Validation d'une base de connaissances hybride avec objets et règles de production

Michèle Courant, Gérard Guevel

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 579

VALIDATION D'UNE BASE DE CONNAISSANCES HYBRIDE AVEC OBJETS ET RÈGLES DE PRODUCTION

Michèle COURANT
Gérard GUEVEL

Novembre 1986

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

Validation d'une base de connaissances hybride avec objets et règles de production

Validation of an hybrid knowledge base including objects and production rules

Michèle COURANT*, Gérard GUEVEL**

Publication Interne n° 316

Octobre 86

30 pages

RESUME:

Face à la diffusion des systèmes experts, la validation des bases de connaissances se pose aujourd'hui en problème crucial, telle la compilation il y a quelques années... Nous abordons ici cette problématique dans le cadre du système expert HAVANE, basé sur une représentation des connaissances hybride greffant des règles de production sur une représentation objet. Dans cette application nous proposons de mettre en oeuvre deux techniques de validation distinctes, associées à chacun des deux niveaux de représentation. Cet article décrit successivement ces deux techniques de validation portant, l'une sur une grammaire hors-contexte, l'autre sur une base de connaissances à règles de production.

ABSTRACT:

Face to the widespread diffusion of expert systems, the validation of knowledge bases has become a compelling necessity, just as compilation did a few years ago... Here we consider this problem for the expert system HAVANE, based upon an hybrid knowledge representation. Within this framework we propose to validate the knowledge base by mean of two distinct verifications, each tied to one level of representation. This article describes successively these two validation methods, related the one to a context-free grammar, the other to a production rules knowledge base.

* Laboratoire d'Informatique - Université de Fribourg - Pérolles - 1700 Fribourg / Suisse
** IRISA - Université de Rennes - Campus de Beaulieu - 35042 Rennes cédex / France

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

SOMMAIRE

INTRODUCTION

La complexité des bases de connaissances allant croissant avec la multiplication des systèmes experts et des modes de représentation de connaissances, la **validation des bases de connaissances** émerge aujourd'hui comme l'un des obstacles majeurs à l'acquisition des connaissances et par suite au développement de systèmes experts réellement utilisables. L'actualité de la problématique soulevée se trouve notamment illustrée dans [ROU 86] à travers une tentative de formalisation du problème en termes logiques, de même que dans [ASS 86].

Nous l'abordons quant à nous dans le cadre du système expert HAVANE [BOS 86, BOS 85], système expert dédié à la mise en relation de Petites Annonces (PA). Dans ce système les PA sont formulées dans le langage usuel des petites annonces de journaux, c'est-à-dire telles que: "Beau T5, jardin, plain-pied, 5mn Rennes, proximité commerce, école, 650 000f". La base de connaissances d'HAVANE relative au langage d'entrée de l'application (à savoir le langage usuel des PA) a pour originalité de greffer à une **représentation objet**, traduisant la sémantique du domaine d'annonces traité, des **règles de production** exprimant l'inférence sous-jacente au domaine. Cette base est issue d'un langage de spécification ad'hoc construit autour des grammaires hors-contexte attribuées [KNU 68], qui comporte deux composants:

- Les grammaires d'Analyse Syntaxique: elles décrivent la syntaxe du langage des PA acceptable par l'application (ce sont des grammaires dont les chaînes terminales sont des éléments textuels des PA, significatifs au regard des objectifs de l'application, à savoir la mise en relation;

- Le Descripteur de Structure: il décrit la sémantique du langage d'entrée de l'application par la définition d'objets sémantiques associés aux annonces.

Le formalisme du Descripteur de Structure a la particularité de produire une **Représentation Centrée Objet** [NIL 80, RIC 83] intégrant la définition de **Règles de Production**. C'est une grammaire hors-contexte attribuée telle que:

- A son axiome "annonce" est associée sous forme d'attributs synthétisés la sémantique d'une PA (retenue au regard des objectifs de mise en relation);

- A chaque règle de grammaire sont associées d'une part des règles de production spécifiant l'inférence à travers du calcul d'attributs, et d'autre part des expressions spécifiant des contraintes d'acceptabilité portant sur les valeurs d'attributs;
- Ses terminaux sont les axiomes des Grammaires d'Analyse Syntaxique.

La validation de la base de connaissances issue du Descripteur de Structure, rejoint pour une large part de nombreux travaux déjà effectués sur les grammaires attribuées [COU 86, DER 86]. Néanmoins l'utilisation de règles de production pour la spécification du calcul d'attributs induit une problématique nouvelle, sur laquelle nous avons voulu focaliser. Plutôt que d'aborder globalement la validation de la grammaire attribuée que constitue le Descripteur de Structure, nous avons ainsi préféré scinder le problème en deux objectifs, à savoir:

- Le contrôle du squelette grammatical hors-contexte du Descripteur de Structure;
- Le contrôle du calcul d'attributs, à travers les règles de production attachées aux règles de grammaire.

Dans ce document nous présentons successivement dans les deux premiers chapitres, notre approche face à ces deux problèmes. Dans un troisième chapitre nous procédons à une évaluation des solutions choisies, qui s'ouvre naturellement sur de nouvelles perspectives de travail.

CHAPITRE 1: Contrôle niveau grammatical

Dans ce chapitre, après une brève introduction au problème posé au niveau grammatical du Descripteur de Structure, nous donnons la représentation choisie pour ce dernier comme préalable à la mise en oeuvre des deux contrôles grammaticaux retenus. Nous abordons ensuite successivement ces deux contrôles.

1.1 - INTRODUCTION

Nous présentons tout d'abord succintement la fonction du composant grammatical hors-contexte du Descripteur de Structure, avant de décrire la problématique posée à ce niveau.

1.1.1 - Fonction du composant grammatical du Descripteur de Structure

Le composant grammatical du Descripteur de Structure a pour fonction de spécifier la sémantique d'un domaine d'annonces sous forme d'une structure arborescente dont les différents niveaux font apparaître des décompositions successives d'un domaine d'annonces. Ainsi le domaine immobilier peut-il être représenté par la figure 1.

