre objectif est de

reproduire cette capacité, importante pour le traitement du langage naturel, dans un système de compréhension de la parole.

2.2 La prise en compte du dialogue

La plupart des systèmes, cités dans le premier paragraphe de ce chapitre, ont été essentiellement réalisés dans le but de comprendre des phrases. Certains ont fonctionné en mode interactif, mais très peu d'entre eux ont été conçus avec un niveau qui prend véritablement en compte le dialogue de façon suffisamment indépendante de l'application.

Ainsi, pour les différentes applications de MYRTILLE I, il a été nécessaire de développer un module de gestion de dialogue ad hoc.

Or le dialogue joue un rôle important dans un système de communication en langage naturel ; parmi eux nous distinguons :

- **la validation de la reconnaissance :** comme nous avons pu le constater, le treillis de phonèmes délivré par le niveau acoustico–phonétique, possède un caractère incertain. L'utilisation de connaissances linguistiques permet, dans une certaine mesure, d'éliminer les ambiguïtés. Mais l'expérience prouve qu'un énoncé ne peut être parfois que partiellement compris. L'information manquante peut alors être complétée à l'aide d'une phase de dialogue.

  Prenons l'exemple suivant, où le caractère ? -point d'interrogation- symbolise une lacune dans la compréhension.

  Utilisateur 1 : *Je voudrais le poste de Monsieur ?*

  Système 1 : *Veuillez répéter le nom de votre correspondant s'il vous plaît*

  Dans un système de compréhension de la parole, ces situations sont généralement provoquées par les limites imposées par la phase de reconnaissance. Il s'agit d'un phénomène propre à l'oral qu'on ne rencontre pas dans une communication écrite.

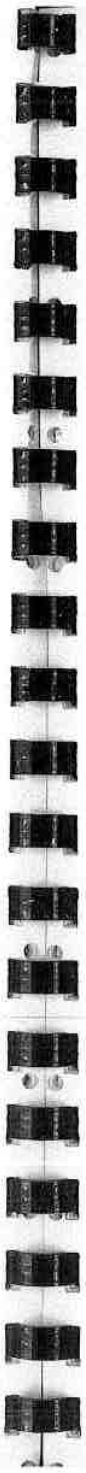
- **la gestion du canal de communication :**

  Nous regroupons, sous cette appelation, plusieurs sortes d'échanges tels que :

  - les énoncés de mise en attente : lorsque le traitement en cours demande un temps relativement long, il convient de faire patienter l'interlocuteur en émettant des messages stéréotypés comme, par exemple :

    Veuillez patienter s'il vous plaît...

  - les énoncés de relance : à l'inverse, en présence d'un long silence anormal, il faut pouvoir relancer l'utilisateur en répétant, par exemple, la dernière intervention.



  - Les énoncés de maintien du canal : pour informer l'interlocuteur de la bonne réception du message en cours, nous rencontrons des énoncés du type :

    Oui,
    bien,
    ok...

- **la gestion de l'historique. :** au cours d'un dialogue, nous observons fréquemment des références à des objets ou à des événements antérieurement cités. Ces références se traduisent par des énoncés elliptiques ou anaphoriques Prenons l'exemple de dialogue suivant :

  U1 : *Quel est le numéro de Monsieur X à Nancy?*

  S1 : *83 33 33 33*

  U2 : *Et celui de Monsieur Y?*

  Nous pouvons constater que l'énoncé noté S1 est une ellipse car, pour être complet, il peut être paraphrasé par l'énoncé suivant :

  Le numéro de Monsieur X à Nancy est le 83 33 33 33.

  Nous pouvons également noter que U2 contient un anaphore car le mot *celui* fait référence à *numéro* qui apparaît dans U1.

  Cet exemple montre qu'il est indispensable de disposer d'une représentation de l'historique du dialogue afin de créer un contexte dans lequel s'effectue une interprétation correcte des références éventuelles.

- **la gestion de la tâche :** une requête, formulée par un locuteur, peut s'avérer incomplète. Le système doit, alors, être capable, en posant les questions appropriées, d'obtenir les éléments manquants à la réalisation de la tâche demandée. Cette situation est illustrée par l'exemple suivant :

  U1 : *Je voudrais le numéro de Monsieur X.*

  S1 : *Où habite votre correspondant?*

  U2 : *A Nancy.*

  S2 : *C'est le 83 33 33 33.*

- **le contrôle du dialogue :** nous pouvons difficilement imaginer un échange où seul l'un des deux intervenants prend l'initiative tout au long d'un dialogue. Dans un dialogue naturel, l'un et l'autre dirigent alternativement l'échange. Il s'agit donc de déterminer de quelle façon et à partir de quel moment la machine peut prendre l'initiative du dialogue.

Voici un exemple de dialogue, où, tour à tour, l'usager puis la machine prend l'initiative du dialogue.

U1 : *Je voudrais un numéro de téléphone.* ← initiative : U

S1 : *Quel est le nom de votre correspondant?* ← initiative : S

U2 : *Monsieur X, il habite à Nancy.* ← initiative : U

S2 : *C'est le 83 33 33 33.*

U3 : *Merci.*

S3 : *Désirez-vous autre chose?* ← initiative : S...

- **la focalisation :** il s'agit d'exploiter les informations acquises en cours de dialogue, pour orienter les traitements sur les interventions futures du locuteur. Cela se traduit par des prédictions d'ordre lexicale, syntaxique, sémantique...sur les énoncés. Il devient possible, par ce biais, de réduire la masse des connaissances à utiliser.

3 LES PREMIERS SYSTÈMES DE DIALOGUE

Dans les premiers systèmes de compréhension de la parole, le dialogue intervient pour une grande part dans la validation de la reconnaissance. Mais déjà en 1975, une solution pour l'interprétation de certaines formes d'ellipses [WALKER 78] a été proposée dans le système du SRI-SDC.

Voici deux exemples de dialogues proposés par les auteurs :

U1 : *Est-que les Britaniques possèdent tous les sous-marins?*

S1 : ...

U2 : *Et les porte-avions?*

S2 : ...

et

U1 : *Est-que les Britaniques possèdent les porte-avions?*

S1 : ...

U2 : *Et les Etats-Unis?*

S2 : ...

Il faut préciser que ce système renferme une représentation de l'univers de l'application sous la forme d'un vaste réseau sémantique [COULON 86] ; chaque objet apparaissant dans la conversation, est une instance d'un noeud de ce réseau. L'énoncé elliptique est alors complété avec des éléments du précédent énoncé du locuteur, en effectuant une comparaison des classes syntaxiques et sémantiques.

Cette solution reste cependant incomplète car elle ne résout qu'une forme restreinte d'ellipses.

En France, on assiste à l'élaboration du logiciel CADI [2] [AN 81] qui, dans la même structure hiérarchique, intègre le modèle général de dialogue et le modèle de la tâche (voir figure 2.2 et figure 2.3).

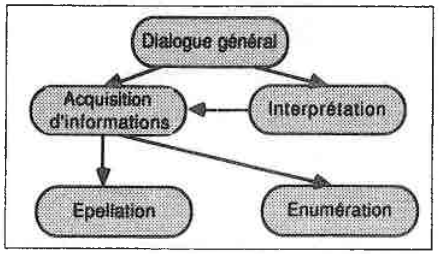


Figure 2.2: Graphe du dialogue dans CADI.

Une *phase* est une représentation instantanée du dialogue ; elle est constituée d'une structure hiérarchique avec un état associé à chaque noeud. A chaque échange, la phase est réexaminée. Cette opération a pour but de modifier l'état des noeuds concernés par le dernier énoncé.

Cette méthode permet de réduire sensiblement le contrôle exercé par la machine car l'utilisateur ne se trouve jamais dans l'obligation de répondre strictement aux questions de la machine ; des informations supplémentaires peuvent naturellement être apportées.

En outre, CADI possède un mécanisme de focalisation qui se traduit par l'émission de prédictions syntaxiques sur le prochain énoncé du locuteur.

Nous pouvons, cependant, remarquer que ce système, bien que prévu pour un dialogue oral, ne prend pas du tout en compte les énoncés de gestion de dialogue ne portant pas sur la tâche tels que les contestations, les demandes de reformulation, les énoncés de satisfaction... Or, une étude de BUNT [BUNT 81] rapporte que ce type d'énoncés -appelé *acte de contrôle de dialogue* par l'auteur- apparaît en grande proportion dans un dialogue naturel.

Parmi les rôles du dialogue que nous avons énumérés dans la section précédente, cer-

[2] CADI : Constructeur Automatique de Dialogue Intelligent

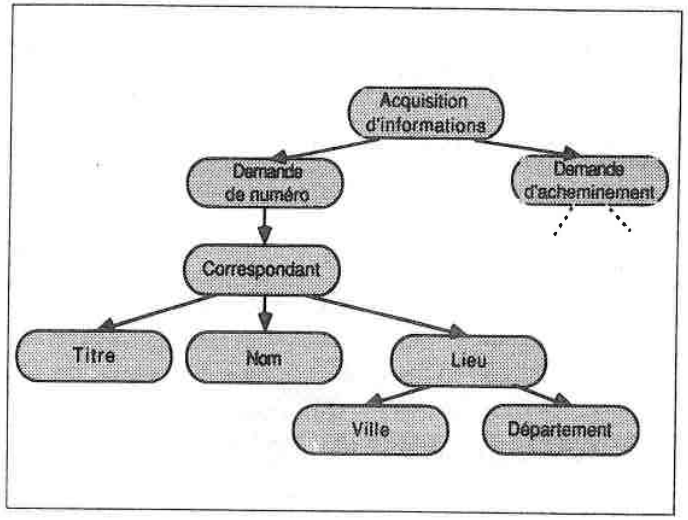


Figure 2.3: Graphe de l'acquisition d'information dans CADI.

taines sont propres à l'oral, tandis que d'autres sont communes avec l'écrit. De ce fait, nous allons également nous intéresser aux travaux sur le dialogue en langage naturel avec l'écrit comme support de communication.

Nous allons limiter ce rapide survol à deux réalisations.

La première est GUS [3] [BOBROW 77a]. Ce système présente un intérêt évident par l'originalité de sa conception. Les *frames* ou schémas, introduite par MINSKY[MINSKY 75], alliant les avantages d'une approche déclarative et l'efficacité d'une approche procédurale, servent de formalisme de base à ce système.

Comme nous pouvons le constater, dans l'exemple de dialogue suivant, GUS permet de simuler un opérateur dans une agence de voyage.

S1 : *Bonjour, mon nom est GUS. Je peux vous aider à planifier un voyage simple en avion. Où désirez-vous aller?*

U1 : *Je voudrais aller à San Diego le 28 Mai.*

S2 : *A quelle heure voulez-vous partir?*

U2 : *Je dois être à San Diego à 10h.*

[3] GUS : Genial Understanding System

S3 : *Le vol PSA qui arrive à 9h15 minutes vous convient-il?*

U3 : *Quel est le vol suivant?*

S4 : *Air California 310 qui part à 8h30 et qui arrive à 10h*

U4 : *Je prendrai le premier.*

S5 : *A quel nom faut-il réserver?*

U5 : *Bobrow...*

Dans cet exemple, nous nous apercevons que le système présente plusieurs avantages :

- l'initiative est mixte : l'utilisateur ne répond pas à la question du système à U3.
- les références sont traitées : il y a une anaphore à l'intervention U4 et une ellipse, à U5.
- le système minimise l'échange en utilisant des valeurs implicites : la ville de départ est, par défaut, *San Jose*.
- il existe une structure du dialogue et elle correspond à la description de la tâche dans un *frame*.

Mis à part le fait que ce système soit destiné à l'écrit, les mêmes critiques concernant les énoncés de gestion de dialogue peuvent être formulées à l'égard de ce système.

La deuxième réalisation que nous voudrions citer, est un système de questions–réponses sur les rubriques professionnelles telles que nous pouvons les trouver dans les pages jaunes de l'annuaire [SABAH 83]. En effet, le dialogue engendré par ce système présente des aspects originaux qui nous semblent intéressants.

Comme le montre l'extrait de dialogue suivant, ce système est capable d'expliquer son propre raisonnement.

U1 : *Je voudrais le numéro de téléphone d'un restaurant d'extrême–orient.*

S1 : *Préférez-vous un restaurant chinois, vietnamien ou japonais?*

U2 : *C a m'est égal.*

S2 : *Dans quel quartier?*

U3 : *Je voudrais d'abord savoir si vous avez le numéro d'un radio-taxi?*

S3 : *Vous avez dans la rubrique taxi :...*

U4 : *Dans ce cas, j'aimerais un restaurant dans le 5ième arrondissement.*

S4 : ...

La particularité de ce système réside dans :

- le raisonnement qui permet de sélectionner une rubrique appropriée.
- l'explication du raisonnement [VILNAT 84],
- et le suivi de thème [GRAU 83].

L'intégration de ces nouvelles fonctionalités conduisent à un comportement plus *intelligent* du système. Cependant, nous pouvons noter qu'ici non plus, le cas des énoncés de contrôle de dialogue n'est pas abordé.

4 LE PROJET "DIALOGUE" ACTUEL

4.1 Les acquis des projets précédents

Notre étude s'inscrit dans le cadre d'un projet de compréhension automatique de dialogues oraux finalisés, actuellement en cours de développement au CRIN.

Ce projet fait logiquement suite aux systèmes MYRTILLE I et MYRTILLE II, comme nous allons pouvoir le constater à travers un bref rappel sur l'évolution, au CRIN, des recherches en compréhension de la parole.

Les premiers travaux, en 1975, ont aboutit à la réalisation du système MYRTILLE I qui possède l'avantage de fonctionner en temps réel sur un calculateur de puissance moyenne (MITRA 15). Si ce système tolère quelques écarts par rapport à la syntaxe, il n'en demeure pas moins que les énoncés traités proviennent d'un langage artificiel, défini à l'aide de grammaires formelles.

Le fait de contraindre un usager à utiliser un langage artificiel peut parfois constituer une barrière insurmontable aux applications destinées au grand public.

Le sytème MYRTILLE II a été développé à la suite de ce constat, dans le but de traiter un sous-ensemble plus étendu du langage naturel. Le système, qui se présente comme un prototype expérimental, a permis de tester quelques idées novatrices dans le domaine.

En effet, les RNPs [4], qui constituent un formalisme de base de ce système, autorisent à la fois une description plus souple du langage et, l'utilisation de contraintes sémantiques qui régissent le langage.

De plus, grâce aux procédures associées aux noeuds des RNPs, MYRTILLE II permet une étroite coopération entre différents niveaux de connaissances.

[4] RNP : Réseau à Noeuds Procéduraux.

Notre propos n'est pas de détailler davantage ces deux systèmes ; une présentation très complète de ces deux réalisations est effectuée dans [PIERREL 81]

Cependant, l'exploitation de MYRTILLE I, ainsi que celle de MYRTILLE II, demande l'élaboration d'un module pragmatique, adapté à l'application, qui puisse traiter le dialogue. Cette approche réduit considérablement les interactions entre le niveau pragmatique et les autres niveaux.

Cela nous a conduit à définir une nouvelle architecture, ayant pour objectif, non plus la compréhension d'une phrase, mais celle d'un dialogue en langage naturel. Nous pensons que cette nouvelle orientation vers le traitement du dialogue va conduire à une communication plus naturelle et, de ce fait, plus confortable pour un utilisateur non-entraîné.

4.2 Le cahier des charges

Notre objectif est la conception et la réalisation d'un système informatique constituant une interface orale homme-machine, efficace et confortable pour l'usager, dans le cadre d'une application destinée au grand public [CARBONELL 84].

Nous nous proposons de traiter les communications où un utilisateur naïf s'adresse à une machine dans l'intention de réaliser une tâche. Nous partons de l'hypothèse que le sujet adopte un comportement coopératif caractérisé par les maximes de GRICE [GRICE 75] que nous reproduisons à figure 2.4.

MAXIME DE QUANTITÉ :	ne donner ni trop, ni trop peu d'informations.
MAXIME DE QUALITÉ :	être sincère et parler à bon escient.
MAXIME DE PERTINENCE :	être pertinent.
MAXIME DE MANIÈRE :	être facilement compréhensible.

Figure 2.4: Les maximes de Grice pour une conversation coopérative.

Exception faite des principes de coopération, nous voulons dispenser le locuteur de toute autre forme de conditionnement préalable. Cela signifie que le locuteur n'effectue pas d'apprentissage et n'a pas, non plus, de consignes.

Ces conditions nous amènent à envisager un système qui :

- **gère et structure un dialogue oral finalisé :** nous voulons éviter que le dialogue se présente comme un *menu* où tout le contrôle serait attribué à la machine. Pour plus de naturel, il nous semble indispensable que le locuteur puisse également prendre l'initiative lorsqu'il le juge opportun.
- **comprend et satisfait les requêtes en provenance d'un usager s'exprimant en français :** la compréhension s'effectue dans un domaine limité par l'application. Par conséquent, sur le plan syntaxique et lexical, le système accepte un sous-ensemble

du Français parlé, suffisant pour formuler n'importe quelle requête dans le cadre de l'application.

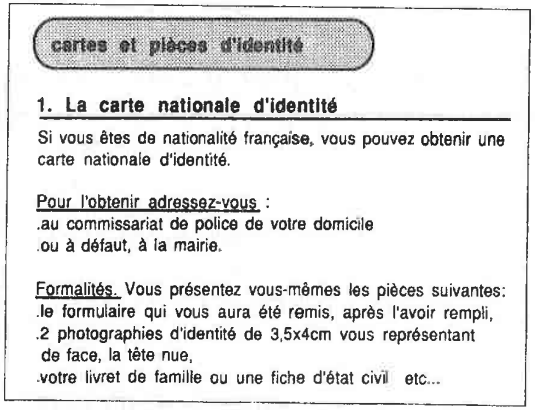
Sur le plan sémantique et pragmatique, l'interprétation d'un énoncé se fait en fonction du contexte créé conjointement par la tâche à accomplir et le dialogue en cours. Le locuteur a ainsi la possibilité de produire des énoncés anaphoriques et elliptiques.

4.3 L'application retenue

Les applications potentielles du système sont :

- la consultation de bases de données telle que les renseignements administratifs, météorologiques, téléphoniques (PTT) ou concernant des horaires (SNCF)...
- les demandes de rendez-vous dans un centre médical...
- les réservations dans une agence de voyages, au cinéma, au théâtre...
- les demandes de locations dans une agence immobilière,
- les commandes d'articles dans un catalogue de vente par correspondance.

Pour tester notre système, nous avons opté pour les renseignements administratifs. Dans cette application, la machine essaye de répondre aux requêtes portant sur les formalités administratives courantes. Notre base de renseignements est directement inspirée des pages roses de l'annuaire téléphonique des PTT (voir figure 2.5).



cartes et pièces d'identité

1. La carte nationale d'identité

Si vous êtes de nationalité française, vous pouvez obtenir une carte nationale d'identité.

Pour l'obtenir adressez-vous :
.au commissariat de police de votre domicile
.ou à défaut, à la mairie.

Formalités. Vous présentez vous-mêmes les pièces suivantes:
.le formulaire qui vous aura été remis, après l'avoir rempli,
.2 photographies d'identité de 3,5x4cm vous représentant de face, la tête nue,
.votre livret de famille ou une fiche d'état civil etc...

Figure 2.5: Un extrait des pages roses de l'annuaire des PTT

Notre choix se justifie par le fait que cette application engendre des dialogues suffisamment variés pour que l'on puisse tester les différentes capacités attendues du système. En effet :

- la population concernée est le grand public,
- l'application se compose d'une variété de tâches suffisamment simples et relativement bien structurées,
- la taille du lexique, qui avoisine 1000 mots, semble réaliste pour un traitement en temps réel. Cette estimation provient de l'étude d'un corpus que nous avons constitué en simulant un centre de renseignements administratifs [ROUSSANALY 86]. Sur les 42 conversations enregistrées, nous comptons l'apparition de 1377 mots différents, dont 868 sont attribués au locuteur et 937 à l'opératrice.
- et enfin, en se basant sur ce même corpus, nous pouvons raisonnablement nous attendre à un dialogue finalisé avec, en outre, un contrôle du dialogue partagé.

  Il est également intéressant de souligner que ce corpus contient, en moyenne, 14 échanges par dialogue en situation homme-homme. Ce nombre, relativement élevé, laisse présager une grande diversité dans les interventions.

4.4 Le dialogue visé

La conversation suivante constitue un exemple de dialogue tel qu'on l'envisage dans le cadre de notre application, en supposant que la compréhension soit correcte.

S : *Ici le centre de renseignements administratifs, bonjour.*

U : *Bonjour, voilà, j'ai égaré ma carte d'identité. Où peut-on me la renouveler?*

S : *Est-ce à la suite d'une perte ou d'un vol?*

U : *Il s'agit certainement d'une perte.*

S : *Dans ce cas, présentez-vous à la mairie avec une déclaration de perte, deux photos d'identité et votre livret de famille ou une fiche d'état civil.*

U : *Comment obtient-on une déclaration de perte?*

S : *Pour une déclaration de perte, adressez-vous à la préfecture.*

U : *Merci.*

S : *Désirez-vous un autre renseignement?*

U : *Non, merci. Au revoir.*

S : *Au revoir.*

Il faut néanmoins rester conscient du fait que ce type de dialogue ne se produit que rarement dans la réalité. Les enregitrements de dialogues oraux montrent que :

- en fonction du locuteur, la structure d'un énoncé est très variable,
- les phénomènes d'hésitations et de reprises sont très fréquents, rendant l'analyse d'une phrase difficile avec les méthodes classiques.

Les spécificités de l'oral entrainent la mise en oeuvre de processus de compréhension assez souples, mais qui risquent d'introduire des erreurs d'interprétation. Au niveau du dialogue, cela se manifeste de différentes manières : contestation de la part du locuteur, demande de reformulation ou de précision de la part de la machine...

D'autre part, il faut tenir compte des limites actuelles de la reconnaissance de la parole. En dépit de l'utilisation de connaissances linguistiques, la reconnaissance reste encore partielle et peut altérer la compréhension. Il s'agit donc de prévoir ces situations, et d'y répondre en instaurant des demandes de confirmation ou de répétition.

En cours de dialogue, nous autorisons également les énoncés de gestion du canal de communication lorsque ce dernier est à l'origine d'une reconnaissance défectueuse. Selon les situations, nous prévoyons alors d'envoyer des consignes telles que :

Veuillez parlez plus fort,
Articulez davantage...

4.5 Les sources de connaissances

Nous comptons mettre à la disposition de ce système toutes les sources d'informations utilisées, à notre connaissance, par l'être humain dans son activité de communication. Rappelons brièvement ces connaissances que nous avons déjà énumérées dans le premier chapitre de ce manuscrit. Il s'agit de l'acoustique, la phonétique, la phonologie, la prosodie, le lexique, la syntaxe, la sémantique et, enfin, la pragmatique.

Les connaissances pragmatiques, auxquelles nous nous intéressons plus particulièrement dans l'étude présente, sont constituées :

- **des connaissances sur l'univers de l'application**, qui regroupent les concepts spécifiques susceptibles d'apparaître dans l'application, les relations mutuelles entre les concepts et les liens entre ces concepts et une structure de surface,

- **des connaissances sur la tâche**, qui se composent de la base de renseignements proprement dite et du modèle de la tâche,
- **des connaissances sur le dialogue**, formées des scénarios possibles de dialogue, incluant les stratégies en fonction de la tâche à accomplir,
- **les connaissances sur l'usager**, qui permettent de choisir les informations implicites sur l'utilisateur telles que, en relation avec notre application, sa nationalité, son lieu de résidence...

Il est vrai que cette classification des connaissances a une influence sur les modèles théoriques qui sont à l'origine de la plupart des systèmes de compréhension de la parole. Néanmoins, elle ne rend pas compte des informations ayant un caractère dynamique qui interviennent dans le processus de traitement.

Dans un souci de paramétrisation du système, nous proposons une approche plus opératoire où nous distinguons les informations selon deux critères :

- **Les connaissances statiques** ; ce sont les connaissances qui varient peu, voire pas du tout, au cours du temps. Elles sont composées :
  - des informations indépendantes de l'application : il s'agit essentiellement des connaissances de bas niveau telles que l'acoustique, la phonétique, la phonologie et la prosodie.
  - des informations communes à une famille d'applications : en l'occurrence, notre application appartient à la famille des demandes de renseignements. Nous y incluons la syntaxe et le modèle statique du dialogue.
  - des informations spécifiques à l'application : elles contiennent le lexique et l'univers de l'application.
- **Les connaissances dynamiques** ; ce sont les informations susceptibles d'évoluer avec le temps. Cette classe comprend :
  - les informations liées à l'évolution de l'application, formées principalement des connaissances sur la tâche. En effet, si nous nous référons à notre application, nous constatons que certaines données, comme, à titre d'exemple, le coût d'un document, peuvent être modifiées.
  - les informations liées au dialogue en cours, à savoir l'historique qui renferme un représentation instantanée du dialogue en cours et qui s'enrichit au fur et à mesure des échanges.

- les informations liées à un énoncé : il s'agit des représentations successives de l'énoncé en cours de traitement (représentations acoustique, phonétique, lexicale...).

Chapitre 3

UNE PROPOSITION D'ARCHITECTURE

1 LES PRINCIPES GÉNÉRAUX

Pour implanter notre système, nous avons choisi une approche de type multi-bases de connaissances, qui se traduit par une organisation modulaire et une base de connaissances spécifique à chaque module. Cette technique, qui est, par ailleurs, couramment appliquée aux travaux se rapportant à l'intelligence artificielle [NII 86b] [LAASRI 87], rajoute deux caractéristiques importantes au système :

- **la modularité :** Notre système comprend plusieurs modules. Chaque processeur opère sur un type particulier de données et utilise une base de connaissances spécifique. Le découpage permet de mener en parallèle la réalisation des différents modules.

  En outre, le fait de discerner le traitement et les connaissances facilite les interventions du concepteur, offrant ainsi la possibilité d'une mise au point incrémentale du système.

- **la flexibilité :** Notre classification opératoire des connaissances permet de définir les différents niveaux de paramétrisation. Cela rend la mise en place d'une nouvelle application plus aisée (voir figure 3.1).

D'autre part, notre option nécessite le développement :

- d'interfaces conviviales afin que le concepteur puisse facilement manipuler les connaissances des différents modules [MOUSEL 86a],
- et d'outils permettant de vérifier leur consistance.

Par ailleurs, d'un point de vue méthodologique, nous avons constitué un corpus [ROUSSANA formé d'enregistrements de dialogues concernant des demandes de renseignements admi-

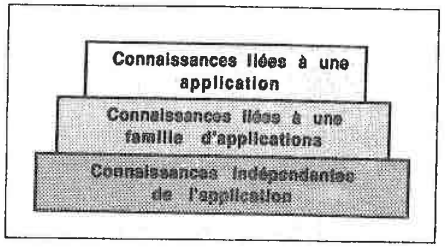


Figure 3.1: Les niveaux de paramétrisation du système.

nistratifs entre un usager et une opératrice. Ce corpus nous sert, d'une part, à valider un modèle de représentation de connaissances, d'autre part, à élaborer les connaissances nécessaires à l'application.

2 LES COMPOSANTES

Notre système est composé de cinq modules : APHON pour le décodage acoustico-phonétique, PROSO pour la prosodie, LEX pour le traitement lexical, SYN-SEM pour la syntaxe et la sémantique et enfin, DIAL pour le dialogue.

Dans ce paragraphe, nous allons passer en revue chacune de ces composantes, exceptée la composante DIAL, que nous allons développer ultérieurement.

2.1 Le décodage acoustico-phonétique : APHON

Ce module forme le premier maillon du processus de compréhension. Il construit la description phonétique d'un énoncé, à partir du signal acoustique numérisé. Le résultat est un treillis de phonèmes, représenté par une suite de segments. A chaque segment, est associée une liste des phonèmes les plus probablement prononcés par le locuteur.

Pour sa réalisation, une approche de type *système expert* a été développée [FOHR 86]. Elle repose sur l'observation d'experts phonéticiens qui sont capables d'établir la description phonétique d'un énoncé à partir d'une représentation visuelle -spectrogramme- de la parole. Il s'agit, par conséquent, de formaliser cette connaissance à l'aide de règles spécifiques et d'utiliser des techniques classiques d'intelligence artificielle pour reconstituer le raisonnement de l'expert.

Cette composante est donc formée :

- d'un ensemble de modules de prétraitements qui permettent d'effectuer une segmentation du signal sous forme de *macro-classes* -détection des noyaux vocaliques, des plosives et des fricatives-,

- d'une base de connaissances qui contient l'ensemble des règles d'expertises
- et d'un moteur d'inférence.

Un exemple de règle est fournit à la figure 3.2. Il s'agit d'une règle qui permet d'éliminer le phonème *n*, lorsqu'il est en concurrence avec le phonème *m*, dans le cas où le phonème suivant est un *i*.

```
R80 :

        RESULTAT [m n]
        CONTEXTE-DROIT [i]

        SI
          transition-F3-montante
          transition-F2-montante
          energie ^(2000-2400)

        ALORS
          [m 1 n -1]
```

Figure 3.2: Une règle de discrimination dans le décodage acoustico-phonétique.

A travers cet exemple, nous pouvons constater que, pour identifier un phonème, le module de décodage acoustico-phonétique combine les connaissances acoustiques -mesure d'énergie, calcul des formants-, phonétiques -conclusion de la règle- et phonologiques -contexte de la règle-.

Il faut noter l'aspect multi-locuteur du décodage acoustico-phonétique, résultant de l'indépendance de l'expertise par rapport aux locuteurs.

Dans sa version actuelle, ce processeur fournit un taux moyen de reconnaissance qui se situe autour de 70%. Cette performance, encore assez faible, est toutefois compensée par le fait que les noyaux vocaliques (>95%), les fricatives (>90%) et les plosives (>85%) soient détectés de façon suffisamment fiable.

2.2 Les traitements prosodiques : PROSO

Ce module détient un double rôle dans le système :

- il contribue, d'une part, à déterminer la signification de l'énoncé par l'émission des hypothèses concernant la forme d'un énoncé -interrogative, énonciative, impérative, exclamative-. Considérons la phrase suivante :

  C'est la préfecture qui délivre les passeports

Prononcée avec une intonation ascendante en fin d'énoncé, cette phrase peut être considérée comme une interrogation. A l'inverse, une intonation descendante indique une énonciation.

Cette information vient en complément des données destinées au processeur syntaxico-sémantique qui s'en sert pour lever certaines ambiguïtés.

- il s'efforce, d'autre part, de localiser des marqueurs prosodiques qui indiquent, sur le plan lexical, une frontière de mot et, sur le plan syntaxique, une frontière de syntagme. Ainsi, les hypothèses lexicales ou syntaxiques, qui entrent en contradiction avec les informations prosodiques, peuvent être abandonnées par le processeur correspondant. Par conséquent, le module PROSO doit retenir uniquement les marqueurs non-ambigus, c'est à dire ceux qui sont détectés avec un taux de certitude suffisamment élevé. Un premier test sur plusieurs corpus [BONIN 87], visant à détecter la dernière voyelle accentuée dans un mot, fournit un taux de succès proche de 95%. Ce taux est calculé de la façon suivante :

$$taux = \frac{nb\ proposé\ avec\ succès}{nb\ total\ proposé}$$

Ce traitement s'effectue séquentiellement de la manière suivante : en premier lieu, le processeur construit les courbes de l'évolution de la fréquence fondamentale à partir du signal. Ensuite, l'interprétation des *pics*, des *vallées* et des *plateaux* de ces courbes permet de déterminer les marqueurs prosodiques.

2.3 Le traitement lexical : LEX

L'objectif de ce module est de détecter des mots dans le treillis de phonèmes fourni en entrée. Cette opération est déclenchée de différentes manières :

- **la vérification de mots :** cela se produit lorsqu'un processeur de niveau supérieur émet une hypothèse concernant la présence d'un mot particulier à un instant donné de l'énoncé. Cette hypothèse, qui constitue une donnée du module LEX, est un couple composé :
  - d'une forme de référence représentant une transcription phonétique du mot-hypothèse
  - et d'un repère temporel indiquant le début du mot dans le treillis de phonèmes.

  Le traitement consiste à apparier la forme de référence et le sous-treillis correspondant au mot à vérifier. Le résultat est une validation ou un rejet de l'hypothèse traitée.

- **la recherche de mots :** lorsque le processeur ne dispose pas d'information temporelle sur un mot à valider, il procède à une recherche itérative en effectuant un balayage du treillis de phonèmes. A chaque itération, un appariement, entre le mot de référence et la partie sélectionnée dans le treillis, est tenté.

  Cette opération séquentielle rend le processus très coûteux en temps de traitement. Pour réduire ce dernier, on peut s'appuyer sur les hypothèses prosodiques qui vont permettre d'éliminer les appariements avec un sous-treillis contenant un marqueur prosodique.

Dans une analyse lexicale, l'appariement consiste à comparer chaque phonème du mot de référence à son vis-à-vis dans le treillis phonétique. Il faut cependant tenir compte des incertitudes et des erreurs introduites par le décodage acoustico-phonétique.

En effet :

- Une segmentation à outrance peut provoquer la duplication de certains phonèmes.
- A l'inverse, un effet de sous-segmentation risque de faire disparaître d'autres phonèmes.
- D'autre part, pour un phonème, le processeur APHON fournit plusieurs solutions possibles.
- De plus, il arrive que le phonème réellement prononcé n'apparaisse pas dans le choix proposé.

Pour résoudre les problèmes posés par les deux premiers points, l'algorithme d'appariement admet les élisions et les insertions de phonèmes. Le résultat n'est plus un booléen indiquant la validation ou le rejet d'une hypothèse, mais un score de reconnaissance qui constitue une mesure de certitude d'apparition du mot de référence. Ce score est calculé en appliquant une pénalité à la suite de chaque resynchronisation provoquée par une élision ou une insertion. Le score initial est donc le score idéal (100%).

Dans le cas où le phonème recherché n'apparaît pas parmi les choix possibles, la solution que nous mettons en oeuvre [HATON 86] repose sur l'idée intuitive qui consiste à pénaliser la reconnaissance en fonction du degré de ressemblance entre le phonène recherché et les phonèmes proposés.

Une première comparaison grossière est effectuée en se basant sur les *patrons phonétiques* des mots.

Un patron phonétique est une suite de macro-phonèmes définis comme des classes de phonèmes créées en fonction de critères de proximité.

Cette première passe permet de rejeter rapidement un mot s'il est trop *éloigné* de la référence. En revanche, s'il est retenu, une analyse plus fine, reposant sur les phonèmes, va être effectuée. Une hypothèse est rejetée quand le score devient inférieur à un seuil fixé.

Le résultat de l'analyse lexicale est un ensemble de mots ; à chaque mot sont associés un score de validité et un repère sur l'axe temporel de l'énoncé. Cette structure est représentée à l'aide d'un *treillis de mots* (voir figure ??).

Ce module utilise un lexique contenant l'ensemble des mots de l'application. Le lexique est utilisé pour extraire la (les) représention(s) phonétique(s) d'un mot à partir de sa forme orthographique. Mais, il arrive que le module SYN-SEM demande la validation d'un ensemble de mots ayant la même fonction syntaxique ou possédant des traits sémantiques communs. Afin de permettre une sélection selon ces critères, il est indispensable d'affecter aux éléments du lexique des attributs syntaxiques et sémantiques.

Dans la suite de ce manuscrit, nous allons faire la distinction entre un *mot validé*, obtenu à la suite de la validation d'une hypothèse syntaxico-sémantique, et un *mot reconnu* résultant d'une recherche effectuée par la propre initiative du processeur lexical.

2.4 Les analyses syntaxico-sémantiques : SYN-SEM

La finalité du module SYN-SEM est de produire une ou plusieurs structures syntaxico-sémantiques correspondant à un énoncé en langage naturel.

La représentation retenue pour les structures trouve son origine dans le modèle des grammaires de cas [FILLMORE 68] [BRUCE 75]. Pour notre application, DEVILLE ET PAULUSSEN [DEVILLE 86] proposent un modèle linguistique de type casuel, partiellement inspiré de la grammaire fonctionnelle de DIK [DIK 79]. Ce modèle a la particularité de proposer une représentation sémantique de chaque énoncé selon une structure composée d'un prédicat -dérivé d'une primitive- et d'un certain nombre d'arguments de cette primitive dont les termes sont spécifiés au moyen de fonctions sémantiques ou *cas* définis comme l'expression d'une relation sémantique entre un terme et une primitive. La structure casuelle d'une primitive correspond donc à une séquence de cas spécifiques -obligatoires ou optionnels- à cette primitive. (voir figure 3.3).

Le résultat de l'analyse correcte de la phrase :

> *Je voudrais savoir où il faut s'adresser pour faire une demande de double nationalité.*

est une structure représentée par le schéma de la figure 3.4.

Le fonctionnement de ce processeur est caractérisé par la simultanéité des traitements syntaxiques et sémantiques. Cela est rendu possible grâce à l'utilisation d'une base d'informations commune appelée *espace des théories* (THEORIES)(voir figure 3.5)

Primitive	agt	pat	obj	ben	src	des	loc	tmp	ins	mes	med	goa	cond	cau
MVT1	+	-	-	-	+	+	-	0	-	-	0	0	0	0
MVT2	+	-	+	-	+	+	-	0	-	-	0	0	0	0
ACTANIM	+	-	-	+	-	-	0	0	0	0	0	0	0	0
ACTOBJ	+	-	+	0	-	-	0	0	0	-	0	0	0	0
ECHPROD	+	-	+	+	-	-	0	0	-	-	0	0	0	0
ECHOBT	0	-	+	+	-	-	0	0	-	-	0	0	0	0
ATRANS	+	-	-	0	-	-	0	0	-	-	0	0	0	0
LOCAT1	-	?	?	-	-	-	+	0	-	-	-	-	0	0
LOCAT2	-	+	+	0	-	-	0	0	-	-	-	-	0	0
EXTEN	-	+	+	-	-	-	-	0	-	-	-	-	0	0
STAT1	-	+	-	-	-	-	-	0	-	-	-	-	0	0
STAT2	-	-	+	-	-	-	0	0	-	0	-	-	0	0
MESU1	-	+	-	-	-	-	-	-	-	+	-	-	0	0
MESU2	-	-	+	0	-	-	0	-	-	+	-	-	0	0
PROCES1	-	+	0	-	-	-	0	0	-	-	-	-	0	0
PROCES2	-	-	+	-	-	-	0	0	0	-	-	-	0	0

Figure 3.3: Les primitives et les cas associés.

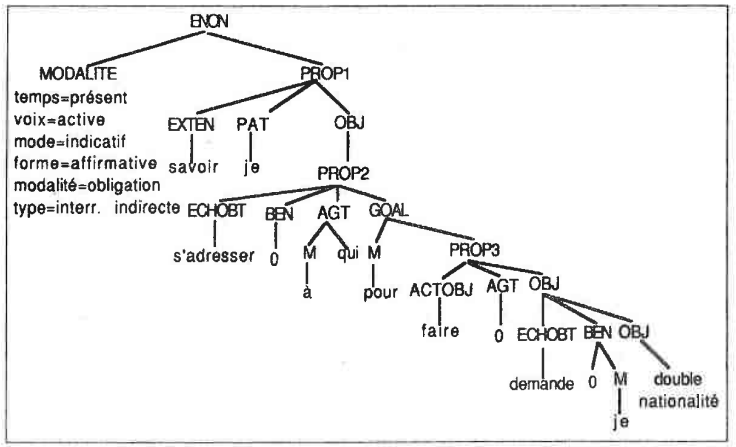


Figure 3.4: La représentation syntaxico-sémantique de la phrase : *Je voudrais savoir à qui il faut s'adresser pour faire une demande de double-nationalité*

[MOUSEL 86a].

Une théorie peut être définie comme l'analyse locale d'un fragment de l'énoncé.

Nous distinguons deux types de théories, caractérisées par la méthode d'analyse qui a

amené leur construction :

- Les théories ascendantes : à l'origine, elles sont construites à partir de mots reconnus provenant du module LEX.
- Les théories descendantes : elles sont engendrées par des hypothèses provenant du module DIAL.

L'analyse syntaxico-sémantique consiste à effectuer les opérations suivantes sur les théories :

- **La création d'une hypothèse-mot :** afin de faire progresser une analyse correspondant à une théorie descendante, on crée des hypothèses lexicales à valider, en s'appuyant sur des connaissances syntaxiques et, éventuellement, sémantiques.
- **La génération d'une théorie descendante :**
  - à partir d'une hypothèse provenant de DIAL : il s'agit d'une manière d'initialiser une théorie descendante.
  - à partir d'une théorie descendante : la progression d'une théorie descendante n'est effective que lorsqu'une une hypothèse lexicale a été validée par le processeur LEX. La validation entraine la génération d'une nouvelle théorie descendante.
  - à partir d'une théorie descendante et d'une théorie ascendante : il arrive qu'une analyse ascendante et une analyse descendante se rejoignent. Dans ce cas, on procède à une unification ; la théorie résultante est de type descendante.
- **La génération d'une théorie ascendante :**
  - à partir d'un mot reconnu : il s'agit également d'un processus d'initialisation des théories ascendantes.
  - à partir de plusieurs théories ascendantes : un ensemble de théories ascendantes peut être assemblé pour former une théorie englobante, également ascendante.

L'organisation de ce module est illustrée à la figure 3.5. Le traitement syntaxique s'appuie sur une base de connaissances définie à l'aide des RNPs. Ce traitement construit une structure syntaxique intermédiaire. Le passage à une représentation conforme au modèle syntaxico-sémantique défini dans la base CGR, est assuré par l'application des règles de correspondance rassemblées dans REGLES.

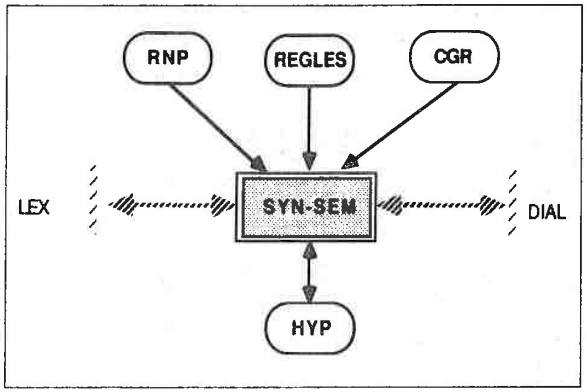


Figure 3.5: Organisation du module SYN-SEM.

3 LE FLUX D'INFORMATIONS

La figure 3.6 présente l'architecture du système.

Nous supposons que chaque processeur dispose de connaissances statiques propres, qui n'apparaissent pas sur cette figure. La circulation des informations est symbolisée par les flèches qui relient les différents modules.

Nous distinguons trois sortes d'informations susceptibles d'être échangées par les différents processeurs :

- **Les hypothèses émises par un processeur de niveau supérieur :** Ces informations présentent un caractère prédictif. Elles demandent à être valider par un processeur de niveau inférieur. Dans notre architecture, il s'agit :
  - des prédictions lexicales, issues du processeur DIAL ; ce sont des mots-clefs sélectionnés pour leur forte probabilité d'apparition dans l'énoncé qui va suivre, en fonction de l'état du dialogue en cours,
  - des prédictions syntaxico-sémantiques, provenant également du module DIAL ; elles vont être utilisées par SYN-SEM pour amorcer le processus d'analyse descendante,
  - des hypothèses-mots, émises par le processeur syntaxico-sémantique au cours d'une analyse descendante et à valider par LEX.
  - des structures phonétiques, construites par LEX et correspondant à un mot à vérifier directement sur le signal par le processeur APHON. Dans la version

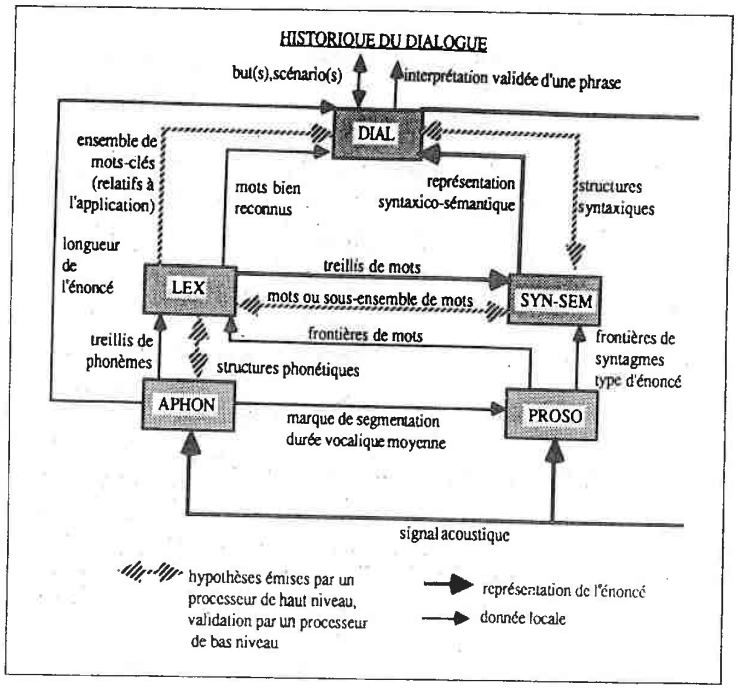


Figure 3.6: Architecture générale du système.

actuelle du système, nous ne prévoyons pas l'implantation de cet échange car il s'avère coûteux en temps de traitement ; mais nous n'excluons pas, à terme, une vérification lexicale ponctuelle directement sur le signal.

Ces hypothèses induisent un fonctionnement descendant du système, car elles permettent de guider un processeur de niveau inférieur en le focalisant sur une partie de ses connaissances.

