Unification d'arborescences : évaluation sémantique d'énoncés en langue naturelle

Vania Joloboff

THESE

présentée à

Université Scientifique et Médicale de Grenoble

Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR INGENIEUR
Informatique

par

Vania JOLOBOFF

UNIFICATION D'ARBORESCENCES.

EVALUATION SEMANTIQUE D'ENONCES EN LANGUE NATURELLE.

Thèse soutenue le 8 septembre 1978 devant la Commission d'Examen :

Président : M. L. BOLLIET

Examinateurs : MM. ALARDO
COURTIN
JORRAND
VEILLON

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Joseph KLEIN : Vice-Président

MEMBRES DU CORPS ENSEIGNANT de l'U.S.M.G.

PROFESSEURS TITULAIRES.

MM	AMBLARD Pierre	Clinique de dermatologie
	ARNAUD Paul	Chimie
	ARVIEU Robert	I.S.N.
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique Expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale A
	BEAUDOING André	Clinique de Pédiatrie et Puériculture
	BELORIZKY Elie	Physique
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques Pures
	BEZES Henri	Clinique chirurgicale et Traumatologie
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Jean-Louis	Clinique Ophtalmologique
	BONNET-EYMARD Joseph	Clinique Hépato-gastro-entérologique
Mme	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BOUTET DE MONVEL Louis	Mathématiques Pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Clinique Oto-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	COUDERC Pierre	Anatomie pathologique
	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie

../..

MM.	DEPORTES Charles	Chimie minérale
	DESRÉ Pierre	Métallurgie
	DODU Jacques	Mécanique appliquée (IUT I)
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	FONTAINE Jean-Marc	Maths pures .
	GAGNAIRE Didier	Chimie Physique
	GALVANI Octave	Mathématiques pures
	GASTINEL Noël	Analyse numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique générale
	KLEIN Joseph	Mathématiques pures
	KOSZUL Jean-Louis	Mathématiques pures
	KRAVTCHENKO Julien	Mécanique
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
Mme	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie Pharmaceutique
	LAURENT Pierre	Mathématiques appliquées
	LEDRU Jean	Clinique médicale B
	LE ROY Philippe	Mécanique (IUT I)
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Jean-Marie	Sciences nucléaires
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques pures
MM.	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Clinique cardiologique
	MAYNARD Roger	Physique du solide
	MAZARE Yves	Clinique Médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MICOUD Max	Clinique Maladies infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEGRE Robert	Mécanique
	NOZIERES Philippe	Spectrométrie Physique
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET Jean	Séméiologie Médicale (Neurologie)
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REVOL Michel	Urologie
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-Chirurgie
	SARRAZIN Roger	Clinique chirurgicale B
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SIBILLE Robert	Construction mécanique (IUT I)
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie

MM.	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique Nucléaire
Mme	VERAIN Alice	Pharmacie galénique
MM.	VERAIN André	Physique - Biophysique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale

PROFESSEURS ASSOCIES

MM.	CRABBE Pierre	CERMO
	SUNIER Jules	Physique

PROFESSEURS SANS CHAIRE

Mlle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Gilbert	Géographie
	BENZAKEN Claude	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (IUT I)
	BUISSON René	Physique (IUT I)
	BUTEL Jean	Orthopédie
	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	CONTE René	Physique (IUT I)
	DELOBEL Claude	M.I.A.G.
	DEPASSEL Roger	Mécanique des fluides
	GAUTRON René	Chimie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biochimie médicale
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Médecine préventive
	IDELMAN Simon	Physiologie animale
	JOLY Jean-René	Mathématiques Pures
	JUILLIEN Pierre	Mathématiques Appliquées
Mme	KAHANE Josette	Physique
MM.	KRAKOWIACK Sacha	Mathématiques Appliquées
	KUHN Gérard	Physique (IUT I)
	LUU DUC Cuong	Chimie organique -Pharmacie
	MICHOULIER Jean	Physique (IUT I)
Mme	MINIER Colette	Physique (IUT I)
MM.	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	PFISTER Jean-Claude	Physique du solide
Mlle	PIERY Yvette	Physiologie animale

../...

MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
MM.	STIEGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	ARMAND Yves	Chimie (IUT I)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro-chirurgie
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (IUT B)(Personne étrangère habilitée à être recteur de thèse).
	BERNARD Pierre	Gynécologie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	COLIN DE VERDIERE Yves	Maths pures
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculai
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FAURE Gilbert	Urologie
	GAUTIER Robert	Chirurgie générale
	GIDON Maurice	Géologie
	GROS Yves	Physique (IUT I)
	GUIGNIER Michel	Thérapeutique
	GUITTON Jacques	Chimie
	HICTER Pierre	Chimie
	JALBERT Pierre	Histologie
	JUNIEN-LAVILLAVROY Claude	O.R.L.
	KOLODIE Lucien	Hématologie
	LE NOC Pierre	Bactériologie-virologie
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et médecine préventive
	MALLION Jean-Michel	Médecine du travail

.../...

MM.	MARECHAL Jean	Mécanique (IUT I)
	MARTIN-BOUYER Michel	Chimie (CUS)
	MASSOT Christian	Médecine interne
	NEMOZ Alain	Thermodynamique
	NOUGARET Marcel	Automatique (IUT I)
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B) (Personnalité étrangère habilitée à être directeur de thèse).
	PEFFEN René	Métallurgie (IUT I)
	PERRIER Guy	Géophysique-Glaciologie
	PHELIP Xavier	Rhumatologie
	RACHAIL Michel	Médecine Interne
	RACINET Claude	Gynécologie et Obstétrique
	RAMBAUD Pierre	Pédiatrie
	RAPHAEL Bernard	Stomatologie
Mme	RENAUDET Jacqueline	Bactériologie (Pharmacie)
	ROBERT Jean-Bernard	Chimie-Physique
	ROMIER Guy	Mathématiques (IUT B) (Personnalité étrangère habilitée à être directeur de thèse.)
	SAKAROVITCH Michel	Maths appliquées
	SCHAEKER René	Cancérologie
Mme	SEIGLE-MURANDI Françoise	Cryptogamie
	STOEBNER Pierre	Anatomie Pathologie
	STUTZ Pierre	Mécanique
	VROUSOS Constantin	Radiologie

MAITRES DE CONFERENCES ASSOCIES.

MM.	DEVINE Roderick	Spectro Physique
	KANEKO AKIRA	Maths pures
	JOHNSON Thomas	Maths appliquées
	RAY Tuhina	Physique

MAITRE DE CONFERENCES DELEGUE

M.	ROCHAT Jacques	Hygiène et Hydrologie (Pharmacie)

Fait à SAINT MARTIN D'HERES, NOVEMBRE 1977

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Monsieur Philippe TRAYNARD : Président

Monsieur Pierre-Jean LAURENT : Vice Président

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BLOCH Daniel	Physique du solide
	BONNETAIN Lucien	Chimie minérale
	BONNIER Etienne	Electrochimie et électrométallurgie
	BOUDOURIS Georges	Radioélectricité
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	COUMES André	Radioélectricité
	DURAND Francis	Métallurgie
	FELICI Noël	Electrostatique
	FOULARD Claude	Automatique
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie-Physique
	PAUTHENET René	Physique du solide
	PERRET René	Servomécanismes
	POLOUJADOFF Michel	Electrotechnique
	SILBER Robert	Mécanique des fluides

PROFESSEUR ASSOCIE

M.	ROUXEL Roland	Automatique

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuel	Electronique
	BOUVARD Maurice	Génie mécanique
	COHEN Joseph	Electrotechnique
	LACOUME Jean-Louis	Géophysique
	LANCIA Roland	Electronique
	ROBERT François	Analyse Numérique
	VEILLON Gérard	Informatique fondamentale et appliquée
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES

MM.	ANCEAU François	Mathématiques appliquées
	CHARTIER Germain	Electronique
	GUYOT Pierre	Chimie minérale
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du solide
	MORET Roger	Electrotechnique nucléaire
	PIERRARD Jean-Marie	Mécanique
	SABONNADIERE Jean-Claude	Informatique fondamentale et appliquée
Mme.	SAUCIER Gabrièle	Informatique fondamentale et appliquée

MAITRE DE CONFERENCES ASSOCIE

M.	LANDAU Ioan	Automatique

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

MM.	FRUCHART Robert	Directeur de Recherche
	ANSARA Ibrahim	Maître de Recherche
	CARRE René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Je tiens à remercier

Monsieur Louis BOLLIET Professeur à l'Université Scientifique et Médicale de Grenoble, qui m'a fait l'honneur de présider le Jury de cette thèse.

Monsieur Gérard VEILLON, Professeur à l'Institut National Polytechnique de Grenoble qui a encouragé ces travaux.

Monsieur ALARDO, Chef de service à la compagnie CII-HB qui a accepté de participer au Jury.

Messieurs Jacques COURTIN, Maître de Conférences à l'Université des Sciences Sociales de Grenoble, et Philippe JORRAND, Maître de Recherches au CNRS, pour l'intérêt qu'ils ont porté à cette thèse et qui ont accepté de la juger.

Je tiens également à remercier Augustin LUX et Ernest GRANDJEAN pour l'aide bienveillante qu'ils m'ont apporté.

Je voudrais aussi remercier Mme Geneviève BICAIS qui a assuré la dactylographie de cette thèse, ainsi que le service de reproduction qui en a réalisé le tirage.