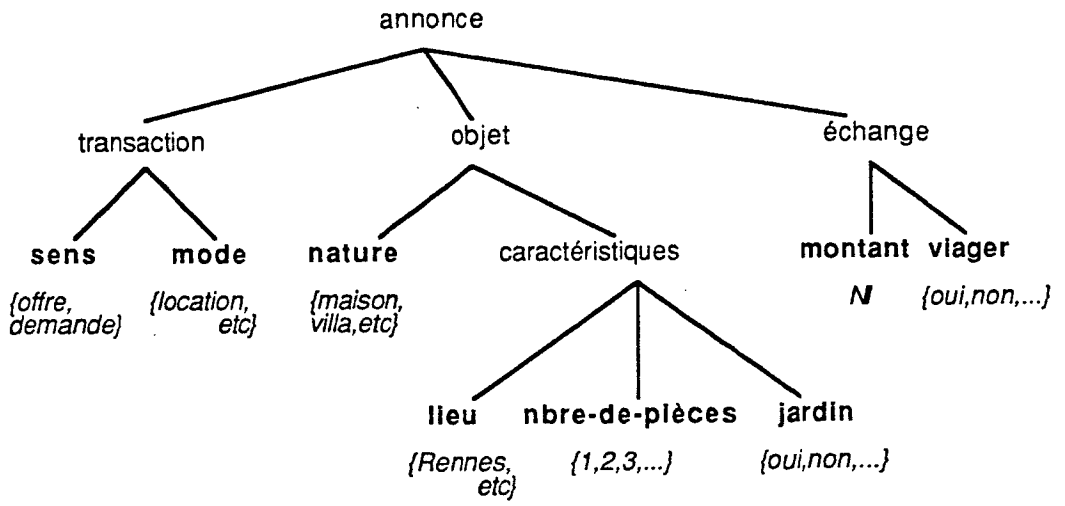

Figure 1: Représentation arborescente du domaine des annonces immobilières.
Les informations notées entre { } sont les domaines de valeurs associées aux feuilles de l'arborescence, qui rappelons-le, correspondent aux Grammaires

d'Analyse Syntaxique, soit à des éléments textuels de base du langage des annonces.

Nous verrons dans le chapitre 2 le détail des noeuds de l'arborescence.

1.1.2 - Problématique du niveau grammatical

Le squelette hors-contexte de la grammaire attribuée que constitue le Descripteur de Structure, se prête à deux contrôles statiques. Il s'agit de vérifier:

- que tout non-terminal de la grammaire se dérive en terminaux (ici les axiomes de Grammaires d'Analyse Syntaxique);
- que tout non-terminal défini est bien accessible depuis l'axiome de la grammaire.

Les contrôles visés à ce niveau sont donc des problèmes extrêmement classiques de validation de grammaires, et de nombreux algorithmes existent pour les résoudre. Cependant le souci de demeurer dans l'environnement PROLOG constitué par la mise en oeuvre préexistante de l'application HAVANE, conjugué à l'aisance avec laquelle ces contrôles pouvaient être réalisés, nous ont conduits à repenser quelque peu ces problèmes.

1.2 - REPRESENTATION DU DESCRIPTEUR DE STRUCTURE

Ce paragraphe décrit la représentation choisie. Il en donne ensuite une illustration, puis indique la méthode de création associée.

1.2.1 - Représentation choisie

Le contexte PROLOG dans lequel nous nous situons suggère bien entendu d'emblée de choisir une représentation du graphe associé à la grammaire du Descripteur de Structure, sous forme d'un terme PROLOG facilitant conjointement la mise en oeuvre des deux contrôles requis. Ceci nécessite une représentation suffisamment riche pour permettre l'émission d'un diagnostic précis en cas d'erreur.

Une première solution consiste à représenter la grammaire sous la forme d'un arbre fini "ET/OU" [BAR 86]. Pour un non-terminal donné:

- les noeuds "OU" correspondent aux alternants (i.e. aux parties droites de dérivations) issus de ce non-terminal;
- les noeuds "ET" dominent un ou plusieurs symboles apparaissant dans une même partie droite de règle ayant pour partie gauche ce non-terminal.

Exemple:

Soit une grammaire hors-contexte et l'arbre associé:

A -> BC / D
B -> ab
C -> c
D -> d

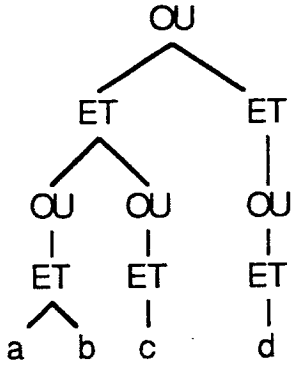

Figure 2

Cette représentation a toutefois l'inconvénient de perdre trace des non-terminaux utilisés dans la grammaire et donc d'interdire la localisation précise des erreurs éventuelles.

Une seconde solution résulte de la substitution dans la représentation précédente, des noeuds "OU" par les non-terminaux parties gauches de règles de grammaire, et des noeuds "ET" par les numéros de règles. A un "OU" correspond donc une partie gauche de règle, tandis qu'à un "ET" correspond une dérivation identifiée par un numéro exclusif.

1.2.2 - Illustration

Soit la grammaire hors-contexte suivante décrivant une structuration possible des annonces immobilières.

Numéro de règle	Règle		
1	annonce	->	annonce1
2	annonce1	->	transaction objet échange
3	transaction	->	**sens mode**
4	objet	->	**nature** caractéristiques
5	échange	->	**montant viager**
6	caractéristiques	->	**lieu nbre-de-pièces jardin**

Première solution:

Nous obtenons l'arbre "ET/OU" classique donné par la figure 3.

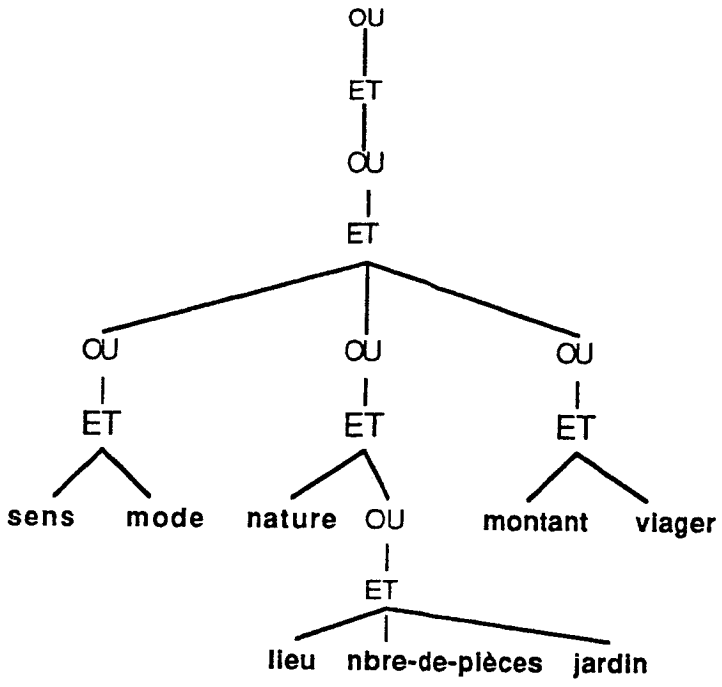

Figure 3

Deuxième solution:

Les noeuds "OU et "ET" sont respectivement substitués par des non-terminaux et numéros de règles.