- **Les représentations de l'énoncé :** à partir du signal acoustique, qui constitue l'entrée du système, un énoncé passe par plusieurs représentations propres au processeur qui le traite, pour aboutir à une représentation interprétative exploitable pour la gestion de la tâche et du dialogue. Ainsi, APHON construit un treillis de phonèmes, LEX, un treillis de mots, SYN-SEM un ensemble de structures syntaxico-sémantiques, et enfin DIAL, une représentation interprétative que nous allons définir ultérieurement.

Ces représentations sont, soit le fruit d'une validation d'une hypothèse descendante, soit le résultat d'un traitement purement ascendant. Cependant, leur circulation dans le système reste hiérarchique dans la mesure où une représentation créée par un processeur quelconque est destinée au module de niveau immédiatement supérieur. C'est la conséquence de notre conception modulaire du système.

- **Les informations locales sur l'énoncé :** Ce sont les données ponctuelles concernant l'énoncé en cours de traitement. Elles possèdent des rôles très divers au sein du système :
  - **La durée vocalique moyenne :** ce facteur est calculé par le processeur APHON pour ses propres traitements ; mais il est également nécessaire au traitement prosodique. Il s'agit donc d'une transmission de résultat intermédiaire, commun aux deux modules.
  - **Les marques de segmentation :** les marques de segmentation, détectées par APHON, sont utilisées par PROSO pour repérer précisément les marqueurs prosodiques.
  - **Les frontières de mots et de syntagmes :** ces informations vont permettre d'éliminer les hypothèses syntaxiques et lexicales qui vont à l'encontre du découpage qu'elles préconisent.
  - **La longueur de l'enoncé :** DIAL s'appuie sur la longueur de l'énoncé, mesurée par APHON, pour restreindre les prédictions syntaxiques.
  - **Les maxima d'énergie :** ces mesures, qui indiquent soit une saturation de l'acquisition la parole, ou, au contraire, une énonciation peu audible, permettent d'envoyer à l'utilisateur des consignes concernant le canal de communication.
  - **Les mots bien reconnus :** dans certaines situations, la détection d'un nombre très restreint de mots-clefs, suffit à la compréhension d'un énoncé. Lorsque ces cas se produisent, on tente de faire l'économie d'une analyse syntaxico-sémantique en envoyant directement les mots bien reconnus au processeur DIAL.

4 LE CONTRÔLE ET LA STRATÉGIE

Dans la conception de notre architecture, nous avons voulu concilier trois points fondamentaux : l'autonomie des processeurs, leur grande coopération et l'exploitation du parallélisme dans les traitements.

L'autonomie est mise en oeuvre :

- de manière interne, par l'utilisation de bases de connaissances spécifiques à chaque processeur,

- de manière externe, par la réalisation de chaque processeur comme une *boîte noire* qui assure les échanges définis dans l'architecture.

La coopération est réalisée par l'existence d'un échange bi-directionnel entre la plupart des processeurs du systèmes. Cet échange est modélisé à l'aide d'une communication de type *producteur-consommateur*.

Ce mécanisme correspond à un contrôle distribué. Il permet donc d'exploiter le parallélisme qui existe entre les différents processeurs.

Dans un traitement ascendant, APHON transmet les informations intermédiaire nécessaires à PROSO. Cette transmission a pour effet d'activer le module prosodique et, dès lors, les deux processeurs deviennent indépendants et peuvent s'éxécuter de façon parallèle.

Il existe également un parallélisme entre le traitement lexical et syntaxico-sémantique. SYN-SEM *produit* des hypothèses lexicales, *consommées* par LEX, qui tente de les valider sur la représentation de l'énoncé qu'il dispose. Réciproquement, LEX transmet les mots validés et reconnus au processeur SYN-SEM qui s'en sert pour bâtir une représentation syntaxico-sémantique.

Un modèle identique de transmission de données existe également entre les processeurs SYN-SEM et DIAL.

Dans un premier temps, nous adoptons un fonctionnement du système selon la stratégie suivante :

- Effectuer une analyse ascendante de APHON vers LEX.
- Parallèlement, il faut effectuer une analyse descendante des processeurs de niveau supérieur vers le niveau lexical. Notons que le processeur LEX, fonctionne en mode à la fois ascendant et descendant.
- En cas d'échec, utiliser les mots reconnus par LEX pour effectuer une analyse ascendante.

L'analyse est arrêtée lorsqu'une ou plusieurs structures complètes ont été construites ou que le processus se trouve bloqué faute d'informations. Cet événement provoque l'activation du processeur DIAL.

Dans cette architecture, la gestion des échanges est complètement décentralisée. Chaque module gère ses propres entrées et communique des informations à un ensemble pré-défini de modules. De cette façon, nous comptons, d'une part, éviter le contrôle trop statique d'un superviseur et, d'autre part, gagner en efficacité par rapport aux structures de type *blackboard*.

5 LES OBJECTIFS ET LA STRUCTURATION DE LA COMPOSANTE DIALOGUE

5.1 Hypothèses de travail

Dans notre système, la composante dialogue constitue le processeur pragmatique. La pragmatique, définie comme l'étude de l'usage de la langue [LEVINSON 83], intervient dans le processus de compréhension du dialogue, à travers les relations entre un message et son contexte d'énonciation.

La notion de contexte est très importante car elle permet de construire un univers dans lequel s'effectue l'interprétation d'un message.

Le contexte est formé par la combinaison de nombreux facteurs. Certains sont liés à l'identité des locuteurs, à leur croyance, à leur intention et à leurs connaissances. D'autres dépendent de l'instant et du lieu de production de l'énoncé.

Dans une perspective d'automatisation, tenter de prendre en compte tous ces facteurs tels qu'ils interviennent dans un dialogue absolument naturel, nous semble une entreprise vouée à l'echec en raison des processus complexes qui sont mis en oeuvre. Cependant, dans la situation qui nous intéresse, leur influence reste heureusement limitée. Cette restriction est rendue possible grâce à la stabilisation de certains paramètres, découlant des hypothèses suivantes :

- en premier lieu, nous considérons qu'il ne peut y avoir d'influences liées à un état affectif de l'utilisateur -et encore moins à celui de la machine-. Ce paramètre nous semble, en effet, très difficile à maîtriser avec une population représentant le *grand public*,
- nous supposons ensuite que l'utilisateur s'adresse à l'ordinateur dans l'intention de lui soumettre une requête. Réciproquement, la machine a pour objectif de satisfaire cette requête.
- La croyance des participants se résume aux deux éléments suivants :
  - le locuteur croit que la machine est capable de satisfaire sa requête ;
  - la machine *croit* que le locuteur a une requête précise à lui soumettre et qu'il va concrétiser sa demande. Elle dispose, d'autre part, d'informations par défaut issues d'une observation statistique ou tout simplement, de suppositions *de bon sens*.

Signalons, toutefois, que toutes ces hypothèses concordent avec les objectifs fixés par le cahier des charges.

Le contexte qui évolue en cours de dialogue et que nous allons avoir à gérer, est donc essentiellement fonction de l'univers de l'application et de l'historique du dialogue.

La modélisation du processus de compréhension du dialogue passe d'abord par le problème de représentation du discours et celui des connaissances pragmatiques. La définition de fonctions qui s'exercent sur ces différentes structures constitue l'étape suivante.

5.2 Analyse de la conversation

Sur la base des conclusions que nous venons de tirer, nous allons maintenant aborder le problème de la représentation du dialogue.

Dans cette perspective, la théorie d'analyse de la conversation, élaborée par des linguistes de Genève [ROULET 80], nous paraît intéressante pour de multiples raisons :

- elle repose sur l'examen de conversations réelles entre un sujet qui désire réaliser une tâche et un professionnel capable de satisfaire la requête : les dialogues étudiés portent sur des enregistrements effectués chez un libraire, à un guichet ... [AUCHLIN 80] ;
- elle aboutit à une représentation fonctionnelle et hiérarchique : le constituant élémentaire dans cette analyse est la fonction correspondant à un acte de dialogue. Une conversation est alors définie comme une structure hiérarchique composée de ces fonctions.
- elle met en évidence des propriétés *programmatrices* de la conversation ; à certains stades du dialogue, il devient possible de prédire les structures de la conversation.

Pour une présentation plus complète de cette théorie, nous renvoyons le lecteur intéressé à [MOESCHLER 85]. Il nous semble néanmoins nécessaire de présenter brièvement quelques aspects fondamentaux de cette analyse qui constitue le point de départ vers une modélisation de la compréhension du dialogue.

Constituants de la conversation

Cette théorie, met en évidence trois constituants majeurs dans la conversation : l'échange, l'intervention et l'acte de langage.

Généralement, une intervention correspond à un *tour de parole* de l'un des participants. L'échange est minimal lorsqu'il est formé de deux interventions successives -exemple : une question suivie d'une réponse-. Mais il peut également possèder une structure enchâssée (voir figure 3.7).

Dans la perspective d'une analyse automatique, nous nous intéressons uniquement aux actes illocutoires [AUSTIN 70].

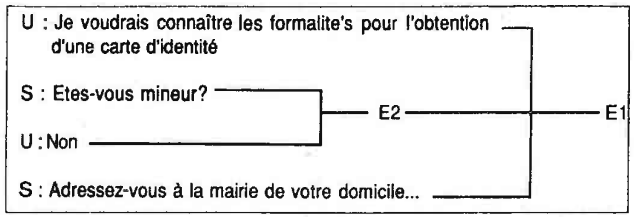


Figure 3.7: Exemple d'échanges enchâssés.

Dans une analyse pragmatique, ces actes sont caractérisés par leur fonction au sein d'une conversation telle que la demande d'information, la requête, la demande de confirmation, la production d'information, l'offre, le refus ...

L'analyse hiérarchique et fonctionnelle reste cependant un modèle statique de représentation de la conversation. Conscient de cette lacune, MOESCHLER [MOESCHLER 85] critique lui-même cette approche en écrivant : *L'analyse hiérarchique et fonctionnelle de la conversation est une analyse interprétative. Mais le type d'interprétation donné à la conversation est fonction de son achèvement... De façon analogue, l'analyse hiérarchique et fonctionnelle est atemporelle. Elle n'est nullement déterminée par le déroulement effectif de la conversation.*

Cette remarque est lourde de conséquence lorsque l'objectif est d'effectuer une analyse de façon automatique. Cependant, l'auteur émet l'hypothèse de propriétés programmatrices des énoncés permettant d'effectuer une analyse plus dynamique. On rend compte de ces propriétés à l'aide de contraintes interactionnelles et structurelles, et de propriétés d'enchaînements.

Vers une typologie opératoire

Dans leur théorie, les auteurs de l'analyse fonctionnelle proposent une typologie des fonctions associées aux actes de langage dans les conversations qu'ils observent. Celle-ci n'est pas adaptée à notre problème car, notre domaine étant limité et certains paramètres du contexte étant fixés, il n'est pas nécessaire de prendre en compte toutes les fonctions susceptibles d'apparaître dans un dialogue homme-homme. Par conséquent, nous prévoyons la suppression de certaines fonctions qui nous semblent inutilisables dans le cadre qui nous préoccupe (exemple : offre, invitation...).

D'autre part, nous devons tenir compte du fait que l'un des intervenants est une machine. En effet, les limites techniques actuelles de compréhension de la parole engendrent des énoncés ayant des fonctions spécifiques -demande de répétition, gestion du canal ...-

Nous sommes donc amenés à construire une nouvelle typologie que nous qualifions

d'opératoire, car nous nous plaçons du point de vue de la machine qui à la charge d'interpréter ou de produire un message. Ce qui nous conduit à distinguer les énoncés du locuteur de ceux de la machine.

Pour établir cette typologie, nous avons recensé les fonctions à partir de notre corpus, auxquelles nous avons rajouté les fonctions spécifiques à une machine qui comprend la parole. Dans la mesure du possible, nous allons nous efforcer de choisir des exemples parmi les dialogues que nous avons enregistrés.

Les énoncés du locuteur

- **Ouverture d'un dialogue :** Il s'agit de formules de prise de contact situées en début de dialogue. Leur forme est très stéréotypée et elles forment un préambule à la présentation de la requête.

  Exemple :
  Allo... Bonjour... Allo bonjour... Bonjour Madame...
  Allo, c'est pour un renseignement...

- **Clôture d'un dialogue :** Ce sont également des formules figées qui permettent de clore un dialogue.

  Exemple :
  Au revoir [merci]... Voilà, au revoir...

- **Production d'information (fait) :** Ce type d'acte correspond à une information de nature factuelle que le locuteur apporte dans le but de compléter les connaissances de l'opérateur. Il peut être matérialisé par :

  – une phrase complète :

  - S'il vous plaît je suis marocain...

  – une réponse elliptique à une question ouverte :

  S- Tout dépend de la nationalité que vous demandez...
  U- La nationalité française...

  – une confirmation ou une infirmation :

  S- Vous avez changé de domicile?
  U- Oui, c'est pour ça...

- **Question :** La question forme un élément important d'une requête car elle détermine précisément la demande de l'utilisateur. L'opérateur est amené à traiter deux types de questions :

– les questions ouvertes qui attendent une réponse sur des *valeurs* ;

Exemple :
-... je voudrais savoir où me renseigner.

L'objet de la question est le lieu.

- Je désire connaître la validité d'une fiche d'état civil.

La valeur demandée est ici la validité.

– les questions fermées qui appellent une confirmation ou une infirmation.

Exemple :
- Et il n'y a pas de document à fournir?

- **Thème (but) :** Dans le cadre des demandes de renseignements, le thème est représenté par le but de l'utilisateur. Il vient compléter la requête.

  Exemples :
  - Ma carte d'identité est périmée au premier Juillet, il faudrait que je la remplace...
  ou encore
  - Je voudrais savoir à qui il faudra s'adresser pour faire une demande de double-nationalité?

  Dans ces exemples, nous remarquons que le but est accompagné d'une information ou d'une question.

- **Demande de répétition :** A la suite d'une mauvaise réception d'un message ou d'une ambiguïté, le locuteur est susceptible de demander la répétition ou la reformulation d'un énoncé. Elle peut être globale ou partielle.

  Exemples :
  Comment?... Pardon?... Pouvez-vous répéter s'il vous plaît?(globale)
  ou
  - Pouvez-vous répéter le lieu s'il vous plaît? (locale)

- **Contestation :** Lorsque le locuteur réalise que son interlocuteur a commis une erreur de compréhension, il réagit en émettant une contestation. Celle-ci est soit générale, soit localisée.

  Exemples :
  Vous n'avez pas compris... Ce n'est pas ce que je vous demande...
  Mais je ne suis pas de nationalité française...

- **Enoncé de satisfaction :** Ce sont des messages qui indiquent le bon déroulement de la tâche.

  Exemple :
  Merci... Parfait... Entendu.. Très bien... Okay...

- **Enoncé de gestion du canal de communication :** Il s'agit de messages de :

  - maintien du canal de communication ;

    Exemple :
    Hmm... Oui (avec une prosodie particulière)...

    Cela se manisfeste également par des répétitions d'une information que le locuteur vient de recevoir ;

    S- Vous vous adressez à la mairie de votre commune
    U- A la mairie...

  - perturbation dans le canal de communication ;

    Exemple :
    Pardon? ... Pouvez-vous répéter?...

Les énoncés de la machine

Certaines fonctions, telles que l'**ouverture d'un dialogue** et la **clôture d'un dialogue** sont les mêmes que celles des énoncés du locuteur.

Exemples :
- *Centre de renseignements administratifs bonjour - A votre service...*

D'autres actes ont des fonctions identiques à celles du locteur, mais leur origine est cependant différente :

- **Production d'information :** La machine fournit une information en réponse à une requête de l'usager. La forme de l'information dépend du type de question posée. A une question ouverte, on répond par une ou plusieurs valeurs correspondant à l'objet de la question ;

  Exemples :
  U- Je désire connaître la validité d'une fiche d'état civil
  S- C'est de l'ordre du mois...
  ou encore
  U- Je veux savoir ce qu'il faut faire pour renouveler une carte de séjour
  S- Vous devez vous adresser à l'Hôtel de Police et vous munir de...

A une question fermée, on répond par une confirmation ou une infirmation.

Exemple :
U- S'il vous plaît, je suis Marocain...
U- Faut-il un visa pour aller au Luxembourg?
S- Oui il en faut...

- **Question :** Certains renseignements dépendent d'informations que le locuteur est susceptible de donner. Par exemple, dans notre application, les formalités de demande de carte d'identité sont différentes selon que l'intéressé est majeur ou non.

  Afin de fournir des renseignements concis et adéquats, la machine doit poser des questions à propos des informations manquantes.

  Exemple :
  U- Je voudrais savoir comment il faut faire pour changer de bureau de vote s'il vous plaît
  S- Vous avez changé de domicile?
  U- Oui c'est pour ça...

- **Demande de répétition :** Cette fonction est liée aux limites de la reconnaissance automatique de la parole. On peut supposer que la machine fera des demandes de répétition plus fréquemment que le locuteur humain. La forme des messages est comparable à celle des énoncés du locuteur.

- **Enoncé de gestion du canal de communication :** La machine produit des énoncés de gestion du canal de communication dans plusieurs cas :

  - L'intensité (énergie) du signal est trop faible. La machine réagit en émettant des consignes telles que :
    - *Parlez plus fort*
    - *Rapprochez-vous du micro*
  - Une saturation du signal est détectée. La consigne est alors de la forme :
    - *Parlez moins fort*
    - *Eloignez-vous du micro*
  - Lorsqu'une reconnaissance peu satisfaisante subsiste, on envoie un message tel que :
    - *Veuillez mieux articuler s'il vous plaît*
  - Il arrive que le traitement d'une requête prenne un temps assez conséquent. Le message de maintien du canal de communication correspondant est par exemple :

- Veuillez patienter s'il vous plaît...

D'autres fonctions sont spécifiques à la machine en raison de son rôle de spécialiste dans un domaine.

- **Relance :** Après la satisfaction d'une requête, la machine relance l'utilisateur en lui offrant la possibilité d'énoncer une nouvelle requête.

  Exemple :
  - Désirez-vous un autre renseignement?

- **Enoncé de gestion de la tâche :** Une requête est composée de plusieurs éléments (un but, une question et éventuellement des informations). La nécessité de présence de l'un de ces éléments est fonction de la nature de la demande. Lorsque l'un d'entre eux est manquant, la machine est amenée à inviter le locuteur à le formuler.

  Exemple :
  U- Je voudrais savoir quelles sont les pièces d'identité obligatoires
  S- Dans quel but s'il vous plaît?

Cette typologie permet de dissocier les énoncés du locuteur, destinés à être interprétés, et ceux de la machine, réservés à la génération.

Nous établissons également une nouvelle classification selon d'autres critères liés au contenu des énoncés, qui sont les énoncés portant sur la tâche, sur le dialogue en cours et sur le canal de communication. Cette répartition complémentaire renforce le caractère opératoire de la typologie, car elle permet d'isoler les traitements à effectuer en fonction de la classe identifiée.

Nous proposons une récapitulation de cette typologie sous la forme d'un tableau à la figure 3.8.

	ENONCÉ LOCUTEUR	ENONCÉ MACHINE
TÂCHE	Fait, Question, But	Information, Question
DIALOGUE	Ouverture, Clôture, Contestation, Demande de répétition Satifaction	Ouverture, Clôture, Relance Demande de répétition Gestion de la tâche
CANAL	Gestion du canal de communication	Gestion du canal de communication

Figure 3.8: Typologie des énoncés dans un dialogue.

La composante DIAL, va s'appuyer sur cette typologie pour représenter les différentes informations contenues dans chaque intervention du dialogue. Nous reviendrons plus en détail sur cette représentation dans le chapitre suivant.

5.3 Rôle de la composante

La composante DIAL assure deux rôles au sein de notre système.

Le premier rôle est externe car il concerne le locuteur. Il consiste à fournir une réaction à chaque intervention de l'usager. Dans le domaine de demandes de renseignements, cette réaction se traduit par l'émission d'un message.

Le second rôle est interne dans la mesure où la finalité est d'émettre, vers les autres modules du systèmes, des hypothèses sur l'enchaînement possible des énoncés.

Extérieurement, on peut considérer le module DIAL comme une *boîte noire* dont les entrées et les sorties sont fournies par le schéma de la figure 3.9.

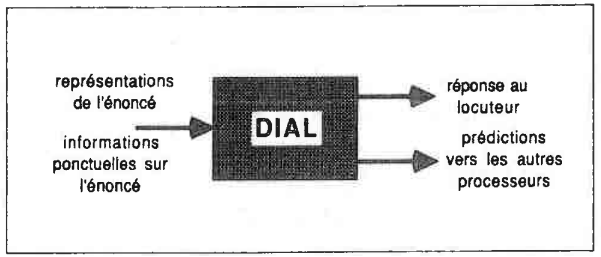


Figure 3.9: Structure externe de la composante DIAL.

Pour parvenir à ces objectifs, le processeur DIAL remplit un ensemble de fonctions que nous allons détailler dans la suite de cette section.

- **L'interprétation contextuelle :** cette fonction constitue une étape du traitement d'un énoncé donné sous sa forme syntaxico-sémantique. Elle mène :
  - à l'identification des actes de langages contenus dans un énoncé et coïncidant avec la typologie que nous avons établie ;
  - à l'unification des entités apparaissant dans un énoncé et de leurs références dans l'historique du dialogue.

- **La gestion du dialogue :** elle consiste à intégrer la forme interprétative de l'énoncé dans une représentation discursive du dialogue en cours, puis de l'évaluer dans le but de réagir *intelligemment* à cet énoncé. Cette fonction peut faire appel à la suivante lorsqu'une activité de raisonnement sur la tâche est nécéssaire.

- **Le raisonnement :** il s'applique aux actes portant sur la tâche. Son objectif est de trouver une solution à la requête du sujet. Lorsque celle-ci peut être satisfaite, cette fonction met en évidence la (les) solution(s) possible(s). Mais elle permet également de détecter :

- une requête incomplète : dans ce cas, elle détermine la poursuite du dialogue afin d'amener l'usager à compléter sa demande ;
- une requête dépassant les compétences du système : elle propose alors un message expliquant les limites du système ;
- et, enfin, des informations incohérentes entre elles suite à une contradiction.

- **La prédiction :** Grâce aux propriétés programmatrices d'un échange, il est possible de prévoir les enchaînements les plus probables des énoncés. Cela conduit à émettre des hypothèses sur la structure syntaxico-sémantique et lexicale (mots-clefs) de l'énoncé attendu du locuteur. Ces hypothèses permettent aux différents processeurs d'amorcer leurs traitements respectifs et de se focaliser sur un sous-ensemble de connaissances.

- **La génération :** il s'agit de construire une structure de surface en langage naturel, à partir d'une représentation interne d'un énoncé.

5.4 Structure interne de la composante

La mise en évidence des fonctions du processeur DIAL nous a permis de découper cette composante en trois modules prenant chacune en charge une ou plusieurs des fonctions que nous venons d'énumérer. Pour mettre en oeuvre ces modules, nous avons conservé l'approche que nous avions choisie pour l'ensemble du système, c'est à dire un système modulaire et à bases de connaissances.

L'organisation interne de DIAL est donnée à la figure 3.10.

Cette organisation fait apparaître les principaux modules de la composante dialogue.

L'interpréteur, qui remplit la fonction d'interprétation, traite toutes les informations provenant des autres modules et fournit une *interprétation contextuelle* du message prononcé par le locuteur.

Le raisonneur regroupe les traitements associés à la tâche à accomplir.

Le gestionnaire de dialogue assure de la conduite du dialogue. Il inclut la fonction de gestion de dialogue, mais aussi celle de prédiction et de génération.

Ces trois modules partagent une structure centrale, **l'historique** du dialogue.

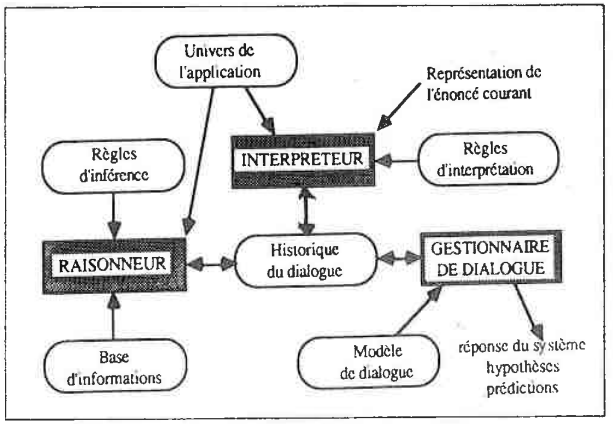


Figure 3.10: Structure interne de la composante DIAL.

Chapitre 4

LES CONNAISSANCES ET LES TRAITEMENTS

1 STRUCTURES DE BASE

Avant d'aborder les modules internes de DIAL et la représentation de leurs connaissances, il nous paraît indispensable de présenter en premier lieu les structures de bases qui vont servir à décrire ces connaissances.

Dans cette optique, nous allons commencer par les notations et le formalisme que nous utilisons.

1.1 Le formalisme de la présentation

Dans notre présentation, nous utilisons la notion de *type* pour caractériser les différentes structures que nous manipulons. Par conséquent, ce terme permet de décrire une famille d'objets possédant des propriétés communes.

Les types de base représentent les objets élémentaires que nous pouvons manipuler. Il permettent de construire des types de plus en plus structurés tels que les *ensembles*, les *suites*, les *produits scalaires* ....

La notation $x : X$ signifie que x est de type X.

Pour décrire les accès, les propriétés et les traitements, nous utilisons une notation fonctionnelle de la forme :

$$f(a_1 : t_1, ..., a_n : t_n) \rightarrow res$$

où f est l'identificateur de la fonction, a_i ($i \in [1, n]$) représentent les paramètres de type t_i de la fonction et enfin, res détermine le type du résultat.

Ainsi, pour le type *produit scalaire* $t = (val : t_1 \times champ_2 : t_2 \times ... \times champ_n : t_n)$, les

différentes projections sont données par les fonctions respectives :

$$val(x:t) \rightarrow t_1,$$
$$champ_2(x:t) \rightarrow t_2,$$
$$\vdots$$
$$champ_n(x:t) \rightarrow t_n.$$

Par abus de langage et pour simplifier les expressions, nous nous autorisons à écrire indifféremment x ou $val(x)$ lorsque le contexte le permet.

1.2 Les concepts

Définition : Un concept est une unité minimale qui sert à décrire un objet (*carte d'identité*), un état (*couleur*), une relation (*nationalité*) ou une action (*se présenter*) dans l'univers d'une application. Cette classe d'objets définit le type de base: *concept*.

L'ensemble des concepts, noté C : *ensemble(concept)*, susceptibles d'intervenir dans un dialogue portant sur un univers d'application donné, est appelé ensemble des concepts modèles.

Lorsque le domaine est clos, nous supposons que l'ensemble C est fini et qu'il est possible d'en énumérer tous les éléments. Cette condition est, en l'occurrence, vérifiée dans les applications visées. En conséquence, l'ensemble des concepts modèles devient une connaissance statique liée à l'univers de l'application.

Il existe, dans cet ensemble, une relation, notée *isa*, représentant les relations de taxonomie entre différents concepts.

Exemples :
`isa(mairie,administration)`
`isa(carte d'identité,carte)`
`isa(passeport,carte)`
`isa(carte,document)`
`isa(document,inanimé)`

Cette relation possède les propriétés suivantes:

1. Réfléxivité : $\forall x \in C, isa(x,x)$
2. Transitivité : $\forall x,y,z \in C, isa(x,y) \wedge isa(y,z) \Rightarrow isa(x,z)$
3. Antisymétrie : $\forall x,y \in C, isa(x,y) \wedge isa(y,x) \Rightarrow x = y$

Par ailleurs, nous définissons une fonction, notée *arg*, qui, à chaque concept associe un n-uplet représentant la liste de ses arguments.

Exemples :
`arg(nationalité) = (personne,pays)`
`arg(se présenter) = (animé,lieu)`

Cette fonction traduit les contraintes sémantiques relatives à l'occurrence d'un concept dans un univers donné.

Un concept *hérite* des arguments de sa classe se traduisant formellement par :

soient $c_1, c_2 \in C$ tels que $arg(c_1) = (a_1,...,a_n)$ et $arg(c_2) = (b_1,...,b_m)$;

$$isa(c_1,c_2) \Rightarrow \forall i\ 1 \leq i \leq m a_i = b_i$$

Cette notion d'héritage est semblable à celle que l'on utilise maintenant classiquement dans les langages orientés objets [COLNET 86].

Exemple :
`isa(nationalité,état civil)` et
`arg(état civil) = (personne)`
`arg(nationalité) = (personne,pays)`

Parallèlement aux concepts modèles, nous utilisons un autre type : les concepts objets. auxquels est associé le type : $noeud = (val : concept \times signe : \{+,-\}, score : [0,100])$.

Ils représentent essentiellement les concepts qui interviennent dans les énoncés et dans les bases de connaissances. Ce sont des instances de concepts modèles symbolisées par la fonction d'accès aux références $val(n : noeud) \rightarrow concept$.

Afin de gagner en lisibilité dans certaines expressions que nous aurons à écrire dans la suite de ce manuscrit, nous étendons la relation *isa* et la fonction *arg* aux concepts modèles en établissant les correspondances suivantes :

- soient n_1, n_2 : $noeud$, $isa(n_1,n_2) \Leftrightarrow isa(val(n_1), val(n_2))$
- soit n : $noeud$, $arg(n) = arg(val(n))$

Nous complétons la construction du type *noeud* par la définition de deux autres fonctions d'accès :

- $signe(n : noeud) \rightarrow \{+,-\}$: la négation est représentée à l'aide de la fonction *signe* qui, à chaque concept attribue un signe symbolisé par $-$, si le concept est nié et par $+$, sinon.
- $score(n : noeud) \rightarrow [0,100]$: un concept est une représentation interprétative construite à partir d'une reconnaissance effectuée sur le signal. La fonction *score* est un attribut supplémentaire qui fournit le taux de certitude de chaque concept, compte tenu des approximations faites sur le signal et sur le contexte.

En raison de la nature même de ses éléments, l'ensemble des concepts objets varie au cours du temps. Il contient des informations dynamiques liées au dialogue et constitue une partie de l'historique.

Nous notons N : *ensemble(noeud)*, l'ensemble des *noeuds* à un instant donné et nous définissons dans cet ensemble les relations suivantes :

- *equ* : une relation d'équivalence dont les classes sont formées de *noeuds* faisant référence à un concept modèle commun.

  Dans une conversation, il existe fréquemment des références à une même entité sous diverses formulations (voir figure 4.1). La relation *equ* permet de les organiser en classe d'équivalence afin d' appliquer tous les traitements concernant les éléments d'une même classe, sur leur représentant principal.

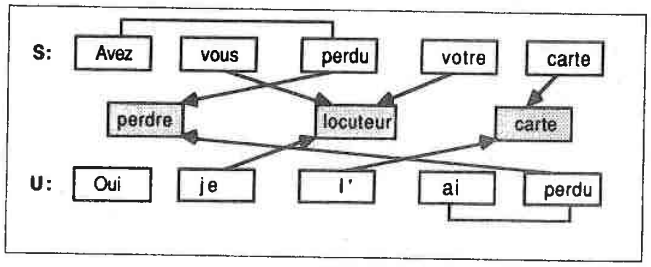


Figure 4.1: Illustration de la relation *equ*.

  Une condition nécessaire pour qu'un concept x appartienne à une classe Cl est : $\forall y \in Cl, isa(x,y) \vee isa(y,x)$.

  Dans la suite de notre présentation, nous ne ferons pas la distinction entre un concept objet et sa classe d'équivalence à laquelle on attribue les propriétés de son représentant principal que nous choisissons selon le critère suivant :

  soit x le représentant de la classe Cl, $\forall y \in Cl, isa(x,y)$.

- *diff* : une relation *d'exclusion mutuelle* qui permet de distinguer deux concepts susceptibles d'appartenir à une même classe. En effet, lorsque la condition nécessaire est vérifiée pour deux concepts x et y , 3 cas se présentent:

  - x et y appartiennent à la même classe,
  - x et y appartiennent peut-être à la même classe,
  - x et y n'appartiennent certainement pas à la même classe.

  La relation *diff* permet de décrire ce dernier cas (voir exemple figure 4.2).

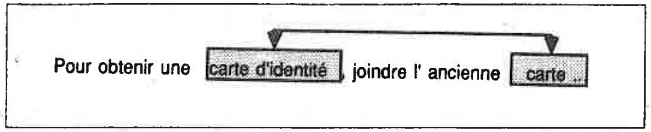


Figure 4.2: Illustration de la relation *diff*.

Les contraintes imposées a *diff* sont les suivantes:

1. $\forall x \in N, \neg diff(x,x)$ (non refléxive),
2. $\forall x,y \in N, diff(x,y) \Rightarrow diff(y,x)$ (antisymétrique),
3. $\forall x,y \in N, diff(x,y) \Rightarrow \forall z / equ(y,z), diff(x,z)$

1.3 Les formules

Une formule élémentaire est définie par un type structuré, noté *fe*. Elle est formée d'une arborescence dont les noeuds sont des objets de type *noeud* qui vérifient la condition suivante :

soit f : *fe* $\forall x \in f$ (cela signifie que x est un noeud de l'arborescence f), on pose $fils(x) = (x_1, ..., x_n)$ et $arg(x) = (a_1, ..., a_n)$, il faut que : $\forall i, isa(x_i, a_i)$.

Exemples : (représentation prédicative)

```
aller(locuteur,mairie)
nationalité(locuteur,France)
ouvrir(robot,porte(rouge, en bois))
```

Exemples : (représentation graphique, voir figure 4.3)

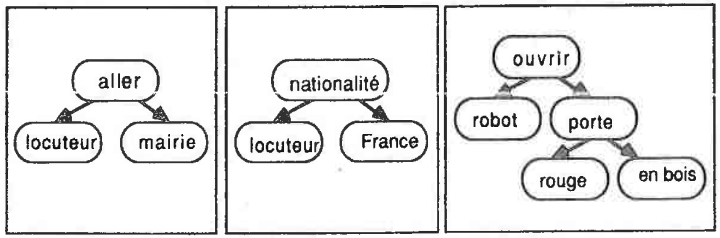


Figure 4.3: Représentation graphique de formules élémentaires.

Une formule conjonctive est également une arborescence (type : *fc*) dont la racine est un concept particulier symbolisé par le caractère & et représentant la conjonction de formules élémentaires.

Autrement dit, une formule conjonctive est de la forme $f = \&(f_1, ..., f_n)$ où les f_i sont des formules élémentaires que nous appelons les *termes* de f notés $terme(f) = \{f_1 ... f_n\}$.

Une formule disjonctive (ou canonique) possède un niveau d'arborescence de plus qu'une formule conjonctive. La disjonction se situe au niveau de la racine (concept |) et les fils sont des objets de type *fc*. Le type correspondant est noté *fd*.

Nous considérons que les formules suivantes sont équivalentes :

$$\begin{cases} \&(f) \Leftrightarrow f, & f : fe \\ |(f) \Leftrightarrow f & f : fc \end{cases}$$

Une formule quelconque est une arborescence dont les noeuds appartiennent à $N \cup \{\&, |\}$ (voir figure 4.4).

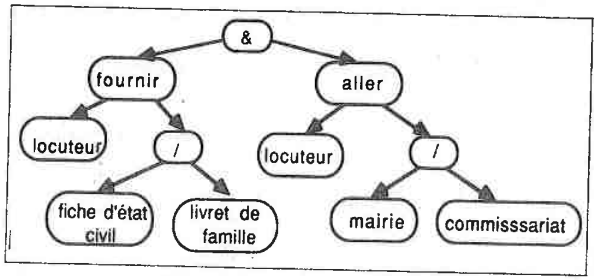


Figure 4.4: Exemple de formule quelconque.

Afin d'éviter les ambiguïtés et d'accélérer les fonctions de gestion des formules nous allons nous ramener à une forme canonique en postulant les équivalences suivantes :

$$\begin{cases} \forall x_i, y_i \ \&(x_1, ..., \&(y1, ...y_m), ..., x_n) \Leftrightarrow \&(x_1, ..., y_1, ..., y_m, ..., x_n) \\ \forall x_i, y_i \ |(x_1, ..., \&(y1, ...y_m), ..., x_n) \Leftrightarrow |(x_1, ..., y_1, ..., y_m, ..., x_n) \\ \forall x_i, y_i, \forall p \ p(x_1, ..., \&(y1, ...y_m), ..., x_n) \Leftrightarrow \&(p(x_1, ..., y_1, ..., x_n), ..., p(x_1, ..., y_m, ..., x_n)) \\ \forall x_i, y_i, \forall p \ p(x_1, ..., |(y1, ...y_m), ..., x_n) \Leftrightarrow |(p(x_1, ..., y_1, ..., x_n), ..., p(x_1, ..., y_m, ..., x_n)) \end{cases}$$

Ce système d'équivalence permet de réécrire une formule quelconque en formule disjonctive qui nous sert de forme canonique. Ainsi, la formule de l'exemple précédent se réécrit en une formule canonique représentée à la figure 4.5.

Remarque : effectuer la transformation d'une formule en vue de la rendre canonique revient d'une certaine façon à l'interpréter. L'interprétation des symbole & et | que nous faisons ne s'applique pas toujours aux conjonctions *et* et respectivement *ou* du langage naturel.

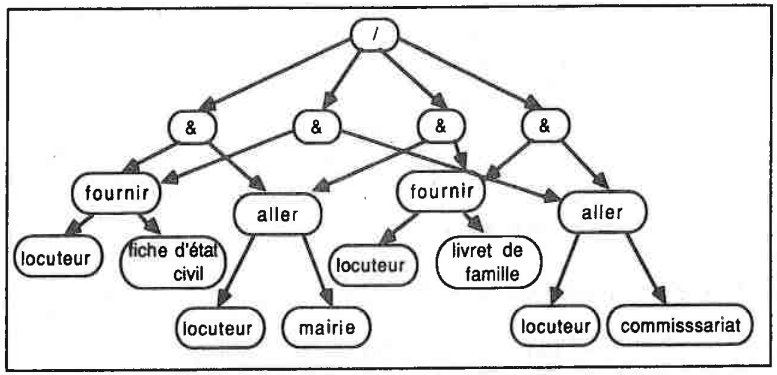


Figure 4.5: Exemple de formule canonique réécrite.

2 L'INTERPRÉTEUR

2.1 L'interprétation contextuelle

La fonction majeure de l'interpréteur est l'interprétation contextuelle d'un énoncé. Pour mieux définir ce que nous entendons par ce terme, il nous faut revenir brièvement sur les relations du module DIAL avec les autres composantes du système.

DIAL reçoit des données provenant des processeurs APHON, LEX et SYN-SEM. Ces informations sont de natures très diverses et peuvent être complémentaires, redondantes ou contradictoires. Pour les gérer efficacement et les faire coopérer, la première tâche de l'interpréteur est de les structurer de façon plus homogène.

D'autre part, ces informations restent très descriptives en ce sens qu'elles indiquent la présence de certains éléments sur le signal et leurs éventuelles relations.

Or, le module DIAL requiert une représentation fonctionnelle sur laquelle il peut s'appuyer pour *raisonner* et pour contrôler le dialogue. Dans cette optique, la structure construite par l'interpréteur a une valeur d'interprétation.

Par ailleurs, les représentations fournies en entrée ne tiennent compte ni de la tâche, ni du dialogue. Examinons l'exemple suivant qui illustre bien ce phénomène.

S: `Avez-vous rempli le formulaire?`

U: `Oui, je l'ai fait.`

Le pronom personnel *l'* se rapporte à *remplir le formulaire* qui existe déjà dans le dialogue. Pour résoudre cette référence, il est indispensable de revenir sur l'historique

géré par DIAL car les autres modules ne possèdent pas de connaissances suffisantes pour le faire.

En restant toujours dans cet exemple, on constate que la réplique *U* est une réponse à la question précédente. Mise dans le contexte de la tâche, elle constitue un apport d'information. On ne saurait détecter cette fonction sans faire appel à l'historique et à l'état courant de la tâche.

Par conséquent, le traitement consiste également à contextualiser l'énoncé; cela justifie la qualification d'interprétation contextuelle.

2.2 Les informations en entrée

Les informations entrée de l'interpréteur sont des représentations globales ou locales de l'énoncé produit par le locuteur. Il faut également envisager le cas des informations partielles.

Provenant du module SYN-SEM, que nous considérons comme un processeur privilégié vis-à-vis de DIAL en raison du flux important d'informations échangées entre ces deux modules (voir Chapitre 3), l'interpréteur reçoit deux types de représentations que nous allons maintenant présenter.

La représentation syntaxique

Elle décrit la structure de surface de l'énoncé. Elle est le résultat d'une analyse syntaxique à l'aide de RNP [PIERREL 81] [MOUSEL 86a]. et est formée d'un arbre syntaxique dont les noeuds sont des non-terminaux de la grammaire et dont les feuilles constituent des entrées dans le lexique de l'application.

La figure 4.6 montre un exemple d'analyse syntaxique sur une phrase se rapportant à notre application.

La représentation sémantique

Contrairement à la précédente, cette représentation ne rend pas compte de la position des mots entre eux dans un énoncé. Mais, elle exprime les relations de natures sémantiques entre les différents constituants [DEVILLE 86].

Nous avons déjà abordé, au Chapitre 3, le formalisme que nous avons adopté pour cette représentation. Pour restituer les grandes lignes de cette approche, nous fournissons à la figure 4.7 la description d'une structure sémantique à l'aide d'une grammaire BNF [1].

Ainsi, l'exemple précédent sera représenté comme il est indiqué à la figure 4.8.

[1] BNF : Bacchus-Naur Form, forme normalisée d'une grammaire hors-contexte.

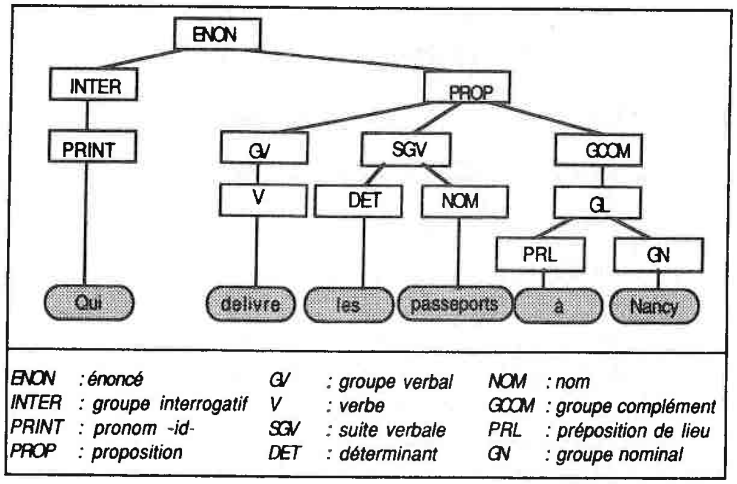


Figure 4.6: Arbre syntaxique de l'énoncé : *Qui délivre les passeports à Nancy?*

ENONCE	→	*MODALITE|PROPOSITION*
MODALITE	→	*TEMPS|VOIX|MODE|FORME|MODAL|TYPE*
TEMPS	→	*présent|passé|futur*
VOIX	→	*active|passive*
MODE	→	*infinitif|indicatif*
TYPE	→	*déclaratif|inter – direct|inter – indirect*
FORME	→	*affirmatif|négatif*
MODAL	→	*obligation|permission|possibilité*
PROPOSITION	→	*PRIMITIVE STRUCTURE – CAS*
PRIMITIVE	→	*verbe|nom|adjectif*
STRUCTURE – CAS	→	*(CAS|ENONCE)*^{n}
CAS	→	*agent|objet|bénéficiaire...*

Figure 4.7: Description des structures sémantiques [DEVILLE ET PAULUSSEN 86].

Les représentations lexicales

Il s'agit d'un ensemble de lexèmes reconnus dans l'énoncé. Ces informations sont souvent redondantes avec celles que fournissent les analyseurs. Par conséquent, l'interpréteur n'en tient compte que lorsque la représentation syntaxico-sémantique est mise à défaut. Cela peut se produire dans deux cas :

- La stratégie appliquée conduit à *inhibition* des analyseurs ; une des raisons que l'on peut invoquer est la prévision d'une réponse très courte (un à deux mots) ne

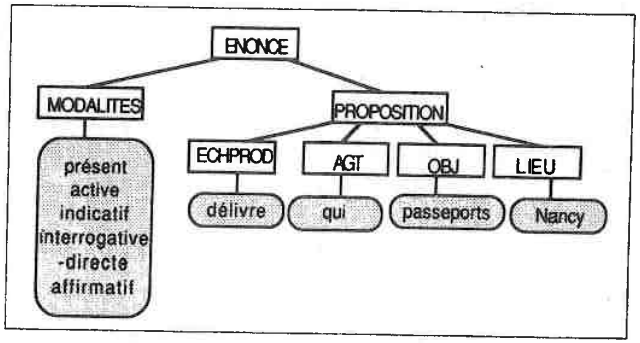


Figure 4.8: Représentation sémantique de l'énoncé : *Qui délivre les passeports à Nancy?*

nécessitant pas une analyse très fine.

- L'analyse est effectivement défaillante à la suite d'une anomalie dans la reconnaissance.

Les informations d'origine acoustico-phonétique

Ce sont des informations locales sur l'énoncé. Dans l'état actuel du système nous prévoyons seulement deux types d'informations : les maxima d'énergie, qui servent à contrôler le canal de communication, et la longueur de l'énoncé (par exemple, en nombre de phonèmes).