TABLE DES MATIERES

PARTIE I - UNIFICATION D'ARBORESCENCES

CHAPITRE - 0 - DEFINITIONS - RAPPELS

CHAPITRE 1 - OPERATIONS ET APPLICATIONS SUR LES RAMIFICATIONS

CHAPITRE 2 - UNIFICATION D'ARBORESCENCES

PARTIE II - EVALUATION SEMANTIQUE D'ENONCES EN LANGUE NATURELLE

CHAPITRE 3 - ANALYSE DE QUELQUES MODELES ET TECHNIQUES

CHAPITRE 4 - UN NOUVEAU MODELE D'EVALUATION ET SA REALISATION

CHAPITRE 5 - EXEMPLES APPLICATIONS

INTRODUCTION

Les travaux présentés ici s'insèrent dans le cadre du traitement automatique des langues. Le terme d'évaluation sémantique ne signifie pas ici une compréhension "absolue" d'un énoncé, mais plutôt une interprétation dans le contexte fourni par l'application informatique considérée. Le modèle présenté se veut donc partiel, issu de considérations pragmatiques, et ne prétend pas à une théorie linguistique. L'interprétation effectuée consiste à vérifier la cohérence d'un énoncé et à le rattacher à des références connues dans le cadre du sous langage visé. Les critères linguistiques utilisés pendant l'analyse sont propres à ce sous langage et sont donc très liés à l'application considérée. Comme on vise une gamme assez large d'applications (commande d'un robot, interrogation d'un système informatique, documentation etc...) les éléments linguistiques du modèle se doivent d'être des paramètres du système, aisément modifiables par un utilisateur sans remettre en cause l'implémentation. Cette démarche a déjà porté ses fruits dans le système PIAF (Co3) utilisé ici comme préprocesseur, effectuant l'analyse morphologique et syntaxique. L'analyse "sémantique" définie ici consiste à établir une structure fonctionnelle de l'énoncé, c'est-à-dire à reconnaître dans la structure syntaxique la fonction des éléments de l'énoncé par rapport à l'univers réel décrit dans cet énoncé.

Le problème qui se pose ici est la reconnaissance d'une structure syntaxique arborescente comme vérifiant certaines propriétés, autrement dit l'unification d'une structure donnée et d'un modèle linguistique. Si les problèmes de transformations arborescentes ont fait l'objet de recherches poussées dans le domaine linguistique [Boi] [Chau] [Gins] [Ros] , il semble que ce problème d'unification n'ait été que peu traité.

Cette technique a été abordée dans les langages spécialisés d'Intelligence Artificielle [QLISP] [PLAN] [Rul] mais la présentation est restée expérimentale et informelle. D'autres algorithmes ont été plus formalisés mais se rangent dans ce qu'on peut appeler "algorithmes du premier ordre", c'est-à-dire où les éléments du modèle recouvrent une partie complète de la structure manipulée (un sommet, un sous-arbre complet) [Vel]

Des algorithmes sophistiqués ont été développés en calcul de prédicats mais opèrent sur des objets déjà formalisés [Huet2] et ne sont pas utilisables ici.

On s'est donc rattaché à définir formellement un algorithme d'unification opérant sur des arborescences, où les éléments des modèles représentent des fragments quelconques de structure. Cet algorithme est défini à l'aide du formalisme sur les ramifications développées par M. Pair.

Cette thèse comprend donc deux parties :

La première est consacrée à l'élaboration de l'algorithme d'unification dans les chapitres 1 et 2, le chapitre 0 étant une présentation des notations utilisées et un rappel sur la notion de ramification.
On verra que cet algorithme est très puissant et peut être utilisé dans un cadre différent comme celui de la transformation de programmes.

Cet algorithme est intégré dans un système d'analyse linguistique dont l'étude est l'objet de la seconde partie. Après un tour d'horizon des modèles et systèmes d'analyse linguistique au chapitre 3, on aborde au chapitre 4, la définition et la réalisation d'un nouveau système.
Deux applications sont commentées au chapitre 5.

CHAPITRE - 0

DEFINITIONS - RAPPELS

I - LISTES

Une liste est une suite ordonnée d'éléments d'un même ensemble.

L_E désigne l'ensemble des listes formées sur un ensemble E, contenant la liste vide Ω .

ϵ est l'élément vide de E.

Une liste $\ell \in L_E$ est notée $\langle e_1\ e_2 \dots e_p \rangle$ $e_1, e_2, \dots, e_p \in E$.

Application sur les listes

- tête $L_E \to E$

  tête(Ω) = ϵ

  tête($\langle e_1\ e_2 \dots e_p \rangle$) = e_1

- reste $L_E \to L_E$

  reste(Ω) = Ω

  reste($\langle e_1\ e_2 \dots e_p \rangle$) = $\langle e_2 \dots e_p \rangle$

- taille $L_E \to N$

  taille(Ω) = 0

  taille(ℓ) = taille (reste(ℓ))+1

- adjonction

  Cette application adjoint un élément en tête d'une liste, elle est notée $\|$.

$\| : E \times L_E \to L_E$

$e_1 \| \Omega = \langle e_1 \rangle$

$e_1 \| \langle e_2 \cdots e_p \rangle = \langle e_1 e_2 \cdots e_p \rangle$

$\varepsilon \| L = L$

- Concaténation

C'est l'opération de L_E notée . qui concatène deux listes

$\Omega \cdot \Omega = \Omega$

$\Omega \cdot \langle e_1 \cdots e_p \rangle = \langle e_1 \cdots e_p \rangle . \Omega = \langle e_1 \cdots e_p \rangle$

$\langle e_1 \cdots e_p \rangle . \langle e'_1 \cdots e'_n \rangle = \langle e_1 \cdots e_p e'_1 \cdots e'_n \rangle$

2 - LANGAGE ALGORITHMIQUE

Les algorithmes exprimés par la suite sont donnés sous forme de schémas de programme dans un langage similaire à un langage de programmation de haut niveau dont les primitives sont :

- l'affectation notée :=
  Elle autorise l'affectation de n-uplets
  $\langle x_1\ x_2\ x_3\rangle := \langle y_1\ y_2\ y_3\rangle$ signifie que l'on affecte aux x_i respectivement les y_i.

- l'instruction conditionnelle où ssi signifie sinon si

- l'itération indiquée par la structure

  boucle <liste de situations>
  corps de l'itération
  répéter
  situation_1 : instruction_1
  ⋮
  situation_n : instruction_n
  finboucle

La sémantique de cette instruction est la suivante :

. les situations sont des étiquettes

. si au cours de l'exécution du corps de l'itération on rencontre une de ces étiquettes situation_i, on sort de la boucle et on exécute immédiatement l'instruction de sortie instruction_i.

. sinon on itère indéfiniment l'exécution du corps de l'itération.

Cette instruction [Zahn], [Knuth] à l'avantage de regrouper les instructions "tantque" et "répéter jusqu'à" en une seule structure et d'éviter les tests multiples en sortie de boucle.

Exemple : procédure de multiplication de deux entiers

```
z ← 0
boucle <zéro déborde sortie>
  co si la multiplication est possible la variable z contient le
     produit des deux variables x et y co
  si x = 0 ou y = 0 alors zéro
  ssi z > limite alors déborde
  ssi y = 1 alors sortie
  sinon
       z ← z+x
       y ← y-1
  fsi
répéter
  zéro : z ← 0
  déborde : erreur ("débordement entier")
  sortie  : z ← z+x
finboucle
```

- Ce symbolisme comprend trois types de procédures :
  - . les actions qui ne délivrent pas de valeur et ne peuvent donc pas être utilisées au sein d'une expression
  - . les fonctions qui possèdent une valeur résultat de leur évaluation
  - . les prédicats qui sont des fonctions à valeur booléenne.

Dans une fonction ou un prédicat, le symbole → indique que l'algorithme s'arrête et que l'expression qui suit est la valeur de la fonction ou du prédicat.

Exemple :

```
fonction max(i,j)
       si i > j alors → i
       sinon → j
       fsi
finfonction
```

- Le symbolisme ne comporte pas de déclarations et autorise la manipulation de structure de données complexes.

3 - RAMIFICATIONS

On trouvera dans [Pa-Qu] [Berli] les définitions et démonstrations suivantes sur les ramifications.

3.1 - DEFINITION D'ARBORESCENCE PAR LA THEORIE DES GRAPHES

3.1.1 - Graphe

Un graphe est un doublet <X,U> ou X est un ensemble de sommets et U une relation binaire sur X.

. $\forall x,y \in X$ on écrira $x \, U \, y$ si le couple (x,y) satisfait la relation U.

. On note U(x) l'ensemble des $y \in X$ tels que $x \, U \, y$.

3.1.2 - Itinéraire

C'est un ensemble ordonné de sommets $x_1 \; x_2 \; \ldots \; x_n$ tel que

$\forall i \in N \quad 1 \leq i \leq n \qquad x_i \, U \, x_{i+1} \qquad$ ou/et $\qquad x_{i+1} \, U \, x_i$

3.1.3 - Circuit

C'est un ensemble ordonné de sommets $x_1 \; x_2 \; \ldots \; x_n$ tel que

- $\forall i \in N \quad 1 \leq i \leq n \qquad x_i \, U \, x_{i+1}$

- $x_1 = x_n$

3.1.4 - Graphe connexe

C'est un graphe tel que pour tout couple de sommets (x_o, x_n) il existe au moins un itinéraire $x_o x_1 \ldots x_n$.

3.1.5 - Arborescence

C'est un graphe fini connexe sans circuit tel que

- $\forall x \in X$ il existe au plus un y tel que y U x
- il existe un sommet et un seul x_o tel qu'il n'existe aucun $y \in X$ tel que y U x.