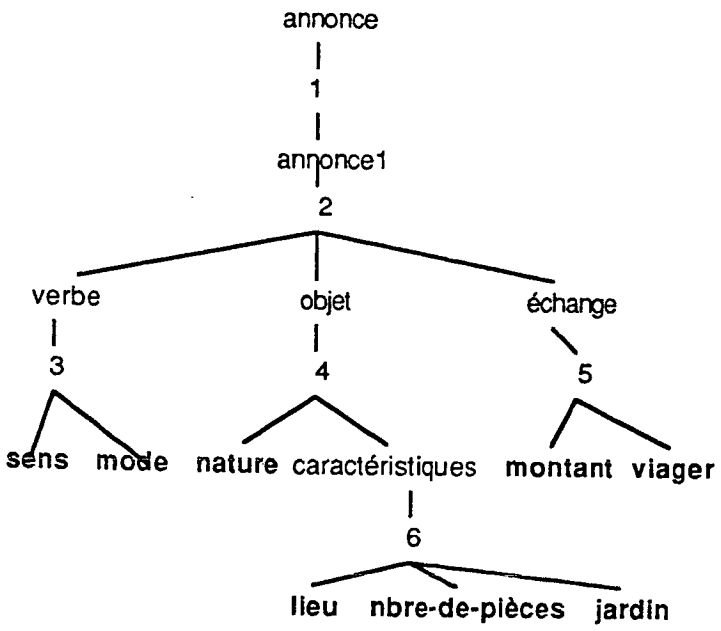

Figure 4

1.2.3 - Méthode de création

1.2.3.1 - Algorithme

Le but est de partir de l'axiome de la grammaire du Descripteur de Structure, de rechercher sa première dérivation, de créer le sous-terme correspondant à cette dérivation et de procéder récursivement sur les autres dérivations. Deux cas peuvent se présenter lors de la rencontre d'un symbole en partie droite de dérivation:

- le symbole est un terminal. Ce terminal constitue une feuille pour la branche étudiée;
- le symbole est un non-terminal. Trois situations sont possibles dans ce cas:
  * le non-terminal considéré n'est pas défini dans la grammaire. Celui-ci constitue une feuille avec l'indication "non-défini" **(a)**
  * le non-terminal considéré apparaît déjà dans l'arbre créé. Cela révèle la présence d'un cycle dans la grammaire **(b)**
  * le non-terminal est défini et non traité. Il est considéré comme la racine du sous-arbre correspondant à la partie de la grammaire qu'il engendre **(c)**.

1.2.3.2 - Terminaison du calcul

La grammaire du Descripteur de Structure possède un nombre fini de règles donc un nombre fini de symboles terminaux et non-terminaux. En partant de l'axiome de la grammaire le nombre de symboles rencontrés est donc fini. La rencontre d'un terminal provoque en effet l'arrêt du calcul, tandis que la rencontre d'un non-terminal présente trois cas:

* le non-terminal est déjà défini: arrêt du calcul **(a)**
* le non-terminal est déjà rencontré: arrêt du calcul **(b)**
* le non-terminal est axiome d'une sous-grammaire: cette sous-grammaire étant définie par des règles incluses dans la grammaire initiale, tous les terminaux sont rencontrés deux fois, d'où arrêt du calcul **(c)**.

1.3 - CONTRÔLE DE LA DERIVABILITE DE TOUT NON-TERMINAL EN TERMINAUX

1.3.1 - Nature du problème

Muni du Descripteur de Structure, l'expert rédige des spécifications relatives à un domaine d'annonces par l'énoncé d'un certain nombre de règles de grammaire hors-

contexte. Pour un non-terminal introduit dans une partie droite de règle, celui-ci peut donc avoir omis de spécifier les dérivations qui en sont issues.

1.3.2 - Méthode

Le terme représentant le Descripteur de Structure étant construit, il est possible de vérifier que tout non-terminal se dérive en terminaux en exécutant une analyse du type "ET/OU". Cette analyse consiste à parcourir l'arbre en profondeur et à gauche d'abord. A chaque noeud de l'arbre est élaborée une réponse résultant d'un traitement effectué sur les réponses provenant de ses noeuds fils.

A un noeud donné de l'arbre correspondant à la définition d'un non-terminal, où se greffent un ou plusieurs sous-arbres, est associé un traitement de type "OU" sur les réponses provenant de l'analyse de ces sous-arbres.

A un noeud correspondant à la définition d'une dérivation, c'est-à-dire d'une partie droite de règle, est associé un traitement de type "ET" sur les réponses issues de l'analyse de ces dérivations.

Les réponses possibles synthétisées aux noeuds sont de quatre types:

- ok: indique pour un noeud "OU" qu'il existe au moins une des branches ayant pour feuilles des terminaux et pour un noeud "ET" que toutes les branches ont pour feuilles des terminaux;
- cy: indique la présence d'un cycle et donc la possibilité pour le non-terminal, de ne pas se dériver en terminaux;
- pd: indique la présence d'un prédicat évaluable PROLOG et donc la non-dérivation en terminaux;
- nd: indique qu'un non-terminal rencontré dans une partie droite de règle de grammaire n'est pas défini.

1.3.3 - Synthèse des résultats

Nous illustrons tout d'abord la synthèse des résultats sur des exemples. Les opérateurs "ET" et "OU" étant associatifs, nous nous restreignons à des cas binaires.

- Ok ET ok **donne** ok (les deux sous-arbres se dérivant en terminaux, le non-terminal racine se dérive également en terminaux);

- Cy ET ok **donne** cy (le cycle rencontré au niveau du sous-arbre gauche est propagé au non-terminal racine);

- Ok OU x (avec x égale ok, cy, pd, nd) **donne** ok (s'il est prouvé que l'un des alternants se dérive bien en terminaux, alors il est prouvé que le non-terminal se dérive lui aussi en terminaux, quels que soient les autres alternants).

En résumé cette synthèse des résultats peut s'exprimer par une notion de "priorité" des valeurs pour la remontée dans l'arbre. On a en fait:

- Pour les "ET", par ordre de priorité décroissante de valeurs: pd, cy,nd, ok;
- Pour les "OU", par ordre de priorité décroissante de valeurs: ok, pd, nd, cy.

L'arbre étant par construction fini, l'étude de cet arbre est finie. Ceci garantit la synthèse en un temps fini d'une réponse attachée à l'axiome de la grammaire. Si cette réponse est ok, la grammaire étudiée possède les propriétés suivantes:

- Il existe au moins une suite de non-terminaux de premier terme l'axiome initial, qui se dérive en terminaux. Cette suite constitue un arbre de dérivation fini, c'est-à-dire décrit un sous-langage correct;
- cet arbre ne contient pas de prédicat évaluable PROLOG;
- tous les terminaux de cet arbre sont correctement définis;
- Il peut exister des cycles dans la grammaire, mais ceux-ci ne sont pas fermés (ils contiennent au moins une dérivation possible en terminaux).

Toute autre réponse traduit la présence d'au moins une erreur fatale vis-à-vis des propriétés requises. La création d'une liste des erreurs accompagne l'analyse "ET/OU" effectuée sur l'arbre. Grâce à celle-ci un verdict peut être établi, indiquant à l'expert:

- La nature des erreurs;
- Les non-terminaux mis en cause;
- Les numéros de règles (en cas d'erreurs survenues en parties droites de règles).

1.4 - CONTRÔLE DE L'ACCESSIBILITE A TOUT NON-TERMINAL

1.4.1 - Nature du problème

Lors de la spécification du niveau grammatical du Descripteur de Structure, l'expert introduit un certain nombre de non-terminaux. Si l'un de ces non-terminaux n'est pas

accessible depuis l'axiome de la grammaire, cela révèle un oubli: celui-ci devrait en effet apparaître dans l'une des parties droites de règles.