Ces données ne sont pas véritablement interprétées. On se contente d'harmoniser leur représentation avec le reste des informations et de les transmettre aux autres composantes par l'intermédiaire de l'historique. L'interpréteur remplit alors uniquement un rôle d'interface.

2.3 Les informations en sortie

A partir des données que nous venons d'énumérer, l'interpréteur construit une représentation interprétative fonctionnelle de l'énoncé.

Chaque énoncé est représenté sous forme d'une suite de fonctions dont certaines possèdent des attributs. L'ensemble des fonctions à identifier par l'interpréteur est donné par la typologie établie au Chapitre 6. La nature des attributs dépend fortement de la fonction à laquelle ils se rattachent. La figure 4.9 fournit la table de correspondance entre fonctions et attributs.

FONCTIONS	ATTRIBUTS
Fait	Formule représentant le fait
But	Formule représentant le but
Question ouverte	Formule représentant la question
Question fermée	Formule représentant la question et noeud indiquant l'objet de la question
Maintien du canal	Pas d'attribut
Perturbation du canal	Saturation ou énergie trop faible
Ouverture	Pas d'attribut
Fermeture	Pas d'attribut
Contestation	Objet de la contestation (s'il y a lieu)
Satisfaction	pas d'attribut
Répétition	Objet de la demande de répetition (s'il y a lieu)

Figure 4.9: Les fonctions et leurs attributs.

2.4 Principes de fonctionnement et connaissances

L'interpréteur est construit à la manière d'un système à base de règles de production [WINSTON 77]. La réalisation de la version que nous allons présenter a fait l'objet d'un stage au CRIN en collaboration avec l'Université de Namur [LORANT 86].

La figure 4.10 permet d'isoler les différents éléments qui interviennent dans son fonctionnement.

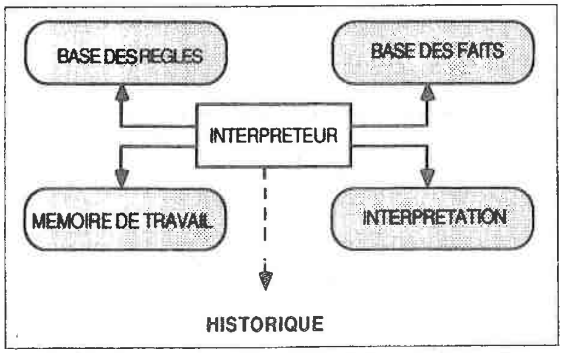


Figure 4.10: Structure de l'INTERPRÉTEUR.

Nous allons passer en revue chacune de ces composantes en introduisant progressive-

ment les structures que nous utilisons pour exprimer les connaissances.

La mémoire de travail (MT)

Initiallement, elle renferme les informations validées sur l'entrée. Cette opération est assurée par un pré-processeur qui produit une description prédicative des différentes données.

Reprenons l'exemple de la phrase :

```
Qui délivre les passeports à Nancy?
```

Le pré-processeur, après transformation des représentations syntaxico-sémantiques de cette phrase, fournit les prédicats suivants :

```
(1) enonce(e)
(2) proposition(e,délivre)
(3) prédicat(e,délivrer)
(4) agent(délivrer,qui)
(5) objet(délivrer,passeport)
(6) lieu(délivrer,Nancy)
(7) présent(délivrer)
(8) voix-active(e)
(9) indicatif(e)
(10) affirmatif(e)
(11) interr-directe(e)
```

Comme nous pouvons le constater, un prédicat traduit une information d'origine syntaxique (7–11) ou sémantique (1–6).

Remarque : Dans notre étude, nous nous bornons à exploiter les structures syntaxico-sémantiques, mais cette méthode peut être étendue aux informations provenant des autres processeurs.

Chaque prédicat est construit selon un schéma qui caractérise les arguments variables.

Exemple :

```
predicat(#v).
```

Un prédicat est construit par instanciation des variables d'un schéma.

Exemples :

```
proposition(marcher),
proposition(aller)...
```

La MT est une structure dynamique. Son évolution est tributaire de l'activation des règles d'interprétation. Nous reviendrons ultérieurement sur ce point.

L'interprétation (I)

Il s'agit d'un espace dans lequel on construit progressivement l'interprétation. En cours de traitement, cette structure renferme une représentation partielle du résultat.

Rappelons que le résultat est composé d'un ensemble de fonctions identifiées sur l'énoncé et, pour chacune d'entre elles, des attributs appropriés.

La construction de l'interprétation s'effectue également par l'activation de certaines règles.

La base de faits

Elle regroupe des connaissances statiques qui décrivent les propriétés des éléments qui apparaissent dans l'entrée. On les représente à l'aide d'un ensemble de prédicats.

De façon pratique, ces connaissances peuvent être de différentes natures :

- **des connaissances syntaxiques :** elles portent sur les unités intervenant dans la description d'un énoncé ;

  Exemples :

  ```
  (1) première-personne(je)
  (2) singulier(je)
  ```

- **des connaissances sémantiques :** elles concernent principalement les prédicats de la grammaire de cas que nous utilisons pour représenter l'énoncé ;

  Exemples :

  ```
  (1) ech-obt(obtenir)
  (2) mvt1(aller)
  ```

- **des connaissances pragmatiques :** elles permettent de relier des mots ou des syntagmes aux concepts auxquels ils se réfèrent (voir exemple suivant (1) (2)). On peut également y faire figurer les relations d'ordre pragmatique ((3)(4)) ;

  Exemples :

  (1) `concept-mairie(mairie)`,
  (2) `concept-mairie(hôtel de ville)`
  (3) `administration(mairie)`
  (4) `bâtiment(mairie)`

La plupart de ces informations sont aussi utilisées dans les autres composantes de DIAL. Elle apparaissent donc plusieurs fois dans le module. Mais leur existence provient d'une nécessité pratique de développement incrémental et indépendant. Cette solution comporte cependant un inconvénient majeur ; on s'expose alors au risque de voir les informations devenir incohérentes entre elles. Pour éviter cette situation, un contrôle statique de cohérence sur les connaissances communes devient indispensable.

La base de règles

Elle est formée d'un ensemble de règles d'interprétations. Elle fait partie des connaissances statiques dépendantes de l'application.

Chaque règle exprime un fragment de connaissances relatif à l'activité d'interprétation. Les règles doivent être indépendantes entre elles; ce qui permet de les donner de façon déclarative.

Le formalisme de description correspond à celui des règles de production, composées d'une conjonction de conditions et d'une suite d'actions. Une règle possède donc la forme générale suivante :

SI: c_1 & c_2 & ... & $c_n \Rightarrow a_1$ & a_2 & ... & a_m

où les c_i sont des schémas et les a_i, des actions pouvant être éxécutées lorsque les conditions correspondantes sont vérifiées.

Dans la version actuelle de l'interpréteur, on distingue six types d'actions :

- **ajouter(x) :** ajoute un prédicat dans MT.
- **supprimer(x) :** supprime un prédicat dans MT.
- **hist-refer() :** ajoute, dans MT, des prédicats de la forme `refer(x)` qui indiquent les concepts existants dans l'historique et susceptibles d'être référencés par des éléments de l'énoncé courant.

La stratégie que nous adoptons consiste à rechercher les références potentielles uniquement dans l'échange précédent.

- **ajout-int(x) :** rajoute une interprétation partielle, x, dans I.
- **en-machine(e) :** crée une interprétation ayant comme fonction *la production d'informatic* construite à partir de la dernière question posée par la machine. Cette action revient à supposer que e est un concept qui répond à la question de manière elliptique.

  Exemple : admettons que la dernière question posée par la machine soit : `Quelle est votre nationalité?` que nous représentons de la façon suivante : `nationalité(locuteur,?x)`.
  `en-machine(France)` a pour effet de ranger dans I l'interprétation qui suit : `nationalité(locuteur,France)`

- **interpréter(x) :** procède à un appel récursif de l'interpréteur avec la sous-structure x. La MT est alors empilée.

Par ailleurs, on autorise trois catégories de conditions :

- **Les conditions sur l'énoncé :** elles peuvent être vérifiées par des prédicats situés dans MT ;
- **Les conditions sur le dialogue :** elles sont testées sur l'historique du dialogue ;
- **Les conditions sur l'univers de l'application :** elles peuvent être satisfaites par des prédicats de la base de faits.

Pour finir, donnons quelques exemples de règles :

```
RÈGLE : 1
si :  proposition(#x)
& objet(#x,#o)
& requête(#x)
& obligation(#x)
=> ajouter(objet-requête(#o))
```

Cette règle signifie que si une proposition qui exprime une requête, dans un énoncé ayant une modalité d'obligation, possède un cas objet, alors ce cas peut représenter l'objet d'une requête.

```
RÈGLE : 2
si :  proposition(#x)
& proposition(#y)
& objet(#x,#y)
& requête(#x)
& objet-requête(#y)
=> ajout-int("but:")
& interpréter(#y)
```

Cette règle signifie que si une proposition exprimant une requête admet comme cas objet une autre proposition et, que celle-ci décrit en outre l'objet de la requête, alors le *but* est l'interprétation de la proposition objet.

Activation des règles et contrôle

Une condition $c = p(\#x_1, ..., \#x_n)$ est vérifiée si $\exists\ v_1, ..., v_n$ tel que $p(v_1, ..., v_n)$ est un prédicat appartenant à MT, à l'historique ou à la base de faits (pour i donné, v_i instancie $\#x_i$).

Une règle devient activable lorsque toutes ses conditions sont vérifiées.

Plusieurs règles peuvent être simultanément activables; se pose alors un problème de priorité qu'il faut résoudre à l'aide d'une stratégie adéquate.

Celle de l'interpréteur fait intervenir des informations sur deux axes ; l'un est lié à la *nature* des règles et l'autre, à leur *forme*.

Pour mettre le premier critère en oeuvre, nous commençons par construire deux classes de règles :

- les règles d'identification, dont les actions mettent la MT à jour,
- et les règles d'interprétation, qui font progresser l'interprétation.

Le processus se déroule en deux étapes conformément à la séquence suivante :

1. sélectionner puis activer les règles d'identification ;
2. sélectionner puis activer les règles d'interprétation.

Le deuxième critère intervient au niveau de chacune des étapes avec l'application des règles de priorités suivantes :

1. privilégier une règle ayant un plus grand nombre de conditions vérifiées ;

2. préférer une règle rendue activable plus récemment.

L'interpréteur se charge du contrôle des activations successives. Le traitement est arrêté pour deux raisons :

- il ne reste plus de règles activables,
- tous les éléments de l'énoncé ont été interprétés.

3 LE RAISONNEUR

3.1 Les fonctions

L'objectif final de ce module consiste à proposer des solutions susceptibles de satisfaire la requête de l'utilisateur. Pour y parvenir, plusieurs fonctions sont mises en oeuvre.

La mise à jour de l'état de la tâche

A partir des informations fournies par le locuteur et de la requête qu'il a émise, le raisonneur accède aux connaissances qui décrivent la tâche (en l'occurrence les renseignements). Lors de cette opération plusieurs situations peuvent se présenter :

- Il n'existe pas de solution : ce cas se produit quand le locuteur ne remplit pas les conditions nécessaires à la réalisation de la tâche ou quand le système ne possède pas les compétences suffisantes pour satisfaire la requête courante ;
- Il existe plusieurs solutions : une seule requête peut parfois admettre plusieurs solutions. Il s'agit de déterminer l'ensemble des choix ;
- Les informations sont insuffisantes : il arrive, en effet, que le locuteur n'ait pas apporté toutes les informations nécessaires à la réalisation de la tâche ;
- La requête a la possibilité d'être satisfaite : toutes les conditions sont réunies et il n'y pas d'ambiguïté dans la réponse. C'est le cas qui pose le moins de problèmes.

Cependant il ne suffit pas seulement d'établir un diagnostic ; on doit, en outre, y adjoindre les directives nécessaires afin que le gestionnaire du dialogue puisse prendre une décision sur la suite du dialogue à mener.

Toutes ces informations sont concentrées dans une structure que nous appelons l'état de la tâche. La réponse du système découle de l'examen de cette structure.

La gestion des informations implicites

Pour satisfaire une requête un certain nombre de conditions doit être préalablement remplie. Les conditions sont satisfaites soient par des informations énoncées par le locuteur, soient par des informations implicites (les *non-dits*).

Dans notre approche, nous traitons deux formes particulières d'implicites :

- **Les valeurs par défauts :** ce sont des informations que l'on suppose valides tant qu'elles ne sont pas explicitement mises à défaut,
- **Les informations déduites :** ce sont des informations obtenues à la d'un raisonnement déductif à partir des informations déjà connues.

Le contrôle dynamique de la cohérence

L'ensemble des informations apportées par le locuteur, que ce soit de manière explicite ou implicite, peut devenir incohérent en cours de dialogue. Une telle situation rend inconsistant l'ensemble des informations sur lequel s'appuient la gestion des implicites et la mise à jour de l'état de la tâche. Il devient alors indispensable de mettre en oeuvre une fonction particulière qui détecte les incohérences, puis qui les traite.

Ce point sera développé ultérieurement, mais nous pouvons d'ores et déjà affirmer qu'il existe des éléments permettant d'éliminer quelques cas d'incohérence. Cela signifie que le raisonneur est doté d'un mécanisme spécifique permettant de juger si l'incohérence peut être immédiatement traitée ou s'il faut la signaler au gestionnaire du dialogue. Ce dernier cas se traduit par un état spécifique de la tâche.

On peut, néanmoins se poser la question de l'utilité de cette fonction car nous avons supposé que l'utilisateur est coopératif. Par conséquent, il ne devrait pas fournir des informations incohérentes. Cependant, en raison du caractère incertain de la reconnaissance de la parole (<100%), il est tout à fait probable de voir une incohérence apparaître dans l'ensemble des informations à cause, par exemple, d'une mauvaise reconnaissance d'un marqueur de négation.

3.2 Les connaissances dynamiques

Plutôt que d'aborder le raisonneur en définissant ses entrées et ses sorties, il nous semble plus approprié de parler des informations dynamiques qu'il utilise. En effet si le raisonneur reçoit à chaque échange de nouvelles informations, il ne les traite pas de façon isolée. Il commence par les intégrer dans des structures plus globales faisant partie de l'historique et opère ensuite sur celles-ci.

De plus, ces structures sont susceptibles d'évoluer au cours des traitements et constituent un des résultats du raisonneur.

Nous pouvons néanmoins, définir ces nouvelles informations comme étant les énoncés portant sur la tâche. Ceux-ci ont été déterminés dans la typologie présentée dans Chapitre 6 ; ils recouvrent les fonctions suivantes : les faits, les buts et les questions.

Ces informations qui constituent les informations dynamiques du raisonneur, sont conservées dans des structures incluses dans l'historique du dialogue.

Elles sont représentées par différents ensembles que nous allons maintenant présenter.

La base de faits (BF)

Cette structure regroupe toutes les informations de nature factuelle du système. Elle est représentée par un ensemble BF dont les éléments sont de type

$$fait = (val : fe \times état : \{énoncé, défaut, donnée\} \times score : [0, 100]$$

dont nous définissons les fonctions qui permettent d'accéder aux différents attributs. Ces fonctions sont :

- $val(f : fait) \rightarrow fe$

  Cette fonction permet d'accéder au contenu d'un fait représenté par une formule.

- $état(f : fait) \rightarrow \{énoncé, défaut, donnée\}$

  Cette fonction fournit l'état d'un fait qui indique son l'origine. Un fait peut provenir d'une déduction, d'une valeur par défaut, d'un énoncé ou d'une donnée statique du système.

- $score(f : fait) \rightarrow [0, 100]$

  Cette fonction calcule le score d'un fait f en combinant les scores des noeuds de $val(f)$. Ce score désigne un taux de certitude de compréhension du fait f. Dans un premier temps, nous allons considérer que $score(f)$ est une donnée.

L'ensemble des buts (EB)

C'est un ensemble qui réunit tous les *buts* depuis le début du dialogue. Il est représenté à l'aide d'un ensemble $EB : ensemble(but)$ où $but = (val : fe \times score : [0, 100])$ est le type défini par :

- $val(b : but) \rightarrow fe$

  C'est une fonction qui permet d'accéder à la formule qui représente un but quelconque b. On l'utilisera de la même façon que dans BF.

- $score(b : but) \rightarrow [0, 100]$

  Cette fonction est analogue à celle qui s'applique sur BF.

L'ensemble des questions (EQ)

Cette structure rassemble les questions posées par le locuteur au cours du dialogue. C'est un ensemble EQ : $ensemble(question)$. Le type

$$question = (val : fe \times type : \{ouverte, fermée\} \times obj : fe \times score : [0, 100])$$

est défini par :

- $val(q : quest) \rightarrow fe$

  De façon similaire aux précédentes, cette fonction constitue l'accès à la formule qui représente la question.

- $type(q : question) \rightarrow \{ouverte, fermée\}$

  Cette fonction détermine si la question est ouverte ou fermée.

- $obj(q : question) \rightarrow fe$

  Dans le cas de question ouverte, $obj(q)$ permet d'accéder au concept sur lequel porte la question. $obj(q)$ est un sous-arbre de $val(q)$.

- $score(q : question) \rightarrow [0, 100]$

  Cette fonction s'interprète de la même manière que les deux précédentes.

3.3 Les connaissances statiques

L'ensemble des concepts (C)

Les connaissances statiques du raisonneur sont stockées dans différentes bases construites en fonction des accès que nous voulons y faire.

Nous avons déjà présenté la première de ces bases au début de ce Chapitre; il s'agit de l'ensemble C des concepts modèles.

En effet, en raison de la nature des contraintes que décrivent les relations définies dans cet ensemble, celui-ci fait partie des connaissances sur l'univers de l'application.

Cet univers sera d'autant plus étendu que l'on accepte un grand nombre de concepts dans C.

Remarque : Lorsque nous avons présenté l'interpréteur, nous avons soulevé le problème de la cohérence des bases de connaissances. Il faut noter à ce sujet, que les informations pragmatiques de la *base de faits* dans l'interpréteur doivent rester cohérentes avec les connaissances représentées par C.

N'ayant pas véritablement approfondi ce sujet dans notre étude, nous nous contenterons de l'effleurer en essayant de montrer la relation entre ces deux bases à travers un exemple simple :

> Dans les connaissances de l'interpréteur, nous avons cité comme exemple :
> `administration(mairie)`
> et dans C : `isa(mairie,administration)`.
> La relation entre ces deux informations apparaît ici de façon évidente.

La base des règles (BR)

Cette base renferme un ensemble de règles qui régissent l'activité de déduction du raisonneur. Le type associé aux éléments est $règle = (pré : fd, post : fc)$ sur lequel on définit les accès suivants :

- $pré(r : règle) \rightarrow fd$

  Cette fonction accède à la formule qui représente la pré-condition de la règle r.

- $post(r : règle) \rightarrow fc$

  Celle-ci accède à la formule représentant la post-condition de r.

Pour décrire une règle r, on utilise la notation : $a \Rightarrow b$ où $a = pré(r)$ et $b = post(r)$.

On remarque que *pré* est une formule quelconque tandis que *post* est restreinte à une formule conjonctive. Nous transformons systématiquement les règles r dont la précondition comporte une racine correspondant à une disjonction ($rac(r) = |$) en les décomposant de la manière suivante :

$$|(p_1, ..., p_n) \Rightarrow q \Leftrightarrow \begin{cases} p_1 \Rightarrow q \\ \vdots \\ p_n \Rightarrow q \end{cases}$$

Par conséquent, les traitements vont s'effectuer sur des règles ayant une formule conjonctive en pré-condition ainsi qu'en post-condition. Ces règles sont comparables aux clauses de Horn [COLMERAUER 83].

Examinons maintenant quelques exemples de règles :

```
(1)-nationalité(personne,étranger) => nationalité(personne,France)
(2)vol(document) => -perte(document)
(3)lieu-de-naissance(locuteur,France) => nationalité(locuteur,France)
```

La règle (1) signifie que si une personne n'est pas étrangère, alors elle est française.

La règle (2) veut dire que s'il s'agit d'un vol de document, ce n'est alors pas un cas de perte. Cette règle permet d'abandonner plus rapidement des solutions potentielles qu'on aurait conservées en mémoire.

Quant à la règle (3), elle concerne un locuteur. Si celui-ci est né en France, alors il est de nationalité française. Cette règle n'est évidemment pas correcte dans l'absolu, mais on peut imaginer une population où elle serait vérifiée.

A travers ces exemples, on constate que les règles peuvent décrire des connaissances, à la fois, sur l'univers de l'application (1), sur la tâche (2) et sur le locuteur(3).

La base de données (BD)

Cette base contient des connaissances sur la tâche et éventuellement sur le locuteur. Elles sont exprimées de manière assertive et sont représentées à l'aide du type *fait*.

Il existe toutefois quelques restrictions par rapport à BF :

- $d \in BD \Rightarrow état(d) \in \{défaut, donnée\}$
- $d \in BD \Rightarrow score(d) = 100$

Exemples et notations :

```
(1) validité(carte-d-identité,dix,ans)
(2) numéro-de-téléphone(CRIN,83-91-20-00)
(3) nationalité(locuteur,France) [défaut]
```

Les exemples (1) et (2) décrivent des renseignements qui constituent des données. En revanche, dans l'exemple (3), on précise que l'assertion est une valeur par défaut.

La base de renseignements (BX)

Toutes les connaissances décrivant la tâche ne peuvent pas être exprimées sous une forme assertive. Dans la famille d'application que nous traitons, il existe des renseignements qui possèdent une structure spécifique. Nous sommes donc amenés à les représenter à l'aide d'une structure de données appropriée.

Nous définissons cette structure à l'aide du type $renseign(but : fd \times rens : cond)$ dont les fonctions d'accès sont les suivantes :

- $but(b : renseign) \rightarrow fd$

  Cette fonction permet de sélectionner un groupe de renseignements au moyen du but (ou thème) d'une requête.

- $rens(b : renseign) \rightarrow cond$

  Il s'agit d'un accès aux renseignements proprement dits. Nous avons introduit un nouveau type *cond* que nous allons définir à son tour.

Le type $cond = (action : fd \times test : fd \times alors : cond \times sinon : cond)$ comprend quatre accès :

1. $action(c : cond) \rightarrow fd$
2. $test(c : cond) \rightarrow fd$
3. $alors(c : cond) \rightarrow cond$
4. $sinon(c : cond) \rightarrow cond$

Les accès récursifs (3) et (4) indiquent la structure arborescente de *cond*. Il s'agit, plus précisément, d'un arbre binaire dont la valeur associée aux noeuds est donnée par les fonctions *action* et *test*.

Le modèle de référence est celui des conditionnelles imbriquées.

- $action(c)$ fournit les renseignements disponibles lorsque les conditions de niveau supérieur sont satisfaites,
- $test(c)$ indique le test à vérifier pour satisfaire la condition c,
- $alors(c)$ détermine la condition imbriquée dans le cas où c est valide et $sinon(c)$ est la fonction complémentaire qui indique la condition enchâssée pour c non-valide.

Voici un exemple de renseignement que nous reprenons sous une forme graphique à la figure 4.11 :

```
but:  obtenir(personne,carte-d-identité)
si:  nationalité(personne,France)
alors:
        &(se-présenter(personne,|(mairie,commissariat)),
        fournir(personne,|(fiche-d-état-civil,livret-de-famille)))
        si:  mineur(personne)
        alors:  accompagner(parent(personne),personne)
```

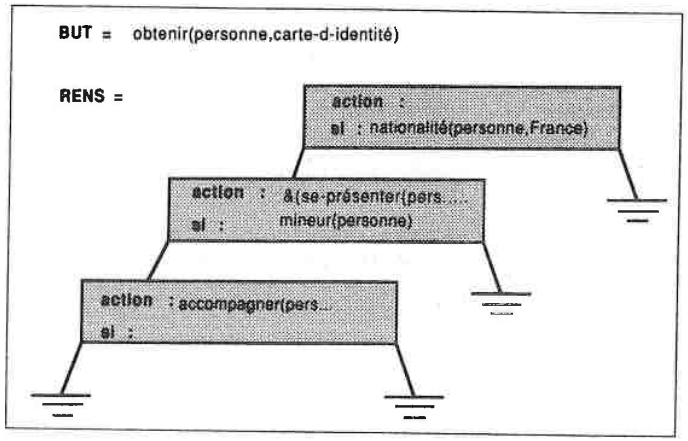


Figure 4.11: Exemple de renseignement.

```
        fsi:
fsi:
```

Nous allons nous ramener à des renseignements où les *buts* et les *tests* sont des formules conjonctives en appliquant les réécritures décrites à la figure 4.12 et 4.13.

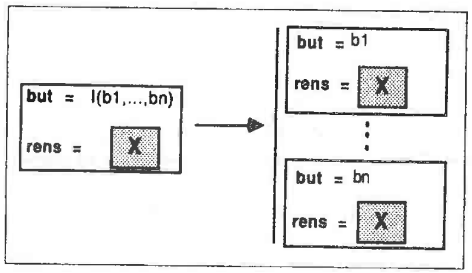


Figure 4.12: Première transformation d'un renseignement.

Les transformations du premier type permettent d'éliminer les disjonctions éventuelles du champ *but*.

Quant aux transformations du second type, elles conduisent à l'élimination des disjonctions dans les champs *test*.

Après avoir décrit les structures utilisées dans le raisonneur, nous allons consacrer les

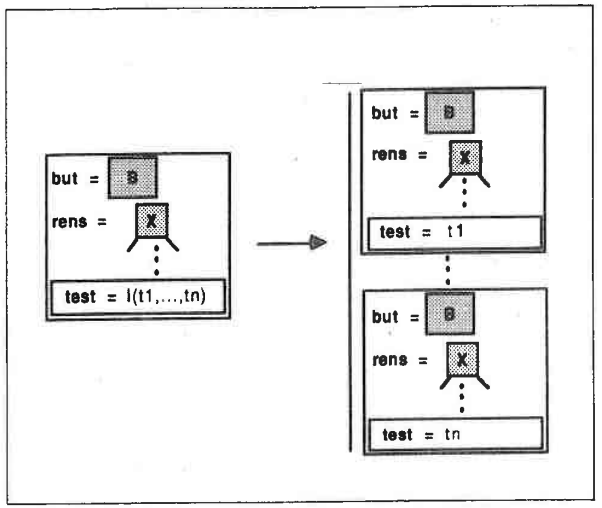


Figure 4.13: Seconde transformation d'un renseignement.

sections suivantes à la présentation des traitements qui agissent dessus. Pour cela, nous tenterons de montrer l'utilisation des différentes connaissances dynamiques et leur lien avec les connaissances statiques du module.

3.4 Les traitements sur BF

L'unification

Nous allons commencer par l'unification qui constitue un principe de base dans l'activité de raisonnement. Celle que nous utilisons s'inspire de la méthode de ROBINSON[ROBINSON 79] [CONDILLAC 86].

Dans les traitements que nous mettons en oeuvre, nous sommes conduits à unifier des éléments de l'ensemble des concepts objets (noté N).

Unifier deux éléments a et b appartenant à N revient à modifier la relation equ en lui rajoutant la relation supplémentaire $equ(a, b)$.

Cette opération n'est pas toujours possible. En effet, nous avons vu que a et b doivent préalablement vérifier les contraintes que nous leur imposons (voir Section 1.2).

Ensemble d'unificateurs : Un ensemble d'unificateurs, de type un, possède des éléments formés d'un couple de concepts ; autrement dit $un = ensemble(x : N \times y : N)$

Les éléments d'un ensemble d'unificateurs $u = \{(a_1, b_1), ..., (a_n, b_n)\}$ sont créés de telle sorte que $\forall i$, $equ(a_i, b_i)$ ne violent pas les contraintes de equ et de dif.

Par convention, on place dans le premier membre de chaque couple le concept le moins général, c'est-à-dire : soit $u : un$ tel que $u = \{(a_1, b_1), ..., (a_n, b_n)\}$, $\forall i, isa(a_i, b_i)$.

On construit la réunion de plusieurs ensembles de type un à l'aide de l'opération n-aire $comb(u_1 : un, .., u_n : un) \rightarrow un$ définie de la manière suivante :

$$comb(u_1, ..., u_n) = \begin{cases} \bigcup_{i=1}^{n} u_i & si \ \bigcup_{i=1}^{n} u_i \ est \ de \ type \ un, \\ \emptyset & sinon \end{cases}$$

La première étape de notre processus d'unification est la recherche d'un ensemble d'unificateurs.

Recherche des unificateurs : La recherche d'unificateurs s'effectue entre les ensembles des noeuds appartenant à deux formules distinctes.

Explicitons cette opération dans le cas de deux formules élémentaires. La notation fonctionnelle correspondante est : $match(f_1 : fe, f_2 : fe) \rightarrow un$.

Soient $f_1, f_2 : fe$, $res = match(f_1, f_2)$ est obtenu par l'algorithme suivant :

posons : $fils(f_1) = (x_1, ..., x_n)$ *et*
$fils(f_2) = (y_1, ..., y_m)$

initialisation : $res \leftarrow \emptyset$
si $isa(f_1, f_2)$
pour i *de* 1 *à* m
$res \leftarrow comb(res, match(x_i, y_i))$
fpour
$res \leftarrow comb(res, \{(f_1, f_2)\})$
fsi

Cet algorithme réalise l'appariement (*matching*) de deux formules. Par convention, un résultat égal à l'ensemble vide ($\emptyset$) indique un échec dans la tentative d'appariement.

Unifier : Nous avons montré qu'unifier deux concepts a et b revient à rajouter la relation $equ(a, b)$ dans N. Nous généralisons cette opération en l'appliquant itérativement sur chaque élément d'un ensemble d'unificateur (notation : $unif(u : un)$).

Sont unifiés deux objets faisant référence à un même concept. La résolution d'une référence anaphorique, le traitement d'une ellipse ou, tout simplement, la mise en correspondance des multiples occurrences d'un mot se rapportant à un même objet dans le dialogue, se concrétise par une unification dans l'ensemble des concepts.

Cependant, l'unification et, surtout, l'appariement interviennent dans d'autres traitements du raisonneur pour comparer une information produite par le locuteur avec certaines connaissances statiques du raisonneur.

L'appariement permet notamment de valider ou non un renseignement (comparé au but), un test (comparé à un fait), une action (comparée à une question).

Nous reviendrons ultérieurement sur ces points, mais nous allons auparavant présenter le mécanisme de déduction utilisé par le raisonneur dans lequel intervient également l'appariement.

La déduction

La déduction s'appuie sur deux bases de connaissances : la base des faits et la base des règles. L'objectif est d'inférer de nouveaux faits à partir de ceux déjà existants dans BF. Par conséquent, le traitement consiste principalement à enrichir BF en ***appliquant*** les règles contenues dans BR (voir figure 4.14).

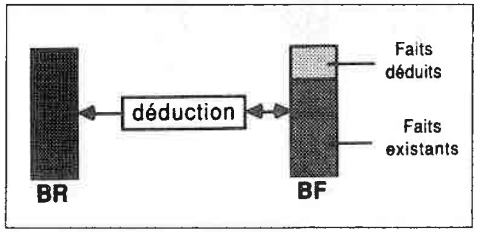


Figure 4.14: Les connaissances intervenant dans la déduction.

Parallèlement, il faut veiller à conserver la cohérence de BF ; la détection de l'incohérence constitue donc un résultat secondaire du processus de déduction.

Cycle de traitement : Concrètement, un cycle correspond à un échange minimal dans le dialogue.

Au début d'un cycle, les informations factuelles (ou faits) qui sont éventuellement apparues dans l'énoncé du locuteur sont placées dans BF. Pour des raisons évidentes de temps de traitements, nous voulons éviter, à chaque cycle, de confronter la totalité de la base des faits avec la base des règles. Pour cela, il faut avoir mémorisé les résultats intermédiaires provenant des cycles précédents.

Les traitements effectués dans un cycle se décomposent selon la séquence qui suit :

1. examen des règles ;
2. recherche des règles activables ;
3. planification des activations ;

4. choix des règles à activer ;

5. activation ;

6. recommencer la séquence tant qu'il existe des faits à examiner.

La figure 4.15 illustre l'évolution de BF lors de la déduction. Nous allons maintenant approfondir davantage chacune des étapes qui font partie d'un cycle.

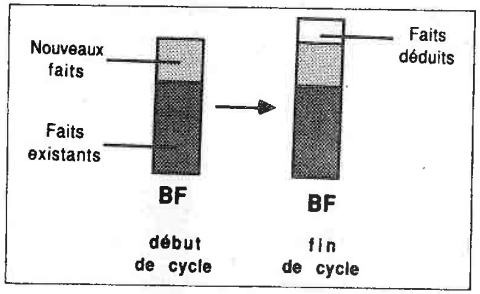


Figure 4.15: L'évolution de BF dans un cycle de déduction.

Examen des règles : Cette étape constitue une confrontation de la base des règles avec les nouveaux faits. Elle permet, à chaque cycle, d'évaluer leur influence mutuelle. Les itérations suivantes décrivent plus précisément le parcours que l'on effectue.

initialisation: soit $Efne$ = ensemble des faits non examinés.
pour chaque $r \in BR$
  pour chaque $f \in Efne$
    $examiner(r, f)$
  fpour
fpour
$Efne$ est maintenant examiné.

Reste encore à expliciter le traitement central noté *examiner*.

Rappelons que $pré(r)$ est une formule conjonctive et f, une formule élémentaire. L'opération *examiner* consiste à mémoriser les informations provenant de l'appariement de chaque *terme* de la pré-condition d'une règle avec un fait. Pour cela, on dispose d'une structure associée à chaque terme d'une précondition. Cette structure est de type $ensemble(f : fait \times u : un)$ accessible à l'aide de la fonction $act(fm : fe)$.

L'opération $examiner(rg : règle, ft : fait)$ est réalisée selon l'algorithme suivant :

initialisation: $tm \leftarrow terme(pré(rg))$
pour chaque $t \in tm$
  si $match(ft, t) \neq \emptyset$
    $ajouter(act(t), (ft, match(ft, t)))$
  fsi
fpour

En résumé, l'examen des règles se ramène à collecter des informations dans les structures de mémorisation associées aux termes de leur pré-condition respective. Une information élémentaire représente un fait pouvant être apparié avec le terme concerné accompagné de l'ensemble des unificateurs correspondant.

Lorsqu'un terme peut être apparier avec un fait, cela signifie que les pré-conditions de la règle correspondante deviennent partiellement valides. On tend, de cette façon, vers l'activation d'une règle.

Exemple :

Admettons qu'on examine le fait : f = `mineur(locuteur)` et la règle : r = `&(mineur(personne),émancipé(personne))=>majeur(personne)`; f vérifie le terme t = `mineur(personne)` de r.
Par conséquent, il valide partiellement la règle r.

Recherche de règles activables : L'étape suivante consiste à déterminer les règles suceptibles d'être activées. Une règle r devient activable (notée $activable(r)$) lorsque :

$$\forall r : règle,\ telle\ que\ pré(r) = \&(c_1, ..., c_n),\ activable(r)$$
$$\Leftrightarrow \exists (a_1, ..., a_n)\ tel\ que \begin{cases} a_1 \in act(c_1) \\ \vdots \qquad\qquad et\ comb(u(a_1), ..., u(a_n)) \neq \emptyset \\ a_n \in act(c_n) \end{cases}$$

Dans un cycle, pour chaque règle, nous recherchons toutes les suites $(a_1, ..., a_n)$ vérifant la condition d'activation précédente. Cette opération équivaut à une recherche de chemins de longueur fixe dans un graphe orienté.

Nous notons $rechrgact()$ la fonction qui,à un instant donné, fournit la totalité des règles activables sous la forme d'un ensemble de type $chemin : (r : règle \times sf : suite(fait) \times u : un)$, c'est-à-dire $rechrgact \rightarrow ensemble(chemin)$.

$rechrgact()$ est construite de la façon suivante : $\forall (a_1, ..., a_m)$ vérifiant la condition d'activation de la règle r_i, $(r_i, (f(a_1), ..., f(a_n)), comb(u(a_1), ..., u(a_n))) \in rechrgact(r)$.

Planification : Observons ce qui se produit lorsque plusieurs règles sont simultanément activables ou lorsqu'une seule règle est rendue activable mais avec plusieurs *chemins* possibles.

Comme nous pourrons le voir plus loin, l'activation d'une règle entraîne des unifications à partir des différents ensembles d'unificateurs associés aux chemins. Il devient donc indispensable de s'assurer auparavant de leur cohérence.

Pour mieux percevoir la présence d'un conflit, nous allons prendre un exemple avec deux règles r_1 et r_2 activables simultanément et s'excluant mutuellement. Ce cas se produit lorsque :

- $\exists(r_1, sf_1, u_1) \in rechrgact()$,
- $\exists(r_2, sf_2, u_2) \in rechrgact()$ et que
- $comb(u_1, u_2) = \emptyset$.

Il traduit la nécessité d'un choix car il est impossible d'effectuer l'unification avec u_1 et u_2 à la fois.

Remarque : Dans cet exemple, si $r_1 = r_2$, nous retrouvons le cas d'une seule règle activable par deux chemins distincts.

Pour gérer ces conflits, nous créons une structure externe supplémentaire dans laquelle sont exprimées les contraintes d'exclusion. Cela permet de disposer de toutes les possibilités avant de choisir les règles les plus pertinentes à activer.

Ce problème entre dans un cadre général de décomposition d'une tâche complexe en sous-tâches que l'on voudrait rendre indépendantes. Cette solution permet de décrire les contraintes dans une structure externe [STEFIK 81] qu'on appelle un *plan*. Dans notre cas, l'ensemble des déductions associées à un cycle constituent la tâche à décomposer. Le plan permet de choisir un sous-ensemble de règles activables simultanément et de les activer successivement et de manière indépendante.

Le plan est représenté par un graphe (voir figure 4.16) dont les noeuds sont de type *chemin*. Il est organisé de telle sorte qu'un chemin dans ce graphe correspond à un ensemble de règles activables sans conflit.

La planification consiste à construire, étape par étape, ce graphe et à y rechercher à chaque cycle le chemin optimal correspondant à un choix pertinent.

Choix des règles à activer : L'activation des règles dépend de la stratégie que l'on adopte dans cette recherche. Celle que nous appliquons repose essentiellement sur les critères suivants :

- Eviter les retours en arrière sur les déductions des cycles précédents : il ne faut pas oublier que le système réagit à la fin de chaque cycle et que pour formuler sa réponse il peut s'appuyer sur des informations déduites. Un retour arrière reflète une raisonnement erroné et engendre une phase de justification qui risque de rendre le dialogue trop lourd.

  Remarque: On n'est cependant pas à l'abri d'une erreur de raisonnement. Par conséquent, un mécanisme de retour arrière reste indispensable. Mais on n'y fait appel que dans des cas très spécifiques d'incohérence ou de contestation.

- Activer une séquence de règles susceptibles d'apporter le maximum d'informations : l'idée intuitive sous-jacente à ce critère est que plus on dispose d'informations, moins on a de questions à poser au locuteur. On tend ainsi vers un système qui accepte de prendre l'initiative.

  A défaut de connaissances permettant de mesurer objectivement l'information déduite, nous utilisons une heuristique qui avantage les chemins les plus *allongés*. Cela signifie que l'on privilégie l'activation du plus grand nombre de règles à chaque cycle.

L'exploitation de contraintes liées au domaine de l'application est une technique couramment pratiquée dans les problèmes d'*intelligence artificielle* [WINSTON 77]. L'utilisation d'une heuristique permet, soit de parvenir plus rapidement à une solution optimale, soit de l'approcher à moindre coût. Dans la mesure où le retour en arrière reste interdit, notre stratégie ne garantit pas l'obtention de la solution optimale (voir figure 4.16).

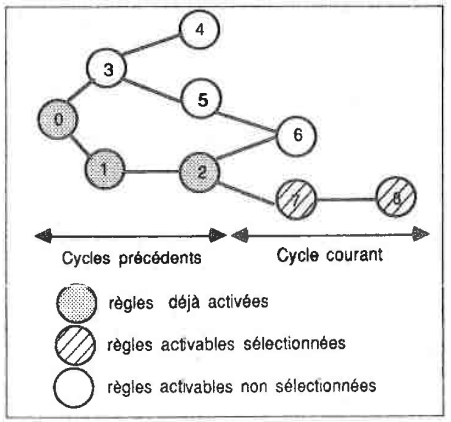


Figure 4.16: Exemple de plan.

En définitive, cette opération, notée *selectrgact()*, produit un ensemble dont les éléments sont de type *chemin* correspondant à un parcours *optimum* du plan, c'est-à-dire : $selectrgact() \rightarrow ensemble(ch : chemin)$

Activation : Cette ultime étape donne lieu à la modification de la base des faits. Les éléments de *selectrgact()* vont servir à créer de nouveaux faits que l'on ajoute à BF.

Le mécanisme d'activation d'un élément $x \in selectrgact()$ comporte deux phases distinctes :

- **La génération de faits semblables :** posons $x = (r, sf, u)$; il s'agit de *dupliquer* chaque terme de la post-condition de r en créant de nouveaux noeuds dans N et de leur donner la même structure arborescente (voir figure 4.17).

  Formellement, cela se traduit par une fonction de création de formule , $semblable(p : fe) \rightarrow fe$, définie selon les critères suivants :

$$\text{posons } p \in terme(post(x)),$$

$$\begin{aligned} semblable(p) = s \quad & tel\ que : \\ & val(p) = val(s)\ et \\ & sachant\ que\ fils(p) = \{p_1, ..., p_n\}\ et fils(s) = \{s_1, ..., s_n\} \\ & \Rightarrow \forall i,\ semblable(p_i, s_i) \end{aligned}$$

  En conséquence, le nouveau fait f que l'on ajoute à BF est tel que $val(f) = p$.

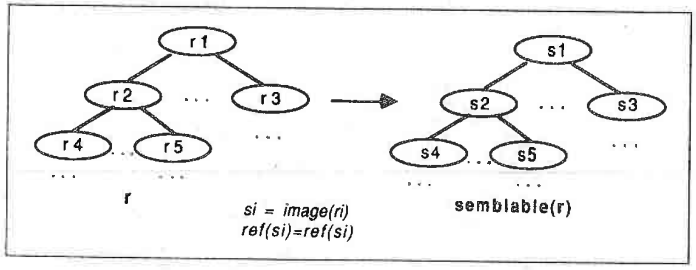


Figure 4.17: Illustration de la fonction *semblable(f)*.

  Chaque noeud $s_i \in s = semblable(p)$ résulte de la duplication d'un noeud de $p_i \in p$; on note $s_i = image(p_i)$ (voir figure 4.17).

- **L'unification des faits semblables :** l'ensemble des unificateurs u s'écrit $u = \{(f_1, p_1), ..., (f_n, p_n)\}$ où $\forall i$, f_i est un noeud d'un fait appartenant à la suite sf et p_i, un noeud appartenant à l'une des pré-conditions de r.

  Nous construisons un nouvel ensemble d'unificateurs u' en substituant, dans u, toutes les occurrences des noeuds p_i par $image(p_i)$. L'unification est alors effectuée à l'aide de u' par la fonction $unif(u')$.

La figure 4.18 représente les différents états d'une règle et de la base des faits lorsqu'on applique la déduction.

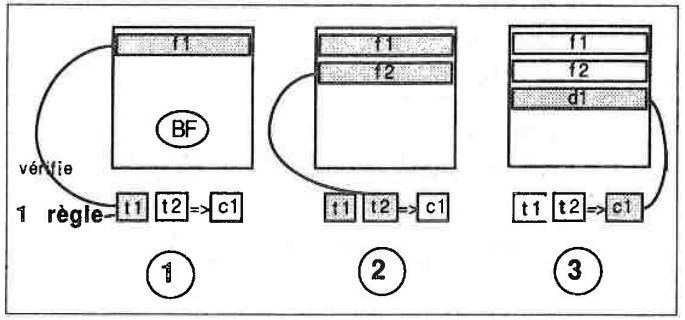


Figure 4.18: Activation d'une règle.

Notre solution présente l'avantage de se dérouler progressivement avec le dialogue et garantit la cohérence de l'ensemble des objets (N) apparaissant dans le dialogue (*univers du dialogue*).

Cependant, la mise en oeuvre de procédés de retour-arrière devient très lourde et coûteuse en temps pour mise à jour des différentes structures de mémorisation. Ce processus intervient lorsque la base des faits vient à contenir une incohérence.

Le contrôle de la cohérence

Dans cette section, nous allons exposer un aspect particulier de l'incohérence caractérisée par la présence d'une contradiction dans BF.

Contrairement à l'incohérence des bases de connaissances que nous avons évoquées dans la section 3.3, il est évident que le traitement des contradictions ne peut s'effectuer que d'une façon dynamique au cours du dialogue.

Détection : Le problème revient à assurer la consistance de la base de faits. La condition suivante doit donc toujours être vérifiée :

$$\forall x, y \in BF,\ match(x, -y) \cup match(-y, x) = \emptyset$$

Dans le raisonneur, la vérification s'effectue de manière dynamique, au moment de l'ajonction d'un nouveau fait f dans BF.

Une incohérence est détectée si $\exists x \in BF$ *tel que* $match(x, -f) \cup match(-f, x) \neq \emptyset$

La figure 4.19 montre une telle détection sur un exemple simple. Nous pouvons constater qu'une étape de déduction permet de conclure à une contradiction entre les faits $f1 = mineur(locuteur)$ et $f2 = majeur(locuteur)$.