Ce sommet est appelé racine de l'arborescence.
Tout élément x d'une arborescence est appelé feuille si il n'existe aucun $y \in X$ tel que x U y

Une arborescence dont la racine est une feuille est dite dégénérée

3.1.6 - Arborescence orientée

C'est un triplet <X,U,O> tel que

- <X U> est une arborescence
- <X O> est un ordre partiel tel que
  - . les restrictions de O à chacun des sous-ensembles U(x), $x \in X$ sont des ordres totaux.
  - . si x U y et $x \not\emptyset z$ $x,y,z \in X$
    y et z ne sont pas comparables.

3.1.7 - Arborescence étiquetée

C'est un triplet <x,U,Δ> où Δ est une application de X dans un ensemble W.
W est le vocabulaire et ses éléments sont les étiquettes.

3.1.8. - Ramification

On appelle ramification toute suite finie d'arborescences orientées étiquetées.

L'ensemble des ramifications est le monoïde libre $\hat{W}$ engendré par les arborescences sur W.

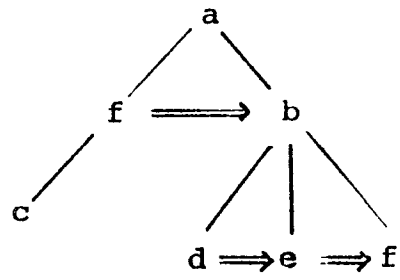
3.1.9. - Sous ramification

Si r = <X,U,O,Δ> une ramification sur W.
On appelle sous ramification de r toute ramification r' = <X',U',O',Δ'> telle que

- $X' \subseteq X$
- $\forall x', y' \in X'$ $\quad X' \; U' \; y' \iff x' \; U \; y'$
- $\forall x', y' \in X'$ $\quad x' \; O' \; y' \iff x' \; O \; y'$
- Δ' est la rectriction de Δ à X'
- $\forall x \in X', \quad y \in X$

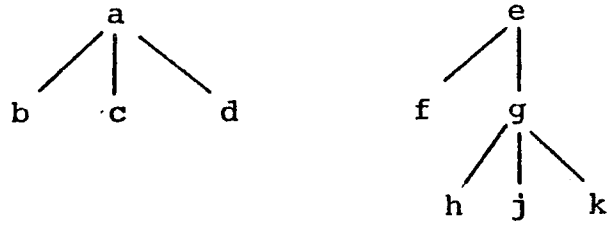
  . $x \; U \; y \Rightarrow y \in X'$
  . $x \; O \; y \Rightarrow y \in X'$
  . $y \; O \; x \Rightarrow y \in x'$

Exemples



est une arborescence orientée étiquetée ou a est une racine et c,d,e,f sont des feuilles

r =



est une ramification

3.2 - DEFINITION ALGEBRIQUE D'UNE RAMIFICATION

3.2.1 - Binoïde

On appelle binoïde sur un ensemble W un ensemble muni

- d'une loi de composition interne associative admettant un élément neutre, notée ici +
- d'une loi de composition externe à opérateur dans W, notée ×

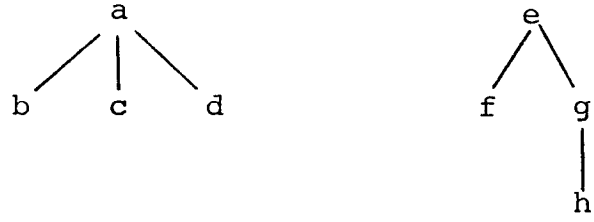
3.2.2 - Structure de binoïde sur $\hat{W}$

3.2.2.1 - Concaténation

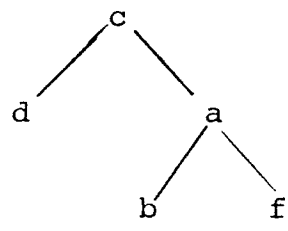
La loi + sur $\hat{W}$ est la concaténation de deux ramification, d'élément neutre la ramification vide Λ

Exemple :

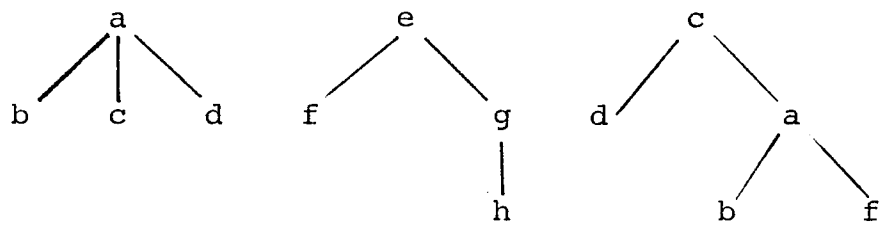
r =



r' =



r + r' =



3.2.2.2. - Enracinement

La loi × est appelée enracinement d'un élément de W sur une ramification

- $\forall a \in W \qquad \Lambda \times a = a$

- si r est une suite non vide d'arborescence $\langle X_i, U_i, O_i, \Delta_i \rangle$ supposant les X_i disjoints deux à deux et désignant par e_i la racine de $\langle X_i, U_i \rangle$, $r \times a$ est l'arborescence $\langle X, U, O, \Delta \rangle$ définie par :

- X est l'union des X_i et d'un élément e n'appartenant à aucun X_i

- $x \; U \; y \iff (\exists i \quad x \; U_i \; y)$ ou $(x = e$ et $\exists i \quad y = e_i)$

- $x \; O \; y \iff (\exists i \quad x \; O_i \; y)$ ou $(\exists i,j \quad i \leq j$ et $(x = e_i$ et $y = e_j))$

- $\Delta(x) = \Delta_i(x)$ si $x \in X_i$, $\Delta(e) = a$.

Réciproquement pour toute arborescence s sur W il existe un élément a de W et un seul, une ramification r sur W et une seule tels que $s = r \times a$.

$\hat{W}$ muni de + et × est un binoïde sur W.

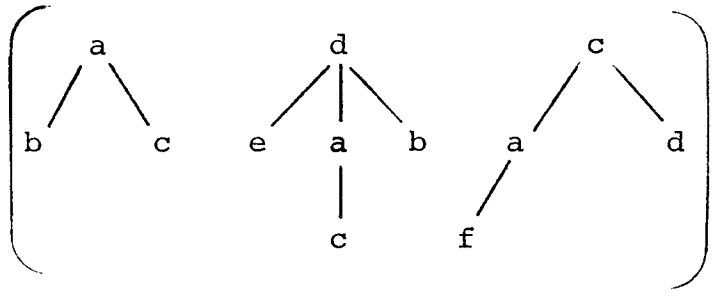
De ces définitions découle :

Pour toute ramification $u \in \hat{W}$ non vide, il existe $a \in W$, r et $s \in \hat{W}$ uniques tels que

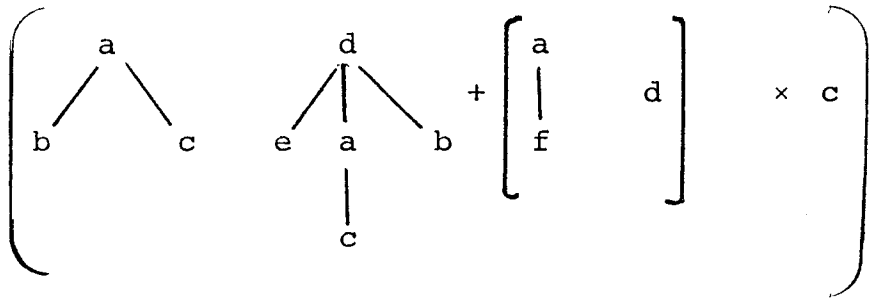
$$u = r + s \times a$$

Exemple :

$u =$



$=$



3.2.3 - Principe de récurrence

Si P est un prédicat tel que

a) $P(\Lambda)$ est vrai

b) $\forall r,s \in \hat{W}$ $(P(r)$ et $P(s)) \Rightarrow \forall a \in W$ $P(r+s \times a)$

alors $P(u)$ est vrai pour tout $u \in \hat{W}$.

3.2.4 - Mot des racines

C'est le mot obtenu en concaténant les racines de la ramification de gauche à droite, par l'application $\mu : \hat{W} \to W^*$.

$\mu(\Lambda) = \varepsilon$

$\mu(r+s\times a) = \mu(r)a$

3.2.5 - Mot des feuilles

C'est le mot obtenu par l'application $\nu : \hat{W} \to W^*$ en concaténant les feuilles de la ramification de gauche à droite

$\nu(\Lambda) = \varepsilon$

$\nu(r+s\times a) =$ si $s = \Lambda$ alors $\nu(r)a$ sinon $\nu(r)\nu(s)$ fsi

3.2.6 - Longueur d'une ramification

C'est le nombre d'éléments de la ramification

longueur : $\hat{W} \to N$

longueur$(\Lambda) = 0$

longueur$(r+s\times a) =$ longueur$(r)+1$.

CHAPITRE - I

OPERATIONS ET APPLICATIONS SUR LES RAMIFICATIONS

I.1 - PRESENTATION

Dans tout ce chapitre et le suivant on cherche à élaborer un algorithme d'unification entre deux types d'arborescences, dites modèles et déterminées. Un arbre modèle est défini comme une ossature comprenant des parties "variables". Ces variables se transforment par une application en une ramification. Le résultat d'une telle application à un arbre modèle est un arbre déterminé. L'unification de ces deux types d'arbres consiste donc à rechercher quelles sont les ramifications devant se substituer aux variables à travers l'application.

Le chapitre I est consacré à la définition des divers types d'arborescences utilisés et de quelques applications, le chapitre 2 étant réservé à l'algorithme lui-même.