1.4.2 - Méthode

L'arbre représentant la grammaire du Descripteur de Structure étant constitué, on dispose de la liste de tous les non-terminaux accessibles depuis l'axiome de la grammaire. Cette liste ayant été créée de manière à éviter un parcours infini de la grammaire, il suffit de vérifier pour chaque non-terminal existant dans le "monde" (au sens PROLOG du terme), s'il appartient à la liste précécente, et dans le cas contraire de le signaler à l'expert.

CHAPITRE 2: Contrôle niveau règles de production

2.1 - INTRODUCTION

Dans ce paragraphe nous décrivons brièvement le rôle des règles de production dans le Descripteur de Structure, avant d'exposer la problématique posée à ce niveau de la base de connaissances.

2.1.1 - Fonction des règles de production

Nous avons vu dans l'introduction que le rôle du Descripteur de Structure est de décrire la sémantique d'une petite annonce, au regard des objectifs de l'application, en l'occurrence la mise en relation de petites annonces. Ceci s'effectue:

- par la structuration du domaine d'annonces traité sous forme arborescente (cf 1.1.1);
- par la synthèse sous forme d'attributs attachés à l'axiome de la grammaire du Descripteur de Structure, de la sémantique d'une annonce donnée.

Ainsi la structure arborescente n'est que le support permettant la synthèse de la sémantique des annonces, depuis les feuilles de l'arbre, en passant par les niveaux successifs, jusqu'à la racine. Cette synthèse consiste en du calcul d'attributs, lequel est exprimé, non pas directement par des relations comme dans les grammaires attribuées classiques, mais par des règles de production, dont les conséquents sont de telles relations.

Dans la règle de grammaire suivante par exemple:

annonce(st,mt,...) -> transaction(st1,mt1) objet(n,c1,c2,...) échange(st2,mt2,...)

le calcul de l'attribut "mt" (attaché au non-terminal partie gauche) pourrait s'exprimer en partie par des règles telles que:

mt1= *inconnu* ^ mt2=*inconnu* ^ n=*maison* -> mt=*achat-vente*

mt1= *inconnu* ^ mt2=*inconnu* ^ n=*appartement* -> mt=*location*

mt1=*chambre* -> mt=*location*

2.1.2 - Problématique du niveau règles de production

Les règles de production expriment par la synthèse d'attributs, l'inférence susceptible d'être effectuée sur la sémantique des annonces, ceci au niveau de chaque règle de la grammaire du Descripteur de Structure. Cette synthèse est réalisée par le calcul des attributs du

non-terminal partie gauche à partir des attributs de la partie droite. Les règles de production se prêtent à deux contrôles de validité sur le calcul des attributs du non-terminal partie gauche. Il s'agit:

- de garantir la **complétude** du calcul spécifié, c'est-à- dire d'assurer qu'un attribut donné est calculé dans tous les cas;
- de détecter **l'indéterminisme** sous-jacent au calcul d'attributs, lié au recouvrement possible de prémisses.

2.2 - Complétude d'un calcul

Dans ce paragraphe nous présentons tout d'abord l'intérêt d'effectuer un contrôle de complétude sur le calcul d'attributs. Nous procédons ensuite à un essai de formalisation du problème, avant d'exposer le principe de la solution choisie et la mise en oeuvre associée.

2.2.1 - Intérêt du problème

L'une des contraintes sémantiques inhérentes au formalisme du Descripteur de Structure est que tout attribut doit être calculé, et ce dans toutes les réalisations du monde (attribut,valeur) dont il est issu. Il s'agit par là d'exclure la "non-instanciation", pour chacun des attributs concrétisant la sémantique d'une Petite Annonce. Nous dirons que pour chaque règle de la grammaire du Descripteur de Structure, le calcul des attributs y figurant doit être complet. Cette contrainte est en effet une condition élémentaire en dehors de laquelle il devient impossible d'effectuer aucun contrôle sur la pertinence et par suite, la validité des dires de l'expert.

2.2.2 - Nature du problème

On considère un ensemble de règles de production $R_1, ..., R_p$, de la forme :
$R_i : P_i \rightarrow C_i$ où $\forall i \in \{1, ..., p\}\ \exists e_{ix}$, expression calculable telle que $(X = e_{ix}) \in C_i$.
($\{R_1, ..., R_p\}$ est donc l'ensemble des règles spécifiant le calcul d'un attribut donné X).
Il s'agit de vérifier que l'ensemble des réalisations pour lesquelles X peut être calculé, c'est-à-dire l'ensemble des réalisations qui sont des modèles de l'une des prémisses au moins sur le monde (attribut, valeur) auquel on se réfère, correspond à l'ensemble des réalisations possibles.
On note M, l'ensemble des réalisations possibles sur le monde (attribut, valeur) auquel on se réfère.
On note $P_1 \cup P_2 \cup ... \cup P_p$, la réunion des modèles des prémisses $P_1, P_2, ..., P_p$.

Définition : Le calcul d'un attribut X est dit "complet"
ssi $P_1 \cup P_2 \cup ... \cup P_p \supseteq M$

2.2.3 - Principe de la solution

On note A l'ensemble des attributs apparaissant dans le monde auquel on se réfère.

Soit $A = \{X_1, ..., X_n\}$ tel que $Card(A) = n$.

On note $D(X_i)$, le domaine des valeurs de l'attribut X_i, $i \in \{1, ..., n\}$.

On suppose disposer de M sous la forme d'un produit cartésien :

$D(X_1) \times D(X_2) \times ... \times D(X_n)$

(autrement dit nous n'envisageons pas le cas où M peut être restreint par des contraintes issues de relations entre attributs).

On suppose disposer de l'ensemble des prémisses P_i des règles concernées R_i, $i \in \{1, ..., p\}$

De même on suppose disposer pour chaque prémisse P_i, de règle R_i concernée, telle que $P_i = m_{i1} \cup ... \cup m_{iqi}$ (m_{ik} = monôme), de chacun de ses monômes m_{ik} ($k \in \{1, ..., qi\}$) sous forme d'un produit cartésien $D_{ik}(X_1) \times D_{ik}(X_2) \times ... \times D_{ik}(X_n)$

où $D_{ik}(X_j)$ = domaine des valeurs de l'attribut X_j, $j \in \{1, ..., n\}$ couvert par le monôme m_{ik} [KOR 66].

L'idée est alors de "soustraire" (au sens ensembliste, par action sur les produits cartésiens) de M, les réalisations correspondant à chacun des monômes m_{ik} ($k \in \{1, ..., qi\}$) de toutes les prémisses P_i ($i \in \{1, ..., p\}$).

A l'issue de cette soustraction, on obtient un ensemble de réalisations résiduelles noté R.

Deux cas se présentent :

- $R = \emptyset$, autrement dit $P_1 \cup P_2 \cup ... \cup P_p \supseteq M$.
  Le calcul de l'attribut X, commun aux conséquents $C_1, ..., C_p$ est complet.
- $R \neq \emptyset$, autrement dit $P_1 \cup P_2 \cup ... \cup P_p \subset M$. Il reste des réalisations pour lesquelles le calcul de l'attribut X n'est pas spécifié.