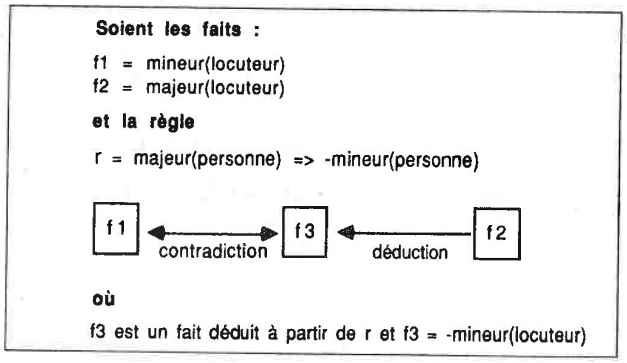


Figure 4.19: Détection d'une contradiction.

Tentative de résolution : Des procédures spécifiques sont déclenchées lorque qu'une incohérence apparaît. Elles ont pour objectif la recherche de l'origine de la contradiction et son élimination si cela s'avère possible.

Voici un recensement des cas qui sont suceptibles de se produire lorsque deux faits x et y sont contradictoires :

- l'un des deux faits est un fait par défaut ; supposons que $état(x) = défaut$. Ce conflit peut être naturellement résolu en supprimant le fait x car l'intérêt d'un fait par défaut est de ne rester valide que jusqu'à preuve du contraire,
- l'un des deux faits est beaucoup plus *incertain* que l'autre ; cette situation se manifeste par une disproportion des scores entre x et y. Supposons que $score(x) \ll score(y)$. La solution que nous avons retenue consiste à supprimer x; cela signifie que nous remettons en cause la compréhension du fait x,
- l'un des deux faits est une donnée (par exemple $état(x) = donnée$). Nous voulons éviter la destruction des propres connaissances du système. En conséquence, dans ce cas, nous choisissons de supprimer le fait y. En effet, dans les applications que nous développons, l'usager n'est pas autorisé à modifier les connaissances statiques du système,

- le dernier cas que nous voudrions évoquer se produit lorsque x et y sont proviennent d'un énoncé du locuteur ($état(x) = énoncé$ et $état(y) = énoncé$) et lorsque, par ailleurs, leurs scores restent comparables ($score(x) \approx score(y)$).

  Si ce cas est détecté à la suite d'une déduction, le raisonneur tente de revenir sur celle-ci en effectuant un retour-arrière dans le plan de déduction. Un autre chemin est alors sélectionné. Ce mécanisme vient compléter la démarche purement progressive de la déduction.
  Remarque : en raison du coût élevé de ce traitement, il devient indispensable de le limiter. La statégie que nous avons adoptée n'autorise les retours-arrière que jusqu'à un état correspondant à celui du début du cycle courant. Cette limite n'est pas bloquante pour le système car si aucun chemin *cohérent* ne peut être sélectionné, il subsiste toujours la solution d'engager une phase de dialogue spécifique permettant de supprimer l'incohérence.

  Lorsque que tous les chemins du plan ont été explorés sans succès, le raisonneur ne possède plus la capacité de résoudre le conflit. Son rôle se borne alors à signaler l'incohérence au module de gestion du dialogue par l'intermédiaire de l'état de la tâche. Cette information peut être exploitée par le gestionnaire si le besoin s'en ressent.

La combinaison des états et des scores

Les actions associées aux cas d'incohérence dépendent essentiellement d'attributs tels que l'état et le score des faits. Cela nous conduit à soulever le problème de la combinaison de ces attributs pour les faits issus de la déduction.

Etat d'un fait déduit : Rappelons qu'un fait peut être déduit à partir de la conjonction de plusieurs autres faits possédant divers états. Il s'agit de choisir une manière de les combiner simplement tout en restant cohérent vis à vis de leur emploi.

Notons + l'opération de combinaison des états et soit $E = \{énoncé, défaut, donnée\}$, l'ensemble des états. Il est commode de trouver une opération + qui soit commutative et associative. La figure 4.20 fournit la table de l'opération que nous avons retenue dans notre système.

Score d'un fait déduit : La combinaison des scores est un problème plus général auquel se heurtent tous les systèmes qui emploient des informations incertaines. Ce problème est

	donnée	défaut	énoncé
donnée	donnée	défaut	énoncé
défaut	défaut	défaut	défaut
énoncé	énoncé	énoncé	défaut

Figure 4.20: Combinaison des états.

étudié sur le plan théorique [DUBOIS 85] et sur le plan pratique [SHORLIFFE 76], et il en ressort que la combinaison dépend fortement des spécificités du problème traité.

Dans l'état actuel du système, le score provenant des niveaux inférieurs est entièrement simulé. L'élaboration d'une fonction de combinaison de score entre dans le cadre d'une stratégie globale du système de compréhension de dialogue. Une reflexion dans ce sens est menée actuellement dans notre groupe [ROMARY 87].

Dans la version actuelle du raisonneur, à un fait déduit, nous attribuons la moyenne des scores des faits qui possédent l'état : *énoncé* et qui ont permis de le déduire. Autrement dit : soit fd, un fait déduit à partir des faits $f_1, ..., fm, ...fn$. Supposons que $\forall i \in [1, m]$, $état(f_i) = énoncé$.

$$score(fd) = frac\sum_{i=1}^{m} score(f_i)m$$

La suppression d'un fait

Nous avons montré, dans la section consacrée au contrôle de la cohérence, que la résolution d'une contradiction pouvait se traduire par la suppression d'un fait. Notons qu'il ne s'agit pas du seul cas où cette action est nécessaire. En effet, le gestionnaire de dialogue peut également y faire appel dans le cas d'un énoncé de contestation.

Or, il existe, entre les faits, des relations de dépendance qui précisent leur origine. Nous les exprimons à l'aide des fonctions suivantes (voir aussi la figure 4.21) :

- $déduitde(f : fait) \rightarrow ensemble(fait)$: soit $f : fait$, $déduitde(f)$ fournit l'ensemble des faits qui ont directement participé à la déduction de f.

  Lorsque f n'est pas un fait obtenu par déduction alors $déduitde(f) = \emptyset$

- $induitpar(f : fait) \rightarrow ensemble(fait)$: soit $f : fait$, $induitpar(f)$ fournit l'ensemble des faits qui ont été obtenus par une déduction faisant intervenir f.

Par conséquent, la suppression d'un fait entraine aussi celle d'autres faits desquels il dépend.

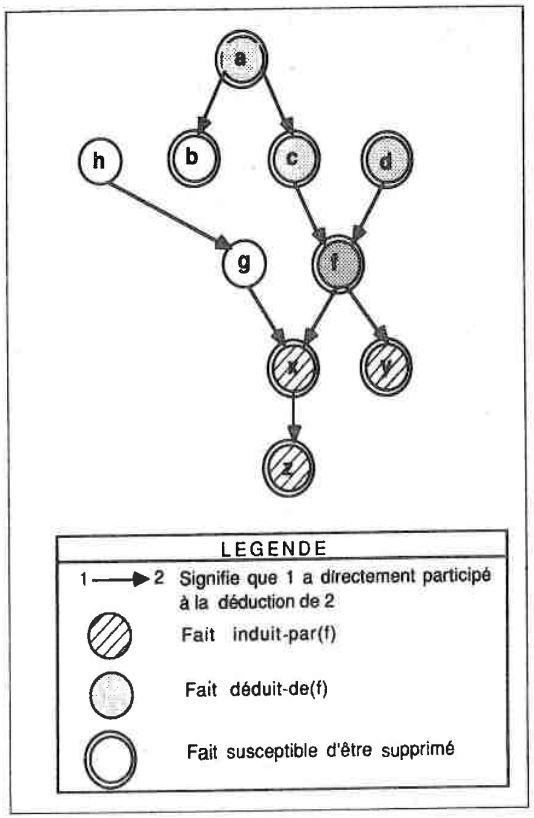


Figure 4.21: Dépendance entre les faits.

Le choix des faits *satellites* qui vont être supprimés en même temps qu'un fait f quelconque est déterminé par l'algorithme suivant :

$supprimer(f) = res$ tel que ;

posons : $déduit(f) = (d_1, ..., d_n)$ *et*
résultat : res : ensemble de faits à supprimer
initialisation : $res \leftarrow f \cup induit(f)$
pour i *de* 1 *à* n *tel que* $état(d_i) \neq donnée$
$res \leftarrow res \cup supprimer(d_i)$
fpour

A la figure 4.21, les faits entourés d'un double anneau dont l'état est différent de *donnée*

seront donc automatiquement détruits lorsqu'on demande la suppression du fait f.

D'autre part, chaque suppression effective d'un fait est obligatoirement accompagnée d'une remise à jour des structures de mémorisation dans lesquelles il est intervenu.

3.5 Les traitements sur EB

Rappelons que EB contient l'ensemble des énoncés possédant la fonction *but* ou *thème* dans notre typologie. Chaque élément est représenté à l'aide d'une formule élémentaire, c'est-à-dire $val(b) : fe$.

Par ailleurs, BX contient des renseignements accessibles à l'aide de l'attribut *but()* qui fournit une formule de type *fc*. Nous supposons ici que les transformations, visant à normaliser la représentation des renseignements, ont été réalisées.

Les traitements sur EB consistent, tout d'abord, à déterminer les renseignements dont le champ *but* est vérifié par un sous-ensemble de EB et, ensuite, pour chacun d'entre eux, à valider les champs *test* en fonction des informations situées dans la base des faits.

Pour traiter EB, nous adoptons la même démarche que pour BF, c'est-à-dire une démarche progressive fondée sur le cycle de traitement.

Un cycle comporte plusieurs étapes que nous représentons par la séquence suivante :

1. examen des renseignements ;
2. recherche des renseignements activables ;
3. choix d'un renseignement ;
4. validation des conditions à l'intérieur du renseignement sélectionné;
5. création d'un état courant de ce renseignement.

Remarque : Cette séquence n'est pas itérative car l'activation d'un renseignement n'engendre pas de nouveaux buts.

Examen des renseignements : Cette étape vise à prendre immédiatement en compte l'influence de nouveaux buts sur BX. Pour cela, nous utilisons une technique semblable au traitement de BF. En effet, à chaque terme de l'accès *but()* d'un renseignement, nous associons une structure identique à celle de *act()* où le type *but* joue le rôle du type *fait*, le type *renseign* celui de *règle* et, enfin, la fonction *but()*, celle de *pré()*.

Prenons un exemple où $b1 \in EB$ est un nouveau but qui vérifie un terme du *but* d'un renseignement r, c'est-à-dire :

$$\text{soit } x1 \in terme(but(r)),\ match(b1, x1) \neq \emptyset.$$

La figure 4.22 illustre la structure de mémorisation résultant de l'examen du but b_1 et du renseignement r.

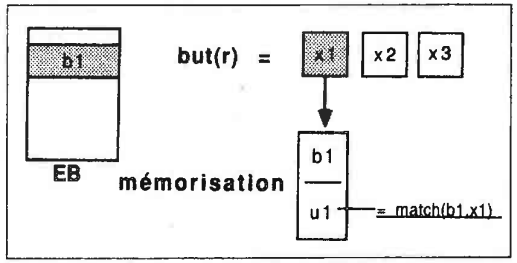


Figure 4.22: Structure de mémorisation au cours de la phase d'examen.

Recherches des renseignements activables : Dans cette étape, nous cherchons à déterminer les renseignements qui sont susceptibles de contenir une solution à la requête de l'utilisateur. Il s'agit de réaliser un premier filtrage dans le but de réduire l'espace de sélection.

Un renseignement r passe à travers ce filtre s'il correspond à un but de l'usager. On dit alors qu'il est activable et on note $activable(r)$. Cette condition se traduit formellement de la façon suivante :

$$\forall r : renseign,\ tel\ que\ but(r) = \&(x_1, ..., x_n),$$
$$activable(r) \Leftrightarrow \exists (b_1, ..., b_n)\ tel\ que \begin{cases} b_1 \in act(x_1) & \\ \vdots & et\ comb(u(b_1), ..., u(b_n)) \neq \emptyset \\ b_n \in act(x_n) & \end{cases}$$

Ce problème est identique à celui de la recherche des règles activables. Par conséquent, nous procédons exactement de la même manière pour le traiter.

Reprenons l'exemple précédent, en rajoutant des buts supplémentaires de telle sorte que le renseignement r soit maintenant activable. La figure 4.23 fournit les différents chemins qui rendent r activable. Nous considérons, bien entendu, que la combinaison des différents unificateurs reste possible.

Le type *chemin* est donc étendu en adaptant le type correspondant à $chemin = (r : renseign, sf : suite(but) \times u : un)$. Nous pouvons alors introduire la fonction $rechrsact(r : renseign) \rightarrow ensemble(chemin)$ qui fournit, à un instant donné, l'ensemble des chemins qui autorisent r à devenir activable.

Choix d'un renseignement : Le fait de choisir un seul renseignement découle de la stratégie générale de dialogue que l'on désire mettre en oeuvre. Nous choisissons de ne

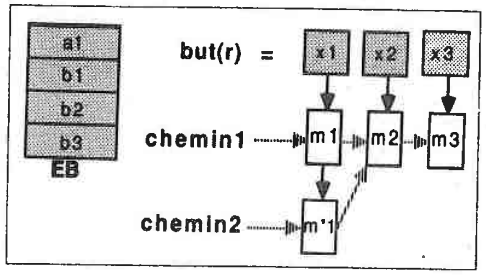


Figure 4.23: Chemins permettant de rendre un renseignement activable.

traiter qu'une seule requête à la fois.

Dans ce cas, en présence de plusieurs renseignements activables, il est indispensable d'opter pour l'un d'entre eux. Il s'agit donc de sélectionner un élément de *rechrsact*(). Pour effectuer ce choix, nous combinons les critères suivants :

- préférer un renseignement correspondant à un but plus précis : ce critère est approché par le choix d'un *chemin* si $x \in rechrsact()$ tel que la longueur de $sf(x)$ soit maximale ;
- opter pour un renseignement répondant à une demande plus *certaine* : cela revient à choisir un renseignement validé par des buts possédant les scores les plus élevés ;
- privilégier la demande la plus récente : cela entraine le choix du *chemin* x tel que $sf(x)$ contienne le but le plus récemment apparu dans EB.

La fonction $selectrsact() \rightarrow chemin$ s'appuie sur ces principes pour délivrer le *meilleur chemin* parmi l'ensemble *rechrsact*().

Remarque : Ce choix n'est pas définitif. En effet, lorsqu'un but concernant le renseignement sélectionné est supprimé, d'autres renseignements, jusque là en attente, peuvent être choisis à leur tour. Cela se produit lorqu'une des requêtes de l'usager est satisfaite et que la stratégie du dialogue mène, dans ce cas, à la suppression du but courant.

Validation des conditions : Cette opération porte sur un renseignement particulier $r = ch.r$ où ch = selectrsact().

Sachant que $c = cond(r)$ est un arbre binaire, il s'agit de vérifier si

$$\exists f \in BF \text{ et } \exists n \in c \ (n \text{ est un noeud de } c)$$
$$\text{tels que } match(f, test(n)) \neq \emptyset \text{ ou } match(f, -test(n)) \neq \emptyset$$

La fonction $valid(r : renseign) \rightarrow ensemble(cv)$ s'efforce de construire un ensemble de quadruplet de type $cv = (f : fait \times n : cond \times signe : \{+,-\} \times u : un)$ en respectant les conditions suivantes :

- **condition locale :**

$$x \in valid(r) \Leftrightarrow \begin{cases} match(f, test(n(x))) \neq \emptyset \text{ et } \subseteq u(x) & si\ signe(x) = + \\ match(f, -test(n(x))) \neq \emptyset \text{ et } \subseteq u(x) & sinon \end{cases}$$

  De cette façon, nous construisons un ensemble contenant le plus grand nombre de conditions vérifiées.

- **condition générale :**

$$\forall x = (f, n, +, u) \in valid(r), \forall f', u';\ y = (f', n, -, u') \notin valid(r) \text{ et}$$
$$\forall x = (f, n, -, u) \in valid(r), \forall f', u';\ y = (f', n, +, u') \notin valid(r)$$

  Nous interdisons ainsi la validation à la fois positive et négative d'une condition dans un même ensemble.

- **autre condition générale :** supposons que r ait été sélectionné à partir du chemin $ch = selecrsact()$.

$$valid(r) = \{v_1, ..., v_n\} \Rightarrow comb(u(v_1), ..., u(v_n), u(ch) \neq \emptyset$$

  Cette condition permet d'assurer la cohérence des unificateurs entre eux et avec ceux qui ont conduit à la sélection du renseignement.

Prenons un exemple concret.

Soit un renseignement r tel que $but(r) = obtenir_r(personne_r, ci_r)$ et $cond(r) = c$ est donné à la figure 4.24

Supposons que $b = obtenir_b(locuteur_b, ci_b) \in EB$ et que
$f = nationalité_f(locuteur_f, France_f) \in BF$. La fonction $valid(r) = \{(f, c, +, u)\}$ avec :

$$u = \{(nationalité_f, nationalité_c), (locuteur_f, personne_c), (France_f, France_c),$$
$$(obtenir_b, obtenir_r), (locuteur_b, personne_r), (ci_b, ci_r)\}$$

Ces informations vont permettre d'accéder aux *actions* qui sont censées renfermer les réponses aux questions de l'utilisateur.

Création d'un état des conditions : Afin de faciliter l'exploitation des résultats obtenus par l'ensemble des traitements sur EB, nous créons une structure conditionnelle intermédiaire dans laquelle nous faisons disparaître toutes les conditions vérifiées. Cette structure représente un état instantané des conditions d'un renseignement. Elle est construite à l'aide de la fonction $etatcond(c : cond) \rightarrow cond$ qui s'écrit de la façon suivante :

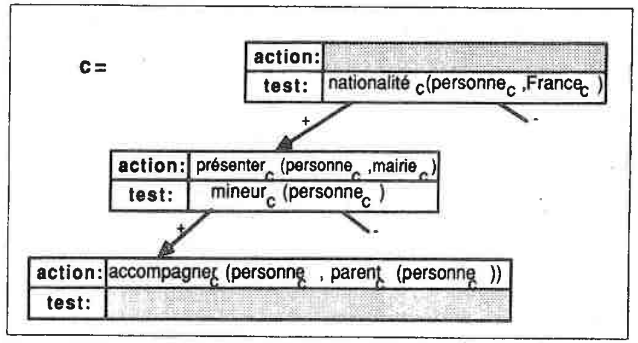


Figure 4.24: Exemple : *cond(r)*.

$$étatcond(c) = res \text{ tel que}$$

si $\exists x \in valid(r)$ *tel que* $n(x) = c$
alors
  cas $signe(x) = +$:
    $action(alors(c)) \leftarrow et(action(alors(c)), action(c))$
    $res \leftarrow étatcond(alors(c))$
  cas $signe(x) = -$:
    $action(sinon(c)) \leftarrow et(action(sinon(c)), action(c))$
    $res \leftarrow étatcond(sinon(c))$
sinon
  $alors(c) \leftarrow étatcond(alors(c))$
  $sinon(c) \leftarrow étatcond(sinon(c))$
  $res \leftarrow c$
fsi

La figure 4.25 permet de visualiser ce traitement à travers un exemple.

3.6 Les traitements sur EQ

EQ contient les énoncés appartenant à la classe *question* dans notre typologie. Nous supposons qu'une question est disponible dans *EQ* tant qu'elle n'a pas été satisfaite. Cela signifie que le gestionnaire de dialogue a la possibilité d'inhiber une question lorque la réponse a été apportée au locuteur.

Deux traitements bien distincts sont à considérer dans *EQ* : la première consiste à sélectionner une question dans *EQ* et la seconde, à rechercher une réponse à cette question dans *BD* ou *BX*.

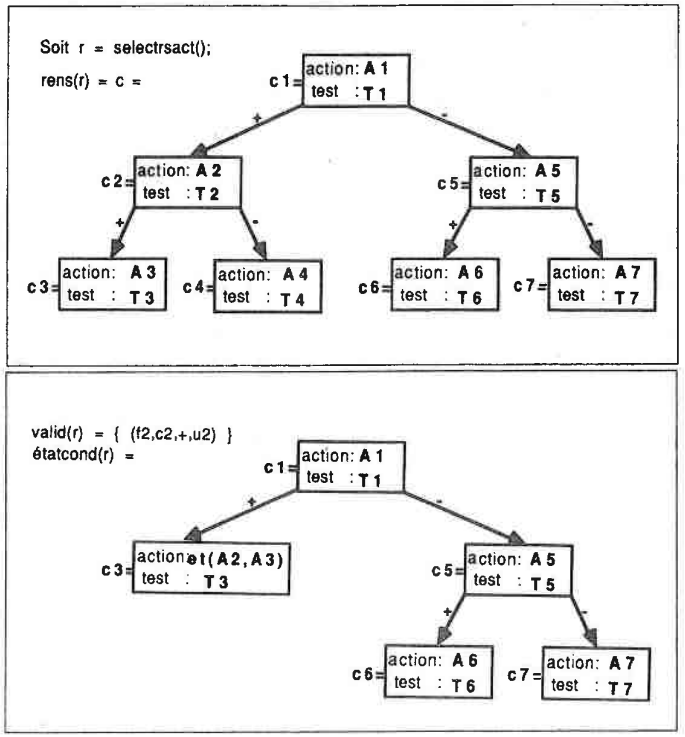


Figure 4.25: Suppression des conditions vérifiées.

Sélection d'une question

Le déroulement du raisonnement est guidé par une question posée par l'usager. En conséquence, pour que le système puisse réagir normalement, il est indispensable d'établir cette question.

Trois cas cas de figures peuvent se présenter :

- *EQ* ne contient qu'une seule question : cet unique élément est forcément sélectionné et nous chercherons à lui apporter une solution ;

- *EQ* est vide : nous avons prévu la possibilité d'associer à chaque renseignement, une question implicite. Il arrive, en effet, qu'une requête en langage naturel ne comporte pas explicitement de question. Celle-ci fait partie des présuppositions des

intervenants.

Exemple de requête :
`J'aimerais renouveler ma carte d'identité.`
sous-entendu `Quelles sont les formalités?`

Par conséquent, s'il est possible de déterminer une question implicite à partir du renseignement courant sélectionné, ce cas se ramène au premier. Dans le cas contraire, le raisonneur ne possède pas la capacité de satisfaire la requête de l'utilisateur ;

- Il existe plusieurs questions concurrentes dans EQ : dans ce cas, un choix intervient. Il repose sur les critères suivants :
  - Une question dont on détient la réponse est préférable ;
  - Une question récente prime sur une ancienne ;
  - Le score sert également de discriminant entre deux questions.

Après avoir déterminé la question à traiter, nous nous attachons à établir une réponse.

Recherche d'une réponse

Dans la pratique, nous distinguons deux catégories de questions : les questions ouvertes et les questions fermées.

Les premières suscitent des réponses sous la forme de *valeurs*, tandis que les dernières attendent une confirmation ou une infirmation.

Exemple :
questions ouvertes :
(1) `?formalité` correspond à une question comme : *quelles sont les formalités...?*
(2) `présenter(locuteur,?lieu)` correspond à : *où faut-il se présenter...?*
questions fermées :
(3) `gratuit(fiche d'état civil)?` est une interprétation de : *la fiche d'état civil est-elle gratuite?*
(4) `munir(locuteur,carte d'identité)?` représente une question comme : *faut-il se munir de la carte d'identité...*

Comme nous pouvons le constater la réponse à une question peut se trouver soit dans BX, soit dans BD. Dans ce dernier cas, la précision d'un but n'est pas nécessaire.

Chaque cas fait appel à des traitements spécifiques que nous allons maintenant présenter.

Recherche d'une solution dans BD

Question ouverte : Rechercher une solution pour une question q telle que $type(q) = ouverte$ consiste à déterminer tous les concepts pouvant s'unifier avec $obj(q)$ et provenant de l'appariement $val(q)$ avec val(d) où d décrit tous les éléments de BD.

La fonction d'appariement $match'(x : fe, y : fe)$ se différencie de la fonction $match(x : fe, y : fe)$ par le traitement particulier appliqué à $obj(q)$. En effet, $\forall x, match'(obj(q), x) = match(x, obj(q))$.

Illustrons cette fonction par un exemple :

soit une donnée $d = cout_d(document, 100F)$ et une question $q = cout_q(ci, ?prix)$;

$$match'(q, d) = \{(cout_q, cout_d), (ci, document), (100F, prix)\}$$

On notera l'*inversion* du dernier élément de cet ensemble d'unificateurs.

Le résultat qu'il faut retenir de la comparaison d'une question et d'une donnée est le concept auquel peut s'unifier l'objet de la question si celui-ci existe. Cette fonction, notée $repouv(f : fe, q : question) \rightarrow fe$ permet de généraliser la recherche dans un ensemble de formules élémentaires de la façon suivante :

$repouv_{et}(ef : ensemble(fe), q : question) \rightarrow fc;$
$repouv_{et}(\{f_1, ..., f_n\}, q) = res$ *tel que*

initialisation $res = fvide$(représente la formule vide)
pour i *de* 1 *à* n
$res = et(res, repouv(f_i, q))$
fpour

Remarques : Nous avons introduit une fonction $et()$ qui effectue la conjonction de deux ou plusieurs formules quelconques. Nous supposons également que cette fonction retourne une formule canonique.

D'autre part, pour des commodités d'initialisation, nous utilisons une formule particulière $fvide$ qui sert à décrire les cas limites et qui se comporte comme un élément neutre pour la plupart des opérations sur les formules.

L'interprétation de la réponse à la question q dans BD est obtenue en substituant, dans $val(q)$, le noeud $obj(q)$ par $repouv_{et}(BD, q)$. Lorsque $repouv_{et}(BD, q) = fvide$, cela signifie que la question n'a pas de réponse dans BD. En revanche, lorsque le résultat est une formule conjonctive (et non élémentaire), il existe plusieurs réponses à la question.

Question fermée : Il s'agit, cette fois-ci, de trouver dans BD, une réponse à une question fermée.

Considérons un élément unique $d \in BD$ et tentons de savoir si d apporte une réponse à une question quelconque q. Il existe alors trois possibilités de réponse : elle est, soit *affirmative*, soit *négative* ou encore *indéfinie*, c'est-à-dire que d n'apporte pas de solution à q.

Ce résultat est fourni par la fonction $repferm(f : fe, q : question) \rightarrow \{oui, non, indéfini\}$ telle que :

$$repferm(f,q) = \begin{cases} oui & si\ match(q,f) \neq \emptyset \\ non & si\ match(-q,f) \neq \emptyset \\ indéfini & sinon \end{cases}$$

Ce traitement est généralisé à toute la base de données en appliquant la fonction $repferm_{et}(ef : ensemble(fe)) \rightarrow \{oui, non, indéfini, indéterminé\}$ qui *combine* les résultats de $repferm()$.

Afin de rendre le traitement indépendant de l'ordre d'arrivée des données, la fonction de combinaison que nous utilisons est commutative et associative. Elle est donnée par la table de la figure 4.26.

	oui	non	indéfini	indeterminé
oui	oui	indeterminé	oui	indeterminé
non	indéterminé	non	non	indeterminé
indéfini	oui	non	indéfini	indeterminé
indéterminé	indéterminé	indéterminé	indéterminé	indeterminé

Figure 4.26: Fonction de combinaison de réponses à une question fermée.

On peut noter l'introduction d'une nouvelle réponse : *indéterminé*. Elle exprime le fait qu'il existe au moins deux données qui répondent, de manière opposée, à la question examinée.

Selon la modalité de la question, ce résultat peut être interprété de différentes manières. Mais cela dépasse le cadre du raisonneur qui s'arrête à la détection des différents cas. La décision du choix de la réponse est reportée au niveau du gestionnaire de dialogue.

Recherche d'une solution dans BX

En réalité, seule une partie de BX est utilisée pour effectuer la recherche. En effet, l'exploration se limite aux données accessibles à partir du renseignement r sélectionné et par l'état de ses conditions. Ces informations sont données par l'accès $action()$ des noeuds de $étatcond(rens(r))$.

Question ouverte : Répondre à une question ouverte q en utilisant les informations fournies par l'état des conditions c revient à établir une nouvelle structure conditionnelle

semblable à c où $\forall n \in c$, l'attribut $action(n)$ est remplacé par le résultat de la recherche locale de solution entre $action(n)$ et q.

Cette recherche locale est toutefois un peu plus complexe que pour les questions à la base de données car $action(n)$ peut être disjonctif. Etudions le cas général, où $action(n) = f = |(\&(a_1, ..., a_a), ..., \&(n_1, ..., n_n))$.

Le résultat de cette recherche est fourni par la fonction $repouv_{ou}(f : fd, q : question) \rightarrow fd$. Appliquée au cas général f,

$repouv_{ou}(f,q) = |(repouv_{et}(terme(\&(a_1, ..., a_a)), q), ..., \&(repouv_{et}(terme(\&(n_1, ..., n_n)), q))$

Remarque : Il arrive qu'un sous-arbre $s \subset c$ ne dispose pas de réponses à la question examinée. Cette situation se manisfeste par :

$$\forall n \in s, action(n) = fvide$$

Un *élagage* visant à supprimer la *branche* inutile, est alors effectué.

Reprenons l'exemple de la figure 4.25.

La figure 4.27 illustre la recherche de solution à une question ouverte en simulant les différents cas possibles.

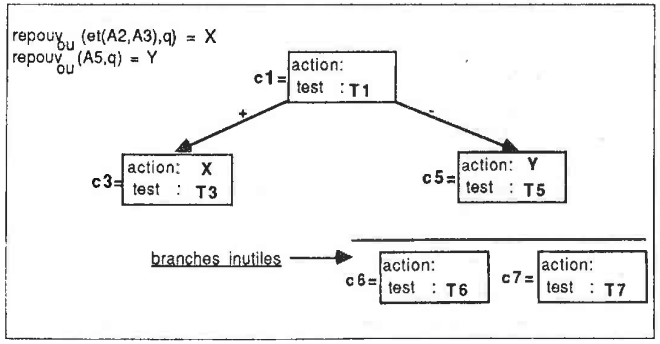


Figure 4.27: Construction d'une réponse à une question ouverte.

Question fermée : Le résultat que nous recherchons ressemble à celui du cas précédent, c'est-à-dire une structure conditionnelle provenant de $étatcond(rens(r))$

La différence se situe au niveau de l'attribut $action()$ qui, pour les questions fermées, est un élément de l'ensemble : $\{oui, non, indéfini, indéterminé\}$

Nous sommes donc amenés à introduire la fonction $repferm_{ou}(f : fd, q : question) \rightarrow \{oui, non, indéfini, indéterminé\}$ où la principale difficulté est de définir une fonction de combinaison traduisant la disjonction des différentes réponses possibles.

Nous proposons une solution, sous forme de table à la figure 4.28. Cependant, nous sommes conscients du fait de cette fonction reste très contestable. Mais, les combinaisons les plus discutables correspondent à des cas marginaux dans les applications qui nous intéressent.

	oui	non	indéfini	indeterminé
oui	oui	indéfini	indéfini	indéfini
non	indéfini	non	indéfini	indéfini
indéfini	indéfini	indéfini	indéfini	indéfini
indéterminé	indéfini	indéfini	indéfini	indéterminé

Figure 4.28: Fonction de combinaison de réponses à une question fermée.

Nous prenons à nouveau, à figure 4.29, l'exemple de la figure 4.25 appliquée cette fois-ci à une question fermée.

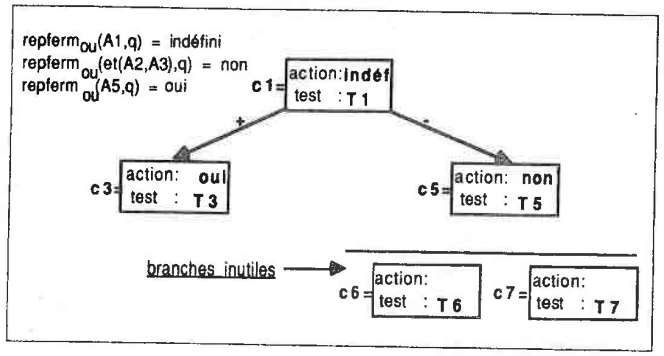


Figure 4.29: Construction d'une réponse à une question fermée.

3.7 Le contrôle dans le raisonneur

Jusqu'à présent nous nous sommes attachés à décrire les traitements locaux sur les différentes structures contenant les connaissances du raisonneur sans tenir compte de l'ordre de leur activation.

L'objet de ce paragraphe est de présenter l'enchaînement de ces traitements dans le processus de raisonnement.

La finalité est de transmettre un diagnostic sur l'état de la tâche au gestionnaire de dialogue. L'organigramme de la figure 4.30 indique toutes les possibilités d'enchaînements. Nous y faisons également figurer, aux feuilles, les différents états de la tâche qui vont être

exploités par le gestionnaire de dialogue.

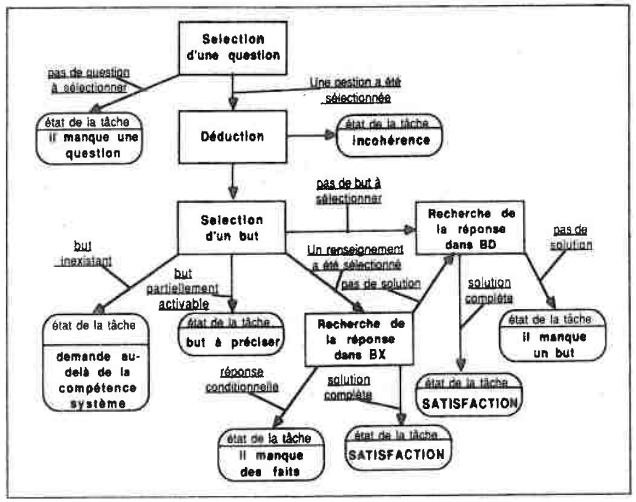


Figure 4.30: Enchaînement des traitements dans le raisonneur.

4 LE GESTIONNAIRE DE DIALOGUE

4.1 Les fonctions

La décision finale sur la réponse à apporter à l'usager appartient au gestionnaire de dialogue. Pour cela, il tient compte non seulement du résultat du raisonneur élaboré à partir des énoncés portant sur la tâche, mais également des énoncés portant sur le dialogue et sur le canal de communication.

En effet, pour répondre à une intervention du locuteur, composée d'actes de langage possédant des fonctions variées, plusieurs solutions peuvent convenir. Un des rôles de ce module est d'effectuer un choix parmi toutes les éventualités.

A cette première fonction appelée **conduite du dialogue**, s'ajoute celle de **prédiction** Elle consiste à communiquer aux processeurs de niveaux inférieurs des hypothèses sur la suite du dialogue. Ces informations sont ensuite utilisées pour initialiser, à chaque cycle, les processus descendants.

4.2 Les connaissances dynamiques

Comme le raisonneur, le gestionnaire s'appuie également sur l'ensemble des interventions pour progresser dans le dialogue. Celles-ci sont regroupées dans des structures incluses dans l'historique. Elles constituent les connaissances dynamiques du module.

Ces informations se composent essentiellement d'une représentation chronologique des interventions et de l'état de la tâche qui évoluent parallèlement.

La représentation chronologique (HISTO)

Cette structure, notée $HISTO$: $suite(intervention)$, est représentée par une suite dont les éléments désignent chacun une intervention.

Une intervention est une entité de type $intervention = (locuteur : \{usager, machine\} \times fn : ensemble(fonction))$ où :

- $locuteur(i : intervention) \rightarrow \{usager, machine\}$: indique l'émetteur de l'intervention;
- $fn(i : intervention) \rightarrow ensemble(fonction)$: fournit l'ensemble des fonctions des actes de langages contenus dans une intervention munies de leurs attributs.

  Par conséquent, le type

  $$fonction = (val : \{but, fait, question, ouverture, \ldots\} \times attr : attribut)$$

  Les informations rangées dans l'attribut dépendent de la fonction associée.

  Les énoncés émis par le système possèdent des attributs qui représentent la réponse fournie au locuteur.

  Pour ceux du locuteur, nous reproduisons à la figure 4.31 les différents cas de fonctions avec leur attribut respectif. Ce tableau reprend plus précisément certaines informations de la figure 4.9.

La représentation sous forme de suite est ici très bien adaptée car elle permet facilement de réaliser les principales requêtes du gestionnaire telles que:

Exemples :
"Existe-t-il un énoncé portant sur la tâche dans la dernière intervention du locuteur?"
ou
"Quels sont les faits de l'avant-dernière intervention du locuteur?"
ou encore
"Quel est le dernier énoncé de la machine?"
ou enfin
"Quelle est la plus récente question?" ...

FONCTIONS	ATTRIBUTS
Fait	fait
But	but
Question ouverte	question + noeud
Question fermée	question
Maintien du canal	
Perturbation du canal	+, −
Ouverture	
Fermeture	
Contestation	intervention ou vide
Satisfaction	
Répétition	intervention ou vide

Figure 4.31: Fonctions et attributs.

L'état de la tâche (ET)

Cette structure rend compte des activités du raisonneur. Elle renferme une variété d'informations susceptibles d'être exploitées lors de la gestion du dialogue.

En premier lieu, nous y trouvons l'*état de la tâche* proprement dit. Il s'agit des conclusions du raisonneur que nous avons précédemment établies, c'est-à-dire :

- **pas de question :** aucune question n'a pu être déterminée ;
- **but inconnu :** une demande contenant un but n'est pas disponible dans les connaissances sur la tâche du système ;
- **but partiellement validé :** il existe un but dont seuls certains termes ont pu être validés ;
- **pas de but :** il n'existe ni but, ni réponse dans BD ;
- **réponse conditionnelle :** la requête est incomplète ;
- **satisfaction :** une réponse a été déterminée ;
- **incohérence :** une contradiction est apparue.

Il est à noter que certains de ces résultats peuvent se combiner entre eux et que d'autres s'excluent mutuellement. Ainsi, les états *satisfaction* et *incohérence* peuvent, par

exemple, coexister, tandis que *satisfaction* et *pas de question* ne peuvent pas apparaître simultanément.

A ces informations, il faut adjoindre quelques résultats intermédiaires du raisonneur qui interviennent également dans la gestion du dialogue tels que :

- **la question active** : il s'agit d'un accès à la question sélectionnée par le raisonneur pour établir une réponse ;
- **les buts actifs** : de la même manière, on peut accèder aux buts ayant permis la sélection d'un renseignement ;
- **l'état de la réponse** : cela correspond, selon le cas, soit à la réponse complète, soit à la réponse conditionnelle proposée par le raisonneur ;
- **les buts inconnus** : Un but ne correspondant à aucun renseignement disponible est signalé dans l'état de la tâche ;
- **les buts incomplets** : un renseignement partiellement validé par un ensemble de buts apparaît également dans *ET* ;
- **les faits contradictoires** : dans le cas d'une contradiction, le raisonneur fournit les faits qui en sont à l'origine ;
- **les faits par défaut** : il s'agit des faits par défaut ayant effectivement participé à la validation des conditions ;
- **les faits de score faible** : ce sont, enfin, les faits dont les scores ont été estimés insuffisants, mais qui ont néanmoins servi à valider des conditions.

Toutes ces informations contribuent à définir une stratégie visant à produire une réponse correcte.

4.3 Le modèle de dialogue

Dans le chapitre précédent, nous avons cité des études linguistiques relativement récentes [ROULET 80] d'où il ressort une organisation hiérarchique du dialogue.

Cette hiérachie est fondée sur l'existence des trois constituants majeurs de la conversation : l'échange, l'intervention et l'acte de langage.

Dans la perspective d'un dialogue homme-machine finalisé et dans le cadre d'une famille d'applications, nous sommes amenés à concevoir un modèle de dialogue plus restrictif où les échanges sont clairement définissables et les possibilités d'enchaînements relativement limitées.

Ainsi, au niveau le plus élevé de cette hiérarchie, nous définissons le dialogue comme la juxtaposition des trois échanges qui suivent (voir également figure 4.32) :

- **Phase 1** : cet échange décrit la phase introductive d'un dialogue. Il renferme les premières interventions d'établissement de la communication, de salutations et de présentation des différents interlocuteurs ;
- **Phase 2** : cet échange contient lui-même des échanges enchâssés qui visent à présenter une requête et à converger vers sa réalisation.
- **Phase 3** : c'est un échange de clôture du dialogue formé des interventions de salutations et de fins de communication.

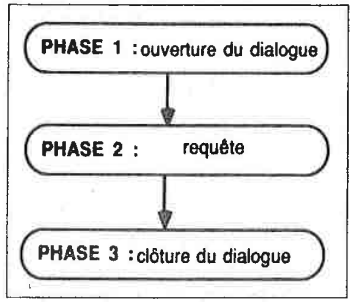


Figure 4.32: Description d'un dialogue.

Ce formalisme est très commode car il permet de concevoir un modèle de dialogue de manière modulaire et descendante. Cependant, dès que nous atteignons les niveaux inférieurs, il devient inopérant pour plusieurs raisons :

- Il n'est pas toujours possible de définir nettement la frontière entre deux échanges : en effet, une même intervention peut terminer un échange et débuter le suivant. C'est le cas de l'exemple qui suit :

  `Bonjour, c'est Mr. X, j'aimerais connaître la durée de validité d'une fiche d'état civil`

  Cette intervention appartient simultanément à la phase introductive et à la présentation de la requête.

- La manière d'effectuer une transition n'apparaît pas dans ce formalisme : pour passer d'une phase à un autre, de nombreux critères entrent en jeu. La transition peut être conditionnée :

- par l'identification ou l'absence de certaines fonctions dans l'intervention courante ;
- par l'historique du dialogue ;
- ou par l'état de la tâche.

- Notre modèle est destiné à être utilisé par le système. Or, ce formalisme ne permet pas de faire la distinction entre les interventions de l'usager et celles de la machine. Les premières sont adaptées à une description *déclarative* tandis que les secondes sont de nature plus *procédurale*.

Compte tenu de ces remarques, il devient opportun d'éviter un modèle totalement descriptif. Pour pallier les faiblesses d'un tel modèle, nous proposons l'approche que COULON ET KAYSER [COULON 86] qualifient de *psycho-cognitive* et qui consiste à considérer une conversation comme un processus respectant un scénario typique. L'intérêt majeur que nous trouvons dans cette approche réside dans la structuration d'un processus sous forme de séquences typiques appelées, selon les auteurs, *stéréotypes* [MINSKY 75] ou *scripts* [SCHANK 77]. Cette approche a débouché sur des formalismes de représentations de connaissances fondés sur les *frames* [BOBROW 77b] qui concilient les aspects déclaratifs et procéduraux des connaissances.

Nous nous inspirons de cette démarche pour structurer le réseau décrivant les différents enchaînements possibles dans un dialogue. Cette organisation repose sur le découpage d'un échange sous forme d'un paquet structuré de connaissances appelé *schéma*.

Le schéma : DIALOGUE

Le schéma du niveau le plus élevé peut alors être exprimé de la façon suivante :

Nom : *DIALOGUE*
Nombre de sorties : 0
Définition des sorties : –
Variables : *p1* : *sortie*;
Procédure :
  PHASE1 → *p1*;
  si (*p1* = *echec*) *alors aller à clôture*;
  PHASE2;
clôture :
  PHASE3;
  retour;

Examinons les différentes composantes ou *attributs* de ce schéma :

- **Le nom :** il s'agit d'un identificateur qui sert à activer un schéma. Dans le cas présent, l'amorçage d'un dialogue s'effectue par l'appel *DIALOGUE*.

- **Les sorties :** les deux attributs, *nombre* et *définition*, concernant les sorties éventuelles d'un schéma permettent, respectivement, de codifier les différents résultats de son activation et d'expliciter la codification.

- **Les variables :** plusieurs *types* de variables peuvent intervenir dans les procédures. Nous avons utilisé ici un type prédéfini représentant une sortie de schéma. Des types tels que *compteurs, booléen, entier, chaîne* ... peuvent également apparaître.

- **Les procédures :** toutes les possibilités de la programmation classique sont utilisées pour décrire une procédure (séquence, test, saut ...). Les actions qui sont exécutées dans une procédure visent :
  - à contrôler le déroulement de la procédure : cette possibilité est inhérente à la notion de procédure ;
  - à enchaîner les schémas : l'enchaînement est assuré par l'appel imbriqué d'un nouveau schéma et par l'examen de sa *sortie* transmise par l'instruction *retour*. Il est à noter que l'enchaînement peut également être conditionné par l'état de la tâche, par l'historique et par la valeur de certaines variables ;
  - à modifier certaines structures de l'historique : cette mise à jour se limite à la suppression de *buts*, de *questions* et de faits nécessaire dans certaines situations ;
  - à engendrer une réponse : pour cela, nous utiliserons la fonction prédéfinie *dire(message)* qui émet un énoncé à l'intention du locuteur ;
  - à se mettre en attente d'un énoncé : nous utiliserons à cet effet une autre fonction prédéfinie *acquérir()* qui donne la parole à l'usager ;
  - à prédire la suite du dialogue : il est également possible d'envoyer un message vers les autres processeurs par l'intermédiaire de la fonction *prédire(processeur, message)*. De cette façon, il devient possible d'émettre des prédictions sur le prochain énoncé du locuteur.

Le schéma : PHASE1

Pour compléter notre exemple, écrivons le schéma *PHASE1* :

Nom : *PHASE1*
Nombre de sorties : 2
Définition des sorties :
succès : la phase introductive s'est bien terminée par l'acquisition d'un énoncé portant sur la tâche.
echec : il est impossible de connaître la requête de l'utilisateur ; la transaction est abondonnée
Variables : *c* : *compteur de valeur initiale* 0;

Procédure :
dire("Bonjour, ici le centre de renseignements...");
prédire(syn,"énoncé introductif + présentation de requête");
prédire(lex,"introduction + requête");
boucle :
acquérir();
si (la dernière intervention du locuteur contient un énoncé portant sur la tâche)
alors retour(succès);
fsi
si (la dernière intervention du locuteur contient des salutations)
alors dire("bonjour");
fsi
incrémenter(c);
si (c < 3)
alors
dire("Veuillez énoncer votre requête svp");
prédire(syn, présentation de requête);
prédire(lex, requête);
aller à boucle;
sinon
dire("Nous sommes obligés d'interrompre...");
retour(echec);
fsi

Le schéma : PHASE2

Le schéma *PHASE2* décrit le modèle de détermination et de la satisfaction d'une requête de l'usager. La figure 4.33 illustre les principales étapes de cet échange.