Par la suite on ne s'intéresse qu'aux ramifications constituées à partir de

- un ensemble de variables V
- un ensemble de constantes C
- un élément spécial noté $*$

On notera :

. $C_+ = C \cup \{*\}$

. $V_+ \quad V \cup \{*\}$

. $W = \quad C \cup V$

. $W_+ = W \cup \{*\}$

. L l'ensemble des listes des ramifications sur $\hat{W}$.

On définit les applications :

- dernier : $\hat{W}_+ \to \hat{W}_+$

  dernier(Λ) = Λ

  dernier(r+s×a) = s × a

- suite : $\hat{W}_+ \to \hat{W}_+$

  suite(Λ) = Λ

  suite(r+s×a) = r

- prac (première racine) : $\hat{W}_+ \to W_+$

  prac(Λ) = ε

  prac(r+s×a) = a

- souram (sous-ramification) : $\hat{W}_+ \to \hat{W}_+$

  souram(Λ) = Λ

  souram(r+s×a) = s

1.2 - RAMIFICATION DETERMINEE

On nomme ainsi toute ramification de $\hat{C}$

. Notation

Dans les exemples les ramifications déterminées sont écrites sous une forme de chaîne de caractères parenthésée correspondant à une notation préfixée. ε désignant l'élément neutre de la concaténation des chaînes de caractères, cette écriture est donnée par l'application

$E\{\Lambda\} = \varepsilon$.

$E\{r+s\times a\} =$

{si s = Λ alors a sinon (a E{s}) fsi E{r}

Cette écriture est également obtenue par la grammaire

<ramification>::=<suite d'arbres>

<suite d'arbres>::=<suite d'arbres><arbre>

|<arbre>

<arbre> ::=(<racine><ramification>)

|<feuille>

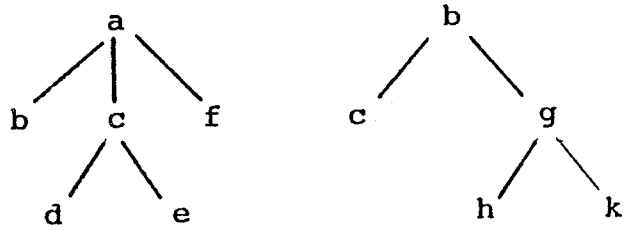
<feuille> ::=<étiquette>

<racine> ::=<étiquette>

Une étiquette est un élément de C .

. Exemple :

La ramification



est notée

(a b (c d e) f) (b c (g h k))

I.2.1 - Ramification simple

C'est une ramification de $\hat{C} \cup V$.

I.3 - SQUELETTES

Intuitivement, un squelette définit une arborescence sur laquelle peuvent venir se greffer des ramifications ceci exclusivement en certains points appelés points de rattachement. Ces points sont les feuilles de l'arborescence qui portent l'étiquette $*$.

L'ensemble S des squelettes résulte de l'union de deux sous-ensembles : S1 ensemble des squelettes déterminés, et S2 des squelettes non déterminés.

1.3.1. - Squelette déterminé

On appelle ramification squelette déterminée toute ramification de $\hat{C}_+$ vérifiant le prédicat

$$\text{squeldet} : \hat{C}_+ \to \{0,1\}$$

$$\text{squeldet}(r+s\times a) =$$

$$\text{squeldet}(r) \text{ et } (\text{si } a = * \text{ alors } s = \Lambda \text{ sinon } \text{squeldet}(s) \text{ fsi})$$

1.3.2. - Squelette non déterminé

C'est une ramification dont le mot des racines contient un seul élément, appartenant à V, et dont tous les éléments autres que la racine sont des feuilles étiquetées $*$.
Elles vérifient le prédicat

$$\text{squelnondet} : \hat{C}_+ \to \{0,1\}$$

$$\text{squelnondet}(r+s\times a) = a \in V \text{ et } r = \Lambda \text{ et } \text{étoiles}(s)$$

$$\text{étoiles}(x+y\times t) = (t=*) \text{ et } (y = \Lambda) \text{ et } \text{étoiles}(x)$$

1.3.3. - Exemple

On appelle ainsi toute arborescence non dégénérée de S .

$$\left.\begin{array}{l} (a * (b\ c * d) * c) \in S1 \\ \\ (A * *),\ (X * * *) \in S2 \end{array}\right\} \text{ sont des arbres squelettes}$$

I.3.4 - Greffe

On définit sur W_+ la fonction nbpa qui donne le nombre de points de rattachement d'une ramification soit le nombre d'occurrences de "*" dans le mot des feuilles.

$$nbpa : \hat{W}_+ \to N$$

$$nbpa(r+s\times a) = nbpa(r) + \underline{si}\ a = * \ \underline{et}\ s = \Lambda\ \underline{alors}\ 1\ \underline{sinon}\ nbpa(s)\ \underline{fsi}$$

L'opération de greffe est l'opération dans $\hat{W}_+$ qui consiste à enraciner une ramification sur un point de rattachement d'une arborescence ce point étant par définition le premier libre dans un parcours postfixé, c'est-à-dire le sommet de l'arborescence correspondant à la première étiquette * dans le mot des feuilles. Cette opération est notée $\otimes$ et définie comme suit :

$\forall u, r+s\times a \in \hat{W}_+$

- $\Lambda \otimes u = u$

- $u \otimes \Lambda = \Lambda$

- $u \otimes r+s\times a =$ $\underline{si}$ $(r = \Lambda$ $\underline{ou}$ $u \otimes r = r)$ $\underline{alors}$
  $\underline{si}$ $(s = \Lambda$ et $a = *)$ $\underline{alors}$ $r+u\times a$
  $\underline{sinon}$ $r+(u \otimes s) \times a$
  $\underline{fsi}$
  $\underline{sinon}$ $(u \otimes r) + s \times a$
  $\underline{fsi}$

Une arborescence ne contenant aucun point de rattachement reste inchangée par l'opération de greffe, la seule modification intervenant si $a = *$ et $s = \Lambda$.

I.4 - ACCROCHAGE

C'est l'itération de l'opération de greffe sur un arbre squelette autant de fois que cet arbre contient de points de rattachements, en greffant à chaque fois une nouvelle ramification. Cette opération est notée $\Gamma : S \times L \to \hat{W}_+$.
On dit que le résultat de Γ est Ψ si Γ n'est pas définie.

$\forall s \in S$, $\forall L = \langle r_1\ r_2 \ldots r_p \rangle \in L$ vérifiant $\forall i \quad 1 \leq i \leq p \quad nbpa(r_i) = 0 \quad r_i \neq \Lambda$

$s\ \Gamma\ \Lambda = s$

$s\ \Gamma\ L =$ si $nbpa(s) \neq taille(L)$ alors Ψ
sinon
$r_p \otimes (r_{p-1} \otimes (\ldots (r_2 \otimes (r_1 \otimes s))\ldots))$
fsi

I.4.1 - Lemme 1

Pour toute arborescence $r \in \hat{W}_+$ résultant d'une opération d'accrochage, il existe un seul $s \in S$ et un seul $L \in L$ tels que $r = s\ \Gamma\ L$.

Considérant une telle arborescence dont un seul sommet est étiqueté *, elle ne peut résulter que de la greffe de l'unique ramification dont ce sommet est la racine sur l'unique squelette dont il est une feuille.

Par récurrence sur ces sommets, le résultat est immédiat.

I.4.2 - Lemme 2

Pour toute ramification squelette $s = x+y\times t \in S$ $t \neq *$, toute suite de ramification $L = \langle r_1 r_2 \ldots r_p \rangle \in L$ tels que $s \Gamma L \neq \Psi$ il existe un entier $K = nbpa(x)$ tel que

$$s \Gamma L = x \Gamma \langle r_1 r_2 \ldots r_k \rangle + (y \Gamma \langle r_{k+1} r_{k+2} \ldots r_p \rangle) \times t$$

Démonstration :

- si $k = nbpa(x) = 0$

On a, par définition de Γ et sachant que la greffe laisse x inchangé

$$s \Gamma L = r_p \otimes (r_{p-1} \otimes (\ldots (r_2 \otimes (x + (r_1 \otimes y) \times t))) \ldots))$$

$$= r_p \otimes (r_{p-1} \otimes (\ldots r_3 \otimes (x+(r_2 \otimes (r_1 \otimes y)) \times t)) \ldots))$$

$$= x + r_p \otimes (r_{p-1} \otimes (\ldots (r_1 \otimes y))) \ldots) \times t$$

$$= x + (y \Gamma L) \times t.$$

- si $nbpa(x) \neq 0$
posant $K_1 = nbpa(x)$, on a

$$s \Gamma r = r_p \otimes (r_{p-1} \otimes (\ldots (r_1 \otimes x + y \times a)) \ldots))$$

et $K_2 = nbpa(r_1 \otimes x) = K_1 - 1$

Il est évident que pour tout i tel que

$$K_i = nbpa(r_{i-1} \otimes (r_{i-2} \ldots \otimes (r_1 \quad a))) \ldots)) > 0$$

on a :

$$s \Gamma r = r_p \otimes (r_{p-1} \otimes (\ldots r_{i+1} \otimes (r_i \otimes (r_{i-1} \otimes (r_{i-2} \otimes \ldots (r_1\ x)))\ldots) + y \times a)\ldots$$
i i

et ce jusqu'à ce que $i = k = nbpa(x)$ pour lequel

$$s \Gamma r = r_p \otimes (r_{p-1} \otimes (\ldots (x \Gamma \langle r_1 r_2 \ldots r_k \rangle) + y \times t$$

Posant $x \Gamma \langle r_1\ r_2 \ldots r_k \rangle = x'$ pour lequel $k' = nbpa(x') = 0$ on est ramené au cas précédent, d'où le résultat.