2.2.4 - Mise en oeuvre

```
début var ensemble-de-réalisations R init D(X1) x D(X2) x ... x D(Xn) ;
    jusqu'à plus-de-monôme-à-traiter
    faire
    co soit m , le monôme à traiter et soit Am , le produit cartésien associé à m fco
        R := diff-ens-réalisations (R, Am)
    fait
    sortie plus-de-monôme-à-traiter :
    si R = ∅
    alors "ok, tous les cas sont prévus pour l'attribut " , X
```

```
    sinon co        R = ensemble des cas non prévus dans le calcul de X fco
    lister-cas-non-prévus (R)
    fsi
fin.
```

Avec diff-ens-réalisations (R_1,R_2) qui est une procédure délivrant un résultat de type ensemble-de-réalisations, représentant la différence ensembliste $R_1 \setminus R_2$ où R_1 et R_2 sont également de type ensemble-de-réalisations.
Et avec lister-cas-non-prévus (R) qui est une procédure permettant de lister les réalisations pour lesquelles le calcul de l'attribut X n'est pas spécifié.

```
procédure diff-ens-réalisations c'est (ensemble-de-réalisations R1, R2)
            résultat ensemble-de-réalisations
début
    co R1 est de la forme D1(X1) x ... x D1(Xn) fco
    co R2 est de la forme D2(X1) x ... x D2(Xn) fco
    var ensemble-de-réalisations R init ∅ ;
    pour I := 1 à n faire
        C:= complémentaire (D2(Xi), D1(Xi))
        si C ≠ ∅ alors
                R := R ∪ ( D1(X1) x ... x D1(Xi-1) x C xD1(Xi+1) x ... x D1(Xn))
        fsi
    fait;
    résultat R
fin.
```

Avec complémentaire (D_1,D_2) qui est une procédure délivrant un résultat de type ensemble-de-réalisations, représentant le complémentaire ensembliste "étendu" de D_1 dans D_2 , c'est-à-dire le retrait dans D_2 des réalisations appartenant à $D_1 \cap D_2$.

Remarque

Ce principe entraine la création de doublons dans R. Il est cependant avantageux par sa simplicité de mise en oeuvre. Il suffit ensuite d'éliminer ces doublons avant de présenter les résultats à l'expert.

Exemple de création de doublons

$R_1 = \{1, 2\} \times \{4, 5\} \times \{6, 7\}$

$R_2 = \{2\} \times \{4, 5\} \times \{6\}$

$R = \emptyset$

$i = 1 \quad C = \{1\} \quad R = \{1\} \times \{4, 5\} \times \{6, 7\}$

$i = 2 \quad C = \emptyset \quad R = \{1\} \times \{4, 5\} \times \{6, 7\}$

$i = 3 \quad C = \{7\} \quad R = (\{1\} \times \{4, 5\} \times \{6, 7\}) \cup (\{1, 2\} \times \{4, 5\} \times \{7\})$

les n-uplets (1, 4, 7) et (1, 5, 7) apparaissent deux fois.

2.3 - Contrôle de l'indéterminisme

Ce paragraphe aborde l'intérêt de l'indéterminisme puis sa formalisation. Ceci amène à une solution et à une mise en oeuvre associée. Enfin une extension est proposée en l'occurrence le contrôle de l'inclusion.

2.3.1 - Intérêt du problème

La distinction déterminisme/non-déterminisme relative au calcul d'un attribut est intéressante sur le plan de la spécification à différents titres. Le non-déterminisme est en effet révélateur d'un certain nombre d'erreurs susceptibles d'être effectuées par l'expert, et dont voici quelques exemples.

On considère les prémisses P_1 et P_2 de deux règles conduisant à des conclusions différentes pour l'attribut X.

1er cas P_1 : _____ l'une des prémisses est totalement
P_2 : _____ incluse dans l'autre.

Interprétation :

Il peut s'agir : - d'une règle générale de prémisse P_1
- d'une règle particulière de prémisse P_2.

En formulant la règle générale, l'expert a oublié d'exclure le cas particulier pour lequel doit s'appliquer la seconde règle.

Conséquence :

L'expert peut vouloir rendre exclusives les deux règles en excluant P_2 de la règle générale, dont la nouvelle prémisse devient $P_3 = P_1 \wedge \neg P_2$

2e cas P_1 : _____ les prémisses possèdent chacune des modèles propres et des
P_2 : _____ modèles communs avec l'autre.

Interprétation :

Ces modèles communs n'avaient pas été "vus" par l'expert.

Conséquences :

1) L'expert souhaite traiter ces modèles autrement :
   - soit en les excluant de l'une ou l'autre prémisse
   - soit en les excluant des deux prémisses et/ou en ajoutant de nouvelles règles pour ces modèles.

2) Ces cas de figure communs à P_1 et P_2 ne peuvent en fait jamais se produire, ceux-ci n'ayant par exemple pas de sens. L'expert souhaite remédier à ce problème :
   - par l'ajout en amont de contrainte(s) d'acceptabilité excluant ces cas, et/ou
   - par l'ajout et/ou la modification de règles en amont , ce qui peut avoir pour effet de supprimer ces réalisations .

2.3.2 - Nature du problème

Soit un ensemble de règles de production $R_1, ..., R_p$, de la forme : R_i : P_i -> C_i où P_i est la prémisse de R_i et C_i son conséquent.

Pour tout C_i ($i \in \{ 1, ..., p\}$) : $\exists\, e_{iX}$ telle que $(X = e_{iX}) \in C_i$, e_{iX} étant une expression calculable.

Autrement dit chacune des règles $R_1, ..., R_p$ spécifie un calcul de l'attribut X. Deux cas peuvent alors se produire, relativement à un sous-ensemble de règles conduisant à des calculs différents de l'attribut X :

- toutes leurs prémisses s'excluent mutuellement : le calcul de X est **déterministe**,
- certaines prémisses ont des réalisations communes : le calcul de X est **non-déterministe**.

On se ramène en fait au cas plus simple de la comparaison de règles prises deux à deux.

Soient deux règles R1 et R2 :

R1 P -> C1 avec $(X = v1) \in C1$

R2 Q -> C2 avec $(X = v2) \in C2$

et $v1 \neq v2$

Parmi toutes les réalisations sur le monde (attribut,valeur) auquel on se réfère

- P et Q n'ont aucun modèle commun
  $P \cap Q = \varnothing$, c'est le déterminisme,
- P et Q possèdent au moins un modèle commun
  $P \cap Q \neq \varnothing$, c'est le non-déterminisme.

2.3.3 - Principe de la solution

Le problème élémentaire à résoudre est le calcul de **l'intersection de deux**

prémisses. Les prémisses en question sont supposées être sous une forme disjonctive quelconque (c'est-à-dire non nécessairement normale et optimale) , i.e sous la forme d'une somme de monômes (non nécessairement canoniques) [KOR 66].