Cela se traduit plus précisément dans notre formalisme par les schémas suivants :

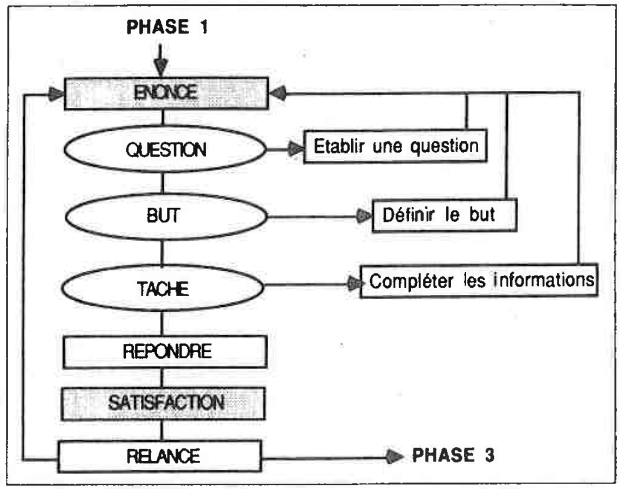


Figure 4.33: Principales étapes de la phase de présentation et satisfaction d'une requête.

Nom : *PHASE2*
Nombre de sorties : 0
Définition des sorties : –
Variables : *q, b : sortie;*
Procédure :
boucle :
QUESTION → q;
si (q = echec) alors retour;
fsi
BUT → b;
si (b = echec) alors retour;
fsi
si (b = inconnu) alors aller à relance;
fsi
TACHE;
si (la dernière intervention du locuteur contient des salutations)
alors retour;
relance :
dire("désirez – vous un autre renseignement ?");
prédire(syn, confirmation/infirmation);
prédire(lex, confirmation/infirmation);
acquérir();
si (la dernière intervention du locuteur contient une infirmation)
alors retour;
aller à boucle;

Ce schéma fait appel à *QUESTION*, *BUT* et *TACHE* que nous explicitons ci-dessous :

Le schéma : QUESTION

Nom : *QUESTION*
Nombre de sorties : 2
Définition des sorties :
succès : Ce résultat est transmis lorsqu'une question a pu être sélectionnée.
echec : Après plusieurs essais, aucune question n'a pu être établie.
Variables : *c* : *compteur de valeur initiale* 0;
Procédure :

```
boucle :
  si (∃une question active) retour(succès);
  incrémenter(c)
  si (c < 3)
  alors
    dire("Quelle est votre question ?");
    prédire(syn, question);
    prédire(lex, question);
    acquérir();
    aller à boucle;
  sinon
    dire("Nous sommes obligés d'interrompre...);
    retour(echec);
```

Le schéma : BUT

Nom : *BUT*
Nombre de sorties : 3
Définition des sorties :
succès : Ce résultat est transmis lorsqu'un renseignement a pu être sélectionnée.
echec : Après plusieurs essais, aucun but n'a pu être intreprété.
inconnu : Il existe un but inconnu.
Variables : *c* : *compteur*

Procédure :

```
  c ← 0;
boucle :
  si (∃un but actif) ou (∃une réponse)
    retour(succès);
  fsi
  si (∃un but inconnu)
    dire("Désolé, nous ne savons pas comment " + but inconnu);
    retour(inconnu);
  fsi
  incrémenter(c)
  si (c < 3)
  alors
    si (∃un but incomplet)
    alors dire("Précisez votre requête svp.");
    sinon dire("Et dans quel but svp ?");
    fsi
    prédire(syn, but);
    prédire(lex, but);
    acquérir();
    aller à boucle;
  sinon
    dire("Nous sommes obligés d'interrompre...);
    retour(echec);
  fsi
```

Le schéma : TACHE

Nom : *TACHE*
Nombre de sorties : 0
Définition des sorties : –
Variables :
Procédure :

```
boucle :
  si (∃ une réponse complète)  REPONDRE;  retour;  fsi
  si (∃ une réponse conditionnelle)  COMPLETER;  aller à boucle;  fsi
  si (∃ une incohérence)  RESOUDRE;  aller à boucle;  fsi
```

Remarque 1 : *TACHE* fait appel à trois autres schémas, *REPONDRE*, *COMPLETER* et *RESOUDRE*. La nature très procédurale de ces schémas risque de nous entraîner dans des descriptions trop techniques. Aussi, nous allons éviter de descendre jusqu'au détail des procédures et nous restreindre à la présentation de leur rôle essentiel.

- **REPONDRE :** Ce schéma détermine la réponse à apporter à l'utilisateur afin de satisfaire sa requête. Cette réponse est fonction de la question posée et de la solution proposée par le raisonneur dans l'état de la tâche

Pour informer davantage l'usager sur les hypothèses faites par la machine, une des méthodes consiste à préciser les valeurs par défaut et les informations de faible score sur lesquelles le raisonneur s'est appuyé pour fournir un résultat.

Exemple :

U : Quelles sont les formalités d'obtention d'une carte d'identité?

S : Si vous êtes de nationalité française, adressez-vous...

Par ailleurs, la satisfaction d'une requête s'accompagne de la suppression de la question active de l'ensemble des questions. Nous parvenons de cette façon à influencer la sélection d'une question au niveau du raisonneur lorsque il en existe plusieurs.

Pour une question fermée, une réponse correspondant à chaque solution proposée par le raisonneur doit être prévue (affirmatif, négatif, indeterminé ou indéfini). Pour une question ouverte, le problème est identique. Il est donc indispensable de disposer d'un générateur qui, à partir d'un message exprimé dans une représentation interne, fournit un énoncé en langage naturel.

Dans le cadre de ce travail, nous contournons ce problème en admettant l'existence d'un tel outil. Cependant, nous restons conscients de sa complexité et de son importance pour le développement futur de notre système.

- **COMPLETER :** Ce schéma reflète une situation où la requête est incomplète. L'état de la tâche contient alors une réponse conditionnelle. Par conséquent, il s'agit dans un premier temps de sélectionner la condition qu'il faudrait valider pour se rapprocher, de façon optimale de la satisfaction de la requête puis, de choisir la formulation de la question correspondante afin d'obtenir une réponse non ambiguë.

  Dans la version actuelle du gestionnaire, la fonction de sélection reste très sommaire; elle fournit la condition associée au noeud de la racine dans l'arborescence formée par la réponse conditionnelle.

  Pour obtenir une question efficace, nous offrons la possibilité d'attribuer à chaque test dans un renseignement, une procédure permettant, à la fois, de formuler une question jugée convenable, et d'émettre les prédictions appropriées. Pour parvenir à la validation de la condition, il suffit de déclencher cette procédure si, toutefois, elle existe. Dans le cas contraire, une confirmation du test est demandée.

  Exemple :

  Si l'objectif est de savoir si le locuteur est de nationalité française ou non, les deux questions suivantes sont possible :

  - Etes-vous de nationalité française?

  ou

  - Quelle est votre nationalité?

- **RESOUDRE :** La stratégie que nous adoptons ne conduit à l'activation de ce schéma que dans le cas où aucune autre possibilité de réponse n'est apparue. Nous supposons que cette situation de bloquage est provoquée par une incohérence.

  Par conséquent, il s'agit de demander l'aide du locuteur pour résoudre cette incohérence. Pour cela une question précise est posée. Pour les mêmes raisons que précédemment, son choix est important. Dans la version actuelle du gestionnaire de dialogue, nous demandons la confirmation de l'un des faits ayant provoqué une contradiction. Une réponse négative signifie que le fait concerné est à supprimer; dans le cas inverse, c'est l'autre fait contradictoire qui disparaît.

Remarque 2 : D'autre part, le modèle que nous avons décrit n'a pas la prétention d'être unique. Notre propos est de montrer qu'à travers le ¡ formalisme des schémas, il est possible de l'améliorer, voire de le reconstruire sans trop de difficultés.

Remarque 3 : Par ailleurs, à tout instant du dialogue, de situations particulières, telles que l'incompréhension, la contestation, la demande de répétition, la gestion du canal, peuvent apparaître. Pour éviter d'alourdir notre formalisme, nous les décrivons à l'aide de schémas qui seront constamment actifs au cours du dialogue.

Les rôles de ces schémas sont les suivants :

- **incompréhension :** nous distinguons trois cas d'incompréhension :
  - l'incompréhension totale : rien ne peut être interprété à partir de l'énoncé du locuteur ; l'action appropriée est, par conséquent, la demande de répétition ;
  - l'incompréhension partielle : une interprétation partielle est obtenue ; une précision concernant les informations manquantes est alors demandée ;
  - l'ambiguïté : plusieurs interprétations sont acceptables ; dans ce cas, une demande de confirmation envoyée.
- **contestation :** la contestation peut être globale, auquel cas nous décidons de supprimer tous les énoncés portant sur la tâche dans l'avant-dernière intervention du locuteur.

  Mais elle peut être également plus ciblée ; elle entraine alors la suppression sélective de faits, de buts ou de questions particuliers.

- **demande de répétition :** l'action correspondante est évidemment la répétition de la dernière intervention de la machine ;

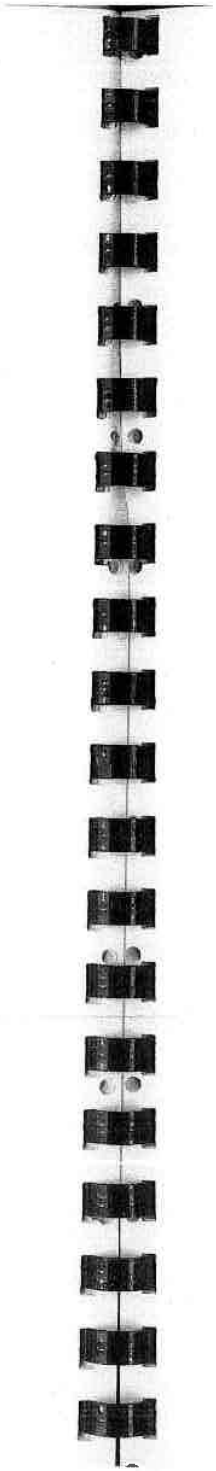
- **gestion du canal :** les messages de maintien du canal de communication sont ignorés. Actuellement, ceux de perturbation le sont également car nous ne disposons pas encore de module de synthèse de la parole.

4.4 Le contrôle dans le gestionnaire

Pratiquement, le gestionnaire est composé d'un ensemble d'outils permettant de communiquer avec les processeurs *SYN-SEM* et *LEX*, d'un générateur d'énoncés en langage naturel et de l'interpréteur de schémas qui assure l'enchaînement des schémas et l'appel des procédures associées.

Son fonctionnement est amorcé par l'exécution du schéma particulier *DIALOGUE*. De plus, à chaque acquisition d'une intervention de l'usager, les schémas correspondant à l'incompréhension, à la contestation, à la demande de répétition et à la gestion du canal sont activés s'il y a lieu.

Le contrôle est effectué dynamiquement par l'éxécution des actions contenues dans les procédures.



Chapitre 5

LES RESULTATS ET LES PERSPECTIVES

1 IMPLANTATION

Pour implanter le processeur DIAL, nous avons utilisé le langage C [KERNINGHAN 78] Ce choix se justifie pour diverses raisons :

- nous développons sous le système UNIX, lui-même écrit en grande partie en C. Le choix de ce langage facilite l'utilisation des outils déjà existants dans l'environnement du système ;

- en raison de la masse des données qu'ils manipulent, les traitements de bas niveau nécessitent des programmes efficaces. A cet effet, C semble un langage approprié. Par conséquent, pour assurer une compatibilité entre les différents processeurs, nous sommes amenés à utiliser le même langage de programmation pour le processseur DIAL.

Sur le plan de l'organisation interne du programme, nous conservons trois grands modules correspondant à l'interpréteur, au raisonneur et au gestionnaire de dialogue. Chaque module est réalisé puis testé séparément. De plus, nous nous efforçons de le concevoir comme un module possédant des bases de connaissances externes qui peuvent évoluer indépendamment des traitements. La figure 5.1 présente schématiquement les différentes composantes qui interviennent dans la réalisation d'un module.

Sur ce schéma, nous voyons apparaître un ensemble de logiciels qui correspondent à :

- une batterie **de compilateurs** (C) qui permet à une personne chargée de mettre en place une application, de constituer et de tester aisément les connaissances pour un module ;

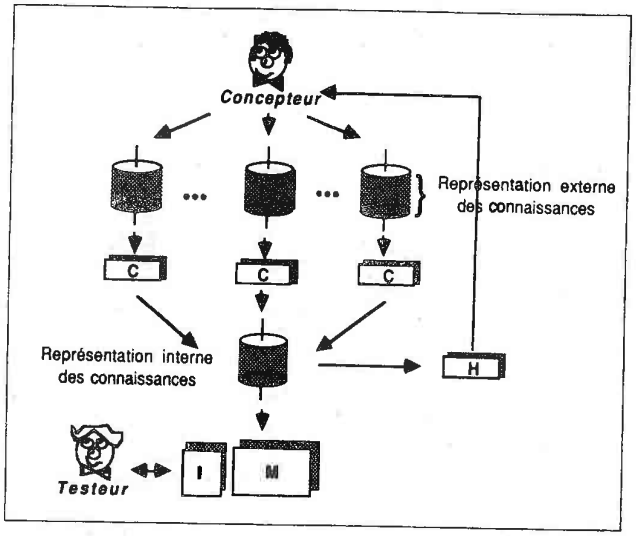


Figure 5.1: Les composantes d'un module.

- **des outils de vérification statique de la cohérence** (H) des différentes bases de connaissances du module.

  Une reflexion concernant ces outils a été menée dans notre équipe à l'occasion d'un stage [LORANT 86]. Pour notre part, nous ne détaillerons pas ici cet aspect du système car les réalisations restent encore très limitées ;

- **une interface** (I) qui est destinée à simuler les entrées du module.

- et enfin **le module de central traitement** (M) qui remplit les principales fonctions du module ;

2 L'INTERPRÉTEUR

2.1 L'organisation

Dans sa version actuelle, l'interpréteur se présente comme un système autonome, possédant ses propres outils de maintenance des bases de connaissances et une interface de test indépendant. La connection avec le reste du module DIAL n'est pas encore effective et constitue la prochaine étape de l'évolution de ce produit. De ce fait, les intéractions avec l'historique sont simulées à l'aide d'un ensemble de fonctions spécifiques.

Son organisation générale est donnée par la figure 5.2.

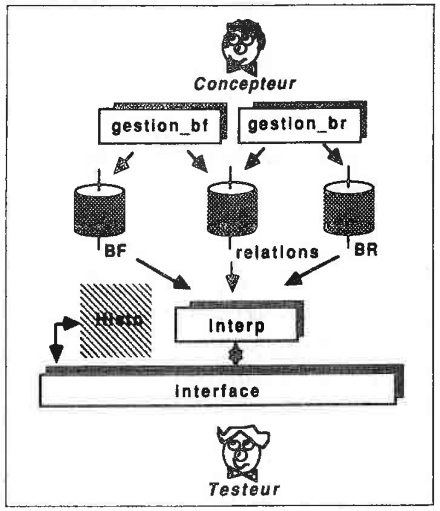


Figure 5.2: Les composantes de l'interpréteur.

Sur cette figure, nous voyons apparaître ses principales composantes parmi lesquelles nous trouvons :

- **gestion-bf et gestion-br :** ce sont des outils interactifs qui servent à créer, à modifier et à visualiser la base de faits (BF) et de la base de règles (BR). Ils assurent également leur cohérence.

  gestion-bf et *gestion-br* agissent respectivement sur les fichiers contenant la représentation interne de BF et de BR.

  Le fichier intermédiaire *relation* regroupe les schémas de relations utilisés dans les différentes bases. Lors de sa construction, la première apparition d'une relation détermine son schéma. Par la suite, toute nouvelle instance doit vérifier le schéma correspondant. Un contrôle est effectué avant chaque création ; la détection d'une incohérence se traduit par le rejet du fait ou de la règle concernés :

  Exemple :

```
introduction des conditions.....

schema >p
variable >#x
```

```
variable >
...
schema >p
variable >#x
variable >#y
variable >
erreur dans le nbre d'arg.  de la relation...
veuillez recommencer, s.v.p...
```

- **interp :**

il s'agit de l'interpréteur lui-même. Il est alimenté par l'intermédiaire d'une interface qui effectue la saisie de l'énoncé sous la forme d'une liste de prédicats, et qui simule les accès à l'historique. Certaines informations issues de l'historique seront introduites systématiquement en même temps que l'énoncé ;

Exemple :

```
enonce-precedent-question()
```
indique que l'énoncé précédent de la machine était une question.

D'autres informations sont explicitement demandées par l'interface ; en effet, l'exécution de l'action *en-machine* mentionnée dans le chapitre précédent engendre l'appel d'une fonction intéractive de simulation de l'historique. Cela se traduit concrètement par une interruption du déroulement de l'interpréteur et par une interrogation de l'utilisateur sur le contenu de l'historique ;

Exemple :

```
Simulation à l'accès à l'historique...
introduisez le dernier enonce-machine, svp.
>...
```

L'interface permet, en outre, de suivre le déroulement de l'interpréteur et de visualiser le résultat de l'interprétation.

2.2 Un exemple

Dans ce paragraphe, nous présentons, à travers un exemple, le fonctionnement de l'interpréteur en commentant les résultats de son déroulement.

Dans cette optique, nous allons restreindre les différentes bases de connaissances aux informations intervenant dans le traitement de l'exemple formé par l'énoncé suivant :

```
Je voudrais savoir à qui il faut s'adresser pour faire une
demande de double-nationalité.
```

Notons que la représentation syntaxico-sémantique de cette phrase a déjà été fournie à la figure 3.4.

Avant d'effectuer l'interprétation, nous devons constituer la base des faits et la base des règles.

La base des faits (BF)

Dans le cadre de notre exemple, la visualisation de *BF* produit la liste suivante :

```
BASE DE FAITS
=============

id : 0   ~_proposition (carte_d_identite)          ←Connaissances de nature
id : 1   ~_proposition (double_nationalite)        syntaxique ;
id : 2   _pers_sing1 (je)                          le symbole ~ exprime
id : 3   _pron_interr (qui)                        la négation.
id : 4   _extension (savoir)                       ←Connaissances de nature
id : 5   _extension (connaitre)                    sémantique.
id : 6   _actobj (faire)
id : 7   _echobt (demande)
id : 8   _echobt (demander)
id : 9   _echobt (s_adresser)
id : 10   _echobt (desirer)
id : 11   _echobt (obtenir)
id : 12   _anime (qui)                             ←Connaissances de nature
id : 13   _carte_d_identite (carte_d_identite)     pragmatique ;
id : 14   _double_nationalite (double_nationalite) reliant un mot
id : 15   _obtenir (obtenir)                       à un concept
id : 16   _demander (demande)
id : 17   _demander (demander)
id : 18   _s_adresser (s_adresser)
```

La base des règles (BR)

Les règles que nous utilisons dans cet exemple sont les suivantes :

```
BASE DE REGLES
==============

identificateur de regle : 0

   _proposition(#x)
 & _predicat(#x,#p1)
 & _extension(#p1)
 & _cas_obj(#p1,#o1)
 & _proposition(#o1)
 & _cas_goal(#p2,#g2)
 & _proposition(#g2)

 => ajouter(_but(#g2))
  & ajouter(_question(#o1))
-------------------------------

identificateur de regle : 1

   _question(#q)
 & _predicat(#q,#p)
 & _echobt(#p)
 & _cas_ben(#p,#b)
 & _cas_agt(#p,#a)
 & _pron_interr(#a)

=> ajout_int("*question:)
 & ajouter(_declencher(#p))
 & interpreter(#p)
 & ajout_int("()
 & ajouter(_declencher(#b))
 & interpreter(#b)
 & ajout_int(",)
 & ajout_int("?)
 & ajouter(_declencher(#a))
```

←Règle d'identification d'un énoncé.

Si l'énoncé possède 3 propositions imbriquées telles que le niveau 2 soit le *cas objet* du niveau 1 et que le niveau 3, le *cas but* du niveau 2, et si, de plus, le prédicat du niveau 1 est de type *extension*

alors la proposition de niveau 3 exprime le *but* et celle du niveau 2, la *question*

←Règle d'interprétation d'une proposition-question.


```
 & interpreter(#a)
 & ajout_int("))
-------------------------------

   identificateur de regle : 2

   _but(#b)
 & _predicat(#b,#p)
 & _actobj(#p)
 & _cas_obj(#p,#o)
 & _proposition(#o)

 => ajouter(_but(#o))
-------------------------------

   identificateur de regle : 3

   _but(#b)
 & _predicat(#b,#p)
 & _echobt(#p)
 & _cas_ben(#p,#n)
 & _cas_obj(#p,#o)
 & ~_proposition(#o)

 => ajout_int("*but:)
  & ajouter(_declencher(#p))
  & interpreter(#p)
  & ajout_int("()
  & ajouter(_declencher(#n))
  & interpreter(#n)
  & ajout_int(",)
  & ajouter(_declencher(#o))
  & interpreter(#o)
  & ajout_int("))
-------------------------------

   identificateur de regle : 4
```

←Règle d'identification de proposition-but de la forme *faire une demande* ou *effectuer une démarche...* Dans ce cas le but véritable est donné par le prédicat imbriqué nominalisé qui constitue le *cas objet.*

←Règle d'interprétation d'une proposition-but

←Règle de construction d'une

```
   _declencher(x)
 & _pers_sing1(x)

 => ajout_int("locuteur)
------------------------------
```

interprétation
à partir de l'unité **je**

```
   identificateur de regle : 5

   _declencher(x)
 & _obtenir(x)

 => ajout_int("obtenir)
------------------------------
```

←Règle de construction d'une interprétation à partir du concept **obtenir**

```
   identificateur de regle : 6

   _declencher(x)
 & _demander(x)

 => ajout_int("obtenir)
------------------------------
```

←Règle de construction d'une interprétation à partir du concept **demander**

```
   identificateur de regle : 7

   _declencher(x)
 & _carte_d_identite(x)

 => ajout_int("carte_d_identite)
------------------------------
```

←Règle de construction d'une interprétation à partir du concept **carte d'identité**

```
   identificateur de regle : 8

 _declencher(x)
```

←Règle de construction d'une interprétation à partir du concept **double-nationalité**


```
 & _double_nationalite(x)

 => ajout_int("double_nationalite)
------------------------------
```

```
   identificateur de regle : 9

   _declencher(x)
 & _s_adresser(x)

 => ajout_int("s_adresser)
------------------------------
```

←Règle de construction d'une interprétation à partir du concept **s'adresser**

```
   identificateur de regle : 10

   _declencher(x)
 & _pron_interr(x)
 & _anime(x)

 => ajout_int("anime)
------------------------------
```

←Règle de construction d'une interprétation à partir de l'unité **qui**

La saisie de l'énoncé

La première étape du traitement consiste à saisir l'énoncé que nous projetons d'interpréter. Pour cela, nous utilisons l'interface qui permet d'introduire le résultat des niveaux inférieurs sous la forme d'une suite de relations avec leurs variables.

Ces données sont stockées dans la mémoire de travail (MT) ; dans le cas présent la *MT* contient initiallement les relations suivantes :

```
MEMOIRE DE TRAVAIL INITIALE
===========================

_proposition (NIVEAU1)
_predicat (NIVEAU1,savoir)
_cas_pat (savoir,je)
```

←Initiallement, la *MT* renferme une représentation de l'énoncé construite à partir de la représentation syntaxico-sémantique de l'énoncé.

```
_cas_obj (savoir,NIVEAU2)
_proposition (NIVEAU2)
_predicat (NIVEAU2,s_adresser)
_cas_ben (s_adresser,je)
_cas_agt (s_adresser,qui)
_cas_goal (s_adresser,NIVEAU3)
_proposition (NIVEAU3)
_predicat (NIVEAU3,faire)
_cas_agt (faire,je)
_cas_obj (faire,NIVEAU4)
_proposition (NIVEAU4)
_predicat (NIVEAU4,demande)
_cas_ben (demande,je)
_cas_obj (demande,double_nationalite)

-----------------------------
```

Ces prédicats expriment les relations syntaxico-sémantiques et hiérarchiques entre les différentes variables.

Cet exemple correspond à l'enoncé :
Je voudrais savoir à il faut s'adresser pour faire une demande de double-nationalité

Le déroulement

La première règle valide est ici la règle 0 ;

```
cycle : 1

debut d'execution de la regle :

id de regle : 0

substitution :
#x  -> NIVEAU1
#p1  -> savoir
#o1  -> NIVEAU2
#p2  -> s_adresser
#g2  -> NIVEAU3

 ajout de l'element suivant en WM :
_but (NIVEAU3)

 ajout de l'element suivant en WM :
_question (NIVEAU2)
```


Comme nous pouvons le constater sur la trace du déroulement, les actions correspondant à cette règle sont exécutées. Par conséquent, deux nouvelles relations sont ajoutées dans *MT*.

Deux nouvelles règles deviennent alors valides, la règle 1 et la règle 2. La stratégie que nous avons adoptée conduit d'abord au choix de la règle 1.

```
cycle : 2

debut d'execution de la regle :

id de regle : 1

substitution :
#q  -> NIVEAU2
#p  -> s_adresser
#b  -> je
#a  -> qui

 ajout de la partie d'interpretation suivante :
*question:

 ajout de l'element suivant en WM :
_declencher (s_adresser)
```

L'exécution de la première action a permis construire un fragment d'interprétation. Un appel récursif à l'interpréteur est maintenant effectué.

```
execution d'un appel recursif a l'interpreteur.
identificateur de variable concernee : 1
```

Cet appel concerne la variable `s_adresser` et valide la règle 9.

```
cycle : 1

debut d'execution de la regle :

id de regle : 9

substitution :
x  -> s_adresser

ajout de la partie d'interpretation suivante :
s_adresser

plus de regles activables...
```

Cette séquence a permis d'enrichir l'interprétation en rajoutant le concept qui sert de prédicat à la question. L'appel récursif est désormais terminé. Le traitement se poursuit

par l'exécution de la suite des actions de la règle 1.

```
 ajout de la partie d'interpretation suivante :
(
 ajout de l'element suivant en WM :
_declencher (je)

execution d'un appel recursif a l'interpreteur.
identificateur de variable concernee : 2
```

Ensuite, nous assistons successivement à deux nouveaux appels récursifs de l'interpréteur qui vont permettre de compléter l'interprétation de la question. A la fin de ces appels, l'interprétation partielle contient l'information suivante :

```
*question:s_adresser(locuteur,?anime)
```

Le déroulement étant comparable à l'étape précédente, nous passerons tout de suite à l'éxecution de la règle 1. A cet instant, la règle 2, toujours valide, reste encore inexploitée.

```
cycle : 3

debut d'execution de la regle :

 id de regle : 2

substitution :
#b  -> NIVEAU3
#p  -> faire
#o  -> NIVEAU4

 ajout de l'element suivant en WM :
_but (NIVEAU4)  1

fin d'execution de la regle courante
```

Son activation conduit à son tour à la validation de la règle 3.

```
cycle : 4

debut d'execution de la regle :

 id de regle : 3

substitution :
#b  -> NIVEAU4
#p  -> demande
#n  -> je
#o  -> double_nationalite
```



```
 ajout de la partie d'interpretation suivante :
*but:

 ajout de l'element suivant en WM :
_declencher (demande)
```

Les actions de cette règle comportent des adjonctions dans l'interprétation et des appels récursifs à l'interpréteur. Finalement, lorsque plus aucune règle ne pourra être activée, le résultat du traitement est rangé dans la structure d'interprétation qui, au terme de notre exemple contient :

```
RESULTAT
========
 *question:s_adresser(locuteur,?anime)
 *but:obtenir(locuteur,double_nationalite)
```

De ce traitement découle un enrichissement de *MT* qui, en fin de traitement, contient les éléments suivants :

```
MEMOIRE DE TRAVAIL FINALE
=========================
_proposition (NIVEAU1)
_predicat (NIVEAU1,savoir)
_cas_pat (savoir,je)
_cas_obj (savoir,NIVEAU2)
_proposition (NIVEAU2)
_predicat (NIVEAU2,s_adresser)
_cas_ben (s_adresser,je)
_cas_agt (s_adresser,qui)
_cas_goal (s_adresser,NIVEAU3)
_proposition (NIVEAU3)
_predicat (NIVEAU3,faire)
_cas_agt (faire,je)
_cas_obj (faire,NIVEAU4)
_proposition (NIVEAU4)
_predicat (NIVEAU4,demande)
_cas_ben (demande,je)
_cas_obj (demande,double_nationalite)
_but (NIVEAU3)                    ←Nouvelles relations engendrées par
_question (NIVEAU2)               par l'application des règles.
_declencher (s_adresser)
_declencher (je)
```

```
_declencher (qui)
_but (NIVEAU4)
_declencher (demande)
_declencher (je)
_declencher (double_nationalite)
```

2.3 Conclusion

Cet exemple demande à être compléter avec le traitement des références anaphoriques par l'utilisation de l'action *hist-refer*, et avec la prise en compte du contenu de l'historique par l'exécution des actions du type *en-machine*. Ces aspects du traitement sont abordés dans [LORANT 86].

D'autre part, l'interface actuelle demeure encore très rudimentaire. Cela est essentiellement dû a' son caractère provisoire. En effet, l'établissement d'une communication réelle entre *DIAL* et les autres processeurs du système entraine la suppression de cet outil. Dans cette éventualité, il devient indispensable de relier l'interpréteur à l'historique commun aux trois modules du processeur *DIAL*. C'est vers cette version que doit évoluer ce module à court terme.

Par ailleurs, les résultats de ce module dépendent fortement du choix des règles d'interprétation. Or, il nous semble qu'un choix approprié ne peut être effectué qu'en étudiant méthodiquement une masse représentative des résultats fournis par les processeurs de niveaux inférieurs.

Par conséquent, l'élaboration de bases de connaissances plus complètes constitue la perspective à plus long terme de ce module.

3 LE RAISONNEUR

3.1 L'organisation

Le raisonneur est également conçu à la manière d'un système autonome possédant ses propres outils de mises à jour des connaissances et de saisie de données [ROUSSANALY 87].

Comme nous pouvons l'observer sur le schéma de la figure 5.3, les principales composantes de ce module sont :

- **cpd :** c'est un compilateur qui traduit un ensemble de fichiers contenant les connaissances exprimées dans une représentation externe, en une représentation interne utilisable par le raisonneur. *cpd* vérifie non seulement la validité syntaxique des connaissances, mais aussi leur cohérence à l'aide de techniques de vérification sémantique [AHO 72]. Nous trouvons une description plus détaillée de cet outil dans [MOUSEL 86a].

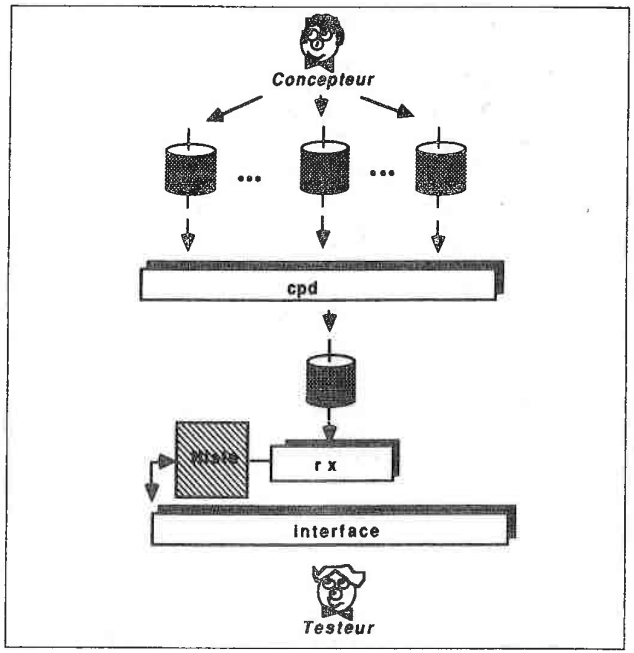


Figure 5.3: Les composantes du raisonneur.

Les fichiers en entrée contiennent :

- **la définition des concepts :** il s'agit de constituer l'ensemble des concepts modèles d'une application et de définir en extension la relation de hiérarchie *isa* et la relation de contraintes sémantiques *arg*.

  Exemple :

```
relation
[relation_physique(objet,etat)
[couleur, forme],
relation_affective(anime,anime)
[aimer,hair]]
```

Cette écriture possède l'avantage d'être compacte. Ce court exemple permet

d'introduire toutes les relations suivantes :

$isa(relation\ physique, relation)$ → Relations *isa* existant explicitement
$isa(relation\ affective, relation)$ dans les expressions de l'entrée.
$isa(couleur, relation\ physique)$
$isa(forme, relation\ physique)$
$isa(aimer, relation\ affective)$
$isa(hair, relation\ affective)$
$isa(couleur, relation)$ → Relations obtenues par un mécanisme
$isa(forme, relation)$ d'héritage de *isa*.
$isa(aimer, relation)$
$isa(hair, relation)$
$arg(relation) = \emptyset$ → Fonctions apparaissant dans les
$arg(relation\ physique) = (objet, etat)$ expressions de l'entrée.
$arg(relation\ affective) = (anime, anime)$
$arg(couleur) = (objet, etat)$ → Fonctions obtenues par héritage
$arg(forme) = (objet, etat)$ sur arg
$arg(aimer) = (anime, anime)$
$arg(hair) = (anime, anime)$

- **la description de la base de données :** cette base renferme toutes les informations statiques de type factuel. Nous les écrivons sous la forme d'une suite de formules de la manière suivante :
  Exemple :

```
prix(carte_ d_identite,"F115");              → 1
prix(carte_de_sejour,"F0");                  → 2
telephone(mairie,"83-54-14-23");             → 3
adresse(mairie,"rue de Parme. Vandoeuvre");  → 4
```

Ces expressions représentent les connaissances factuelles de la tâche. Nous pouvons y rajouter les informations que nous désirons valider par défaut.
Exemple :

```
nationalite(locuteur,France)*  → 5
```

Les informations par défaut se différencient des données certaines par l'apparition d'un caractère spécial, *, en fin d'expression.

Remarque : Les noeuds représentés à l'aide d'une chaîne de caractères entre deux double-quotes désignent une classe particulière de concepts, notée `VAL`. Ils n'apparaissent pas dans la définition des concepts et sont unifiés avec des concepts de même classe ayant la même orthographe.

- **la description de la base de renseignements :** cette base est décrite à l'aide d'une suite de renseignements formés d'une partie *but* et d'une partie *renseignement conditionnel*. Nous voyons apparaître ces deux constituants sur l'exemple ci-dessous :


```
but: obtenir(personne1,carte_d_identite) ← partie1

  si: nationalite(personne1,france)[2] ← partie2
  alors:
    &(aller(personne1,|(mairie,commissariat)),
      munir(personne1,|(livret_de_famille,fiche_d_etat_civil)),
      fournir(personne1,&(timbre,formulaire))
      prix(timbre,"F115"))
    si: mineur(personne1)[1]
    alors: &(accompagner(personne1,personne2),majeur(personne2))
    sinon:
  sinon:
 avec: personne1#personne2 ← diff(personne1,personne2)
;
```

Cet exemple est la traduction du renseignement suivant :

Pour obtenir une carte d'identité, si l'on est de nationalité française, il faut aller, soit à la mairie, soit au commissariat, muni du livret de famille ou de la fiche d'état civil et il faut également fournir un timbre et un formulaire.

Pour un mineur, il faut être accompagné d'une personne majeure.

Le champ *but* est décrit à l'aide d'une formule précédée du mot-clef `but:` tandis que le champ *renseignement* possède une structure enchâssée formée de *conditions*.

Remarque 1 : Au niveau du *but* et des *actions* (précédées de `alors:` ou `sinon`), nous autorisons des formules quelconques. De plus, les *tests* peuvent être des formules conjonctives. Un des rôles de *cpd* est d'effectuer les transformations afin d'aboutir à une forme normalisée exploitable par le raisonneur.

Remarque 2 : A la fin de la description, il est possible d'ajouter une clause indiquant des contraintes d'exclusion mutuelle entre différents noeuds se référant à un même concept. Cette clause, annoncé par le mot-clef `avec:`, enrichit la relation $diff$ et constitue, comme nous l'avons montré au chapitre précédent, une contrainte supplémentaire pour l'unification.

Remarque 3 : De manière optionnelle, une indirection vers une procédure peut être associée à chaque test ; elle est décrite à l'aide d'un nombre entre crochets (`[1]`). La procédure en question est activée lorsque le gestionnaire de dialogue désire poser une question dans le but de valider ce test. Ce dispo-

sitif permet de prévoir des procédures qui émettent des questions en langage naturelle et qui effectuent éventuellement des prédictions sur la réponse.

- **la description de la base de règles du raisonneur :** cette base renferme les règles de déduction utilisées par le raisonneur. Elles sont décrites à l'aide d'une suite d'expressions composées d'une *pré-condition* et d'une *post-condition*.

  Exemple :

```
prix(document,"FO") => gratuit(document);          ← 1
majeur(personne) => -mineur(personne);             ← 2
nationalite(personne,etranger)                     ← 3
 => -nationalite(personne,france);
pere(personne1,personne2) => masculin(personne1)   ← 4
      avec: personne1#personne2;
```

  Les formules apparaissant en *post-condition* sont, tout au plus, conjonctives tandis que celles de la *pré-condition* peuvent être quelconques. Le compilateur se charge alors de normaliser la représentation.

  D'autre part, les clauses $diff$ sont également acceptées (voir règle 4).

- **rx :** il s'agit du module central de traitement, relié à l'interface par l'intermédiaire de l'historique.

  En effet, l'interface permet d'une part, d'insérer dans l'historique, une simulation des résultats de l'interpréteur en cours de dialogue et, d'autre part, de tracer à chaque cycle, l'évolution de l'état de la tâche et de certaines structures importantes de l'historique.

  Il faut cependant noter qu'une version du gestionnaire du dialogue opère également sur l'historique et propose une réponse à l'utilisateur. Par conséquent, pour effectuer les tests sur le raisonneur, il n'est pas nécessaire de simuler les actions du gestionnaire de dialogue.

3.2 Exemples de déduction

Pour les exemples que nous allons développer dans la suite de cette présentation, nous considérons que la base de données BD, la base de renseignements BX et la base des règles BR ont été constituées à partir des connaissances qui nous ont précédemment servi à les illustrer. En conséquence, nous construisons un ensemble minimal de concepts modèles qui autorise leur utilisation. Cet ensemble est exprimé de la manière suivante :


```
piece[
   formulaire,
   timbre,
   document[
     carte_d_identite,
     carte_de_sejour,
     fiche_d_etat_civil,
     livret_de_famille]],

lieu[
  administration[mairie,commissariat],
  nation[france,etranger]],

personne[
  locuteur,tiers,machine,
  administration],

mesure(piece)[prix(VAL),gratuit],

relation(personne,personne)[pere],

etat(personne)[
  majeur,
  mineur,
  masculin,
  nationalite(nation)],

coordonnee(administration)[telephone(VAL),adresse(VAL)],

formalite(personne)[
  aller(administration),
  munir(document),
  fournir(piece),
  accompagner(personne)],

but(personne)[obtenir(document)]
```

Exemple 1

Afin d'activer la regle 2, nous introduisons dans l'historique le fait suivant :

(loc)-> *majeur(locuteur)*.

A la fin du cycle, l'interface fournit alors l'état des principales structures :

ETAT DES INFO DYNAMIQUES	
Ensemble de questions:	vide
Ensemble de buts:	vide
Base des faits:	(6) [sc=100] .majeur[63](locuteur[64]) (7) [sc=100] .-mineur[65](locuteur[64])

Chaque élément de la base des faits apparaît sur cette trace avec un numéro d'index dans cette base, son score (par défaut = 100) et la formule élémentaire correspondante. A chaque noeud de cette formule, est associée une référence à son représentant principal dans l'ensemble des concepts objets.

Dans cet exemple, nous pouvons remarquer que l'application de la règle 2 a amené la déduction du fait (7).

Exemple 2

Cet exemple comporte plusieurs cycles. Dans un premier temps, nous saisissons le fait suivant :

(loc)->*pere(tiers1,tiers2).*

Nous obtenons alors la trace suivante :

ETAT DES INFO DYNAMIQUES	
Ensemble de questions:	vide
Ensemble de buts:	vide
Base des faits:	(6) [sc=100] .pere[63](tiers[64],tiers[65]) (7) [sc=100] .masculin[66](tiers[64])

Le fait (7) résulte de l'application de règle 4.

Puis, nous rangeons successivement les faits suivants dans l'historique :

(loc)->*mineur([64])*

(loc)->*majeur([65]).*

Comme nous pouvons le remarquer, il est possible, au niveau de l'interface, de désigner un noeud par sa référence dans l'historique.

La trace correspondante est reproduite ci-dessous :

ETAT DES INFO DYNAMIQUES	
Ensemble de questions:	vide
Ensemble de buts:	vide
Base des faits:	(6) [sc=100] .pere[63](tiers[64],tiers[65]) (7) [sc=100] .masculin[66](tiers[64]) (8) [sc=100] .mineur[68](tiers[64]) (9) [sc=100] .majeur[69](tiers[65]) (10) [sc=100] .-mineur[70](tiers[65])

Dans ce cas, la règle 2 permet de déduire le fait (10) qui, en raison de la relation d'exclusion entre les noeuds [64] et [65] provenant de la règle 4, n'est pas en contradiction avec le fait (8). Cela se traduit par une absence d'incohérence dans l'état de la tâche suivant fourni par le raisonneur :

ETAT DE LA TACHE	
INCOHERENCE:	**ras
QUESTION:	**Pas de question
BUT:	**but non necessaire pour la reponse
REPONSE:	** pas de reponse:
DEFAUT:	**pas de valeur par defaut

L'état de la tâche regroupe, de manière synthétique, les principales conclusions du raisonneur. L'interface les classe en cinq rubriques :

- **Incohérence :** indique si incohérence a été détectée ou non. Dans le cas où elle l'est, les faits incriminés sont signalés ;
- **Question :** fournit la question active courante s'il en existe ;
- **But :** fournit le but actif courant s'il existe et s'il est nécessaire ;
- **Réponse :** indique l'état de la réponse lorsque que celle-ci existe ;
- **Défaut :** liste les faits par défaut utilisés dans le raisonnement.

3.3 Exemples d'incohérences

Exemple 1

Etudions maintenant le cas d'une incohérence. Pour cela, nous introduisons tour à tour les faits suivants :

(loc)->*mineur(locuteur).*

(loc)->*majeur(locuteur).*

Les états respectifs de l'historique et de la tâche sont alors :

ETAT DES INFO DYNAMIQUES	
Ensemble de questions:	vide
Ensemble de buts:	vide
Base des faits:	(6) [sc=100] .mineur[63](locuteur[64]) (7) [sc=100] .majeur[65](locuteur[66]) (8) [sc=100] .-mineur[67](locuteur[66])

ETAT DE LA TACHE	
INCOHERENCE:	**incoherence entre: .mineur[63](locuteur[64]) .-mineur[67](locuteur[66])
QUESTION:	**Pas de question
BUT:	**but non necessaire pour la reponse
REPONSE:	** pas de reponse:
DEFAUT:	**pas de valeur par defaut

Comme nous pouvons en juger par ces résultats, les faits (6) et (8) sont en contradiction.

Exemple 2

Reprenons le même exemple en imposant cette fois-ci un score à chaque fait ;

(loc)->*mineur(locuteur)[%10].*

(loc)->*majeur(locuteur)[%90].*

Au terme des traitements, nous obtenons les traces suivantes :

ETAT DES INFO DYNAMIQUES	
Ensemble de questions:	vide
Ensemble de buts:	vide
Base des faits:	(7) [sc= 90] .majeur[65](locuteur[66]) (8) [sc= 90] .-mineur[67](locuteur[66])

ETAT DE LA TACHE	
INCOHERENCE:	**ras
QUESTION:	**Pas de question
BUT:	**but non necessaire pour la reponse
REPONSE:	** pas de reponse:
DEFAUT:	**pas de valeur par defaut

Dans ce cas, il n'y a plus d'incohérence. En réalité, la contradiction entre les faits (6) et (8) a bien été détectée. Mais le raisonneur a pris la décision de supprimer le fait (6) en raison de son faible score par rapport celui du fait (8).

Observons, au passage, le score du fait déduit 8 calculé à partir de celui de (7). La combinaison des scores que nous avons utilisé consiste à effectuer la moyenne des scores des préconditions.

3.4 Evolution de l'état de la tâche

Après cette première partie d'exemples axés sur la base des faits, examinons maintenant l'évolution des autres rubriques de l'état de la tâche.

Exemple 1

Commençons par déterminer un but :

(loc)->*but:obtenir(locuteur,carte-d-identite).*

ETAT DES INFO DYNAMIQUES	
Ensemble de questions:	vide
Ensemble de buts:	(0) [sc=100] .obtenir[63](locuteur[64],carte_d_identite[65])
Base des faits:	vide

ETAT DE LA TACHE	
INCOHERENCE:	**ras
QUESTION:	**Pas de question
BUT:	**but actif: .obtenir[32](personne[33],carte_d_identite[34])
REPONSE:	** pas de reponse:
DEFAUT:	**pas de valeur par defaut

La requête n'est pas complète car il manque une question (voir rubrique *Question*). Introduisons-en une dans l'historique à l'aide de l'interface.