I.5 - ARBRE MODELE

On appelle arbre modèle toute arborescence de l'ensemble obtenu par fermeture de l'opération d'accrochage sur l'ensemble des ramifications simples à partir de S . On appelle ramification modèle toute sous ramification d'un arbre modèle. A designe l'ensemble des ramifications modèles.

I.5.1 - Caractéristique : il résulte de la construction des arbres modèles que :

$\forall u = r+s\times a \in A \quad a \in V \Rightarrow r = \Lambda$

si $a \in V$ la ramification est dite flexible

si $a \in C$ la ramification est dite rigide.

On définit donc sur A les deux prédicats

- flexible

flexible(Λ) = faux
flexible($r+s\times a$) = si $a \in V$ alors vrai sinon faux fsi

- rigide

rigide(Λ) = vrai
rigide($r+s\times a$) = si $a \in C$ alors rigide(r)
sinon faux fsi

I.5.2 - Notation

L'élément * n'étant qu'un indicateur structurel, on ne le représente pas dans la représentation parenthésée adoptée pour les exemples mais on le symbolise par les éléments [et] .

L'écriture d'un arbre modèle est donnée par la fonction EM

EM{Λ} = ε
EM{r+s×a} =
{si s = Λ alors a ssi a = * alors[EM{s}]
sinon (a EM{s})
fsi } EM{r}

ou par la grammaire

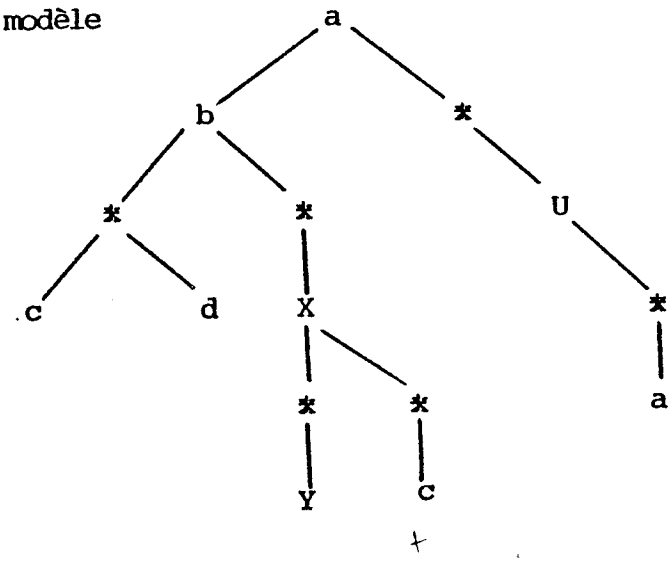
<ramification modèle>::=<ramification modèle><branche>
|<branche>
<branche> ::=<arbre modèle>
|<rameau>
|<feuille>
<arbre modèle> ::=(<racine><ramification modèle>)
<rameau> ::=[<ramification modèle>]
<racine> ::=<étiquette>
<feuille> ::=<étiquette>

Une étiquette est un élément de W.

On appelle rameau toute ramification enracinée sous l'élément *.

Exemple :

L'arbre modèle



s'écrit :

(a (b[c d] [(X [Y][c])]) [(U[a])])

I.6 - ARBRES QUASI DETERMINES

C'est un élément de la partie Q de A obtenue par fermeture de la restriction de Γ à l'ensemble des squelettes déterminés, sur $\hat{C}$. Un arbre quasi déterminé est donc un arbre modèle dont toutes les étiquettes distinctes sont constantes

(a [b c] d(e [(f b)])) est un arbre quasi déterminé.

I.6.1 - Fonction de rattachement

C'est une application qui rigidifie un arbre quasi déterminé en le ramenant à un arbre déterminé par suppression des points de rattachement.
Sur la notation préfixée adoptée ceci revient tout simplement à supprimer les symboles [et] .

Cette application est notée R : $Q \to \hat{C}$ et ainsi définie

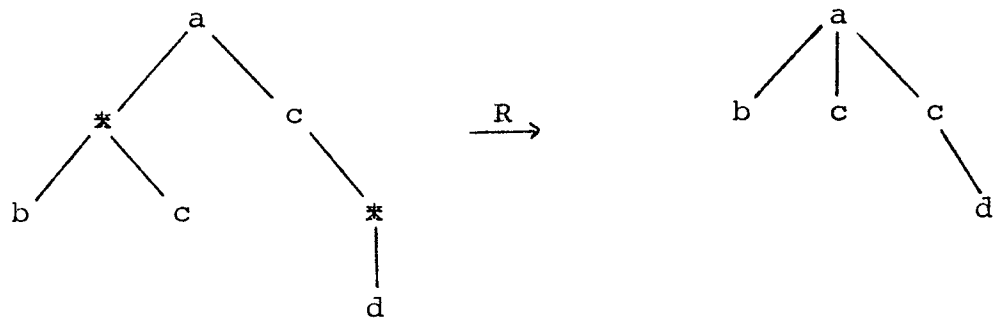
- $R(\Lambda) = \Lambda$

- $R(r+s\times a) = R(r) +$ si a = * alors R(s) sinon R(s) × a fsi

Exemple :

R{(a [b c] (c [d]))} = (a b c (c d))

soit :



I.6.2 - Equivalence d'arbres quasi déterminés

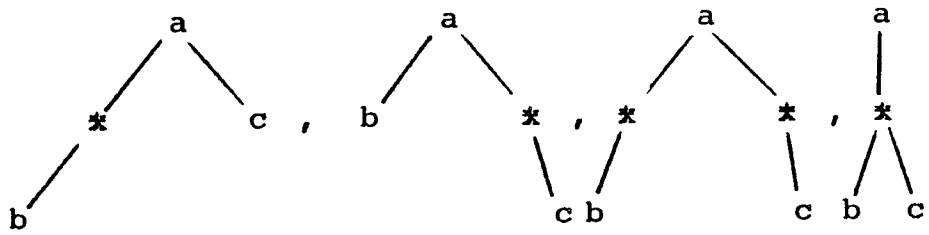
On définit la relation ≡ sur $Q \times Q$ comme suit :

$\forall x,y \in Q$ $\qquad x \equiv y \iff R(x) = R(y)$

On choisit évidemment pour représentant d'une classe d'équivalence l'élément déterminé de cette classe : $D = Q / \equiv$

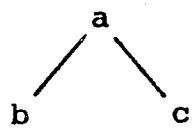
Cette relation exprime les différentes décompositions possibles d'un arbre déterminé en arbres quasi déterminés équivalents, en fonction de la combinaison de rameaux possible

Exemple :



qui s'écrivent (a b [c]) , (a [b] [c]) , (a [b c])
sont équivalent à (a b c)

soit



Un arbre déterminé peut se décomposer en un nombre fini d'arbres quasi déterminés équivalents.

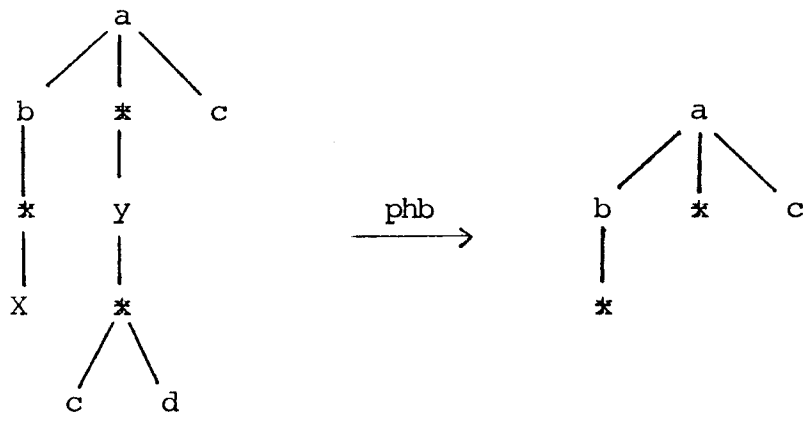
I.7 - PLUS HAUTE BRANCHE D'UN ARBRE MODELE

C'est l'arbre squelette sommet d'un arbre modèle, c'est-à-dire celui sur lequel s'est effectué l'accrochage qui a produit l'arbre modèle, donné par l'application phb : $A \to S$

- phb(Λ) = Λ

- phb(r+s×a) = phb(r) + si a = * alors a sinon phb(s)× a fsi

Exemple :



soit phb{(a (b ⌈X⌉) [(Y ⌈c d⌉)] c)} = (a (b *) * c)

I.8 - BASSES BRANCHES D'UN ARBRE MODELE

Corollairement à la définition de la plus haute branche, c'est le résultat de l'application bb qui produit la suite de ramifications qui a été la source de l'accrochage ayant produit l'arbre modèle auquel elle s'applique.