Définition : Deux règles de prémisses P_1 et P_2 sont dites "disjointes" **SSI** $P_1 \cap P_2 = \emptyset$

Théorème 1 : soit $P_1 = m_1 \cup ... \cup m_p$ et $P_2 = n_1 \cup ... \cup n_q$
où les m_i et les n_i sont des monômes.
$P_1 \cap P_2 = \emptyset$ **SSI** $\forall i \in \{1, ..., p\}$ et $\forall j \in \{1, ..., q\}$ $m_i \cap n_j = \emptyset$

Démonstration :

1) Dans le sens "⇒" (on démontre la contraposée)
Supposons que $\exists i \in \{1, ..., p\}$, $\exists j \in \{1, ..., q\}$
tels que $m_i \cap n_j \neq \emptyset$.
Alors $P_1 \cap P_2 = (m_1 \cup ... \cup m_p) \cap (n_1 \cup ... \cup n_q)$
$= (m_1 \cap n_1) \cup ... \cup (m_1 \cap n_q) \cup ... \cup$
$(m_i \cap n_1) \cup ... \cup (m_i \cap n_q) \cup ... \cup$
$(m_p \cap n_1) \cup ... \cup (m_p \cap n_q)$
Or $m_i \cap n_j \subseteq P_1 \cap P_2$ et $m_i \cap n_j \neq \emptyset$
Donc $P_1 \cap P_2 \neq \emptyset$.

2) Dans le sens "⇐"
$P_1 \cap P_2 = (m_1 \cup .... \cup m_p) \cap (n_1 \cup ... \cup n_q)$
$= (m_1 \cap n_1) \cup ... \cup (m_1 \cap n_q) \cup ... \cup$
$(m_i \cap n_1) \cup ... \cup (m_i \cap n_q) \cup ... \cup$
$(m_p \cap n_1) \cup ... \cup (m_p \cap n_q)$
D'où $P_1 \cap P_2 = \emptyset \Rightarrow \forall i \in \{1, ..., p\}$ et $\forall j \in \{1, ..., q\}$ $m_i \cap n_j = \emptyset$

Théorème 2 : Soient deux monômes m_1 et m_2 et A_c l'ensemble
des attributs communs à m_1 et m_2.
$m_1 \cap m_2 = \emptyset$ **SSI** $\exists a \in A_c$ tel que $D(a)/m_1 \cap D(a)/m_2 = \emptyset$
où $D(X)/m_i$ désigne le domaine de l'attribut X dans le monôme m_i .

Démonstration :

Soit $A_C = \{a_1, ..., a_n\}$,

1) Dans le sens "$\Rightarrow$" (on démontre la contraposée)
Supposons que $\forall a \in A_C \; D(a)/m_1 \cap D(a)/m_2 \neq \emptyset$,
on note $D(a)/m_1 \cap m_2 = D(a)/m_1 \cap D(a)/m_2$.
Alors $m_1 \cap m_2 = D(a_1)/m_1 \cap m_2 \times ... \times D(a_n)/m_1 \cap m_2$,
avec $D(a_1)/m_1 \cap m_2 \neq \emptyset$ et ... et $D(a_n)/m_1 \cap m_2 \neq \emptyset$.
D'où $m_1 \cap m_2 \neq \emptyset$.

2) Dans le sens "$\Leftarrow$"
On note $D(a_i)/m_1 \cap m_2 = D(a_i)/m_1 \cap D(a_i)/m_2$ pour
$i \in \{1, ..., n\}$. Supposons $\exists i \in \{1, ..., n\}$ tel que $D(a_i)/m_1 \cap m_2 = \emptyset$
Alors $m_1 \cap m_2 = D(a_1)/m_1 \cap m_2 \times ... \times D(a_i)/m1 \cap m2 \times ... \times D(a_n)/m_1 \cap m_2$
$= D(a_1)/m_1 \cap m_2 \times ... \times D(a_{i-1})/m_1 \cap m_2 \times \emptyset \times D(a_{i+1})/m_1 \cap m_2 \times ... \times D(a_n)/m_1 \cap m_2$
$= \emptyset$.

En vertu de ces deux théorèmes la mise en oeuvre pour le calcul de l'intersection de deux prémisses découle naturellement.

2.3.4 - Mise en oeuvre

2.3.4.1 - Structures de données

Soient deux règles permettant le calcul de l'attribut a :

$R_1 : P_1 \rightarrow C_1$

$R_2 : P_2 \rightarrow C_2$ avec $a \in C_1$ et C_2

Les deux prémisses P_1 et P_2 sont sous forme disjonctive quelconque, ie sous forme d'une somme de monômes non nécessairement canoniques.

$P_1 = m_{11} \cup ... \cup m_{1P}$

avec $m_{1i} = a_{1i1} \cap ... \cap a_{1iR}$ pour $i = 1, ..., P$

avec $a_{1ik} = (att_{1ik}, dom_{1ik})^i$

$P_2 = m_{21} \cup ... \cup m_{2Q}$

avec $m_{2j} = a_{2j1} \cap ... \cap a_{2jS}$ pour $j = 1, ..., Q$

avec $a_{2jl} = (att_{2jl}, dom_{2jl})^j$

2.3.4.2 - Algorithme

```
début
    i := 1;              co indice du monôme courant m1 dans P1 fco
    indéter := faux      co booléen à vrai si les deux prémisses sont
    faire                    co indéterministes fco
        j := 1               co indice du monôme courant m2 dans P2 fco
        faire
            k := 1;      co indice de l'attribut courant dans m1 fco
            intermonôme := faux      co booléen à vrai si deux monômes
            faire                    co ont une intersection vide fco
                l := 1;      co indice de l'attribut courant dans m2 fco
                interatt := vrai co booléen à faux si deux attributs
                faire                co ont une intersection non vide fco
                    si att1ik = att2jl alors
                        intersection (dom1ik, dom2jl, résul)
                        si résul = Ø alors
                            intermonôme := vrai
                        sinon
                            interatt := faux
                        fsi
                    sinon
                        l := l + 1
                    fsi
                jusqu'à l > Sj ou intermonôme ou ⌉ interatt
                sortie
                        l > Sj ou ⌉ interatt : k := k + 1;
                        intermonôme : vide
            jusqu'à k > Ri ou intermonôme
            sortie
                    k > Ri : indéter := vrai;
                    intermonôme : j := j + 1
        jusqu'à j > Q ou indéter
        sortie
                j > Q : i := i + 1;
                indéter : vide
    jusqu'à indéter ou i > P
    sortie
            i > P : "les deux règles sont déterministes";
            indéter : "les deux règles sont indéterministes"
fin.
```

Remarque :

La détection d'une seule réalisation commune aux deux prémisses provoque l'arrêt du calcul.