(loc)->*?formalite(locuteur).*

Il s'agit d'une question de *type* ouvert dont l'*objet* est le concept *formalité* caractérisé par la présence du symbole ?.

Les différents états évoluent en conséquence :

ETAT DES INFO DYNAMIQUES	
Ensemble de questions:	[o](0) [sc=100] .formalite[66](locuteur[67])
Ensemble de buts:	(0) [sc=100] .obtenir[63](locuteur[64],carte_d_identite[65])
Base des faits:	vide

ETAT DE LA TACHE	
INCOHERENCE:	**ras
QUESTION:	**question ouverte: .formalite[66](locuteur[67]) **objet de la question: .formalite[66](locuteur[67])
BUT:	**but actif: .obtenir[32](personne[33],carte_d_identite[34])
REPONSE:	**reponse conditionnelle:
DEFAUT:	defaut: .nationalite[35](personne[33],france[36])

La détermination de la question a engendré la possibilité d'une réponse qui reste toutefois conditionnelle. Cela signifie que les informations apportées par le locuteur sont encore insuffisantes, et cela malgré l'hypothèse par défaut faite par le raisonneur et signalée dans la rubrique correspondante.

L'information manquante ici, pour répondre avec précision à la requête, est dûe à l'indéterminisme de la validité du fait *mineur(locuteur)*.

La saisie du fait suivant permet de compléter définitivement la requête :

(loc)->*-mineur(locuteur)*

Voici l'état final de la tâche concernant cette requête :

ETAT DE LA TACHE	
INCOHERENCE:	**ras
QUESTION:	**question ouverte: .formalite[66](locuteur[67]) **objet de la question: .formalite[66](locuteur[67])
BUT:	**but actif: .obtenir[32](personne[33],carte_d_identite[34])
REPONSE:	**reponse complete dans BX:
DEFAUT:	defaut: .nationalite[35](personne[33],france[36])

Exemple 2

Dans cet exemple, nous allons inverser l'ordre du but et de la question.

(loc)->*?formalite(locuteur)*.

L'état de la tâche correspondant est alors le suivant :

ETAT DE LA TACHE	
INCOHERENCE:	**ras
QUESTION:	**question ouverte: .formalite[63](locuteur[64]) **objet de la question: .formalite[63](locuteur[64])
BUT:	**il manque un but:
REPONSE:	** pas de reponse:
DEFAUT:	**pas de valeur par defaut

La présence d'un but n'est pas toujours obligatoire pour rendre une requête complète. En effet, dans l'éventualité d'une question à la base de donnée, celui-ci n'est plus nécessaire. La rubrique *Réponse* indique alors que la réponse est complète dans *BD*.

Or ce n'est pas le cas de notre exemple. Aucune réponse n'a pu être déterminée dans *BD* ; nous concluons donc à l'absence d'un but (voir rubrique *But*).

Nous ne développerons pas davantage cet exemple car si nous introduisons ensuite, le même but que dans l'exemple précédent, nous retomberons dans un même état et la suite de l'état de la tâche évoluera de manière identique.

Exemple 3

Examinons, enfin, un exemple qui comporte un but inexistant dans la description de la tâche.

(loc)->*but:obtenir(locuteur,carte-de-sejour).*

Sans pour autant être incohérent, ce but ne peut pas être satisfait car il ne correspond à aucun renseignement dans *BX*. L'état de la tâche est donc le suivant :

ETAT DE LA TACHE	
INCOHERENCE:	**ras
QUESTION:	**Pas de question
BUT:	**but inconnu: .obtenir[63](locuteur[64],carte_de_sejour[65])
REPONSE:	** pas de reponse:
DEFAUT:	**pas de valeur par defaut

Il s'agit du cas où la demande dépasse les compétences du système.

3.5 Conclusion

Dans l'état actuel des travaux, pour tester le raisonneur, nous avons créé les différentes bases de connaissances à l'aide de *cpd*. La figure 5.4 donne une idée de leur taille actuelle.

Avec ces connaissances, le traitement s'effectue en temps réel. Elles restent cependant

	Nombre d'éléments
Concepts modèles	environ 200
Base de données	environ 20
Base de renseignements	environ 5
Base de règles	environ 25

Figure 5.4: Estimation de la taille des connaissances.

incomplètes pour rendre le système opérationnel.

Par conséquent, un des axes de développement futur de ce module concerne l'enrichissement des bases de connaissances qu'il utilise.

D'autre part, dans notre étude, l'interprétation que nous faisons des scores communiqués par les autres modules reste assez superficielle. Il nous semble qu'une étude plus fondamentale est à envisager dans ce domaine.

En définitive, le raisonneur est une composante importante du module *DIAL* car la stratégie de gestion du dialogue repose en grande partie sur les résultats qu'il fournit.

4 LE GESTIONNAIRE DE DIALOGUE

4.1 L'organisation

La figure 5.5 illustre l'organisation de la première version du gestionnaire de dialogue. Examinons de plus près les principales composantes de ce module :

- **gd et bib :** il s'agit d'une version entièrement procédurale ; cela signifie que le module **gd** doit être un programme écrit en C. Précisons tout de même que le concepteur dispose, à cet effet, d'une bibliothèque (**bib**) de fonctions qui lui en facilite la construction.

  Ces fonctions effectuent des traitements élémentaires intervenant dans la gestion du dialogue tels que :

  - la mise à jour et les tests dans l'historique -ensemble des interventions et état de la tâche ;
  - l'acquisition d'un énoncé du locuteur ;
  - l'émission d'une réponse ;
  - l'envoi d'une prédiction vers les autres processeurs.

  Pour parvenir à la version test actuelle de ce module, nous avons procédé à une traduction systématique des schémas que nous avons définis pour notre modèle de

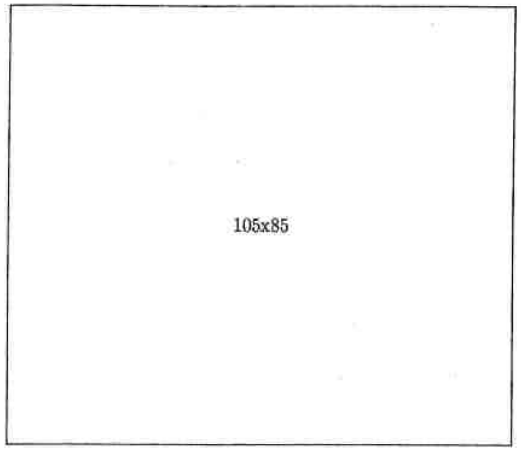


Figure 5.5: Les composantes du gestionnaire de dialogue

dialogue. Cette méthode consiste à faire correspondre une fonction du langage C à un schéma du modèle. Notre objectif est, à terme, d'engendrer automatiquement *gd* à partir d'un modèle de dialogue exprimé à l'aide de schémas.

- **l'interface et gen :** les entrées acceptées par l'interface sont identiques à celles du raisonneur. La différence se situe au niveau de la trace qu'elle fournit car elle intègre -de manière encore assez sommaire- certaines fonctionalités du module de génération que nous n'abordons pas dans notre étude. Pour cela, elle dispose de connaissances représentées par **gen** qui contient :

  - **un ensemble de formules figées** correspondant à des situations bien précises de dialogue.

    Exemple :

    (1) `Renseignements administratifs, je vous ecoute.`

    (2) `$1 est un renseignement que nous ne possedons pas encore, desole.`

    L'énoncé de l'exemple (1) sert à établir la communication et celui de l'exemple (2) permet d'informer le locuteur d'un cas de dépassement des compétences du système.

    Comme nous pouvons le noter dans le second exemple, nous offrons la possibilité de laisser des *trous* dans les énoncés -symbolisés par `$1`- qui peuvent être

instanciés au moment de l'énonciation.

- **un ensemble de procédures** : il s'agit des procédures associées aux conditions d'un renseignement que nous avons déjà signalé plus haut. Rappelons que ces procédures permettent, à la fois de poser une question visant à valider une condition et d'émettre des prédictions sur la réponse.

4.2 Exemple 1

Pour construire les exemples qui vont suivre, nous nous sommes servis de l'application précédente du raisonneur.

Le dialogue théorique correspondant au premier exemple est le suivant :

S1 : *Renseignements administratifs, je vous écoute*

U1 : *Allo, bonjour*

S2 : *Bonjour, ici le centre expérimental de renseignements administratifs. Nous pouvons fournir des renseignements concernant les démarches administratives les plus courantes. Quelle est votre requête?*

U2 : *Voilà, je voudrais savoir quelles sont les formalités pour l'obtention d'une carte d'identité*

S3 : *Etes-vous majeur?*

U3 : *Je suis [mineur]* (← reconnaissance défectueuse)

S4 : *Si vous êtes de nationalité française, vous devez aller à la mairie ou au commissariat, vous munir de votre livret de famille ou d'une fiche d'état civil, fournir un timbre fiscal. Vous devez de plus être accompagné de vos parents.*

U4 : *Mais je suis majeur*

S5 : *Veuillez m'excuser mais est-ce que vous êtes mineur ou non?*

U5 : *Non*

S6 : *Si vous êtes de nationalité française, vous devez aller à la mairie ou au commissariat, vous munir de votre livret de famille ou d'une fiche d'état civil, fournir un timbre fiscal.*
Un autre renseignement?

U6 : *Oui*

S7 : *Continuez svp*

U7 : *Où se trouve la mairie?*

S8 : *Rue de Parme à Vandoeuvre*
Désirez-vous autre chose?

U8 : *Et quel est le numéro de téléphone?*

S9 : *83-54-14-23*
Désirez-vous autre chose?

U9 : *Non merci, au revoir*

S10 : *Au revoir*

Nous reproduisons ci-dessous la trace correspondant à l'intervention *S1* délivrée par l'interface :

```
(syst)->Renseignements administratifs, je vous ecoute.

|        MESSAGE VERS LEX          |
-------------------------------------
 (salutations|requete)
-------------------------------------
|        MESSAGE VERS SYN-SEM      |
-------------------------------------
 (premier_enonce)
-------------------------------------
```

Parallèlement à l'émission du message de prise de contact, des prédictions vers les processeurs LEX et SYN-SEM sont envoyées. Ces prédictions apparaissent ici à titre indicatif car le module *gd* est testé indépendamment des autres processeurs.

L'échange *U1-S2* se déroule comme suit :

```
(loc)->bonjour:.

(syst)->Bonjour
(syst)->Ici le centre experimental de renseignements administratifs,
nous pouvons fournir des renseignements concernant les demarches
administratives les plus courantes, quelle est votre requete?
```

La fonction d'ouverture de dialogue est introduite ici par le terme *bonjour:*. Au niveau de l'interface, nous étendrons cette notation à toutes les fonctions des énoncés portant sur le dialogue. D'autre part chaque fonction sera introduite sur une ligne et la fin d'une intervention sera signalée par le caractère . -point-.

Puisque le dialogue reste dans la phase introductive, les prédictions restent inchangées.

L'intervention suivante permet de passer à la phase de requête.

```
(loc)-> but: obtenir(locuteur,carte-d-identite)
(loc)-> ?formalite(locuteur).
```

Notation : Voici les conventions de notation concernant les énoncés portant sur la tâche :

- un but est précédé du terme *but:*,
- une question ouverte est représentée par une formule, l'objet de la question est précédé du caractère ?,
- une question fermée est également une formule suivie du caractère ?,
- et enfin, un fait est une formule simple.

Si nous nous référons à la base de renseignements, la réponse à cette requête dépend de deux conditions :

1. `nationalite(locuteur,france)`
2. et `mineur(locuteur)`

La première est vérifiée par défaut ; ce qui explique la question posée par le système :
`(syst)->Etes-vous majeur?`

```
|        MESSAGE VERS LEX          |
-------------------------------------
 (confirmation|age)
-------------------------------------
|        MESSAGE VERS SYN-SEM       |
-------------------------------------
 (confirmation|mesure1)
-------------------------------------
```

Cette question fermée est engendrée par une procédure associée à la condition à valider. Dans ce cas les prédictions sont plus précises.

Une reconnaissance défectueuse est simulée dans l'interprétation de l'intervention $U3$:

`(loc)->`*mineur(locuteur).*

La réponse du système tient compte de cette information :


```
(syst)->Si nationalite(personne,france)
est vrai alors
(syst)->La reponse est:
ou
    et
        accompagner(personne,personne)
        aller(personne,mairie)
        munir(personne,livret_de_famille)
        fournir(personne,formulaire)
        fournir(personne,timbre)
   )
    et
        accompagner(personne,personne)
        aller(personne,mairie)
        munir(personne,fiche_d_etat_civil)
        fournir(personne,formulaire)
        fournir(personne,timbre)
   )
    et
        accompagner(personne,personne)
        aller(personne,commissariat)
        munir(personne,fiche_d_etat_civil)
        fournir(personne,formulaire)
        fournir(personne,timbre)
   )
    et
        accompagner(personne,personne)
        aller(personne,commissariat)
        munir(personne,livret_de_famille)
        fournir(personne,formulaire)
        fournir(personne,timbre)
   )
)
```

Il s'agit d'une représentation canonique de la réponse utilisée par le raisonneur. Pour parvenir à une formulation en langage naturel, il est indispensable de faire appel à un module de génération.

L'intervention $U4$ apporte une information qui contredit les hypothèses de la base des faits :

`(loc)->`*majeur(locuteur).*

La stratégie consiste ici à faire confirmer une des informations :

```
(syst)->Est-ce que mineur(locuteur) est vrai?
      |        MESSAGE VERS LEX          |
      -------------------------------------
       (confirmation|infirmation)
      -------------------------------------
      |        MESSAGE VERS SYN-SEM       |
      -------------------------------------
       (apport_information)
      -------------------------------------
```

(loc)->*non.*

Nous pouvons remarquer qu'une réponse de confirmation ou d'infirmation peut être directement interprétée par l'interface sans passer par l'interpréteur.

La requête précédente reste valide, mais la réponse tient compte cette fois-ci de la modification de la base des faits.

```
(syst)->valeurs par defaut utilisees:
nationalite(personne,france)

(syst)->La reponse est:
ou
    et
        fournir(personne,formulaire)
        fournir(personne,timbre)
        aller(personne,commissariat)
        munir(personne,fiche_d_etat_civil)
   ) ...
(syst)->Un autre renseignement?
```

A la suite de la satisfaction de la requête, le système enchaîne sur une relance.

L'utilisateur confirme son intention de demander un autre renseignement ; le système se met alors en attente d'une nouvelle requête. Cela se traduit par l'échange suivant :

```
(loc)-> oui.
(syst)->Continuez svp.
```

Deux nouveaux échanges correspondant à deux couples de questions/réponses, viennent ensuite dans le dialogue -(*U7*,*S8*) et (*U8*,*S9*)- :

```
(loc)->adresse(mairie,?$x).

(syst)->La reponse est:
"rue de Parme. Vandoeuvre"
(syst)->Desirez-vous autre chose?

(loc)->telephone(mairie,?$y).

(syst)->La reponse est:
"83-54-14-23"
(syst)->Desirez-vous autre chose?
```


A la suite de la satisfaction de la dernière requête, le dialogue passe dans la phase de clôture qui se termine par l'arrêt du système :

```
(loc)->non
(loc)->au-revoir.
(loc)->.

(syst)->Au revoir
```

Cet exemple permet de montrer l'alternance du contrôle du le dialogue entre les deux intervenants. Nous pouvons, en effet, attribuer l'initiative du dialogue à la machine dans les interventions *S2*, *S3*, *S5* et dans les énoncés de relance, tandis que le locuteur prend le contrôle dans les interventions *U4*, *U7*, *U8* et *U9*.

La structure de ce dialogue est fournie à la figure 5.6.

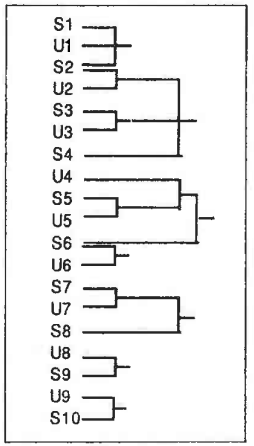


Figure 5.6: Structure du dialogue de l'exemple 1.

4.3 Exemple 2

Commençons à nouveau par reconstituer le dialogue théorique correspondant à ce nouvel exemple :

S1 : *Renseignements administratifs, je vous écoute*

U1 : *Bonjour, je voudrais savoir si la carte de séjour est gratuite?*

S2 : *Oui, elle est gratuite. Désirez-vous autre chose?*

U2 : *Pouvez-vous répéter svp?*

S3 : *Oui, elle est gratuite. Désirez-vous autre chose?*

U3 : *Quelles sont les formalités pour l'obtenir?*

S4 : *Nous ne pouvons pas satisfaire votre demande, elle dépasse notre compétence*

U5 : *Tant pis. Merci et au revoir*

S1 : *Au revoir*

Le dialogue avec l'interface est le suivant :

```
(syst)->Renseignements administratifs, je vous ecoute.

(loc)->bonjour:
(loc)->gratuit(carte-de-sejour)?.

(syst)->La reponse est:  OUI
(syst)->Desirez-vous autre chose?

(loc)->repeter:.

(syst)->La reponse est: OUI
Desirez-vous autre chose?

(loc)->but:obtenir(locuteur,carte-de-sejour)
(loc)->?formalite(locuteur).

(syst)->Nous ne pouvons satisfaire votre demande, elle depasse nos
competences.

(loc)->au-revoir:.

(syst)->Au revoir
```

Cet exemple fait apparaître :

- une question fermée dans l'intervention $U1$. Nous pouvons remarquer que la réponse positive est obtenue à la suite d'une déduction à partir d'une information de la base de données ;
- un énoncé portant sur dialogue -$U2$-. Il s'agit d'une demande de répétition de la part du locuteur ;
- un état de dépassement des limites de la connaissance du système $S4$ qui se manifeste par la non-satisfaction de la requête.

4.4 Conclusion

Le module de gestion de dialogue que nous avons présenté est une première version qui a pour objectif de recenser les besoins nécessaires à une implantation d'une stratégie de dialogue de plus en plus efficace. Les perspectives de développement de ce module portent sur plusieurs axes :

- la mise en oeuvre de stratégies de dialogue face à de situations que nous n'avons pas encore traitées telles que la contestation ;
- la définition précises des prédictions vers les autres processeurs à chaque étape du dialogue ;
- l'écriture d'outils permettant le passage automatique d'un modèle de dialogue au module de traitement *gd* ;
- l'élaboration d'un module de génération des messages en langage naturel.

CONCLUSION

Notre objectif a été l'étude d'une composante dialogue dans un système de communication finalisée homme-machine.

Les premiers chapitres de ce manuscrit nous ont permis de replacer cette étude dans un contexte plus général et de souligner l'importance de la composante dialogue dans un tel système.

Nous avons ensuite présenté le projet dans lequel s'inscrit ce travail en évoquant tour à tour les différentes composantes qui interviennent dans l'architecture de notre système et qui font parallèlement l'objet d'une réalisation.

Nous nous sommes enfin attachés plus particulièrement à la composante dialogue qui représente l'essentiel de notre contribution à la réalisation de ce système.

Cette étude a débouché sur la réalisation d'un processeur *DIAL* composé de trois principaux modules qui remplissent des fonctions bien précises au sein du processus de compréhension de dialogue. Il s'agit :

- du module d'interprétation contextuelle, *INTERPRETEUR*,
- du module de raisonnement, *RAISONNEUR*,
- du module de gestion du dialogue, *GESTIONNAIRE*.

Ces trois trois modules communiquent par l'intermédiaire d'une structure hétérogène : l'*HISTORIQUE*.

Cette solution présente l'avantage de s'intégrer plus facilement dans l'architecture générale du système et de respecter les critères de conception que nous nous sommes définis au départ, à savoir :

- *DIAL* est un processeur autonome susceptible de communiquer avec les autres processeurs,
- *DIAL* est un système à bases de connaissances.

Comme nous le signalons dans le dernier chapitre de ce manuscrit, de nombreux problèmes restent encore ouverts. Cependant, ce projet n'est pas encore arrivé à son terme.

En effet, l'inter-connection des différents processeurs, qui constitue le prochain stade de ce projet, va permettre d'effectuer une première évaluation du système. Mais il nous semble d'ores et déjà indispensable de développer des études plus fondamentales impliquant des linguistes, des psychologues et des ergonomes, afin d'accroître les connaissances du système et d'établir ses limites.

D'autre part, si ce travail s'est déroulé dans la perspective d'une communication homme-machine en langage naturel avec des applications touchant une population étendue, un certain nombre de concepts développés au cours de cette étude peuvent être exploités pour la réalisation de systèmes opérationnels utilisant un langage artificiel dans le cadre d'applications plus restreintes [MORIN 87].

Bibliographie

[AHO 72] V.A. AHO et J.D. ULLMAN. *The Theory of Parsing. Translation and Compiling.* Prentice Hall, Englewood Cliffs, New Jersey, 1972.

[ALINAT 87] P. ALINAT, E. GALLAIS, J.P. HATON, J.M. PIERREL, et P. RICHARD. A continuous speech dialog system for oral control of sonar console. *Proceedings of IEEE ICASSP-87*, 1987.

[AMALBERTI 84] R. AMALBERTI, N. CARBONNEL, et P. FALZON. Stratégie de contrôle du dialogue en situation d'interrogation téléphonique. *Actes du séminaire GRECO sur le dialogue oral homme-machine*, pages –, Nancy, 1984.

[AMALBERTI 86] R. AMALBERTI, N. CARBONELL, P. FALZON, et C. JOLIVET. La communication orale homme-homme : un modèle de référence pour la communication orale homme-machine? Rapport d'ATP, ARI communication, 1986.

[AN 81] BELLEC A. NOUHEN et J. SIROUX. Cadi: constructeur automatique de dialogue intelligent integré à un système de reconnaissance et de compréhension de la parole. Thèse de 3ème cycle, Rennes 1, 1981.

[ANGELETTI 84] T. ANGELETTI et B. BEYSSON. Receuil d'un groupe de questions écrit dans le cadre d'un projet d'interrogation de bases de données textuelles en langage naturel. *Actes du séminaire GRECO sur le dialogue oral homme-machine*, pages 258–279, Nancy, 1984.

[AUCHLIN 80] A. AUCHLIN et A. ZENONE. Conversations, actions, actes de langage; éléments d'un système d'analyse. *Cahiers de la linguistique française*, (1):6–41, 1980.

[AUSTIN 70] J.L. AUSTIN. *Quand dire c'est faire.* Edition du seuil, Paris, 1970.

[BBN 76] BBN. *Speech Understanding System.* Final report no. 3438, BBN, vol.1–4, Cambridge, Massasuchetts, 1976.

[BENNANI 83] M. BENNANI et J.M. PIERREL. Adjonction d'entrées-sorties vocales à un système d'EAO. *Le journal de la formation continue et de l'EAO*, (176):4–10, 1983.

[BOBROW 77a] D.G. BOBROW, R.M. KAPLAN, M. KAY, D.A. NORMAN, H. THOMPSON, et T. WINOGRAD. GUS: a frame driven dialog system. *Artificial Intelligence*, 8-2:155–173, 1977.

[BOBROW 77b] D.G. BOBROW et T. WINOGRAD. An overview of KRL, a knowledge representation language. 1977.

[BONIN 87] J.J. BONIN. Détection d'indices prosodiques linguistiquement significatifs. Rapport de DEA, Nancy 1, 1987.

[BOYER 87] A. BOYER. Application des techniques de programmation dynamique et de quantification vectorielle à la reconnaissance des mots isolés et des mots enchainés. Thèse d'université de Nancy 1, 1987.

[BRUCE 75] B. BRUCE. Cases systems for natural language. *Artificial Intelligence*, 6:327–360, 1975.

[BUNT 81] H. BUNT. *Conversational Principles in Question-Answer Dialogues in Zur Theorie der Frage.* Krallman et. Stickel ed., Gunter Narr Verlag, 1981.

[CAELEN 79] J. CAELEN. Un modèle d'oreille de la parole continue, reconnaissance phonémique. Thèse d'état, université de Toulouse, 1979.

[CARBONELL 84] N. CARBONELL, F. CHARPILLET, J.P. HATON, B. MANGEOL, P. MOUSEL, J.M. PIERREL, et A. ROUSSANALY. Dialogue oral homme-machine : bilan du projet MYRTILLE et perspectives. *Actes du séminaire GRECO sur le dialogue oral homme-machine*, pages 91–122, Nancy, 1984.

[CHARPILLET 85] F. CHARPILLET. Un système de reconnaissance de la parole continue pour la saisie de textes lus. Thèse de doctorat de l'université de Nancy 1, 1985.

[CHOMSKY 71] N. CHOMSKY. *Aspects de la théorie syntaxique.* Editions du Seuil, Paris, 1971.

[COLMERAUER 78] A. COLMERAUER. *Les grammaires de métamorphose in Natural Language with Computers*, pages 133–189. E. Bolc ed., Lecture notes in computer science n.63, Springer-Verlag, 1978.

[COLMERAUER 83] A. COLMERAUER, H. KANOUI, et M. VAN CANEGHEM. PROLOG: bases théoriques et développements actuels. *Techniques et Sciences Informatiques*, 2-4:271–311, 1983.

[COLNET 86] D. COLNET, G. MASINI, A. NAPOLI, Y. NOIRET, et K. TOMBRE. Les langages orientés objets. Rapport CRIN n. 86-R-077, 1986.

[CONDILLAC 86] M. CONDILLAC. *PROLOG : fondements et applications.* Dunod, Paris, 1986.

[COULON 86] D. COULON et D. KAYSER. Informatique et langage naturel : présentation générale des méthodes d'interprétation de textes écrits. *Techniques et Sciences Informatiques*, 5(2):103–128, 1986.

[DANLOS 85] L. DANLOS. *Génération automatique de textes en langues naturelles.* Masson, Paris, 1985.

[DESCLES 82] J.P. DESCLES. Langages quasi-naturels et catégories grammaticales. *Actes du colloque ARC Domaines et objectifs de la recherche cognitive*, pages 109–140, Pont-à-Mousson, 1982.

[DEVILLE 86] G. DEVILLE et H. PAULUSSEN. A case grammar as an original linguistic model for the semantic representation of utterances in a man-machine dialog system. Thesis of computational linguistics, University of Antwerpen, Belgique, 1986.

[DEVILLE 87] G. DEVILLE. *Corpus de dialogues oraux finalisés en situation réelle : Demande de renseignements auprès du ministère de l'emploi et du travail (Cellule Action-Travail).* ed G. Deville, FUNDP Namur, 1987.

[DIK 79] S. DIK. *Functional Grammar.* North-Holland, Amsterdam, 1979.

[dM 84] J. di MARTINO. Contribution à la reconnaissance globale de la parole : mots isolés et mots enchainés. Thèse de Docteur ingénieur, Nancy 1, 1984.

[DUBOIS 85] D. DUBOIS et H. PRADE. *Théorie des possibilités : applications à la représentation des connaissances en informatique.* Masson, 1985.

[DULEY 58] H. DULEY et S. BALASHEK. Automatic recognition of phonetic patterns in speech. *JASA*, 30:721–732, 1958.

[ERMAN 77] L.D. ERMAN. A functional description of the HEARSAY II speech understanding system. *Proceedings of IEEE ICASSP-77*, 1977.

[FALZON 84] P. FALZON. Les langages opératifs. *Actes du séminaire GRECO sur le dialogue oral homme-machine*, pages –, Nancy, 1984.

[FILLMORE 68] C. FILLMORE. *The Case for Case in Universals in Linguistic Theory*. E. Bach and R.T. Harm eds. Rinehart and Winston, 1968.

[FOHR 86] D. FOHR. APHODEX : un système expert de décodage acoustico-phonétique de la parole continue. Thèse de doctorat de l'université de Nancy 1, 1986.

[GRAU 83] B. GRAU. Analyse et représentation d'un texte d'après le thème du discours. Thèse de 3ème Cycle, Paris 6, 1983.

[GRECO 85] GRECO. Dialogue oral homme–machine en situation assitée par l'action. *Actes du 5ème Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle*, pages 281–296, Grenoble, 1985.

[GRENIER 78] Y. GRENIER. Speaker identification from linear prediction. *Proceedings of 4th IJCPR*, pages 1019–1021, Kyoto, 1978.

[GRICE 75] H.P. GRICE. *Logic and Conversation in Syntax and Semantic 3: Speech Acts*, pages 41–58. E. Cole and J.L. Morgan eds,Academic Press, 1975.

[HATON 77] J.P. HATON, J.F. MARI, et J.M. PIERREL. Research towards speech understanding models for artificial and natural language. *Proceedings of IEEE ICASSP-77*, pages 807–810, 1977.

[HATON 80] J.P. HATON. *Speech Recognition Work in Western Europe in Trends in Speech Recognition,*. W. Lea ed., Prentice Hall, 1980.

[HATON 85a] J.P HATON. Intelligence artificielle en compréhension automatique de la parole: état des recherches et comparaison avec la vision par ordinateur. *Techniques et Sciences Informatiques*, 4-3:265–287, 1985.

[HATON 85b] M.C HATON. Contribution à l'éducation vocale assitée par ordinateur: étude des voix et réalisation du système SIRENE. Thèse d'état, université de Nancy 1, 1985.

[HATON 86] J.P. HATON, B. MANGEOL, et J.M. PIERREL. Organisation et fonctionnement d'une composante lexicale dans un système de dialogue oral homme-machine. *Acte du séminaire GRECO-GALF*, pages 259–267, Toulouse, 1986.

[HORMANN 72] H. HORMANN. *Introduction à la psycholinguistique. Langue et langage*, Larousse, 1972.

[JELINEK 76] F. JELINEK. Continuous speech recognition by statistical methods. *Proceedigs of IEEE*, 64-4:532–556, 1976.

[KERNINGHAN 78] B.W. KERNINGHAN et D.M. RITCHIE. *The C Programming Language*. Prentice Hall, 1978.

[KITTERIGDE 84] R. KITTERIGDE. Textual cohesion within sublanguages : implication for automatic analysis and synthesis. *Actes du colloque IRIALISH, représentation des connaissances et raisonnement pour les sciences de l'homme*, pages 42–72, St Maximin, 1984.

[LAASRI 87] H. LAASRI, B. MAITRE, et J.P. HATON. ATOME : outil d'aide au développement des systèmes multi-experts. *Actes du 6ème Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle*, pages 749–760, Antibes, 1987.

[LAFONT 83] S. LAFONT. Etude et mise en oeuvre d'outils informatiques pour non-voyants: console braille et commande vocale. Thèse de Docteur Ingenieur, Nancy 1, 1983.

[LEVINSON 83] S. LEVINSON. *Pragmatics*. Cambridge university press, 1983.

[LIBERMAN 70] A.M. LIBERMAN. The grammar of speech and language. *Cognitive Psychology*, 1:301–323, 1970.

[LORANT 86] M. LORANT et S. SIMONET. Contributions à un système de dialogue homme-machine à composante orale. Mémoire de licence et maîtrise en informatique. Facultés N.D de la paix, Namur, Belgique, 1986.

[LOWERRE 76] B.T. LOWERRE. The HARPY speech recognition system. Technical report, Carnegie-Mellon university, 1976.

[MARIANI 82] J. MARIANI. ESOPE: un système de compréhension de la parole continue. Thèse de Doctorat ès Sciences, Paris 6, 1982.

[MERCIER 77] G. MERCIER, P. QUINTON, et R. VIVES. Dialogue homme–machine avec KEAL. *Recherches acoustiques*, 4:15–22, 1977.

[MINSKY 75] M. MINSKY. *A Framework for Representing Knowledge The Psychology of Computer Vision*, pages 211–277. P.H. Winston ed., Mc Graw Hill, 1975.

[MOESCHLER 85] J. MOESCHLER. *Argumentation et conversation. Eléments pour une analyse pragmatique du discours.* Hatier, 1985.

[MOREL 85] M.A. MOREL et al. Analyse linguistique d'un corpus d'oral finalisé. Rapport final d'étude, CNRS-GRECO communication parlée, 1985.

[MORIN 87] P. MORIN et J.M. PIERREL. PARTNER: un système de dialogue oral homme machine. *Actes du congrès COGNITIVA*, pages 354–361, Paris, 1987.

[MOUSEL 86a] P. MOUSEL. Implémentation des réseaux à noeuds procéduraux sous UNIX BSD 4.2 sur le VAX 11/785 du CRIN. Rapport interne CRIN, n.86-R-123, 1986.

[MOUSEL 86b] P. MOUSEL, J.M. PIERREL, et A. ROUSSANALY. Logiciels d'aide à la création de bases de connaissances linguistiques. Rapport interne CRIN, n.86-R-029, 1986.

[NII 86a] H. P. NII. Blackboard systems: blackboard application systems, blackboard systems from a knowledge engineering perspective. *The AI magazine*, 82–106, 1986.

[NII 86b] H. P. NII. Blackboard systems: the blackboard model of problem solving and evolution of blackboard architectures. *The AI magazine*, 38–53, 1986.

[PIERREL 75] J.M. PIERREL. Contribution à la compréhension automatique du discours continu. Thèse de 3ième cycle, université de Nancy 1, 1975.

[PIERREL 81] J.M. PIERREL. Etude et mise en oeuvre de contraintes linguistiques en compréhension automatique du discours continu. Thèse d'état, Nancy 1, 1981.

[PIERREL 87] J.M. PIERREL. *Dialogue oral homme-machine.* Hermès, Paris, 1987.

[PISTER 84] C. PISTER. Adaptation au locuteur par apprentissage automatique, application à un système de reconnaissance automatique de la parole. Thèse de 3ième cycle, Nancy 1, 1984.

[QUINTON 82] P. QUINTON. Utilisation des contraintes syntaxiques pour la reconnaissance de la parole continue. *Techniques et Sciences Informatiques*, 1-3:233–238, 1982.

[ROBINSON 79] J.A. ROBINSON. *Logic: Form and Function.* Edimburgh University Press, 1979.

[ROMARY 87] L. ROMARY. Rapport de DEA Nancy 1, 1987.

[ROSSI 81] M. ROSSI, A. DI CHRISTO, D. HIRST, P. MARTIN, et Y. NISHINUMA. *L'intonation : de l'acoustique à la sémantique.* Méridiens–Klincksieck, Paris, 1981.

[ROULET 80] E. ROULET. Stratégies d'interactions, modes d'implications et marqueurs illocutoires. *Cahiers de la linguistique française*, (1):80–103, 1980.

[ROUSSANALY 86] A. ROUSSANALY, P. MOUSEL, N. CARBONELL, B. MANGEOL, et J.M. PIERREL. Réalisation d'un corpus de dialogues oraux. application aux renseignements administratifs. Rapport interne CRIN, n.86-R-083, 1986.

[ROUSSANALY 87] A. ROUSSANALY et J.M. PIERREL. Le raisonneur dans un système de dialogue oral homme-machine. *Actes du 6ème Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle*, pages 53–64, Antibes, 1987.

[SABAH 83] G. SABAH. Un système de questions-réponses sur les rubriques professionnelles de l'annuaire téléphonique. Rapport final ATP Intelligence artificielle, 1983.

[SCAPIN 86] D.L. SCAPIN. Guide ergonomique de conception des interfaces homme-ordinateurs. Rapport INRIA, 1986.

[SCHANK 77] R.C. SCHANK et R.P. ABELSON. *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Ass., 1977.

[SHORLIFFE 76] SHORLIFFE. *Computer-Based Medical Consultations : MYCIN.* American Elsevier, 1976.

[SOWA 84] J.F. SOWA. *Conceptual Structures. Information Processing in Mind and Machine.* Addison-Wesley, 1984.

[STEFIK 81] M. STEFIK. Planning with contraints (MOLGEN : part 1). *Artificial Intelligence*, 16(2):112–139, 1981.

[TUBACH 70] J.P. TUBACH. Reconnaissance autaomatique de la parole. Thèse d'état, université de Grenoble, 1970.

[VILNAT 84] A. VILNAT. L'élaboration d'interventions pertinentes dans une conversation homme-machine. 1984.

[VINCENS 69] P. VINCENS. Aspect of speech recognition by computer. Ph.D Thesis, Stanford university, USA, 1969.

[WAKITA 80] H. WAKITA et S. MAKINO. *Recent work on speech recognition in Japan in Trends in speech recognition.* W. Lea ed., Prentice Hall, 1980.

[WALKER 78] D.E. WALKER, éditeur. *Understanding spoken language.* North-Holland, 1978.

[WINOGRAD 83] T. WINOGRAD. *Language as a Cognitive Process. Volume 1: Syntax.* Addison-Wesley, 1983.

[WINSTON 77] P. WINSTON. *Artificial Intelligence.* Addison-Wesley, 1977.

[WOLF 80] J. WOLF et W.A. WOODS. *The HWIM speech understanding system*, pages 316–339. W.A. LEA editor, Prentice-Hall, 1980.

[WOODS 70] W.A. WOODS. Transition network grammars for natural language analysis. *Comunication of ACM*, 13-10:591–602, 1970.

Annexe A

L'INTERPRÉTEUR

Cette annexe contient :

- d'une part, la description des différentes connaissances utilisées actuellement par le module INTERPRÉTEUR (base des faits, base des règles...)
- et, d'autre part, les traces d'exécution concernant l'interprétation de quelques énoncés typiques.

1 La base des faits

gestion-bf est un outil qui permet de gérer, de manière interactive, la base des faits. Une des fonctions de cet outil est de fournir une visualisation complète de la base des faits utilisée. Nous reproduisons, ci-dessous, le résultat de cette visualisation :

```
--------------BASE DE FAITS---------------
id : 0    _pers_sing1 (je)
id : 1    _pron_interr (qui)
id : 2    _pron_interr (ou)
id : 3    _positif (oui)
id : 4    _positif (ok)
id : 5    _positif (bien_sur)
id : 6    _positif (c_est_ca)
id : 7    _positif (naturellement)
id : 8    _negatif (non)
id : 9    _negatif (pas_du_tout)
id : 10   _extension (savoir)
id : 11   _extension (connaitre)
id : 12   _actobj (faire)
id : 13   _actobj (remplacer)
id : 14   _actobj (remplir)
id : 15   _actobj (renouveler)
id : 16   _echobt (s_adresser)
id : 17   _echobt (demande)
id : 18   _echobt (demander)
id : 19   _echobt (desirer)
id : 20   _echobt (obtenir)
```

```
id : 21   _echobt (se_renseigner)
id : 22   _statut2 (perime)
id : 23   _anime (qui)
id : 24   _lieu (Nancy)
id : 25   _lieu (Vandoeuvre)
id : 26   _lieu (Villers)
id : 27   _lieu (Paris)
```

Pour compléter cette base des faits, il faut rajouter les connaissances qui permettent d'associer un *concept* à une *entrée lexicale*. Celles-ci sont stockées dans le fichier nommé **concepts.file** dont le contenu est le suivant :

```
s_adresser s_adresser
carte_d_identite carte_d_identite
carte_de_sejour carte_de_sejour
demander obtenir
demande obtenir
double_nationalite double_nationalite
faire formalite
formalite formalite
je locuteur
le !REF
la !REF
mariage se_marier
moi locuteur
obtenir obtenir
Nancy nancy
ou lieu
Paris paris
perime perime
qui anime
remplacer renouveler
remplir remplir
renouveler renouveler
se_renseigner s_adresser
Vandoeuvre vandoeuvre
Villers villers
```

2 La base des règles

C'est l'outil de gestion interactive de la base des règles **gestion-br**, qui permet de fournir la liste suivante des règles d'interprétation que nous avons utilisées dans le cadre des tests que nous avons effectués :

```
----------BASE DE REGLES----------

identificateur de regle : 0

   _proposition(#x)
 & _predicat(#x,#p1)
 & _extension(#p1)
 & _cas_obj(#p1,#o1)
 & _proposition(#o1)
 & _cas_goal(#p2,#g2)
 & _proposition(#g2)

 => ajouter(_but(#g2))
  & ajouter(_question(#o1))
------------------------------

   identificateur de regle : 1

   _question(#q)
 & _predicat(#q,#p)
 & _echobt(#p)
 & _cas_ben(#p,#b)
 & _cas_agt(#p,#a)
 & _pron_interr(#a)

 => ajout_int("|question:)
  & ajout_int(#p)
  & ajout_int("()
  & ajout_int(#b)
  & ajout_int(",)
  & ajout_int("?)
  & ajout_int(#a)
  & ajout_int("))
------------------------------

   identificateur de regle : 2

   _but(#b)
 & _predicat(#b,#p)
 & _actobj(#p)
 & _cas_agt(#p,#a)
 & _cas_obj(#p,#o)
 & _proposition(#o)

 => ajouter(_but(#o))
------------------------------

   identificateur de regle : 3

   _but(#b)
 & _predicat(#b,#p)
 & _echobt(#p)
 & _cas_ben(#p,#n)
 & _cas_obj(#p,#o)

 => ajout_int("|but:)
  & ajout_int(#p)
  & ajout_int("()
  & ajout_int(#n)
  & ajout_int(",)
  & ajout_int(#o)
  & ajout_int("))
------------------------------
```

```
   identificateur de regle : 4

   _proposition(#x)
 & _predicat(#x,#p1)
 & _echobt(#p1)
 & _cas_obj(#p1,#o1)
 & _cas_pat(#p1,#a1)
 & _proposition(#o1)

 => ajouter(_but&question(#o1))
-------------------------------

   identificateur de regle : 5

   _proposition(#x)
 & _predicat(#x,#p1)
 & _echobt(#p1)
 & _cas_obj(#p1,#o1)
 & _cas_pat(#p1,#a1)

 => ajout_int("|but: )
  & ajout_int(#o1)
  & ajout_int("()
  & ajout_int(#a1)
  & ajout_int("))
  & ajout_int("|question:)
  & ajout_int("?formalite)
-------------------------------

   identificateur de regle : 6

   _but&question(#o1)
 & _predicat(#o1,#p2)
 & _actobj(#p2)
 & _cas_agt(#p2,#a2)
 & _cas_obj(#p2,#o2)
 & _cas_goal(#p2,#g2)

 => ajout_int("|question:)
  & ajout_int("?)
  & ajout_int(#o2)
  & ajout_int("|but:)
  & ajout_int(#g2)
  & ajout_int("()
  & ajout_int(#a2)
  & ajout_int("))
-------------------------------

   identificateur de regle : 7

   _3proposition(#p1,#p2,#p3)

 => interpreter(s/#p1)
```

```
  & interpreter(s/#p2)
  & interpreter(s/#p3)
-------------------------------

   identificateur de regle : 8

   _proposition(#x)
 & _predicat(#x,#p)
 & _statut2(#p)
 & _cas_obj(#p,#o)

 => ajout_int("|fait:)
  & ajout_int(#p)
  & ajout_int("()
  & ajout_int(#o)
  & ajout_int("))
-------------------------------

   identificateur de regle : 9

   _proposition(#x)
 & _modalite(#x,"obligation)
 & _predicat(#x,#p)
 & _actobj(#p)
 & _cas_agt(#p,#a)
 & _cas_obj(#p,#o)

 => ajout_int("|but:)
  & ajout_int(#p)
  & ajout_int("()
  & ajout_int(#a)
  & ajout_int(",)
  & ajout_int(#o)
  & ajout_int("))
-------------------------------

   identificateur de regle : 10

   _proposition(#x)
 & _predicat(#x,#p1)
 & _extension(#p1)
 & _cas_obj(#p1,#o1)
 & _proposition(#o1)

 => ajouter(_question(#o1))
-------------------------------

   identificateur de regle : 11

   _question(#q)
 & _predicat(#q,#p)
 & _actobj(#p)
 & _cas_agt(#p,#a)
 & _cas_obj(#p,"ce_que)
```

```
 => ajout_int("|question:)
  & ajout_int("?)
  & ajout_int(#p)
  & ajout_int("()
  & ajout_int(#a)
  & ajout_int("))
------------------------------

   identificateur de regle : 12

   _but(#b)
 & _predicat(#b,#p)
 & _actobj(#p)
 & _cas_agt(#p,#a)
 & _cas_obj(#p,#o)

 => ajout_int("|but:)
  & ajout_int(#p)
  & ajout_int("()
  & ajout_int(#a)
  & ajout_int(",)
  & ajout_int(#o)
  & ajout_int("))
------------------------------

   identificateur de regle : 13

   _attendu("conf@quest)
 & _reponse(#x)
 & _positif(#x)

 => ajout_int("|fait:)
  & en_machine(")
------------------------------

   identificateur de regle : 14

   _attendu("conf@quest)
 & _reponse(#x)
 & _negatif(#x)

 => ajout_int("|fait:)
  & en_machine("-)
------------------------------

   identificateur de regle : 15

   _attendu("val@lieu)
 & _reponse(#x)
 & _lieu(#x)

 => ajout_int("|fait:)
  & en_machine(#x)
```

Pour élaborer ces règles, nous avons essentiellement considéré quelques énoncés typiques provenant de notre corpus [ROUSSANALY 86] et étudiés dans [DEVILLE 86].

3 Quelques exemples

Pour fonctionner, l'interpréteur (**interp**) a besoin des informations suivantes :

- la représentation (le plus souvent syntaxico-sémantique) de l'énoncé à interpréter,
- éventuellement d'hypothèses internes sur le contenu de l'énoncé attendu,
- et parfois d'accès spécifiques à l'historique.

Ces informations seront trouvées dans des fichiers préalablement préparés ou rentrées de manière intéractive.