$bb : A \rightarrow L$

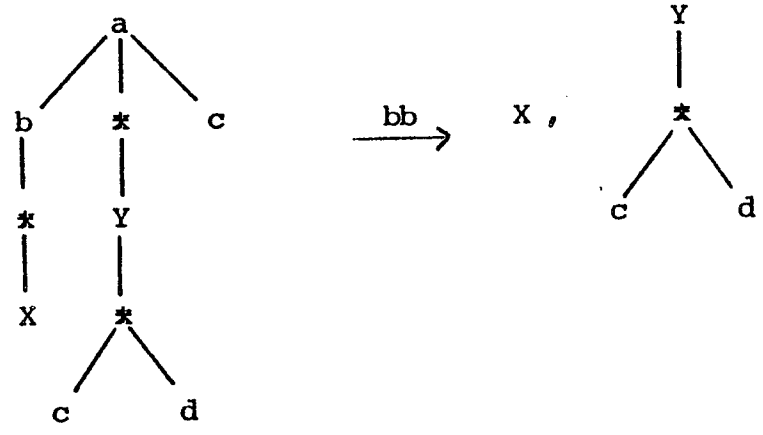
$bb(\Lambda) = \Omega$

$bb(r \dagger s \times a) = bb(r)$
. (si a = * alors ⟨s⟩
sinon bb(s) fsi)

Exemple :

bb{ (a (b [X]) [(Y [c d])] c)} = {[X] , [(Y [c d])]}

ou encore :



I.9 - PROPRIETES

1 Il résulte de la définition de phb que pour tout $u \in A$ phb(u) est un arbre dont tous les sommets étiquetés * sont des feuilles, donc que phb(u) est un arbre squelette.

2 Il est trivial de montrer par récurrence que
nbpa(phb(u)) = taille (bb(u)).

3 Pour tout arbre modèle $r \in A$ phb(r) Γ bb(r) = r
- si $r = \Lambda$ phb(Λ) Γ bb(Λ) = Λ Γ Ω = Λ
- si $r \neq \Lambda$ ∃x,y,t tels que r = x+y × t
Par définition de ces fonctions on a :

. si t = *

phb(r) Γ bb(r) = phb(x+y×t) Γ bb(x+y×t)

= (phb(x)+t) Γ bb(x) . <y>

Posant bb(x) = $\langle r_1 r_2 \ldots r_k \rangle$

phb(r) Γ bb(r) = $y \otimes (r_k \otimes (\ldots (r_1 \otimes (phb(x)+t))\ldots))$

= $y \otimes (r_k \otimes (\ldots (r_1 \otimes phb(x)))))\ldots))+t$

puisque nbpa (phb(x)) = k

= y ⊗ (phb(x) Γ bb(x)+t)

= phb(x) Γ bb(x)) + y × t

. si t ≠ *

phb(x+y×t) Γ bb(x+y×t) = (phb(x)+phb(y)×t) Γ <bb(x) bb(y)>

= (phb(x) Γ bb(x)) + (phb(y) Γ bb(y)) × t

d'après le lemme 2.

D'où le résultat par récurrence.

4 Lemme 3

Il résulte de la propriété précédente et du lemme 1

$$r = x \Gamma y = x' \Gamma y' \iff \begin{cases} x = x' = phb(r) \\ y = y' = bb(r). \end{cases}$$

I.10 - EXPANSION - SUBSTITUTION

I.10.1 - Définition

On cherche à définir un moyen de transformer un arbre modèle en un arbre quasi déterminé. Ceci se réalise en remplaçant chaque variable de l'arbre modèle par une ramification ne contenant pas de variables.

Les ramifications qui doivent remplacer les variables sont données par une substitution.

Etant donné une ramification modèle A, soit VA le sous-ensemble de V des étiquettes variables de A.

On appelle substitution tout sous-ensemble de $V \times \{\hat{C} \cup Sl\}$ défini par une application de VA dans $\hat{C} \cup Sl$.
Une substitution est donc un ensemble de couples et chacun de ces couples est appelé règle de substitution.
Φ désigne l'ensemble des substitutions.

ρ est l'application qui donne la ramification associée à une variable V pour une certaine substitution σ .

$$\rho : \Phi \times \quad \to \hat{C} \cup Sl$$

$$\rho(\sigma, v) = \text{si } \exists \langle v\ r \rangle \in \sigma \text{ alors } r$$
$$\text{sinon } \Lambda$$

L'expansion notée $\circ$ est alors définie comme suit

$$\circ : \Phi \times A \to Q$$

$$\sigma \circ \Lambda = \Lambda$$

$$\sigma \circ (r+s\times a) = \underline{\text{si}}\ a \in V\ \underline{\text{alors}}$$
$$\underline{\text{si}}\ s = \Lambda\ \underline{\text{alors}}$$
$$\rho(\sigma,a)$$
$$\underline{\text{sinon}}$$
$$\rho(\sigma,a)\ \Gamma\ \lambda(\sigma,bb(s\times a))$$
$$\underline{\text{fsi}}$$
$$\underline{\text{ssi}}\ \lambda(\sigma,bb(s\times a)) = \Psi \text{ ou } \sigma \circ r = \Psi \text{ alors } \Psi$$
$$\underline{\text{sinon}}$$
$$\sigma \circ r + phb(s\times a)\ \Gamma\ \lambda(\sigma,bb(s\times a))$$
$$\underline{\text{fsi}}$$

où λ est l'application $\Phi \times L \to L$ ainsi définie

$$\lambda(\sigma,L) = \underline{\text{si}}\ L = \Omega\ \underline{\text{alors}}\ \Omega$$
$$\underline{\text{sinon}}\ \ \sigma \circ \text{tête}(L)\ \|\ \lambda(\sigma,\ \text{reste}(L))$$
$$\underline{\text{fsi}}$$

De façon moins formelle, on peut dire que l'expansion s'applique de manière similaire à une grammaire de macro-expansion sur la notation parenthèsée, en mode inside-outside [Fisch] , c'est-à-dire en appliquant d'abord la règle associée à la variable la plus interne du parenthésage.

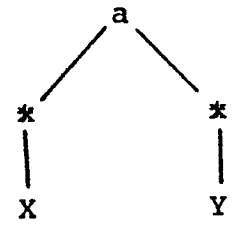
Chaque règle s'applique alors comme suit :

- si cette variable est une feuille, on la remplace simplement par la ramification qui lui est associée.

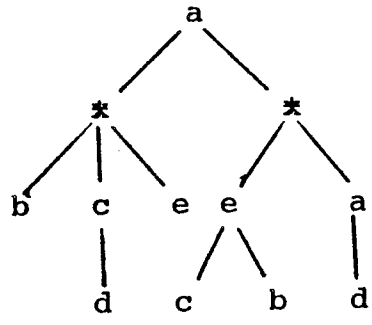
- si cette variable est racine on accroche à l'arbre squelette auquel cette variable est associée, les basses branches de l'arbre dont elle est la racine.

Exemple :

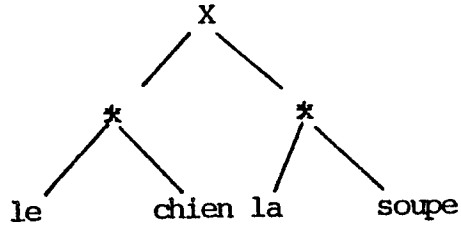
. L'expansion de



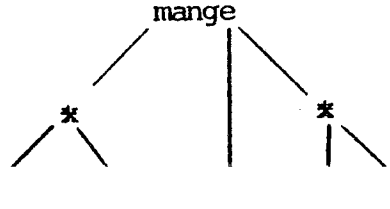
par σ ={<X b (c d) e>,<Y (e c b) (a d)>}
donne



. L'expansion de



par σ = {<X (mange * vite *)>}
donne



I.10.2 - Propriétés

Si il existe un arbre modèle $M = x+y\times t \in A$, un arbre quasi déterminé $Q = r+s\times a \in \mathcal{Q}$ et une substitution $\sigma \in \Phi$ tels que

$$\sigma \circ M = Q$$

alors

1. $M \neq \Lambda \iff Q \neq \Lambda$

2. si $t \in V$ (alors $r = \Lambda$ puisque $M \in A$)

- si $y = \Lambda$ alors $\rho(\sigma,t) = r+s\times a$
- si $y \neq \Lambda$

$$\sigma \circ M = \rho(\sigma,t) \; \Gamma \; \lambda(\sigma,bb(y)) = r+s\times a$$

$$\Rightarrow \begin{cases} \rho(\sigma,t) = phb(r+s\times a) \\ \text{et} \\ \lambda(\sigma,bb(y)) = bb(r+s\times a) \end{cases} \qquad \text{d'après le lemme 3}$$

3. $t \in C$

$$\sigma \circ M = \sigma \circ x + phb(y\times t) \; \Gamma \; \lambda(\sigma,bb(y)) = r+s\times a$$

$$\Rightarrow \begin{cases} \text{longueur } (\sigma \circ x) = \text{longueur}(x) = \text{longueur}(r) \\ phb(y\times t) = phb(s\times a) \\ \lambda(\sigma,bb(y\times t)) = bb(s\times a) \\ \sigma \circ x = r \end{cases}$$

Exemple 2 : Application de SUB2 à (H [(F [(E [X])])] [(D [X])])

SUB2 {
H → (append *'*)
D → (cons (car * nil))
F → (reverse *)
E → (cdr *)
X → ℓ

La première règle appliquée est la règle X

(H [(F [(E [ℓ])])] [(D [ℓ])])

Ensuite viennent les règles D et E

(H [(F [(cdr [ℓ])])] [(cons (car [ℓ]) nil)])

Puis la règle F

(H [(reverse [(cdr [ℓ])])] [(cons (car [ℓ] nil)])

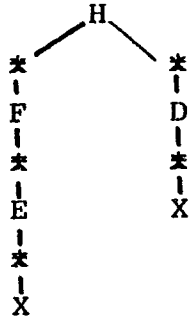
Et enfin la règle H qui donne

(append [(reverse [(cdr [ℓ]))] [(cons (car [ℓ] nil)])

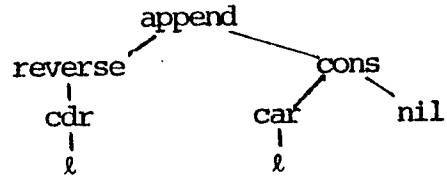
Et après rattachement on obtient

(append (reverse (cdr ℓ)) (cons car ℓ nil)))

SUB2 a donc eu pour effet de transformer



en



A partir d'un même arbre modèle, différentes substitutions peuvent produire par expansion des arbres quasi déterminés équivalents.