2.3.5 - Extension : intersection de deux prémisses

La solution proposée précédemment pour le contrôle de l'indéterminisme entre deux prémisses ne permet pas d'indiquer à l'expert les modèles communs à ces deux prémisses (cf 2.3.3). Il est cependant possible de détecter l'inclusion d'une prémisse dans une autre. Le principe utilisé pour la mise en oeuvre de ce contrôle est celui employé pour le problème de la complétude (cf 2.2.3). En effet ce principe étant basé sur le retrait de n-uplets (modèles des prémisses) dans un ensemble de n-uplets (réalisations possibles sur le monde (attribut,valeur) auquel on se réfère), il est aisé de considérer les n-uplets correspondants aux modèles d'une prémisse et d'en ôter les n-uplets correspondants aux modèles de l'autre. Si cette différence ensembliste produit l'ensemble vide, on peut affirmer que la première prémisse est incluse dans la seconde. Si ce n'est pas le cas on refait le calcul en intervertissant les deux ensembles de modèles pour voir si la seconde prémisse est incluse dans la première. Il est à noter qu'un tel test d'inclusion est rendu possible par l'extension faite à la notion de différence ensembliste. Si l'on désire ôter un n-uplet d'un ensemble qui ne le contient pas il suffit d'y passer outre. Ainsi la différence $R_1 \setminus R_2$ a un sens même si R_2 possède des modèles n'apparaissant pas dans R_1.

Exemple

R1 = { (1, 2), (1, 3), (2, 4) }

R2 = { (1, 3), (2, 4), (3, 5) }

R1 \ R2 = { (1, 2) }

Le n-uplet (3, 5) n'appartient pas à R1. On n'en tient simplement pas compte.

CHAPITRE 3: Evaluation et perspectives

Dans ce chapitre nous présentons un certain nombre d'observations sur la mise en oeuvre actuelle des contrôles et proposons quelques améliorations ainsi que des contrôles complémentaires.

3.1 - INTRODUCTION

La mise en oeuvre actuelle connaît deux grandes limites liées:

- Aux problèmes matériels (c'est-à-dire Macintosh), mais pour une large part imputables à la complexité spatio-temporelle intrinsèque des algorithmes. Ainsi la place mémoire fait défaut lors de calculs importants, et les délais d'exécution peuvent prohiber certains des principes utilisés, en l'occurrence la méthode préconisée pour la vérification de la complétude;
- A la diversité des contrôles requis pour la mise en place d'une interface expert.

Ce chapitre procède à une critique des contrôles présents en suggérant quelques améliorations susceptibles d'effacer les points négatifs, et propose un certain nombre de contrôles complémentaires souhaitables dans le cadre d'un environnement de spécification pour l'expert.

3.2 - A PROPOS DE LA COMPLETUDE

Nous présentons ici quatre extensions au calcul de complétude.

Une limitation essentielle de la mise en oeuvre présente est sa "localité". Celle-ci consiste actuellement en une analyse locale à une règle de la grammaire du Descripteur de Structure, avec l'hypothèse que tous les attributs de la partie droite sont toujours calculés. Pour un attribut donné, la complétude, telle que nous l'avons définie en 2.2.2, présuppose en effet la complétude des calculs des attributs utilisés dans sa valuation. Une mise en oeuvre conforme au principe de complétude, requiert donc en fait le déclenchement du contrôle de complétude sur tous les attributs situés en amont du calcul concerné,et ce jusqu'au niveau terminal du Descripteur de Structure, à savoir les Grammaires d'Analyse Syntaxique.

La seconde extension réside en la prise en compte d'une facilité offerte par le Descripteur de Structure non évoquée jusqu'ici, pour le cas où la synthèse d'un attribut se réduit au transport d'un attribut de la partie droite de la règle de grammaire. Dans ce cas il est possible d'exprimer la synthèse

directement au niveau grammatical, sans recours aux règles de production. Dans la règle suivante par exemple, pour transporter l'attribut x de B en A, il suffit simplement d'écrire:

A(x,y') -> B(x) C(y,z).

Il convient donc de pouvoir traiter un tel cas par l'analyse de la règle de grammaire elle-même, en sus des règles de production. Dans l'exemple ci-dessus cette analyse devrait ensuite déboucher sur l'étude de l'attribut x de B, ainsi que le préconise déjà l'extension précédente.

Une troisième extension, en partie dérivée de la première, consiste en la prise en compte des alternants du composant grammatical. Ceux-ci sous-tendent un certain non-déterminisme potentiel, susceptible d'interférer fortement avec les domaines d'attributs. Le cas extrême est celui de la totale exclusivité "réelle", laquelle devrait induire un partitionnement des domaines d'attributs. Toutefois, la difficulté de maîtriser (en particulier pour l'expert !) ce double non-déterminisme offert par le Descripteur de Structure, tend à remettre en cause l'intérêt des alternants dans les règles de grammaire...

La quatrième extension est liée à la représentation du monde (attribut,valeur) sur lequel porte le calcul de complétude. L'idée est de permettre une représentation des réalisations possibles coïncidant exactement avec la puissance du langage de spécification. Nous avons en effet supposé que le monde (cf 2.2) auquel se réfère une règle du Descripteur de Structure, peut être décrit par un produit cartésien. Or le langage permet l'affectation multiple de valeurs sur les attributs, que ce soit dans le Descripteur de Structure ou dans les Grammaires d'Analyse Syntaxique. Tous les cas ne sont donc pas nécessairement possibles. De même les contraintes d'acceptabilité, également attachées aux règles du Descripteur de Structure, réduisent les domaines des attributs. L'étude d'une règle devrait donc s'effectuer sur la base, non pas d'un produit cartésien, mais de l'intersection d'un produit cartésien et de la représentation ensembliste d'une contrainte, soit éventuellement, d'une union de produits cartésiens.

3.3 - A PROPOS DE L'INDETERMINISME

Nous avons déjà vu (cf 2.3.5) que la vérification du non-déterminisme entre deux règles de production concernant un même attribut (cf 2.3.3) ne permet pas la donnée explicite à l'expert des modèles communs aux prémisses des deux règles. Ceci peut s'avérer gênant dans le cas de prémisses complexes et demander à l'expert un travail fastidieux pour créer par exemple, deux prémisses disjointes. Une première idée serait de soumettre à l'expert l'ensemble "intersection" suggéré au paragraphe 2.3.5, si possible moyennant abstraction de celui-ci en terme de propriétés plutôt qu'en un listage exhaustif des n-uplets le constituant. Une alternative serait encore de

proposer directement deux prémisses disjointes à l'approbation de ce dernier en effectuant le retrait des modèles communs à chacune des deux prémisses suivant le principe de la complétude.

Dans le cas extrême d'indéterminisme où une prémisse P1 est incluse dans une autre prémisse P2, on pourrait également proposer à l'expert la modification standard visant à penser que l'erreur provient du non-retrait de la prémisse P2 de la prémisse P1 (cf 2.3.1 - 1er cas), et de ce fait créer automatiquement la prémisse P3 remplaçant la prémisse P1, avec P3 = P1 ^ ¬P2. Ce retrait des modèles de P2 dans P1 s'effectuerait à l'aide du principe utilisé pour le calcul de la complétude.