3.1 Exemple 1

Enoncé : Il s'agit d'interpréter l'énoncé suivant :

JE VOUDRAIS SAVOIR À QUI IL FAUT S'ADRESSER POUR FAIRE UNE DEMANDE DE DOUBLE-NATIONALITÉ

Hypothèses internes : Nous supposons qu'il n'existe pas d'hypothèses internes du gestionnaire de dialogue.

Représentation de l'énoncé : La représentation de cet énoncé est rangée dans le fichier **2-nationalite** de la manière suivante :

```
_proposition NIVEAU1
_predicat NIVEAU1 savoir
_cas_pat savoir je
_cas_obj savoir NIVEAU2
_proposition NIVEAU2
_predicat NIVEAU2 s_adresser
_cas_ben s_adresser je
_cas_agt s_adresser qui
_cas_goal s_adresser NIVEAU3
_proposition NIVEAU3
_predicat NIVEAU3 faire
_cas_agt faire je
_cas_obj faire NIVEAU4
_proposition NIVEAU4
_predicat NIVEAU4 demande
_cas_ben demande je
_cas_obj demande double_nationalite
```

Interprétation : Le résultat fourni par l'interprétation est le suivant :

```
----------------------------------------
-           Interpretation             -
----------------------------------------

question:s_adresser(locuteur,?anime)
but:obtenir(locuteur,double_nationalite)
```

Trace : Les différentes étapes qui ont conduit à ce résultat sont décrites dans la trace d'exécution ci-après.

```
--------------- TRACE D'EXECUTION ---------------

============================================================
#            HYPOTHESES DU DIALOGUEUR                      #
============================================================
 En attendant le pipe ...
Simuler les hypotheses du gestionnaire du dialogue.
Entrer le nom du fichier-hypothese~:

============================================================
#            REPRESENTATION DE SYN-SEM                     #
============================================================
La traduction de l'enonce n'etant pas implementee, le
contenu de la memoire de travail doit etre introduit a la main.
Entrer le nom du fichier-enonce :  2-nationalite

----------------------------------------
~          Memoire de travail          ~
----------------------------------------

_proposition (NIVEAU1)  0
_predicat (NIVEAU1,savoir)  1
_cas_pat (savoir,je)  2
_cas_obj (savoir,NIVEAU2)  3
_proposition (NIVEAU2)  4
_predicat (NIVEAU2,s_adresser)  5
_cas_ben (s_adresser,je)  6
_cas_agt (s_adresser,qui)  7
_cas_goal (s_adresser,NIVEAU3)  8
_proposition (NIVEAU3)  9
_predicat (NIVEAU3,faire)  10
_cas_agt (faire,je)  11
_cas_obj (faire,NIVEAU4)  12
_proposition (NIVEAU4)  13
_predicat (NIVEAU4,demande)  14
_cas_ben (demande,je)  15
_cas_obj (demande,double_nationalite)  16

----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------

  id de regle : 0
substitution :
(#x=NIVEAU1)(#p1=savoir)(#o1=NIVEAU2)(#p2=s_adresser)(#g2=NIVEAU3)

  id de regle : 10
```



```
substitution :
(#x=NIVEAU1)(#p1=savoir)(#o1=NIVEAU2)

----------------------------------------
~           Cycle :   1                ~
----------------------------------------


 Debut d'execution de la regle    0:
 ----------------------------------

 ajout de l'element suivant en WM :
_but (NIVEAU3)  17

 diagnostic d'execution :  O.K.

 ajout de l'element suivant en WM :
_question (NIVEAU2)  18

 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-------------------------------------------------


----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------

  id de regle : 2
substitution :
(#b=NIVEAU3)(#p=faire)(#a=je)(#o=NIVEAU4)

  id de regle : 12
substitution :
(#b=NIVEAU3)(#p=faire)(#a=je)(#o=NIVEAU4)

  id de regle : 1
substitution :
(#q=NIVEAU2)(#p=s_adresser)(#b=je)(#a=qui)

----------------------------------------
~           Cycle :   2                ~
----------------------------------------


 Debut d'execution de la regle    2:
 ----------------------------------

 ajout de l'element suivant en WM :
_but (NIVEAU4)  19

 diagnostic d'execution :  O.K.
fin d'execution de la regle courante
```

```
diagnostic d'execution :  O.K.
-----------------------------------------------


----------------------------------------
-  Regles activables du conflict-set   -
----------------------------------------
  id de regle : 1
substitution :
(#q=NIVEAU2)(#p=s_adresser)(#b=je)(#a=qui)

  id de regle : 3
substitution :
(#b=NIVEAU4)(#p=demande)(#n=je)(#o=double_nationalite)

----------------------------------------
-            Cycle :   3               -
----------------------------------------


Debut d'execution de la regle    1:
-----------------------------------

ajout de la partie d'interpretation suivante :
|question:
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
s_adresser
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
(
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
locuteur
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
,
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
?
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
anime
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
)
```

```
diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-----------------------------------------------


----------------------------------------
-  Regles activables du conflict-set   -
----------------------------------------
  id de regle : 3
substitution :
(#b=NIVEAU4)(#p=demande)(#n=je)(#o=double_nationalite)

----------------------------------------
-            Cycle :   4               -
----------------------------------------


Debut d'execution de la regle    3:
-----------------------------------

ajout de la partie d'interpretation suivante :
|but:
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
obtenir
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
(
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
locuteur
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
,
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
double_nationalite
diagnostic d'execution :  O.K.

ajout de la partie d'interpretation suivante :
)
diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-----------------------------------------------
```

```
----------------------------------------
-  Regles activables du conflict-set   -
----------------------------------------

 plus de regles activables...



----------------------------------------
-            Interpretation            -
----------------------------------------

question:s_adresser(locuteur,?anime)
but:obtenir(locuteur,double_nationalite)

----------------------------------------
-          Memoire de travail          -
----------------------------------------

_proposition (NIVEAU1)  0
_predicat (NIVEAU1,savoir)  1
_cas_pat (savoir,je)  2
_cas_obj (savoir,NIVEAU2)  3
_proposition (NIVEAU2)  4
_predicat (NIVEAU2,s_adresser)  5
_cas_ben (s_adresser,je)  6
_cas_agt (s_adresser,qui)  7
_cas_goal (s_adresser,NIVEAU3)  8
_proposition (NIVEAU3)  9
_predicat (NIVEAU3,faire)  10
_cas_agt (faire,je)  11
_cas_obj (faire,NIVEAU4)  12
_proposition (NIVEAU4)  13
_predicat (NIVEAU4,demande)  14
_cas_ben (demande,je)  15
_cas_obj (demande,double_nationalite)  16
_but (NIVEAU3)  17
_question (NIVEAU2)  18
_but (NIVEAU4)  19
```

3.2 Exemple 2

Enoncé :

JE VOUDRAIS ME RENSEIGNER SUR LES FORMALITÉS À REMPLIR POUR LE MARIAGE

Hypothèses internes : Pas d'hypothèses internes .

Représentation de l'énoncé : fichier **mariage**.

Interprétation :

```
----------------------------------------
-            Interpretation            -
```



```
----------------------------------------

question:?formalite
but:se_marier(locuteur)
```

Trace :

```
--------------- TRACE D'EXECUTION ---------------

============================================================
#            HYPOTHESES DU DIALOGUEUR                      #
============================================================
 En attendant le pipe ...
Simuler les hypotheses du gestionnaire du dialogue.
Entrer le nom du fichier-hypothese":

============================================================
#            REPRESENTATION DE SYN-SEM                     #
============================================================
La traduction de l'enonce n'etant pas implementee, le
contenu de la memoire de travail doit etre introduit a la main.
Entrer le nom du fichier-enonce :  mariage

----------------------------------------
-          Memoire de travail          -
----------------------------------------

_proposition (NIVEAU1)  0
_predicat (NIVEAU1,se_renseigner)  1
_cas_pat (se_renseigner,je)  2
_cas_obj (se_renseigner,NIVEAU2)  3
_proposition (NIVEAU2)  4
_predicat (NIVEAU2,remplir)  5
_cas_agt (remplir,je)  6
_cas_obj (remplir,formalite)  7
_cas_goal (remplir,mariage)  8

----------------------------------------
-  Regles activables du conflict-set   -
----------------------------------------

  id de regle : 4
substitution :
(#x=NIVEAU1)(#p1=se_renseigner)(#o1=NIVEAU2)(#a1=je)

  id de regle : 5
substitution :
(#x=NIVEAU1)(#p1=se_renseigner)(#o1=NIVEAU2)(#a1=je)

----------------------------------------
-            Cycle :   1               -
----------------------------------------


 Debut d'execution de la regle    4:
 ----------------------------------
```

```
 ajout de l'element suivant en WM :
_but&question (NIVEAU2)  9

 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-----------------------------------------------


----------------------------------------
-  Regles activables du conflict-set   -
----------------------------------------

  id de regle : 6
substitution :
(#o1=NIVEAU2)(#p2=remplir)(#a2=je)(#o2=formalite)(#g2=mariage)

----------------------------------------
-            Cycle :   2               -
----------------------------------------


 Debut d'execution de la regle    6:
 ----------------------------------


 ajout de la partie d'interpretation suivante :
|question:
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
?
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
formalite
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
|but:
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
se_marier
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
(
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
locuteur
 diagnostic d'execution :  O.K.
```



```
 ajout de la partie d'interpretation suivante :
)
 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-----------------------------------------------


----------------------------------------
-  Regles activables du conflict-set   -
----------------------------------------


 plus de regles activables...



----------------------------------------
-            Interpretation            -
----------------------------------------


question:?formalite
but:se_marier(locuteur)

----------------------------------------
-          Memoire de travail          -
----------------------------------------

_proposition (NIVEAU1)  0
_predicat (NIVEAU1,se_renseigner)  1
_cas_pat (se_renseigner,je)  2
_cas_obj (se_renseigner,NIVEAU2)  3
_proposition (NIVEAU2)  4
_predicat (NIVEAU2,remplir)  5
_cas_agt (remplir,je)  6
_cas_obj (remplir,formalite)  7
_cas_goal (remplir,mariage)  8
_but&question (NIVEAU2)  9
```

3.3 Exemple 3

Enoncé :

MA CARTE D'IDENTITÉ EST PÉRIMÉE. IL FAUDRAIT QUE JE LA REMPLACE. JE VOUDRAIS SAVOIR OÙ JE DOIS ME RENSEIGNER

Hypothèses internes : Pas d'hypothèses internes .

Représentation de l'énoncé : fichier ci.

Interprétation :

```
----------------------------------------
-            Interpretation            -
----------------------------------------
```

```
fait:perime(carte_d_identite)
but:renouveler(locuteur,!REF)
question:s_adresser(locuteur,?lieu)
```

Trace :

```
---------- TRACE D'EXECUTION ----------

============================================================
#            HYPOTHESES DU DIALOGUEUR                      #
============================================================
 En attendant le pipe ...
Simuler les hypotheses du gestionnaire du dialogue.
Entrer le nom du fichier-hypothese~:

============================================================
#            REPRESENTATION DE SYN-SEM                     #
============================================================
La traduction de l'enonce n'etant pas implementee, le
contenu de la memoire de travail doit etre introduit a la main.
Entrer le nom du fichier-enonce : ci

----------------------------------------
~          Memoire de travail          ~
----------------------------------------
_3proposition (P1,P2,P3)  0
_proposition (P1)  1
_predicat (P1,perime)  2
_cas_obj (perime,carte_d_identite)  3
_proposition (P2)  4
_modalite (P2,obligation)  5
_predicat (P2,remplacer)  6
_cas_agt (remplacer,je)  7
_cas_obj (remplacer,la)  8
_proposition (P3)  9
_predicat (P3,savoir)  10
_cas_pat (savoir,je)  11
_cas_obj (savoir,P32)  12
_proposition (P32)  13
_predicat (P32,se_renseigner)  14
_cas_ben (se_renseigner,je)  15
_cas_agt (se_renseigner,ou)  16

----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------
  id de regle : 7
substitution :
(#p1=P1)(#p2=P2)(#p3=P3)

  id de regle : 8
substitution :
(#x=P1)(#p=perime)(#o=carte_d_identite)
```



```
  id de regle : 9
substitution :
(#x=P2)("obligation=obligation)(#p=remplacer)(#a=je)(#o=la)

  id de regle : 10
substitution :
(#x=P3)(#p1=savoir)(#o1=P32)

----------------------------------------
~            Cycle :   1               ~
----------------------------------------


 Debut d'execution de la regle    7:
 ----------------------------------

Execution d'un appel recursif a l'interpreteur.
Identificateur de variable concernee : 0
concerne egalement la sous-structure qui en depend

----------------------------------------
~          Memoire de travail          ~
----------------------------------------
_3proposition (P1,P2,P3)  0
_proposition (P1)  1
_predicat (P1,perime)  2
_cas_obj (perime,carte_d_identite)  3

----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------
  id de regle : 8
substitution :
(#x=P1)(#p=perime)(#o=carte_d_identite)

----------------------------------------
~            Cycle :   1               ~
----------------------------------------


 Debut d'execution de la regle    8:
 ----------------------------------

 ajout de la partie d'interpretation suivante :
|fait:
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
perime
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
(
```

```
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
carte_d_identite
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
)
 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-----------------------------------------------


----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------
 plus de regles activables...

fin de l'appel recursif du cycle 1
diagnostic d'execution de cette action : O.K.

Execution d'un appel recursif a l'interpreteur.
Identificateur de variable concernee : 1
concerne egalement la sous-structure qui en depend

----------------------------------------
~          Memoire de travail          ~
----------------------------------------
_3proposition (P1,P2,P3)  0
_proposition (P2)  4
_modalite (P2,obligation)  5
_predicat (P2,remplacer)  6
_cas_agt (remplacer,je)  7
_cas_obj (remplacer,la)  8
_cas_pat (savoir,je)  11
_cas_ben (se_renseigner,je)  15

----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------
  id de regle : 9
substitution :
(#x=P2)("obligation=obligation)(#p=remplacer)(#a=je)(#o=la)

----------------------------------------
~            Cycle :   1               ~
----------------------------------------


 Debut d'execution de la regle    9:
 ----------------------------------
```



```
 ajout de la partie d'interpretation suivante :
!but:
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
renouveler
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
(
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
locuteur
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
,
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
!REF
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
)
 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-----------------------------------------------


----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------
 plus de regles activables...

fin de l'appel recursif du cycle 1
diagnostic d'execution de cette action : O.K.

Execution d'un appel recursif a l'interpreteur.
Identificateur de variable concernee : 2
concerne egalement la sous-structure qui en depend

----------------------------------------
          Memoire de travail
----------------------------------------
_3proposition (P1,P2,P3)  0
_proposition (P3)  9
_predicat (P3,savoir)  10
_cas_pat (savoir,je)  11
_cas_obj (savoir,P32)  12
_cas_agt (remplacer,je)  7
```

```
_cas_ben (se_renseigner,je)  15
_proposition (P32)  13
_predicat (P32,se_renseigner)  14
_cas_agt (se_renseigner,ou)  16

----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------

  id de regle : 10
substitution :
(#x=P3)(#p1=savoir)(#o1=P32)

----------------------------------------
~            Cycle :   1               ~
----------------------------------------


 Debut d'execution de la regle   10:
 -----------------------------------

 ajout de l'element suivant en WM :
_question (P32)  17

 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-------------------------------------------------


----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------

  id de regle : 1
substitution :
(#q=P32)(#p=se_renseigner)(#b=je)(#a=ou)

----------------------------------------
~            Cycle :   2               ~
----------------------------------------


 Debut d'execution de la regle    1:
 ----------------------------------

 ajout de la partie d'interpretation suivante :
|question:
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
s_adresser
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
```



```
(
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
locuteur
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
,
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
?
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
lieu
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
)
 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-------------------------------------------------


----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------

 plus de regles activables...

fin de l'appel recursif du cycle 2
diagnostic d'execution de cette action : O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-------------------------------------------------


----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------

 plus de regles activables...



----------------------------------------
~            Interpretation            ~
----------------------------------------
```

```
fait:perime(carte_d_identite)
but:renouveler(locuteur,!REF)
question:s_adresser(locuteur,?lieu)

----------------------------------------
-          Memoire de travail          -
----------------------------------------

_3proposition (P1,P2,P3)  0
_proposition (P1)  1
_predicat (P1,perime)  2
_cas_obj (perime,carte_d_identite)  3
_proposition (P2)  4
_modalite (P2,obligation)  5
_predicat (P2,remplacer)  6
_cas_agt (remplacer,je)  7
_cas_obj (remplacer,la)  8
_proposition (P3)  9
_predicat (P3,savoir)  10
_cas_pat (savoir,je)  11
_cas_obj (savoir,P32)  12
_proposition (P32)  13
_predicat (P32,se_renseigner)  14
_cas_ben (se_renseigner,je)  15
_cas_agt (se_renseigner,ou)  16
```

3.4 Exemple 4

Enoncé :

JE VEUX SAVOIR CE QU'IL FAUT FAIRE POUR RENOUVELER LA CARTE DE SÉJOUR

Hypothèses internes : Pas d'hypothèses internes .

Représentation de l'énoncé : fichier **cs**.

Interprétation :

```
----------------------------------------
-           Interpretation             -
----------------------------------------

fait:perime(carte_d_identite)
but:renouveler(locuteur,!REF)
question:s_adresser(locuteur,?lieu)
```

Trace :

```
---------- TRACE D'EXECUTION ----------

============================================================
#            HYPOTHESES DU DIALOGUEUR                      #
============================================================
 En attendant le pipe ...
Simuler les hypotheses du gestionnaire du dialogue.
Entrer le nom du fichier-hypothese~:
```



```
============================================================
#            REPRESENTATION DE SYN-SEM                     #
============================================================
La traduction de l'enonce n'etant pas implementee, le
contenu de la memoire de travail doit etre introduit a la main.
Entrer le nom du fichier-enonce :  cs

----------------------------------------
-          Memoire de travail          -
----------------------------------------

_proposition (NIVEAU1)  0
_predicat (NIVEAU1,savoir)  1
_cas_pat (savoir,je)  2
_cas_obj (savoir,NIVEAU2)  3
_proposition (NIVEAU2)  4
_predicat (NIVEAU2,faire)  5
_cas_agt (faire,je)  6
_cas_obj (faire,ce_que)  7
_cas_goal (faire,NIVEAU3)  8
_proposition (NIVEAU3)  9
_predicat (NIVEAU3,renouveler)  10
_cas_agt (renouveler,je)  11
_cas_obj (renouveler,carte_de_sejour)  12

----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------

  id de regle : 0
substitution :
(#x=NIVEAU1)(#p1=savoir)(#o1=NIVEAU2)(#p2=faire)(#g2=NIVEAU3)

  id de regle : 10
substitution :
(#x=NIVEAU1)(#p1=savoir)(#o1=NIVEAU2)

----------------------------------------
~            Cycle :   1               ~
----------------------------------------


 Debut d'execution de la regle    0:
 ----------------------------------

 ajout de l'element suivant en WM :
_but (NIVEAU3)  13

 diagnostic d'execution :  O.K.

 ajout de l'element suivant en WM :
_question (NIVEAU2)  14

 diagnostic d'execution :  O.K.
fin d'execution de la regle courante
```

```
diagnostic d'execution :  O.K.
-----------------------------------------------


----------------------------------------
"  Regles activables du conflict-set   "
----------------------------------------
  id de regle : 12
substitution :
(#b=NIVEAU3)(#p=renouveler)(#a=je)(#o=carte_de_sejour)

  id de regle : 11
substitution :
(#q=NIVEAU2)(#p=faire)(#a=je)("ce_que=ce_que)

----------------------------------------
"           Cycle :   2               "
----------------------------------------


 Debut d'execution de la regle   12:
 ----------------------------------

 ajout de la partie d'interpretation suivante :
|but:
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
renouveler
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
(
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
locuteur
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
,
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
carte_de_sejour
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
)
 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
```



```
-----------------------------------------------


----------------------------------------
"  Regles activables du conflict-set   "
----------------------------------------
  id de regle : 11
substitution :
(#q=NIVEAU2)(#p=faire)(#a=je)("ce_que=ce_que)

----------------------------------------
"           Cycle :   3               "
----------------------------------------


 Debut d'execution de la regle   11:
 ----------------------------------

 ajout de la partie d'interpretation suivante :
|question:
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
?
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
formalite
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
(
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
locuteur
 diagnostic d'execution :  O.K.

 ajout de la partie d'interpretation suivante :
)
 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
-----------------------------------------------


----------------------------------------
"  Regles activables du conflict-set   "
----------------------------------------


 plus de regles activables...
```

```
----------------------------------------
-           Interpretation             -
----------------------------------------

but:renouveler(locuteur,carte_de_sejour)
question:?formalite(locuteur)

----------------------------------------
-         Memoire de travail           -
----------------------------------------
_proposition (NIVEAU1)  0
_predicat (NIVEAU1,savoir)  1
_cas_pat (savoir,je)  2
_cas_obj (savoir,NIVEAU2)  3
_proposition (NIVEAU2)  4
_predicat (NIVEAU2,faire)  5
_cas_agt (faire,je)  6
_cas_obj (faire,ce_que)  7
_cas_goal (faire,NIVEAU3)  8
_proposition (NIVEAU3)  9
_predicat (NIVEAU3,renouveler)  10
_cas_agt (renouveler,je)  11
_cas_obj (renouveler,carte_de_sejour)  12
_but (NIVEAU3)  13
_question (NIVEAU2)  14
```

3.5 Exemple 5

Enoncé :

NON

en réponse à la question :

EST-CE QUE VOUS ÊTES MINEUR?

Hypothèses internes : fichier **dialconf** de contenu :

```
_attendu conf@quest
```

traduisant le fait que l'on s'attend à une confirmation à une question fermée.

Représentation de l'énoncé : fichier **inf.**

Interprétation :

```
----------------------------------------
-           Interpretation             -
----------------------------------------

fait:-mineur(locuteur)
```

Trace :


```
---------- TRACE D'EXECUTION ----------

================================================================
#            HYPOTHESES DU DIALOGUEUR                          #
================================================================
 En attendant le pipe ...
Simuler les hypotheses du gestionnaire du dialogue.
Entrer le nom du fichier-hypothese~:  dialconf

================================================================
#            REPRESENTATION DE SYN-SEM                         #
================================================================
La traduction de l'enonce n'etant pas implementee, le
contenu de la memoire de travail doit etre introduit a la main.
Entrer le nom du fichier-enonce :  Einf

----------------------------------------
-         Memoire de travail           -
----------------------------------------
_attendu (conf@quest)  0
_reponse (non)  1

----------------------------------------
-  Regles activables du conflict-set   -
----------------------------------------
  id de regle : 14
substitution :
("conf@quest=conf@quest)(#x=non)

----------------------------------------
-           Cycle :  1                 -
----------------------------------------


 Debut d'execution de la regle   14:
 -----------------------------------

 ajout de la partie d'interpretation suivante :
|fait:
 diagnostic d'execution :  O.K.

================================================================
#            SIMULATION DE L'ACCES A L'HISTORIQUE              #
================================================================
 Introduisez la derniere question-machine ,s.v.p.
 > ?mineur(locuteur)
 ajout de la partie d'interpretation suivante :
_
 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
------------------------------------------------
```

```
----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------

 plus de regles activables...



----------------------------------------
~            Interpretation            ~
----------------------------------------

fait:-mineur(locuteur)

----------------------------------------
~          Memoire de travail          ~
----------------------------------------
_attendu (conf@quest)  0
_reponse (non)  1
```

3.6 Exemple 6

Enoncé :

A NANCY

en réponse à la question :

OÙ EST-CE QUE VOUS HABITEZ?

Hypothèses internes : fichier **dialreponse** de contenu :

```
_attendu val@lieu
```

Représentation de l'énoncé : fichier **reponse**.
Interprétation :

```
----------------------------------------
~            Interpretation            ~
----------------------------------------

fait:habiter(locuteur,nancy)
```

Trace :

```
---------- TRACE D'EXECUTION ----------

================================================================
#            HYPOTHESES DU DIALOGUEUR                          #
================================================================
```



```
 En attendant le pipe ...
Simuler les hypotheses du gestionnaire du dialogue.
Entrer le nom du fichier-hypothese~:  dialreponse

================================================================
#            REPRESENTATION DE SYN-SEM                         #
================================================================
La traduction de l'enonce n'etant pas implementee, le
contenu de la memoire de travail doit etre introduit a la main.
Entrer le nom du fichier-enonce :  reponse

----------------------------------------
~          Memoire de travail          ~
----------------------------------------

_attendu (val@lieu)  0
_reponse (Nancy)  1

----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------

  id de regle : 15
substitution :
("val@lieu=val@lieu)(#x=Nancy)

----------------------------------------
~            Cycle :   1               ~
----------------------------------------


 Debut d'execution de la regle   15:
 ----------------------------------

 ajout de la partie d'interpretation suivante :
|fait:
 diagnostic d'execution :  O.K.

================================================================
#            SIMULATION DE L'ACCES A L'HISTORIQUE              #
================================================================
 Introduisez la derniere question-machine ,s.v.p.
 > habiter(locuteur,?lieu)
 ajout de la partie d'interpretation suivante :
nancy
 diagnostic d'execution :  O.K.
fin d'execution de la regle courante

diagnostic d'execution :  O.K.
------------------------------------------------


----------------------------------------
~  Regles activables du conflict-set   ~
----------------------------------------
```

```
 plus de regles activables...



----------------------------------------
-           Interpretation             -
----------------------------------------

fait:habiter(locuteur,nancy)

----------------------------------------
-         Memoire de travail           -
----------------------------------------
_attendu (val@lieu)  0
_reponse (Nancy)  1
```

Annexe B

LE RAISONNEUR

Cette annexe contient :

- la représentation externe et interne des connaissances statiques du module RAISONNEUR,
- des traces d'exécution qui mettent en évidence les problèmes liés à la déduction (planification, incohérence),
- et des traces d'éxécution qui montrent l'évolution de l'état de la tâche au cours d'un dialogue.

1 Les connaissances statiques

Ces connaissances sont données sous la forme d'une représentation externe. Pour une information plus complète à propos de la syntaxe exacte de la représentation externe, nous renvoyons le lecteur à [MOUSEL 86b].

Ces connaissances sont divisées en quatre grandes catégories :

- les concepts,
- la base de règles,
- la base de données
- et la base de renseignements.

Nous reproduisons, ci-dessous, l'image des différents fichiers contenant la représentation externe de chacune de ces catégories.

1.1 Les concepts

Rappelons qu'il existe deux relations dans l'ensemble des concepts : *isa* et *arg* (voir chapitre 4). La relation *isa* est représentée à l'aide d'un parenthésage avec les symboles [et]. Quant à la relation *arg*, elle est donnée, pour chaque concept, par la liste des concepts entre parenthèses placée immédiatement derrière celui-ci.

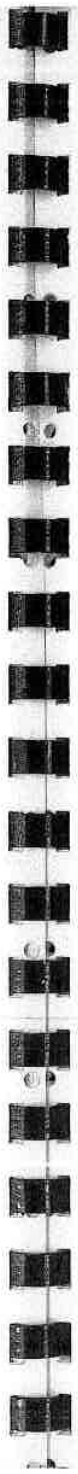
Pour notre application, cela se présente de la manière suivante :

```
/*-------------------------------------------------------*/
/*----------la partie declaration: concepts--------------*/
/*-------------------------------------------------------*/
/*S*/
inanime
[lieu
 [ville
  [nancy,
   vandoeuvre,
   villers,
   laxou,
   maxeville,
   malzeville,
   metz,
   paris
  ],
  departement
  [moselle,
   meurthe_et_moselle,
   meuse,
   vosges,
   region_parisienne
  ],
  commerce
  [bureau_de_tabac,
   super_marche
  ],
  pays
  [france,
   etranger
   [cee
    [belgique,
     luxembourg,
     rfa,
     italie,
     pays_bas,
     grande_bretagne
     [angleterre,
      ecosse,
      irlande,
      galles
     ],
     espagne,
     grece
    ],
    non_cee
    [usa,
     urss,
     madagascar,
     maroc,
     algerie
    ]
   ]
  ],
```



```
  domicile (personne,ville),
  lieu_event(evenement,ville),
  lieu_adm(administration,ville),
  administration
  [sp_p
   [prefecture,
    sous_prefecture
   ],
   mairie,
   hotel_de_ville,
   commissariat,
   gendarmerie,
   consulat[consulat_de_rfa, consulat_de_maroc, consulat_d_algerie],
   cira,
   cpam,
   asina,
   ass_consomateur,
   ophlm
  ],
  coordonnee(anime)
  [residence(ville),
   telephone(VAL),
   adresse(VAL,ville)
  ]
 ],
 document
 [timbre,
  photo,
  formulaire,
  lettre,
  certificat_medical,
  piece_administrative
  [carte_d_identite,
   passeport,
   carte_grise,
   fiche_d_etat_civil,
   livret_de_famille,
   carte_de_sejour,
   certificat_de_scolarite,
   autorisation_de_sortie,
   declaration_de_vol,
   declaration_de_perte,
   visa,
   carte_ss
  ]
 ],
 taxe
 [impot,
  tva,
  impots_locaux
 ]
],
/*S*/
anime
```

```
[personne
 [locuteur,
  machine,
  tiers
 ],
 animal
 [domestique
  [chien,
   chat,
   poisson
  ],
  sauvage
 ],
 administration
],
/*S*/
unite
[unite_prix
 [franc,
  centime
 ],
 unite_taille
 [metre,
  centimetre
 ],
 unite_temps
 [an,
  mois,
  jour,
  heure
 ]
],
/*S*/
mesure
[validite (document,VAL,an),
 delai(action_but,VAL,jour),
 age (personne,VAL,an),
 prix (inanime,VAL,unite_prix),
 gratuit (inanime),
 nb (inanime,VAL),
 taille(inanime,VAL,unite_taille),
 date(VAL)
],
/*S*/
identite (personne,VAL)
[nom,
 nom_de_jf
],
/*S*/
religion(personne)
[chretien,
 musulman,
 juif,
 athe,
```




```
 hindou,
 boudhiste
],
/*S*/
etat
[couleur (inanime)
 [bleu,
  rouge,
  jaune,
  rose,
  gris,
  noir,
  blanc
 ],
 qualificatif
 [ancien(document),
  nouveau(document),
  grand(inanime),
  moyen(inanime),
  petit(inanime),
  prorogable(document),
  renouvelable(document),
  perime(document),
  valide(document)
 ],
 etat_legal(personne)
 [responsabilite
  [majeur,
   mineur
  ],
  nationalite(pays),
  situation
  [marie,
   divorce,
   veuf,
   remarie,
   celibataire
  ],
  sexe
  [masculin,
   feminin
  ]
 ],
 changement(changeable)
],
/*S*/
evenement
[deces(personne),
 naissance(personne),
 mariage(personne),
 perte(document),
 vol(document)
],
/*S*/
```

```
action
[action_but
 [aller(personne,lieu),
  renouveler(personne,document),
  modifier(personne,document),
  etablir(administration,document),
  obtenir(personne,document),
  declarer(personne,evenement)
 ],
 action_administrative
 [autorisation(action),
  degrevement(taxe),
  recours(taxe),
  annulation(action),
  elire(personne,administration),
  prouver(document,etat)
 ],
 action_formalite
 [aller,
  remplir(personne,formulaire),
  munir(personne,document),
  fournir(personne,document),
  presenter(personne,piece_administrative),
  accompagner(personne,personne),
  joindre(anime,anime)
  [ecrire,
   appeler(telephone),
   rencontrer
  ]
 ],
 action_autre
 [ecouter(anime,anime),
  donner(anime,anime,inanime)
 ]
],
etre_departement(ville,departement),
obl_fac(action)
[obligatoire,
 facultatif
],
relation(anime,anime)
[famille
 [parent
  [pere,
   mere
  ],
  enfant
  [fils,
   fille
  ],
  fraternite
  [frere,
   soeur
  ],
```



```
  epoux
  [mari,
   epouse
  ]
 ],
 tuteur,
 connaissance
],
changeable
[coordonnee,
 situation,
 identite
]
```

1.2 La base de règles

Une règle est formée de deux formules séparées par le symbole **=>**. La partie droite représente la pré-condition et la partie gauche, la post-condition.

L'ensemble actuel des règles pour l'application est donné ci-dessous sous sa forme externe :

```
/*_____________________________________________________________*/
/*__________________Les regles_____________________________*/
/*_____________________________________________________________*/

/*___________________*/
/*Theme = cout       */
/*___________________*/

/*0*/ prix(inanime,"0",unite_prix) => gratuit(inanime);

/*___________________*/
/* Generalites       */
/*___________________*/

/*1*/ facultatif(action) => -obligatoire(action);
/*2*/ perime(document) => &(-vol(document),-perte(document));

/*___________________*/
/*Theme = nationalite*/
/*___________________*/

/*3*/ nationalite(personne,-france) => nationalite(personne,etranger);

         /* Une personne  dont la nationalite n'est pas francaise */
         /* est un etranger.                                      */

/*4*/ nationalite(personne,-etranger) => nationalite(personne,france);

         /* Une personne dont la nationalite n'est pas etrangere  */
         /* est francais.                                         */

/*5*/ nationalite(personne,etranger) => -nationalite(personne,france);
```

```
/*--------------------*/
/*Theme = age*/
/*--------------------*/

/*6*/ -majeur(personne) => mineur(personne);

          /* Une personne mineure n'est pas majeure.              */

/*7*/ -mineur(personne) => majeur(personne);

          /* Une personne mineure n'est pas majeure.              */

/*8*/ majeur(personne) => -mineur(personne);

/*9*/ &(nationalite(personne,france),marie(personne)) =>
       majeur(personne);
/*--------------------*/
/*Theme = relation*/
/*--------------------*/

/*10*/ parent(personne1,personne2) => enfant(personne2,personne1)
      avec: personne1#personne2;

/*11*/ pere(personne1,personne2) => masculin(personne1)
      avec: personne1#personne2;

/*12*/ mere(personne1,personne2) => feminin(personne1)
      avec: personne1#personne2;

/*13*/ fils(personne1,personne2) => masculin(personne1)
      avec: personne1#personne2;

/*14*/ fille(personne1,personne2) => feminin(personne1)
      avec: personne1#personne2;

/*15*/ soeur(personne1,personne2) => feminin(personne1)
      avec: personne1#personne2;

/*16*/ frere(personne1,personne2) => masculin(personne1)
      avec: personne1#personne2;

/*17*/ epouse(personne1,personne2) => feminin(personne1)
      avec: personne1#personne2;

/*18*/ mari(personne1,personne2) => masculin(personne1)
      avec: personne1#personne2;

/*19*/ famille(personne1,personne2) => famille(personne2,personne1)
      avec: personne1#personne2;

/*20*/ connaissance(personne1,personne2) => -famille(personne1,personne2)
      avec: personne1#personne2;
```



```
/*21*/ remarie(personne) => marie(personne1);

/*22*/ |(celibataire(personne),divorce(personne)) => -marie(personne);


/*--------------------*/
/*Theme = lieu       */
/*--------------------*/

/*24*/ adresse(anime,$x,ville) => residence(anime,ville);

/*--------------------*/
/*Theme = cout       */
/*--------------------*/

/*25*/ prix(inanime,"0",unite_prix) => gratuit(inanime)
```

1.3 La base de données

Cette structure contient à la fois les connaissances sur la tâche et les informations par défaut repérées par une étoile * en fin de formule.

Voici le contenu de cette base :

```
/*------------------------------------------------------------*/
/*-----------------La Base de Donnees-------------------------*/
/*------------------------------------------------------------*/

/*------------------------*/
/* FAIT-SYSTEMES          */
/*------------------------*/

/*1*/ date("850520");

          /* La date courante = 20 mai 1985.                  */

/*2*/ age(machine,"20",an);

/*------------------------*/
/* COUT                   */
/*------------------------*/

/*1*/ prix(carte_d_identite,"115",franc);

/*2*/ prix(carte_de_sejour,"0",franc);

/*------------------------*/
/* VALIDITE               */
/*------------------------*/

/*1*/ validite(carte_d_identite,"10",an);
```

```
/*2*/ &(validite(passeport,"5",an),prorogable(passeport));

/*3*/ &(validite(carte_de_sejour,"1",an),renouvelable(carte_de_sejour));

/*________________________*/
/* DELAI D'OBTENTION     */
/*________________________*/

/*1*/ delai(obtenir(personne,carte_d_identite),"10",jour);

/*________________________*/
/* ADRESSE et TELEPHONE  */
/*________________________*/

/*1*/ adresse(mairie,"place stanislas",nancy);

/*2*/ adresse(mairie,"bld des aiguilletes",villers);

/*3*/ &(adresse(cira,"100 rte de thionville",metz),
        telephone(cira,"87 31 91 91"));

/*4*/ &(adresse(prefecture,"1 rue M. Barres",nancy),
        telephone(prefecture,"83 35 61 20"));

/*5*/ adresse(commissariat,"bld Lobau",nancy);

/*6*/ &(adresse(ass_consomateur,"5 rue St. Leon",nancy),
        telephone(ass_consomateur,"83 28 02 68"));

/*7*/ &(adresse(consulat_de_rfa,"15 r Buthegnemont",nancy),
        telephone(consulat_de_rfa,"83 96 12 43"));


/*8*/ &(adresse(cpam,"a completer",nancy),
        telephone(cpam,"83 xx xx xx"));

/*____________________*/
/* DEPARTEMENT       */
/*____________________*/

/*1*/ etre_departement(nancy,meurthe_et_moselle);

/*2*/ etre_departement(vandoeuvre,meurthe_et_moselle);

/*3*/ etre_departement(metz,moselle);

/*____________________*/
/* DEFAUT             */
/*____________________*/

/*1*/ nationalite(locuteur,france)*
```


1.4 La base de renseignements

La base de renseignements renferme les connaissances structurées de l'univers de la tâche. Voici comment se présente actuellement cette base :

```
/*_____________________________________________________________*/
/*__________________Les renseignements_____________________*/
/*_____________________________________________________________*/

/*1*/  but: obtenir(personne1,carte_d_identite)
/*_____________________________________________*/
       si: nationalite(personne1,france)[2]
       alors:
            &(aller(personne1,|(mairie,commissariat)),
              munir(personne1,|(livret_de_famille,fiche_d_etat_civil)),
              fournir(personne1,&(photo,timbre,formulaire)),
              nb(photo,"2"),
              prix(timbre,"115",franc))
            si: mineur(personne1)[1]
            alors: &(accompagner(personne1,personne2),majeur(personne2))
            sinon:
       sinon:
      avec: personne1#personne2
;
         /* Pour obtenir une carte d'identite, si on est francais  */
         /* alors il faut aller a la mairie ou au commissariat;    */
         /* Pour les mineurs, il faut etre accompagne d'une autre  */
         /* personne majeure.                                      */
         /* Pour les non-francais, il n'y a pas de solution.       */

/*2*/ but:
!(obtenir(personne,carte_de_sejour),renouveler(personne,carte_de_sejour))
/*_____________________________________________________________________*/
      si: nationalite(personne,france)
      alors:
      sinon: &(aller(personne1,commissariat),
               domicile(personne1,ville),
               munir(personne,&(passeport,certificat_medical,
                                carte_ss,certificat_de_scolarite)))

;
/*3*/  but: renouveler(personne1,carte_d_identite1)
/*_____________________________________________*/
       &(aller(personne1,|(mairie,commissariat)),
         fournir(personne1,&(photo,timbre)),
         nb(photo,"2"),
         prix(timbre,"115",franc))
         si: |(perte(carte_d_identite2),vol(carte_d_identite2))
         alors: fournir(personne1,declaration_de_vol)
                si: mineur(personne1)
                alors: &(accompagner(personne1,personne2),majeur(personne2))
                sinon:
         sinon: &(fournir(personne1,carte_d_identite2),
                  perime(carte_d_identite2))
```

```
                si: mineur(personne1)
                alors: &(accompagner(personne1,personne2),majeur(personne2))
                sinon:
        avec: personne1#personne2
;

/*4*/ but: obtenir(personne,|(declaration_de_perte,declaration_de_vol))
/*-------------------------------------------------------------*/


        aller(personne,|(sp_p,commissariat,gendarmerie))
        si: lieu_event(|(vol(carte_d_identite),perte(carte_d_identite)),ville)
        alors: lieu_adm(|(sp_p,commissariat,gendarmerie),ville)
        sinon:
```

1.5 La compilation des connaissances

Le passage de la représentation externe vers la représentation interne est effectué par un compilateur, **cpdial**. Cet outil fournit, sous une forme relativement lisible la représentation interne des différentes bases que nous venons de citer.