On aurait pu par exemple obtenir ici avec la substitution SUB3 suivante le même arbre déterminé après rattachement qu'avec SUB2

SUB3 {
H → (append (cons *) nil)
D → (car *)
F → (reverse *)
E → (cdr *)
X → ℓ

I.10.3 - Exemples

Exemple 1

Application de la substitution SUBI à l'arbre :

(mange [(SUJ[MOD])] [ADV] [OBJ])

SUBI
- SUJ → (chien le *)
- MOD → [beau petit]
- ADV → [vite]
- OBJ → [(soupe la chaude)]

La première règle appliquée est la règle MOD qui produit

(mange [(SUJ [beau petit])] [ADV] [OBJ])

Application de ADV

(mange [(SUJ [beau petit])] [vite] [OBJ])

Application de OBJ

(mange [(SUJ [beau petit])] [vite] [(soupe la chaude)])

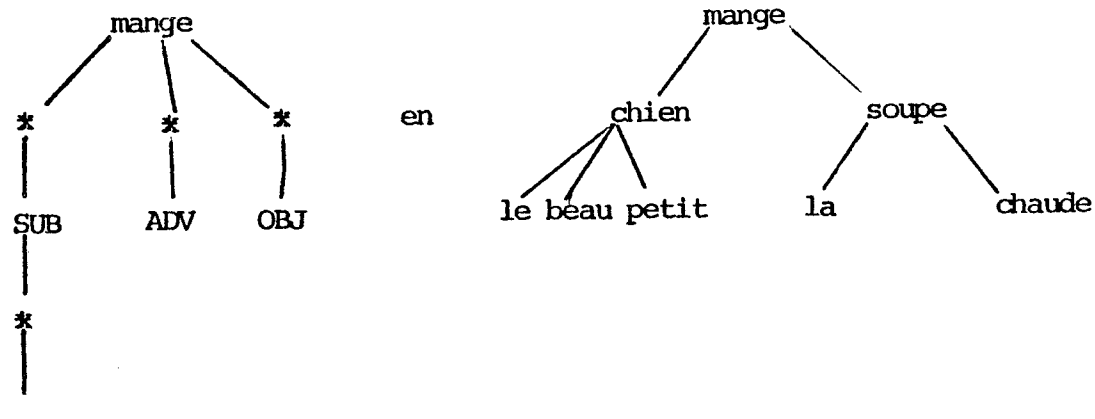
Application de SUJ

(mange [(chien le [beau petit])] [vite] [(soupe la chaude)])

Effectuant l'opération de rattachement on obtient finalement

(mange (chien le beau petit) vite (soupe la chaude))

L'expansion, suivie du rattachement, a donc instantié



CHAPITRE - II

UNIFICATION D'ARBORESCENCES

On s'intéresse maintenant à la construction de substitutions permettant le passage d'un arbre modèle à un arbre déterminé donné. Si une telle substitution existe, elle permet de reconnaître l'arbre déterminé comme une particularisation spécifique du modèle.

2.1 - UN PREMIER ALGORITHME

2.1.1 - Définition

Etant donné un arbre modèle appelé source et un arbre déterminé appelé cible, on cherche à construire la ou les substitutions produisant à partir de la source, un arbre quasi déterminé équivalent à la cible.

A cet effet, on construit dans un premier temps la classe d'équivalence de l'arbre déterminé, puis pour chaque élément de cette classe, on applique l'algorithme suivant, utilisant les propriétés de l'application o.

```
fonction construiresub (source, cible)
  co source est une ramification modèle x+y×t
     cible est une ramification quasi déterminée r+s×a
     Le résultat de la fonction est soit la substitution σ qui se présente sous
     forme d'une liste de couples <variable, ramification>, soit la valeur fail
  co
  si source = Λ et cible = Λ alors sigma := Ω
  ssi  source = Λ ou cible = Λ alors
      co longueur (source) ≠ longueur cible) co
         sigma := fail
  ssi variable (prac(source)) alors
      co t ∈ V et on utilise la propriété 2 de σ co
      si souram(source) = Λ alors co y = Λ co
          sigma := <<prac(source) cible>>
      sinon
         subpart := loop (bb(source),bb(cible))
         co on recherche la substitution partielle, union des substitutions
            de chacune des basses branches de source et cible co
         si subpart = fail alors
               sigma := fail
         sinon
               sigma := <prac(source) phb(cible)>||subpart
         fsi
      fsi
  sinon
      subpart := construiresub(suite(source),suite(cible))
      co on recherche la substitution entre x et r puis on utilise la propriété
         3 co
      si subpart = fail alors
               sigma := fail
      ssi phb(dernier(source)) ≠ phb(dernier(cible)) alors
               sigma := fail
      sinon
          sigma := subpart
          subpart := loop (bb(dernier(source)),bb(dernier(cible)))
          si subpart = fail alors
               sigma := fail
          sinon
               sigma := sigma.subpart
          fsi
      fsi
  fsi
  → sigma
```

```
fonction loop(listesource, listebut)
  si taille (listesource) ≠ taille(listebut) alors → fail
  sinon result := Ω
      boucle <sortie échec>
          si listesource = Ω alors sortie
          sinon
               subpart := construiresub(tete(listesource),tete(listebut))
               si subpart = fail alors échec
               sinon
                    result := subpart . result
                    listesource := reste(listesource)
                    listebut := reste(listebut)
               fsi
          fsi
      répeter
          sortie : → result
          echec  : → fail
      finboucle
  fsi
finfonction
```

2.1.2 - Exemple

On recherche les substitutions permettant de passer de

(VERBE[(et[SUJ1][SUJ2])][OBJ])

à

(mangent(et(chien le) (chat le))(soupe la))

Il est évident que parmi tous les arbres quasi déterminés équivalents, seuls ceux résultant d'un accrochage sur un squelette à deux points d'attachements sont susceptibles de fournir un résultat. Parmi ceux-ci sont encore éliminés ceux ne contenant pas la sous structure [(et * *)]

Seulement deux décompositions peuvent fournir un résultat

Q1 : (mangent [(et [(chien le)][(chat le)][(soupe la)])

Q2 : (mangent [(et [(chien le)][(chat le)] (soupe[la]))

Ces deux décompositions mènent à :

pour Q1

On construit <VERBE (mangent * *)>

et on traite les listes

<(et[SUJ1][SUJ2])>

<(et[(chien le)][(chat le)]), (soupe la)>

d'où les nouvelles listes

<SUJ1, SUJ2> et <(chien le) , (chat le)>

et finalement

<SUJ1 (chien le)>

<SUJ2 (chien le)>

<OBJ (soupe la)>

Pour Q2

On construit <VERBE (mangent * (soupe *))>

La branche 'et' amène au même résultat que Q1 mais on construit <OBJ la>

On obtient donc finalement deux substitutions

S1
<VERBE (mangent * *)>
<SUJ1 (chien le)>
<SUJ2 (chat le)>
<OBJ (soupe la)>

S2
<VERBE (mangent * (soupe *))>
<SUJ1 (chien le)>
<SUJ2 (chat le) >
<OBJ la >

2.1.3 - Conclusion

Parmi toute la classe d'équivalence d'un arbre déterminé donné, seuls quelques arbres peuvent fournir une substitution. La construction de toute la classe et l'itération de l'algorithme sur tous ses éléments pour trouver les substitutions possibles est donc une méthode lourde.

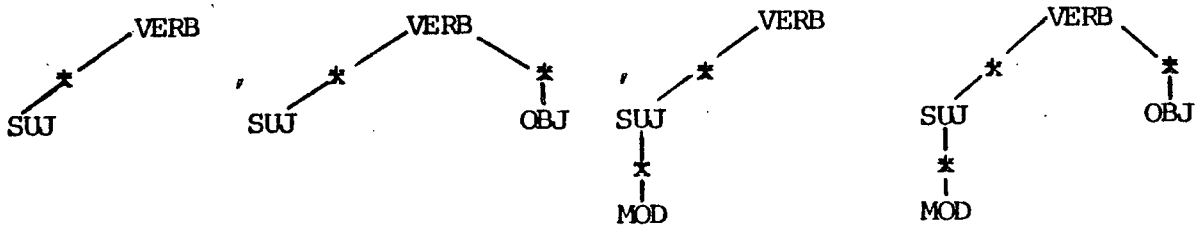
D'autant plus que l'on est à même de détecter, en se servant du modèle comme guide, les décompositions intéressantes. On va donc chercher à reformuler cet algorithme, en ne considérant plus toute la classe d'équivalence, mais en construisant à chaque étape seulement les décompositions intéressantes.

Il est à remarquer également que le formalisme tel qu'il est défini impose des contraintes inutiles.

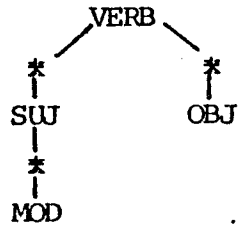
Si l'on veut par exemple construire les arbres modèles recouvrant les structures syntaxiques de dépendance équivalent à la grammaire

<phrase>::=<groupe sujet><groupe verbal>
< groupe sujet>::= groupe nominal
groupe nominal<modulateur>
< groupe verbal>::=<verbe>
|<verbe><groupe objet>

on est amené à définir les modèles



alors qu'il est beaucoup plus simple de pouvoir définir le seul modèle



dans lequel certaines parties peuvent être facultatives.