3.4 - A PROPOS DES DOMAINES DES ATTRIBUTS

Deux questions peuvent être formulées quant aux domaines des attributs: d'une part comment les obtenir, d'autre part comment les représenter.

En ce qui concerne le premier point, il serait regrettable d'être amené à exiger de telles connaissances de l'expert lui-même. Une construction automatique des domaines paraît nettement préférable. Celle-ci pourrait s'effectuer de proche en proche, à partir des domaines des attributs des Grammaires d'Analyse Syntaxique et en vertu de l'inférence exprimée dans les règles de production. D'où la nécessité: premièrement d'étendre l'analyse de l'inférence au-delà du Descripteur de Structure, jusqu'au niveau des Grammaires d'Analyse Syntaxique; deuxièmement de mettre en oeuvre une synthèse automatique des domaines d'attributs à partir des règles terminales.

En ce qui concerne la représentation des domaines de définition, ceux-ci sont actuellement présentés sous forme extensive, non pas valeur par valeur, mais par intervalles. De fait il est impossible de spécifier des domaines infinis. Or il serait agréable de pouvoir dire qu'un attribut prend ses valeurs dans l'ensemble des entiers plutôt que de devoir le restreindre à l'intervalle allant de 0 à 999 999 par exemple. Ceci nécessiterait la définition d'un symbole "infini" et des traitements en conséquence. Dans le même ordre d'idées il serait souhaitable de pouvoir accéder au type "réel". Ceci entrainerait de nombreuses modifications notamment pour le compactage d'un ensemble d'intervalles en un ensemble d'intervalles disjoints.

3.5 - A PROPOS DES AFFECTATIONS DANS LES REGLES DE PRODUCTION

Dans HAVANE, l'inférence exprimée dans les règles de production correspond à des affectations de valeurs à des attributs. Une difficulté apparaît donc, vu la puissance du langage mis à la disposition de l'expert pour l'écriture des expressions calculables figurant en partie droite de ces affectations: celle d'appréhender les domaines de valeurs induits par ces expressions, pour l'attribut figurant en partie gauche de l'affectation.

Par ailleurs il conviendrait d'assurer que les valuations d'attributs issues des règles de production ne sont pas en conflit avec les contraintes d'acceptabilité attachées à la même règle de grammaire. Ceci débouche en fait sur une étude plus approfondie de la cohérence entre règles de production exprimant l'inférence, et contraintes d'acceptabilté.

3.6 - A PROPOS DU DIALOGUE AVEC L'EXPERT

Le dialogue actuel est encore très succint. L'expert n'a par exemple pas la possibilité de spécifier l'attribut qu'il veut étudier. Il peut seulement désigner une règle du Descripteur de Structure à contrôler, ce qui entraîne un traitement global de tous les attributs y figurant. Une autre approche, probablement plus adaptée au contexte d'une interface expert, serait de pouvoir traiter directement un attribut. Ceci confèrerait en outre une plus grande souplesse au module de vérification, lui permettant d'être utilisé à la fois sur les directives de l'expert et de façon automatique pour les besoins d'un contrôle plus général.

CONCLUSION

Cette première approche de la validation de bases de connaissances appliquée au système HAVANE a donné lieu à deux réalisations:

- La validation d'une grammaire hors-contexte, soit un problème tout à fait classique;
- La validation (partielle) d'un ensemble de règles de production, qui dans HAVANE traduisent de l'inférence par le biais de calcul d'attributs. Celle-ci a été envisagée à travers deux aspects: la **complétude** d'un calcul (à savoir la "complète instanciation" d'un système d'attributs), qui est une contrainte inhérente au langage de spécification utilisé, et le **non-déterminisme**, qui nous paraît un révélateur clef des erreurs susceptibles d'être effectuées par un expert.

Au terme de cette étude, concrétisée par un programme écrit en PROLOG II sur Macintosh [GUE 86], il se dégage deux conclusions explicitées au chapitre 3. Il s'agit de:

- La complexité du problème au niveau règles de production. Les structures de données imposées par le monde (attribut,valeur), auquel se réfère la base de connaissances (i.e. des produits cartésiens), sont extrêmement lourdes et coûteuses à gérer. Bien que des optimisations soient envisageables, il est probable qu'elles n'affecteront que très superficiellement les résultats;
- La diversité des outils à mettre en place pour tenter non pas de garantir la validité des connaissances, mais de s'en approcher en s'appuyant sur des critères de cohérence.

Les techniques développées en compilation ont récemment débouché en programmation sur des environnements et des systèmes d'EAO/CAO relativement puissants et confortables. Bien que la problématique soit ici notablement plus large, la connexité des deux domaines reste très forte. Il s'agit donc pour nous d'atteindre des objectifs analogues. En conséquence, outre le fait de dégager de nouveaux critères de cohérence,nos perspectives de travail résident essentiellement dans l'intégration des différents outils proposés (notamment au chapitre 3), en vue de constituer un véritable environnement pour expert.

BIBLIOGRAPHIE

[ASS 86] C. ASSEMAT, A. BONNET, "Conceptualisation, cohérence et validation des Bases de Connaissances", Sixièmes Journées Internationales "Les systèmes experts et leurs applications", Avril 1984, Avignon (France),.pp 299-320.

[BAR 86] A. BARR, E.A. FEIGENBAUM, "Le manuel de l'intelligence artificielle", Tome 1, Eyrolles 1986, pp 45-48.

[BOS 85] P. BOSC, M. COURANT, S. ROBIN, "Spécification de connaissances pour une interface à grande liberté syntaxique", 5ème Congrès Afcet "reconnaissance des Formes et Intelligence Artificielle", Grenoble, Novembre 1985.

[BOS 86] P. BOSC, M. COURANT, S. ROBIN, "CALIN: a user interface based on a simple natural language", ACM Conference on Research and Development in Information Retrieval, Pise, 8-10 Septembre 1986.

[COU 84] B. COURCELLE, P. DERANSART, "Proofs of correctness for attribute grammars and recursive procedures", Rapport de Recherche INRIA N° 322, Juillet 1984.

[DER 86] P. DERANSART, M. JOURDAN, B. LORHO, "A survey of attribute grammars. Part 1: main results on attribute grammars", rapport de Recherche INRIA N° 485, Janvier 1986.

[GUE 86] G. GUEVEL, "Cohérence des spécifications dans le système HAVANE", Rapport de DEA, Université de Rennes, 1986.

[KNU 68] D.E. KNUTH, "Semantics of context-free languages", Mathematical System Theory, Vol. 2, N° 2, 1968, pp127-145.

[KOR 66] R. KORFHAGE, "Logic and algorithms", John Wiley & Sons Ed. 1966, pp 42-75.

[NIL 80] N.J. NILSSON, "Principles of Artificial Intelligence", Springer-verlag 1980,pp 361-415.

[RIC 84] E. RICH, "Artificial Intelligence", McGraw-Hill 1983, pp201-242.

[ROU 86] M.C. ROUSSET, "Consistance et validité des bases de connaissances", Journées du PRC "Intelligence Artificielle" - Pôle "Méthodologie des Applications", Avril 1986, Le bourget du Lac (Savoie).