Nous livrons, ci-dessous, le résultat de la visualisation des connaissances qui nous serviront de référence pour les exemples d'exécution qui vont suivre.

```
        ****************************
        * ETAT DES INFOS STATIQUES *
        ****************************

                              ======================================
                                        LA BASE DES REGLES
                              ======================================
Regle n.  0
.prix0[96] (inanime0[97]  , "0"[98]  , unite_prix0[99] )
=> .gratuit0[100] (inanime0[97] )

Regle n.  1
.facultatif0[101] (action0[102] )
=> .-obligatoire0[103] (action0[102] )

Regle n.  2
.perime0[104] (document0[105] )
=> .&[106](
-vol0[107] (document0[105] ) ,
-perte0[108] (document0[105] ))

Regle n.  3
.nationalite0[109] (personne0[110]  , -france0[111] )
=> .nationalite0[112] (personne0[110]  , etranger0[113] )

Regle n.  4
.nationalite0[114] (personne0[115]  , -etranger0[116] )
=> .nationalite0[117] (personne0[115]  , france0[118] )

Regle n.  5
```



```
.nationalite0[119] (personne0[120]  , etranger0[121] )
=> .-nationalite0[122] (personne0[120]  , france0[123] )

Regle n.  6
.-majeur0[124] (personne0[125] )
=> .mineur0[126] (personne0[125] )

Regle n.  7
.-mineur0[127] (personne0[128] )
=> .majeur0[129] (personne0[128] )

Regle n.  8
.majeur0[130] (personne0[131] )
=> .-mineur0[132] (personne0[131] )

Regle n.  9
.&[133](
nationalite0[134] (personne0[135]  , france0[136] ) ,
marie0[137] (personne0[135] ))
=> .majeur0[138] (personne0[135] )

Regle n. 10
.parent0[139] (personne1[140]  , personne2[141] )
=> .enfant0[142] (personne2[141]  , personne1[140] )

Regle n. 11
.pere0[143] (personne1[144]  , personne2[145] )
=> .masculin0[146] (personne1[144] )

Regle n. 12
.mere0[147] (personne1[148]  , personne2[149] )
=> .feminin0[150] (personne1[148] )

Regle n. 13
.fils0[151] (personne1[152]  , personne2[153] )
=> .masculin0[154] (personne1[152] )

Regle n. 14
.fille0[155] (personne1[156]  , personne2[157] )
=> .feminin0[158] (personne1[156] )

Regle n. 15
.soeur0[159] (personne1[160]  , personne2[161] )
=> .feminin0[162] (personne1[160] )

Regle n. 16
.frere0[163] (personne1[164]  , personne2[165] )
=> .masculin0[166] (personne1[164] )

Regle n. 17
.epouse0[167] (personne1[168]  , personne2[169] )
=> .feminin0[170] (personne1[168] )

Regle n. 18
```

```
.mari0[171] (personne1[172]  , personne2[173] )
=> .masculin0[174] (personne1[172] )

Regle n. 19
.famille0[175] (personne1[176]  , personne2[177] )
=> .famille0[178] (personne2[177]  , personne1[176] )

Regle n. 20
.connaissance0[179] (personne1[180]  , personne2[181] )
=> .-famille0[182] (personne1[180]  , personne2[181] )

Regle n. 21
.remarie0[183] (personne0[184] )
=> .marie0[185] (personne1[186] )

Regle n. 22
.|[187](
celibataire0[188] (personne0[189] ) ,
divorce0[190] (personne0[189] ))
=> .-marie0[191] (personne0[189] )

Regle n. 23
.adresse0[192] (anime0[193]  , [194]  , ville0[195] )
=> .residence0[196] (anime0[193]  , ville0[195] )

Regle n. 24
.prix0[197] (inanime0[198]  , "0"[199]  , unite_prix0[200] )
=> .gratuit0[201] (inanime0[198] )

Regle n. 25
.celibataire0[188] (personne0[189] )
=> .-marie0[191] (personne0[189] )

Regle n. 26
.divorce0[190] (personne0[189] )
=> .-marie0[191] (personne0[189] )
```

LA BASE DE DONNEES

```
Donnee n. 0
 .date0[1] ("850520"[2] )

Donnee n. 1
 .age0[3] (machine0[4]  , "20"[5]  , an0[6] )

Donnee n. 2
 .prix0[7] (carte_d_identite0[8]  , "115"[9]  , franc0[10] )

Donnee n. 3
 .prix0[11] (carte_de_sejour0[12]  , "0"[13]  , franc0[14] )
```



```
Donnee n. 4
 .validite0[15] (carte_d_identite0[16]  , "10"[17]  , an0[18] )

Donnee n. 5
 .&[19](
validite0[20] (passeport0[21]  , "5"[22]  , an0[23] ) ,
prorogable0[24] (passeport0[21] ))

Donnee n. 6
 .&[25](
validite0[26] (carte_de_sejour0[27]  , "1"[28]  , an0[29] ) ,
renouvelable0[30] (carte_de_sejour0[27] ))

Donnee n. 7
 .delai0[31] (obtenir0[32] (personne0[33]  , carte_d_identite0[34] ) , "10"[35]
  , jour0[36] )

Donnee n. 8
 .adresse0[37] (mairie0[38]  , "place stanislas"[39]  , nancy0[40] )

Donnee n. 9
 .adresse0[41] (mairie0[42]  , "bld des aiguilletes"[43]  , villers0[44] )

Donnee n.10
 .&[45](
adresse0[46] (cira0[47]  , "100 rte de thionville"[48]  , metz0[49] ) ,
telephone0[50] (cira0[47]  , "87 31 91 91"[51] ))

Donnee n.11
 .&[52](
adresse0[53] (prefecture0[54]  , "1 rue M. Barres"[55]  , nancy0[56] ) ,
telephone0[57] (prefecture0[54]  , "83 35 61 20"[58] ))

Donnee n.12
 .adresse0[59] (commissariat0[60]  , "bld Lobau"[61]  , nancy0[62] )

Donnee n.13
 .&[63](
adresse0[64] (ass_consomateur0[65]  , "5 rue St. Leon"[66]  , nancy0[67] ) ,
telephone0[68] (ass_consomateur0[65]  , "83 28 02 68"[69] ))

Donnee n.14
 .&[70](
adresse0[71] (consulat_de_rfa0[72]  , "15 r Buthegnemont"[73]  , nancy0[74] ) ,
telephone0[75] (consulat_de_rfa0[72]  , "83 96 12 43"[76] ))

Donnee n.15
 .&[77](
adresse0[78] (cpam0[79]  , "a completer"[80]  , nancy0[81] ) ,
telephone0[82] (cpam0[79]  , "83 xx xx xx"[83] ))

Donnee n.16
 .etre_departement0[84] (nancy0[85]  , meurthe_et_moselle0[86] )
```

```
Donnee n.17
 .etre_departement0[87] (vandoeuvre0[88]  , meurthe_et_moselle0[89] )

Donnee n.18
 .etre_departement0[90] (metz0[91]  , moselle0[92] )

Donnee n.19
 .nationalite0[93] (locuteur0[94]  , france0[95] )
```

LA BASE DES RENSEIGNEMENTS

```
Rens  0:
but: .obtenir0[202] (personne1[203]  , carte_d_identite0[204] )
si:
V
alors:
 si:
 .nationalite0[205] (personne1[203]  , france0[206] )
 alors:.|[207](
 &[208](
 aller0[209] (personne1[203]  , mairie0[210] ) ,
 munir0[211] (personne1[203]  , livret_de_famille0[212] ) ,
 nb0[213] (photo0[214]  , "2"[215] ) ,
 prix0[216] (timbre0[217]  , "115"[218]  , franc0[219] ) ,
 fournir0[220] (personne1[203]  , photo0[214] ) ,
 fournir0[221] (personne1[203]  , timbre0[217] ) ,
 fournir0[222] (personne1[203]  , formulaire0[223] ))  ,
 &[224](
 aller0[209] (personne1[203]  , mairie0[210] ) ,
 munir0[225] (personne1[203]  , fiche_d_etat_civil0[226] ) ,
 nb0[213] (photo0[214]  , "2"[215] ) ,
 prix0[216] (timbre0[217]  , "115"[218]  , franc0[219] ) ,
 fournir0[220] (personne1[203]  , photo0[214] ) ,
 fournir0[221] (personne1[203]  , timbre0[217] ) ,
 fournir0[222] (personne1[203]  , formulaire0[223] ))  ,
 &[227](
 aller0[228] (personne1[203]  , commissariat0[229] ) ,
 munir0[211] (personne1[203]  , livret_de_famille0[212] ) ,
 nb0[213] (photo0[214]  , "2"[215] ) ,
 prix0[216] (timbre0[217]  , "115"[218]  , franc0[219] ) ,
 fournir0[220] (personne1[203]  , photo0[214] ) ,
 fournir0[221] (personne1[203]  , timbre0[217] ) ,
 fournir0[222] (personne1[203]  , formulaire0[223] ))  ,
 &[230](
 aller0[228] (personne1[203]  , commissariat0[229] ) ,
 munir0[225] (personne1[203]  , fiche_d_etat_civil0[226] ) ,
 nb0[213] (photo0[214]  , "2"[215] ) ,
 prix0[216] (timbre0[217]  , "115"[218]  , franc0[219] ) ,
 fournir0[220] (personne1[203]  , photo0[214] ) ,
```



```
 fournir0[221] (personne1[203]  , timbre0[217] ) ,
 fournir0[222] (personne1[203]  , formulaire0[223] )) )
  si:
  .mineur0[231] (personne1[203] )
  alors:.&[232](
  accompagner0[233] (personne1[203]  , personne2[234] ) ,
  majeur0[235] (personne2[234] ))
  sinon:
 sinon:
sinon:
```

```
Rens  2:
but: .renouveler0[257] (personne1[258]  , carte_d_identite1[259] )
si:
V
alors:.|[260](
&[261](
nb0[262] (photo0[263]  , "2"[264] ) ,
prix0[265] (timbre0[266]  , "115"[267]  , franc0[268] ) ,
fournir0[269] (personne1[258]  , photo0[263] ) ,
fournir0[270] (personne1[258]  , timbre0[266] ))  ,
&[271](
nb0[262] (photo0[263]  , "2"[264] ) ,
prix0[265] (timbre0[266]  , "115"[267]  , franc0[268] ) ,
fournir0[269] (personne1[258]  , photo0[263] ) ,
fournir0[270] (personne1[258]  , timbre0[266] )) )
 si:
 .perte0[273] (carte_d_identite2[274] )
 alors:.fournir0[276] (personne1[258]  , declaration_de_vol0[277] )
  si:
  .mineur0[278] (personne1[258] )
  alors:.&[279](
  accompagner0[280] (personne1[258]  , personne2[281] ) ,
  majeur0[282] (personne2[281] ))
  sinon:
 sinon:.&[283](
 fournir0[284] (personne1[258]  , carte_d_identite2[274] ) ,
 perime0[285] (carte_d_identite2[274] ))
  si:
  .mineur0[286] (personne1[258] )
  alors:.&[287](
  accompagner0[288] (personne1[258]  , personne2[281] ) ,
  majeur0[289] (personne2[281] ))
  sinon:
sinon:
```

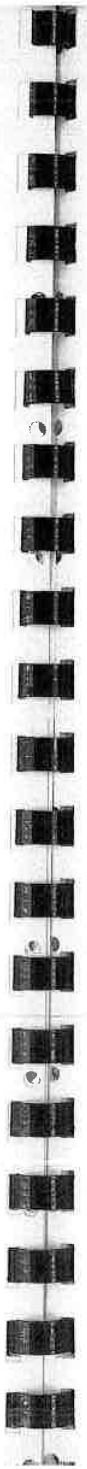
```
Rens  4:
but: .obtenir0[237] (personne0[238]  , carte_de_sejour0[239] )
si:
V
```

```
alors:
 si:
 .nationalite0[241] (personne0[238]  , france0[242] )
 alors:
 sinon:.&[243](
 aller0[244] (personne1[245]  , commissariat0[246] ) ,
 domicile0[247] (personne1[245]  , ville0[248] ) ,
 munir0[249] (personne0[238]  , passeport0[250] ) ,
 munir0[251] (personne0[238]  , certificat_medical0[252] ) ,
 munir0[253] (personne0[238]  , carte_ss0[254] ) ,
 munir0[255] (personne0[238]  , certificat_de_scolarite0[256] ))
sinon:
```

```
Rens  5:
but: .renouveler0[240] (personne0[238]  , carte_de_sejour0[239] )
si:
V
alors:
 si:
 .nationalite0[241] (personne0[238]  , france0[242] )
 alors:
 sinon:.&[243](
 aller0[244] (personne1[245]  , commissariat0[246] ) ,
 domicile0[247] (personne1[245]  , ville0[248] ) ,
 munir0[249] (personne0[238]  , passeport0[250] ) ,
 munir0[251] (personne0[238]  , certificat_medical0[252] ) ,
 munir0[253] (personne0[238]  , carte_ss0[254] ) ,
 munir0[255] (personne0[238]  , certificat_de_scolarite0[256] ))
sinon:
```

```
Rens  6:
but: .obtenir0[291] (personne0[292]  , declaration_de_perte0[293] )
si:
V
alors:.|[296](
aller0[297] (personne0[292]  , sp_p0[298] ) ,
aller0[299] (personne0[292]  , commissariat0[300] ) ,
aller0[301] (personne0[292]  , gendarmerie0[302] ))
 si:
 .lieu_event0[304] (vol0[305] (carte_d_identite0[306] ) , ville0[307] )
 alors:.|[310](
 lieu_adm0[311] (sp_p0[298]  , ville0[307] ) ,
 lieu_adm0[312] (commissariat0[300]  , ville0[307] ) ,
 lieu_adm0[313] (gendarmerie0[302]  , ville0[307] ))
 sinon:
sinon:
```

```
Rens  7:
```




```
but: .obtenir0[294] (personne0[292]  , declaration_de_vol0[295] )
si:
V
alors:.|[296](
aller0[297] (personne0[292]  , sp_p0[298] ) ,
aller0[299] (personne0[292]  , commissariat0[300] ) ,
aller0[301] (personne0[292]  , gendarmerie0[302] ))
 si:
 .lieu_event0[304] (vol0[305] (carte_d_identite0[306] ) , ville0[307] )
 alors:.|[310](
 lieu_adm0[311] (sp_p0[298]  , ville0[307] ) ,
 lieu_adm0[312] (commissariat0[300]  , ville0[307] ) ,
 lieu_adm0[313] (gendarmerie0[302]  , ville0[307] ))
 sinon:
sinon:
```

```
Rens  8:
but: .renouveler0[257] (personne1[258]  , carte_d_identite1[259] )
si:
V
alors:.|[260](
&[261](
nb0[262] (photo0[263]  , "2"[264] ) ,
prix0[265] (timbre0[266]  , "115"[267]  , franc0[268] ) ,
fournir0[269] (personne1[258]  , photo0[263] ) ,
fournir0[270] (personne1[258]  , timbre0[266] ))  ,
&[271](
nb0[262] (photo0[263]  , "2"[264] ) ,
prix0[265] (timbre0[266]  , "115"[267]  , franc0[268] ) ,
fournir0[269] (personne1[258]  , photo0[263] ) ,
fournir0[270] (personne1[258]  , timbre0[266] )) )
 si:
 .vol0[275] (carte_d_identite2[274] )
 alors:.fournir0[276] (personne1[258]  , declaration_de_vol0[277] )
  si:
  .mineur0[278] (personne1[258] )
  alors:.&[279](
  accompagner0[280] (personne1[258]  , personne2[281] ) ,
  majeur0[282] (personne2[281] ))
  sinon:
 sinon:.&[283](
 fournir0[284] (personne1[258]  , carte_d_identite2[274] ) ,
 perime0[285] (carte_d_identite2[274] ))
  si:
  .mineur0[286] (personne1[258] )
  alors:.&[287](
  accompagner0[288] (personne1[258]  , personne2[281] ) ,
  majeur0[289] (personne2[281] ))
  sinon:
sinon:
```

```
Rens  9:
but: .obtenir0[291] (personne0[292]  , declaration_de_perte0[293] )
si:
V
alors:.|[296](
aller0[297] (personne0[292]  , sp_p0[298] ) ,
aller0[299] (personne0[292]  , commissariat0[300] ) ,
aller0[301] (personne0[292]  , gendarmerie0[302] ))
 si:
 .lieu_event0[308] (perte0[309] (carte_d_identite0[306] ) , ville0[307] )
 alors:.|[310](
 lieu_adm0[311] (sp_p0[298]  , ville0[307] ) ,
 lieu_adm0[312] (commissariat0[300]  , ville0[307] ) ,
 lieu_adm0[313] (gendarmerie0[302]  , ville0[307] ))
 sinon:
sinon:

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

Rens 10:
but: .obtenir0[294] (personne0[292]  , declaration_de_vol0[295] )
si:
V
alors:.|[296](
aller0[297] (personne0[292]  , sp_p0[298] ) ,
aller0[299] (personne0[292]  , commissariat0[300] ) ,
aller0[301] (personne0[292]  , gendarmerie0[302] ))
 si:
 .lieu_event0[308] (perte0[309] (carte_d_identite0[306] ) , ville0[307] )
 alors:.|[310](
 lieu_adm0[311] (sp_p0[298]  , ville0[307] ) ,
 lieu_adm0[312] (commissariat0[300]  , ville0[307] ) ,
 lieu_adm0[313] (gendarmerie0[302]  , ville0[307] ))
 sinon:
sinon:

-------------------------------------------------------------------------
```

2 Quelques exemples

Nous allons maintenant présenter plusieurs exemples afin d'illustrer les points suivants :

- la déduction,
- la planification,
- l'incohérence,
- l'influence du score et des valeurs par défaut,
- la mise à jour de l'état de la tâche.

Mais il nous semble indispensable de commenter auparavant les notations utilisées pour tester le raisonneur assossié à la commande rx.

2.1 Notations

Prompts : L'introduction d'une information à destination du raisonneur se trouve derrière le *prompt* `(loc)->`. Le *prompt* `(syst)->` précède la réponse fournie par le système.

Les références aux noeuds (généralement entre crochets `[]`) et aux connaissances (exemple : numéros de règles) se font par rapport aux bases de connaissances (voir plus haut) et aux informations dynamiques correspondant à la visualisation de l'état des infos dynamiques.

Les règles activables sont représentées par :

- la référence à la règle concernée et un rang séparés par le caractère **#**,
- et l'ensemble des unificateurs qui sont des couples de noeuds.

Le plan est une arborescence dont les noeuds sont des références aux règles activables. Le symbole * indique le chemin courant parcouru depuis la racine.

L'état des infos dynamiques est une visualisation de l'ensemble des questions, de l'ensemble des buts et de la base des faits.

L'état de la tâche permet de visualiser de manière syntéthique les différentes informations disponibles pour le gestionnaire du dialogue.

2.2 Exemple 1 : déduction

Commentaires : Dans cet exemple, l'arrivée de l'énoncé :

`nationalite(locuteur,luxembourg)`

entraîne l'activation de la règle 5 et la déduction de :

`-nationalite(locuteur,france)`

dans la base des faits.

Trace :

`(loc)->` *`nationalite(locuteur,luxembourg).`*

```
***Suppression du fait 26 (defaut)
                              ========================================
                                    Nouvelles regles activables
                              ========================================
  5#10 ([343],[120])([344],[121])([342],[119])

                              ========================================
                                           PLAN
                              ========================================
 5#10*
        ****************************
        * ETAT DES INFO DYNAMIQUES *
        ****************************

                              ========================================
                                    ENSEMBLE DE QUESTIONS
                              ========================================
vide
```

```
                                  ========================================
                                          L'ENSEMBLE DES BUTS
                                  ========================================
vide

                                  ========================================
                                          LA BASE DES FAITS:
                                  ========================================
 (37) [sc=100] .nationalite0[342] (locuteur0[343]  , luxembourg0[344] )
 (38) [sc=100] .-nationalite0[345] (locuteur0[343]  , france0[347] )

(loc)->stop:.

(syst)->Au revoir
```

2.3 Exemple 2 : planification

Commentaires : Dans cet exemple, à la suite de la quatrième intervention du locuteur, il existe deux ensembles d'unificateurs mutuellement exclusifs qui permettent d'activer la règle 9. Comme on peut le constater à travers le plan, un choix arbitraire a été effectué.

Trace :

```
(loc)-> frere(locuteur1,locuteur2).

                                  ========================================
                                        Nouvelles regles activables
                                  ========================================
 16#10 ([343],[164])([344],[165])([342],[163])
 19#11 ([343],[176])([344],[177])([342],[175])

                                  ========================================
                                                  PLAN
                                  ========================================
16#10-19#11*
          ****************************
          * ETAT DES INFO DYNAMIQUES *
          ****************************

                                  ========================================
                                        ENSEMBLE DE QUESTIONS
                                  ========================================
vide

                                  ========================================
                                          L'ENSEMBLE DES BUTS
                                  ========================================
vide

                                  ========================================
                                          LA BASE DES FAITS:
                                  ========================================
 (37) [sc=100] .frere0[342] (locuteur1[343]  , locuteur2[344] )
```



```
 (38) [sc=100] .masculin0[345] (locuteur1[343] )
 (39) [sc=100] .famille0[347] (locuteur2[344]  , locuteur1[343] )

(loc)-> marie([343])

                                  ========================================
                                        Nouvelles regles activables
                                  ========================================
  9#12 ([94],[135])([95],[136])([93],[134])([343],[135])([350],[137])
  8#13 ([352],[131])([351],[130])
  7#14 ([354],[128])([353],[127])

                                  ========================================
                                                  PLAN
                                  ========================================
16#10-19#11- 9#12- 8#13- 7#14*

(loc)-> marie([344]).

                                  ========================================
                                        Nouvelles regles activables
                                  ========================================

                                  ========================================
                                                  PLAN
                                  ========================================
16#10-19#11- 9#12- 8#13- 7#14*
          ****************************
          * ETAT DES INFO DYNAMIQUES *
          ****************************

                                  ========================================
                                        ENSEMBLE DE QUESTIONS
                                  ========================================
vide

                                  ========================================
                                          L'ENSEMBLE DES BUTS
                                  ========================================
vide

                                  ========================================
                                          LA BASE DES FAITS:
                                  ========================================
 (37) [sc=100] .frere0[342] (locuteur0[343]  , locuteur0[344] )
 (38) [sc=100] .masculin0[345] (locuteur1[94] )
 (39) [sc=100] .famille0[347] (locuteur2[344]  , locuteur1[94] )
 (40) [sc=100] .marie0[350] (locuteur0[343] )
*(41) [sc=100] .majeur0[351] (locuteur0[94] )
*(42) [sc=100] .-mineur0[353] (locuteur0[94] )
*(43) [sc=100] .majeur0[355] (locuteur0[94] )
 (44) [sc=100] .marie0[357] (locuteur0[344] )

(loc)-> nationalite(locuteur,france).
```

```
                                  ========================================
                                        Nouvelles regles activables
                                  ========================================
  9#15 ([343],[135])([350],[137])([359],[135])([360],[136])([358],[134])
  9#16 ([344],[135])([357],[137])([359],[135])([360],[136])([358],[134])
  8#17 ([362],[131])([361],[130])
  7#18 ([364],[128])([363],[127])

                                  ========================================
                                                PLAN
                                  ========================================
16#10-19#11- 9#16- 8#17- 7#18-
            |
             9#12- 8#13- 7#14- 9#15- 8#17- 7#18*
          ****************************
          * ETAT DES INFO DYNAMIQUES *
          ****************************

                                  ========================================
                                        ENSEMBLE DE QUESTIONS
                                  ========================================
vide

                                  ========================================
                                        L'ENSEMBLE DES BUTS
                                  ========================================
vide

                                  ========================================
                                         LA BASE DES FAITS:
                                  ========================================
 (37) [sc=100] .frere0[342] (locuteur0[343]  , locuteur0[344] )
 (38) [sc=100] .masculin0[345] (locuteur1[94] )
 (39) [sc=100] .famille0[347] (locuteur2[344]  , locuteur1[94] )
 (40) [sc=100] .marie0[350] (locuteur0[343] )
*(41) [sc=100] .majeur0[351] (locuteur0[94] )
*(42) [sc=100] .-mineur0[353] (locuteur0[94] )
*(43) [sc=100] .majeur0[355] (locuteur0[94] )
 (44) [sc=100] .marie0[357] (locuteur0[344] )
 (45) [sc=100] .nationalite0[358] (locuteur0[359]  , france0[360] )
 (46) [sc=100] .majeur0[361] (locuteur0[94] )
 (47) [sc=100] .-mineur0[363] (locuteur0[94] )
 (48) [sc=100] .majeur0[365] (locuteur0[94] )

(loc)->stop:.

(syst)->Au revoir
```

2.4 Exemple 3 : incohérence

Commentaires :

Cet exemple met en évidence la manière de détecter une incohérence dans les énoncés.

Cette détection coincide avec l'apparition d'une contradiction dans la base des faits. En effet, on peut remarquer que le fait (38) :

`-nationalite(locuteur,france)`

lui-même déduit du fait (37) :

`nationalite(locuteur,luxembourg)`

contredit le fait énoncé (39) :

`nationalite(locuteur,france)`.

L'incohérence est alors notée dans l'état de la tâche.

Trace :

```
(loc)-> nationalite(locuteur,luxembourg).

***Suppression du fait 26 (defaut)
          ****************************
          * ETAT DES INFO DYNAMIQUES *
          ****************************

                                  ========================================
                                        ENSEMBLE DE QUESTIONS
                                  ========================================
vide

                                  ========================================
                                        L'ENSEMBLE DES BUTS
                                  ========================================
vide

                                  ========================================
                                         LA BASE DES FAITS:
                                  ========================================
 (37) [sc=100] .nationalite0[342] (locuteur0[343]  , luxembourg0[344] )
 (38) [sc=100] .-nationalite0[345] (locuteur0[343]  , france0[347] )
          ****************************
          *    ETAT DE LA TACHE      *
          ****************************
------------------------------

INCOHERENCE:
**ras
------------------------------
QUESTION:
**Pas de question
------------------------------
BUT:
**but non necessaire pour la reponse
------------------------------
REPONSE:
** pas de reponse:
------------------------------
DEFAUT:
**pas de valeur par defaut
------------------------------
```

```
(loc)-> nationalite(locuteur,france).
         ****************************
         * ETAT DES INFO DYNAMIQUES *
         ****************************

                                 ========================================
                                          ENSEMBLE DE QUESTIONS
                                 ========================================

vide

                                 ========================================
                                          L'ENSEMBLE DES BUTS
                                 ========================================

vide

                                 ========================================
                                          LA BASE DES FAITS:
                                 ========================================
 (37) [sc=100] .nationalite0[342] (locuteur0[343]  , luxembourg0[344] )
 (38) [sc=100] .-nationalite0[345] (locuteur0[343]  , france0[347] )
 (39) [sc=100] .nationalite0[348] (locuteur0[349]  , france0[350] )
         ****************************
         *    ETAT DE LA TACHE      *
         ****************************
------------------------------

INCOHERENCE:
**incoherence entre:
.nationalite0[348] (locuteur0[349]  , france0[350] )
.-nationalite0[345] (locuteur0[343]  , france0[347] )
------------------------------
QUESTION:
**Pas de question
------------------------------
BUT:
**but non necessaire pour la reponse
------------------------------
REPONSE:
** pas de reponse:
------------------------------
DEFAUT:
**pas de valeur par defaut
------------------------------

(loc)->stop:.

(syst)->Au revoir
```

2.5 Exemple 4 : valeurs par défaut

Commentaires :

Examinons le début des traces des exemples 1 et 3 ; nous pouvons remarquer le message

suivant :

```
***Suppression du fait 26 (defaut)
```

En effet, il s'agit de la suppression de la valeur par défaut correspondant à la **Donnee: 19** :

```
nationalite0[93] (locuteur0[94]  , france0[95] ).
```

Ceci illustre le fait qu'une valeur par défaut ne peut en aucun cas engendrer une situation d'incohérence.

2.6 Exemple 5 : scores

Commentaires :

Rappelons que le score d'une information traduit sa mesure de bonne compréhension.

L'exemple suivant montre l'initiative que prend le RAISONNEUR d'ignorer une information de faible score lorque celle-ci est en condradition avec une information de score beaucoup plus élevé. Dans notre cas, cela se manifeste par la suppression du fait (37), empêchant de ce fait l'apparition d'une incohérence (voir état de la tâche).

Trace :

```
(loc)-> mineur(locuteur)[%10].

         ****************************
         * ETAT DES INFO DYNAMIQUES *
         ****************************

                                 ========================================
                                          ENSEMBLE DE QUESTIONS
                                 ========================================

vide

                                 ========================================
                                          L'ENSEMBLE DES BUTS
                                 ========================================

vide

                                 ========================================
                                          LA BASE DES FAITS:
                                 ========================================
 (37) [sc= 10] .mineur0[342] (locuteur0[343] )

         ****************************
         *    ETAT DE LA TACHE      *
         ****************************
------------------------------

INCOHERENCE:
**ras
------------------------------
```

```
QUESTION:
**Pas de question
------------------------------
BUT:
**but non necessaire pour la reponse
------------------------------
REPONSE:
** pas de reponse:
------------------------------
DEFAUT:
**pas de valeur par defaut
------------------------------

(loc)-> majeur(locuteur)[%80].

***Suppression du fait 37 (score)

          ****************************
          * ETAT DES INFO DYNAMIQUES *
          ****************************

                                ======================================
                                        ENSEMBLE DE QUESTIONS
                                ======================================
vide

                                ======================================
                                        L'ENSEMBLE DES BUTS
                                ======================================
vide

                                ======================================
                                        LA BASE DES FAITS:
                                ======================================
 (38) [sc= 80] .majeur0[344] (locuteur0[345] )
 (39) [sc= 80] .-mineur0[346] (locuteur0[345] )
 (40) [sc= 80] .majeur0[348] (locuteur0[345] )

          ****************************
          *    ETAT DE LA TACHE      *
          ****************************
------------------------------

INCOHERENCE:
**ras
------------------------------
QUESTION:
**Pas de question
```



```
------------------------------
BUT:
**but non necessaire pour la reponse
------------------------------
REPONSE:
** pas de reponse:
------------------------------
DEFAUT:
**pas de valeur par defaut
------------------------------

(loc)->stop:.

(syst)->Au revoir
```

2.7 Exemple 6 : mise à jour de l'état de la tâche

Commentaires :

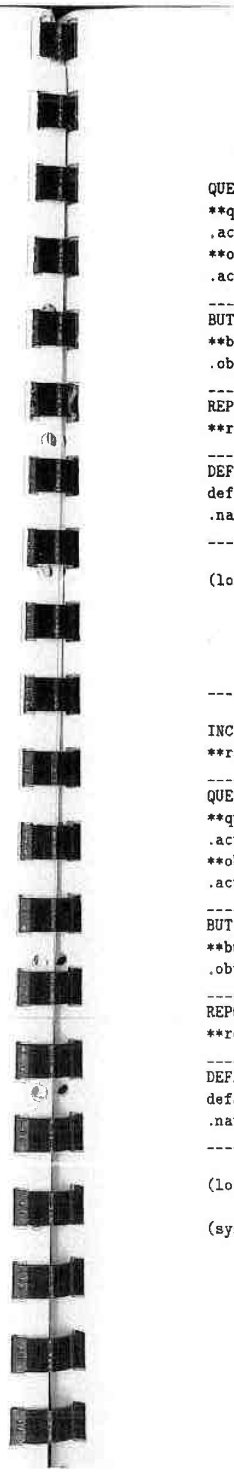
L'exemple ci-après a pour objectif de montrer l'évolution de l'état de la tâche en fonction des interventions successives dans un dialogue. Examinons cette évolution énoncé par énoncé :

1. `(loc)->`.
   Il s'agit de l'enoncé vide qui signifie qu'aucune information n'a été extraite de l'énoncé. Elle permet de visualiser l'état initial de la tâche.
2. `(loc)->`*but: obtenir(locuteur,carte-d-identite).*
   Cet énoncé constitue la partie *but* de la requête ; cependant il manque toujours la partie *question.*
3. `(loc)->`*?action-formalite.*
   La question est maintenant précisée ; une recherche de la réponse indique qu'il manque une (des) information(s) pour avoir une réponse complète dans la base de renseignements *BX* (réponse conditionnelle).
4. `(loc)->`*mineur(locuteur).*
   Si on examine la base de renseignements concernant le but actif courant, on s'aperçoit que l'information manquante est `mineur(locuteur)` ou non. Cet énoncé vient compléter la requête et une réponse est alors proposée.
   On remarque que l'élaboration de cette réponse a nécessité l'utilisation effective de la valeur par défaut `nationalite(locuteur,france)`
5. `(loc)->`*stop:.*
   On termine de cette façon la transaction.

Trace :

```
(loc)-> .

          ****************************
```

```
          *    ETAT DE LA TACHE      *
          ****************************
 ------------------------------

 INCOHERENCE:
 **ras
 ------------------------------
 QUESTION:
 **Pas de question
 ------------------------------
 BUT:
 **but non necessaire pour la reponse
 ------------------------------
 REPONSE:
 ** pas de reponse:
 ------------------------------
 DEFAUT:
 **pas de valeur par defaut
 ------------------------------

 (loc)-> but: obtenir(locuteur,carte-d-identite).

          ****************************
          *    ETAT DE LA TACHE      *
          ****************************
 ------------------------------

 INCOHERENCE:
 **ras
 ------------------------------
 QUESTION:
 **Pas de question
 ------------------------------
 BUT:
 **but actif:
 .obtenir0[202] (personne1[203]  , carte_d_identite0[204] )
 ------------------------------
 REPONSE:
 ** pas de reponse:
 ------------------------------
 DEFAUT:
 **pas de valeur par defaut
 ------------------------------

 (loc)-> ?action-formalite.

          ****************************
          *    ETAT DE LA TACHE      *
          ****************************
 ------------------------------

 INCOHERENCE:
 **ras
 ------------------------------
```




```
 QUESTION:
 **question ouverte:
 .action_formalite0[345]
 **objet de la question:
 .action_formalite0[345]
 ------------------------------
 BUT:
 **but actif:
 .obtenir0[202] (personne1[203]  , carte_d_identite0[204] )
 ------------------------------
 REPONSE:
 **reponse conditionnelle:
 ------------------------------
 DEFAUT:
 defaut:
 .nationalite0[205] (personne1[203]  , france0[206] )
 ------------------------------

 (loc)-> mineur(locuteur).

          ****************************
          *    ETAT DE LA TACHE      *
          ****************************
 ------------------------------

 INCOHERENCE:
 **ras
 ------------------------------
 QUESTION:
 **question ouverte:
 .action_formalite0[345]
 **objet de la question:
 .action_formalite0[345]
 ------------------------------
 BUT:
 **but actif:
 .obtenir0[202] (personne1[203]  , carte_d_identite0[204] )
 ------------------------------
 REPONSE:
 **reponse complete dans BK:
 ------------------------------
 DEFAUT:
 defaut:
 .nationalite0[205] (personne1[203]  , france0[206] )
 ------------------------------

 (loc)-> stop:.

 (syst)->Au revoir
```

Annexe C

LE GESTIONNAIRE DE DIALOGUE

Cette annexe contient :

- les différents schémas actuellement utilisés,
- l'ensemble des énoncés figés qui servent de réponses à la machine,
- les procédures de réponses associées aux conditions dans la base de renseignements,
- des traces d'exécution sur des exemples théoriques de dialogues.

1 Les schémas de dialogue

Nom : DIALOGUE
Variable :
c : compteur;

Procédure :
SLOGAN;
acquérir();
PHASE1;
dialogue1:
    REQUETE
    si ∃ "au revoir"
    alors PHASE3;
    sinon
        RELANCE;
        acquérir();
        si refus
        alors PHASE3;
        sinon
            c ← 0;
            tant que (c < 3) et (non ∃ énoncé portant sur la tâche)
            faire
                MAINTIEN;
                acquérir();
                incrémenter c;

si (c ≥ 3) INTERRUPTION; fsi
fait
fsi
fsi
aller à dialogue1;

Nom : AMBIGUITE

Procédure :
dire(message$_8$);
construire la prédiction interne;
prédire(lex,confirmation ou infirmation);
prédire(syn,apport informations);

Nom : AU-REVOIR

Procédure :
dire(message$_4$);
prédire(lex et syn,"clôture");

Nom : BONJOUR

Procédure :
dire(message$_9$);

Nom : BUT-INCOMPLET

Procédure :
dire(message$_5$);

Nom : BUT-INEXISTANT

Procédure :
dire(message$_{15}$);

Nom : CONFIRMATION

Procédure :
dire(message$_8$):
construire la prédiction interne;
prédire(lex,confirmation ou infirmation);
prédire(syn,apport informations);

Nom : CONFIRMATION-COHER

Procédure :

cas
∃ une confirmation
alors
ajouter un fait = confirmation de la dernière question
(activation du raisonneur)
∃ une infirmation
ajouter un fait = infirmation de la dernière question
activation du raisonneur)
fcas

Nom : CONFIRMATION-TACHE

Procédure :
cas
∃ une confirmation
alors
ajouter un fait = confirmation de la dernière prédiction interne
(activation du raisonneur)
∃ une infirmation
ajouter un fait = infirmation de la dernière prédiction interne
activation du raisonneur)
fcas

Nom : DEFAUT

Procédure :
dire(message$_{18}$);
dire(*Les valeurs par défaut de l'état de la tâche*);

Nom : INTERRUPTION

Procédure :
dire(message$_{14}$);
PHASE3;

Nom : LIMITE-SYSTEME

Procédure :
dire(message$_2$);

Nom : MAINTIEN

Procédure :
dire(message$_1$);
prédire(syn,"demande ou apport d'informations");

Nom : PAS-DE-REPONSE

Procédure :

si ∃ question fermée
alors
dire(message$_{12}$);
sinon
dire(message$_{13}$);

Nom : PHASE1
Variable :
c : compteur;

Procédure :
c ← 0;
tant que (non ∃ énoncé portant sur la tâche)
faire
si (c ≥ 3)
alors INTERRUPTION;
sinon
si (*exists* "bonjour") alors BONJOUR; fsi
fsi
incrémenter c;
SLOGAN; fait

Nom : PHASE3

Procédure :
AU-REVOIR;
exit();

Nom : QUEST-INEXISTANT

Procédure :
dire(message$_{16}$);

Nom : REQUETE
Variable :

Procédure :
requete1:
REQUETE
si ∃ un nouveau but
alors supprimer tous les autres buts
fsi
TACHE;
cas
1) ∃ une réponse (complète, conditionnelle, approximative ou à la limite des connaissances)
alors
FinDeProcédure;
2) ∃ une question fermée à la suite d'une incohérence
alors
acquérir():

CONFIRMATION-COHER;
3) ∃ une question fermée sur la tâche
alors
acquérir();
CONFIRMATION-TACHE;
4) ∃ une question ouverte sur la tâche
alors
acquérir();
VALEUR-TACHE;
fcas aller à requete1;

Nom : RELANCE;

Procédure :
dire(message$_3$);
prédire(lex,"confirmation infirmation");
prédire(syn,"demande d'informations, satisfaction, clôture");

Nom : SATISFAIRE-BD

Procédure :
dire(message$_{17}$);
prédire(lex,"satisfaction");
prédire(syn,"satisfaction ou contestation ou demande informations");

Nom : SATISFAIRE-RS

Procédure :
dire(message$_{17}$);
prédire(lex,"satisfaction");
prédire(syn,"satisfaction ou contestation ou demande informations");

Nom : SLOGAN

Procédure :
dire(message$_0$);
prédire(lex,"salutation+requete");
prédire(syn,"premier énoncé");

Nom : TACHE

Procédure :
Analyse de l'Etat de la Tâche
si *exists* une question active
si *exists* un but actif
alors
si *exists* une incohérence
alors
AMBIGUITE;

```
        sinon
            cas (examen du champ réponse de l'état de la tâche)
            .pas de réponse possible
                PAS-DE-REPONSE;
            .réponse complète
                DEFAUT;
                SATISFAIRE-RS ou SATISFAIRE-BD;
                suppression de la question active;
            .réponse possible (conditionnelle)
                si exists une procédure associée à la condition
                alors déclencher cette procédure;
                sinon CONFIRMATION;
                fsi
            .but à compléter ou
            .but à préciser
                BUT-INCOMPLET;
            .but impossible
                LIMITE-SYSTEME;
            fcas
        fsi
    sinon
        si exists une incohérence
        alors
            cas (examen du champ réponse de l'état de la tâche)
            .pas de réponse possible
                BUT-INEXISTANT;
            .réponse complète
                SATISFAIRE-BD;
                suppression de la question active;
                suppression de l'incohérence;
            fcas
        sinon
            cas (examen du champ réponse de l'état de la tâche)
            .pas de réponse possible
                BUT-INEXISTANT;
            .réponse complète
                SATISFAIRE-BD;
                suppression de la question active;
            .but à compléter
                BUT-INEXISTANT;
            fcas
        fsi
    fsi
sinon
    si exists un but actif
    alors
        si exists une incohérence
        alors
            QUEST-INEXISTANT;
            suppression de l'incohérence;
        sinon
            QUEST-INEXISTANT;
        fsi
    sinon
```



```
        si exists une incohérence
        alors
            QUEST-INEXISTANT;
            suppression de l'incohérence;
        sinon
            MAINTIEN;
        fsi
    fsi
fsi
```

Nom : VALEUR-TACHE

Procédure :
si ∃ une réponse susceptible d'instancier l'objet de la question fermée
alors
    ajouter un fait = construit à partir de la question par instanciation de l'objet
    activation du raisonneur)
fsi

2 Les énoncés figés en langage naturel

Afin de produire une réponse en langage naturel, le système utilise actuellement des messages figés. Le formalisme que nous avons mis en oeuvre correspond à des *messages à trous*, ce qui nous amène à faire un certain nombre de remarques :

- toutes les réponses ne peuvent pas être raisonnablement engendrées de cette manière,
- un mécanisme d'instanciation de variables dans un message donné permet le rendre paramétrable,
- afin de donner plus de naturel aux réponses, il est possible de se définir plusieurs variantes pour un message, le gestionnaire en choisira une au hasard.

Voici l'ensemble de ces messages regroupés par variantes :

0	.Renseignements administratifs, je vous ecoute. .Ici le centre experimental de renseignements administratifs du CRIN, nous pouvons fournir des renseignements concernant les demarches administratives les plus courantes, quelle est votre requete? .Il est temps d'enoncer votre veritable requete
1	.Continuez svp. ... .Oui...
2	.Nous ne pouvons satisfaire votre demande, elle depasse nos competences. .$1 est un renseignement que nous ne possedons pas encore, desole.
3	.Desirez-vous autre chose? .Un autre renseignement?
4	.Au revoir
5	.Veuillez completer votre requete svp. .Preciser votre but svp.
6	.Oui, $1 .Effectivement $1 est vrai. .Nous repondons a votre question $1 par l'affirmative.
7	.Non, non $1 est faux .Non, ce n'est pas $1
8	.Est-ce que $1 est vrai? .Est-ce que $1 ?
9	.Bonjour
10	.Voulez vous parler plus fort svp? .J'ai une banane dans l'oreille, voulez-vous parler plus fort?
11	.Eloignez vous un peu du micro svp.
12	.Non, pas que nous sachions.
13	.Nous n'avons pas de renseignements correspondant a votre requete dans notre base.
14	.Nous ne pouvons pas continuer cette conversation, veuillez vous adresser au CIRA METZ tel.87-60-90
15	.Veuillez enoncer votre but svp.
16	.Quelle est votre question?

17	.La reponse est: \n $1
18	.Si $1 est vrai alors \n .valeurs par defaut utilisees: \n $1
19	.Etes-vous majeur? .Est-ce que vous etes majeur?
20	.Quelle est votre nationalite? .De quelle nationalite etes-vous?

Remarque : La séquence formée du symbole **$** suivi d'un nombre représente une variable à instancier.

3 Les procédures de réponses

Dans l'annexe B, dans le paragraphe concernant la base de renseignements, nous pouvons remarquer qu'à une condition `si:` peut être associée une procédure qui se charge de poser la question correspondante en langage naturel et d'effectuer les prédictions nécessaires.

Il existe actuellement deux procédures ; la première permet de savoir si `mineur(personne)` et la seconde, si `nationalite(personne,france)`.

Voici les différentes étapes de ces procédures :

Procédure 1

- dire($message_{19}$)
- mettre à jour l'historique : `question fermee`
- prédiction à l'interpréteur : `attendu(majeur(locuteur))`
- prédiction aux analyseurs : `confirmation&information`
- prédiction à la composante lexicale : `confirmation&age`

Procédure 2

- dire($message_{20}$)
- mettre à jour l'historique : `question ouverte`
- prédiction à l'interpréteur : `attendu(nationalite(locuteur,?))`
- prédiction aux analyseurs : `information`
- prédiction à la composante lexicale : `nationalite`

4 Deux exemples

4.1 Exemple 1 : dialogue dirigé

Le dialogue théorique

(syst)→ Renseignements administratifs, je vous écoute.

(loc)→ *Bonjour.*

(syst)→ Bonjour , ici le centre experimental de renseignements administratifs du CRIN, nous pouvons fournir des renseignements concernant les démarches administratives les plus courantes, quelle est votre requête?

(loc)→ *Voilà, ma carte d'identité a été volée.*

(syst)→ Continuez svp.

(loc)→ *Et je voudrais la renouveler.*

(syst)→ Quelle est votre question?

(loc)→ *Je voudrais savoir quelles sont les formalités.*

(syst)→ Etes-vous majeur?

(loc)→ *Oui.*

(syst)→ Il faut fournir un timbre, une déclaration de vol et des photos.

(syst)→ Désirez-vous autre chose?

(loc)→ *Ou peut-on obtenir une déclaration de vol?*

(syst)→ Il faut aller à la gendarmerie ou à la préfecture ou au commissariat. Un autre renseignement?

(loc)→ *Non merci, au revoir.*

(syst)→ Au revoir

Trace

```
(syst)->Renseignements administratifs, je vous ecoute.

(loc)-> bonjour:.

(syst)->Bonjour

(syst)->Ici le centre experimental de renseignements administratifs du CRIN,
nous pouvons fournir des renseignements concernant les demarches
administratives les plus courantes, quelle est votre requete?

(loc)-> vol(carte'd'identite).

(syst)->Continuez svp.

(loc)-> but:renouveler(locuteur,carte'd'identite).

(syst)->Quelle est votre question?

(loc)-> ?action'formalite.
```



```
(syst)->Etes-vous majeur?

(loc)-> oui.
(interpretation:)-> majeur(locuteur).

(loc)->majeur(locuteur).

(syst)->La reponse est:
et
    fournir ( personne , timbre )
    fournir ( personne , declaration_de_vol )
    fournir ( personne , photo )
)


(syst)->Desirez-vous autre chose?

(loc)-> but:obtenir(locuteur,declaration'de'vol)

(loc)-> aller(locuteur,?lieu).

(syst)->La reponse est:
ou
    gendarmerie
    sp_p
    commissariat
)


(syst)->Un autre renseignement?

(loc)-> au'revoir:.

(syst)->Au revoir
```

4.2 Exemple 2 : dialogue à initiative mixte

Le dialogue théorique

(syst)→ Renseignements administratifs, je vous écoute.

(loc)→ *Quel est le prix de la carte d'identité?*

(syst)→ 115 francs. Désirez-vous autre chose?

(loc)→ *Où peut-on l'obtenir?*

(syst)→ Si vous êtes de nationalité française, vous pouvez aller au commissariat ou à la mairie. Un autre renseignement?

(loc)→ *Non merci.*

(syst)→ Au revoir

Trace

```
(syst)->Renseignements administratifs, je vous ecoute.

(loc)-> prix(carte'd'identite,?£x,franc).

(syst)->La reponse est:
"115"


(syst)->Desirez-vous autre chose?

(loc)-> but:obtenir(locuteur,carte'd'identite)

(loc)-> aller(locuteur,?lieu).

(syst)->Si nationalite ( personne , france )
est vrai alors


(syst)->La reponse est:
ou
    commissariat
    mairie
)


(syst)->Un autre renseignement?

(loc)-> non.

(syst)->Au revoir
```

NOM DE L'ETUDIANT : ROUSSANALY Azim

NATURE DE LA THESE : DOCTORAT DE L'UNIVERSITE DE NANCY I
en Informatique

VU, APPROUVE ET PERMIS D'IMPRIMER

NANCY, le 28 MARS 1988 n° 491

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

UNIVERSITE NANCY I
LE PRESIDENT
R.ZMAINARD

Université de Nancy I

Centre de Recherche en Informatique de Nancy Inria Lorraine

Thèse de l'Université de Nancy 1
soutenue le 1er Avril 1988 par Azim Roussanaly [1]

DIAL, la composante dialogue d'un système de communication orale homme-machine finalisée en langage naturel.

Résumé

L'objet de cette étude est l'élaboration de la composante dialogue d'un système de communication orale homme–machine. Ce travail s'inscrit dans le cadre d'un projet, actuellement en cours de développement au CRIN, dont l'objectif est de réaliser une interface vocale qui soit à la fois efficace et confortable pour un usager peu familiarisé aux systèmes informatiques. Les dialogues que nous envisageons de traiter sont caractérisés par :

- **le langage** autorisé qui correspond à un *sous–langage* limité par l'univers de l'application,
- **la tâche** à réaliser qui appartient à une famille d'applications : *les demandes de renseignements.* L'application test retenue est la demande de renseignements administratives sur la base des *pages roses de l'annuaire téléphonique*,
- **la population** visée qui est le grand public, c'est-à-dire une population a priori non–entrainée.

La variété des sources de connaissances intervenant dans le processus de compréhension d'un dialogue nous ont conduit à la définition d'une architecture formée de cinq composantes qui reposent sur des bases de connaissances spécifiques et qui interagissent par des échanges d'informations entre elles.

DIAL est l'une de ces cinq composantes. Son rôle consiste à fournir une interprétation contextuelle de l'énoncé de l'usager, à effectuer un raisonnement sur la tâche, à gérer le dialogue, à engendrer une réponse et à prédire les interventions futures. Cette composante s'appuie sur un ensemble de connaissances pragmatiques qui incluent des connaissances sur l'univers de l'application, sur la tâche et sur la structure du dialogue.

Nous proposons un formalisme de représentation de ces connaissances qui permet d'appréhender efficacement les problèmes de la compréhension des dialogues que nous envisageons de traiter et, grâce à la distinction qui apparaît entre les connaissances et les traitements, nous parvenons à un système paramétré et incrémental.

Mots-clefs : Dialogue, raisonnement, compréhension automatique de la parole, langage naturel

---

[1] CRIN - BP 239 - 54506 Vandoeuvre-lès-Nancy cedex (tel.83 91 20 00)