A cet effet, on introduit une règle particulière dans les substitutions dite

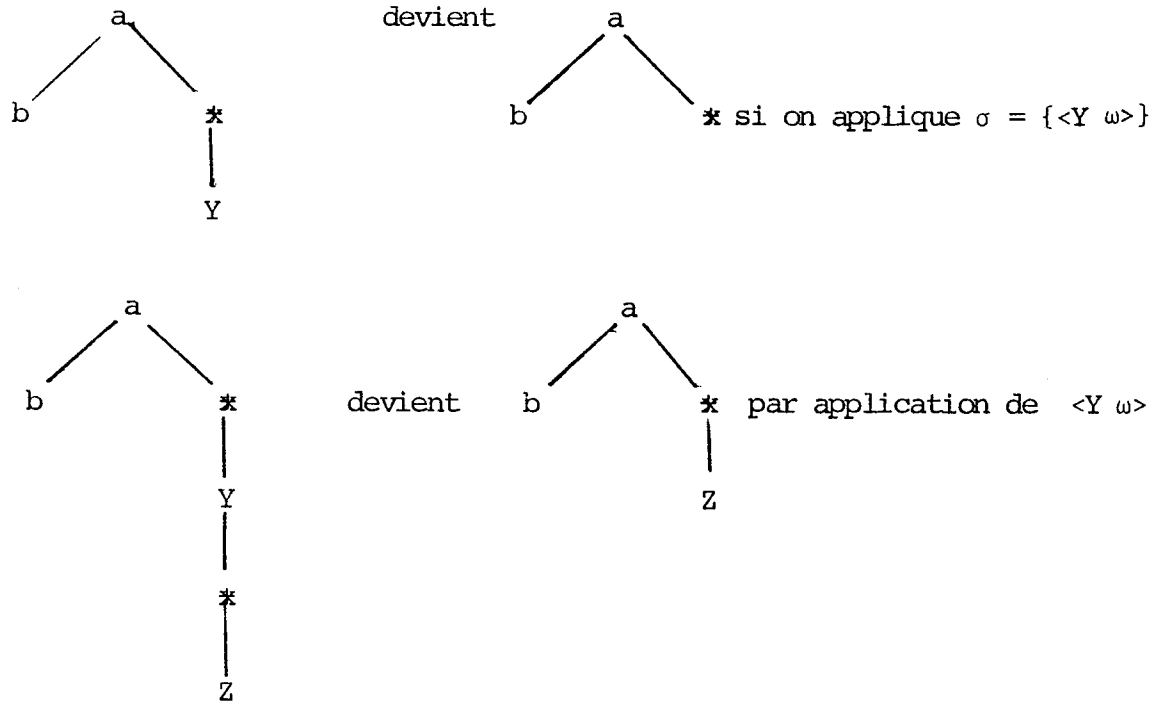
2.2 - REGLE VIDE

Si, dans une substitution, une variable V est associée à la règle vide on note cette règle <V ω>

Dans le cas où cette variable est une feuille, l'application de la règle vide a pour effet de supprimer cette feuille.

Dans le cas où cette variable est une racine, la règle vide a pour effet de substituer au point de rattachement sous lequel la variable est enracinée le point de rattachement dont elle est la racine, à condition que ce dernier soit unique.

Exemple :



2.2.1 - Nouvelle définition de O

L' introduction de la règle vide a pour effet de modifier l'opération o et les propriétés qui lui étaient associées. On introduit donc une nouvelle définition

$$o : \Phi \times A \to Q$$

$\forall \sigma \in \Phi$, $r+s\times a \in A$, on dit que σ o $(r+s\times a) = \Psi$ si o n'est pas définie et

σ o $(r+s\times a)$ =

si flexible $(r+s\times a)$
  si s = Λ alors $\rho(\sigma,a)$
  ssi $\rho(\sigma,a) = \omega$ alors
    si s $\neq$ * alors Ψ
    sinon σ o bb(s×a)
    fsi
  ssi $\lambda(\sigma,bb(s\times a)) = \Psi$ alors Ψ
  sinon $\rho(\sigma,a)$ Γ $\lambda(\sigma,bb(s\times a))$
  fsi
ssi $\lambda(\sigma,bb(s\times a)) = \Psi$ ou σ o r = Ψ alors Ψ
sinon
  σ o r + phb(s×a) Γ $\lambda(\sigma,bb(s\times a))$
fsi

Cette définition entraîne également la modification de l'opération de rattachement afin de supprimer les feuilles * qui subsistent après application d'une règle vide.

$$R : A \to \hat{C}$$

R(r+s×a) = R(r) + si a = * alors si s = ω alors Λ
                                sinon R(s)
                                fsi
                  sinon R(s)×a
                  fsi

2.2.2 - Exemples

Application de :

$$SUB \begin{cases} \langle H\ (mult * *)\rangle \\ \langle F\ (fact *)\rangle \\ \langle D\ \omega\rangle \\ \langle E\ (moins * 1)\rangle \\ \langle X\ x\rangle \end{cases}$$

à $\quad M = (H[(D[X])][(F[(E[X])])])$

$SUB \circ M = (mult * *)\ \Gamma\ \lambda\{SUB, \langle (D[X]), (F[(E[X])])\rangle\}$

$SUB \circ (D[X]) = SUB \circ X = x$

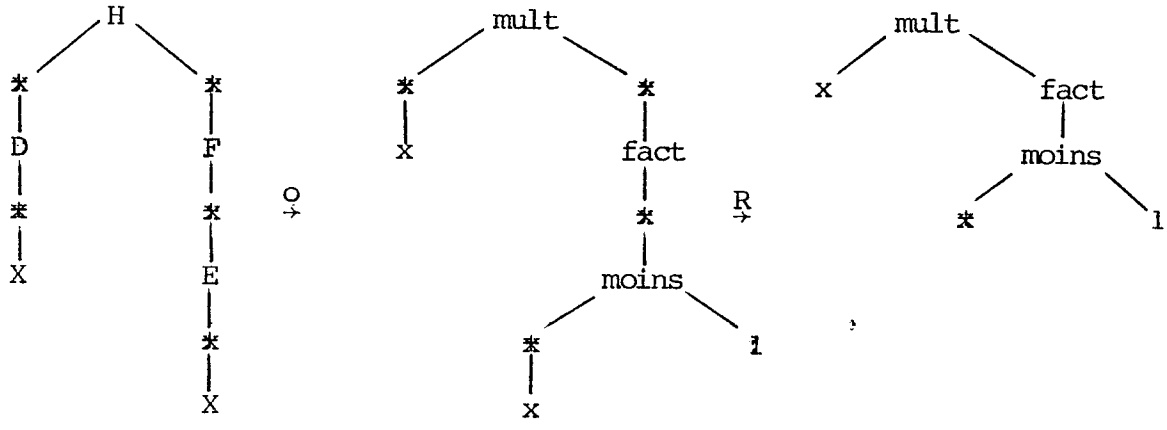
(1) $\quad SUB \circ (F[(E[X])]) = (fact *)\ \Gamma\ \lambda\{SUB, \langle (E[X])\rangle\}$

$SUB \circ (E[X]) = (moins * 1)\ \Gamma\ \lambda\{SUB, \langle X\rangle\}$
$= (moins[x]1)$

d'où (1) $\rightarrow$ (fact [(moins[x]1)])

et $SUB \circ M = (mult[x][(fact[(moins[x]1)])])$

et $\mathcal{R}(SUB \circ M) = (mult\ x\ (fact\ (moins\ x\ 1)))$



Application de

$$SUBV1 \begin{cases} <\text{VERBE (mange } * *)> \\ <\text{SUJ } \omega> \\ <\text{MOD le(grand très)}> \\ <\text{OBJ } \omega> \end{cases}$$

à $M = (\text{VERBE}[(\text{SUJ}[\text{MOD}])][\text{OBJ}])$

$SUBV1 \circ M$

$= (\text{mange } * *) \Gamma \lambda(SUBV1, <(\text{SUJ}[\text{MOD}])\ \text{OBJ}>)$

$SUBV1 \circ (\text{SUJ}[\text{MOD}]) = SUBV1 \circ MOD$

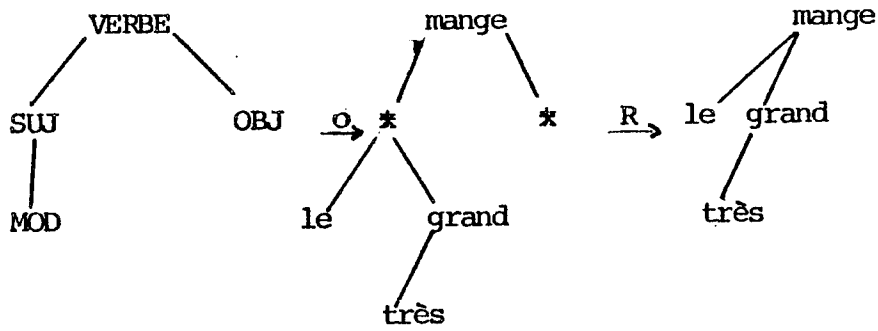
$= \text{le (grand très)}$

$SUBV1 \circ \text{OBJ} = \omega$

$SUVB1 \circ M = (\text{mange } * *) \Gamma <\text{le(grand très)}>.<\omega>$

$= (\text{mange [le(grand très)][}\omega\text{]})$

$R(SUBV1 \circ M) = (\text{mange le (grand très)})$



2.2.3 - Propriété de o

2.2.3.1 - Propriété 1

La plus haute ramification d'une ramification rigide reste inchangée par application d'une substitution (si o est définie pour cette substitution).

On appelle plus haute ramification d'une ramification modèle la ramification obtenue par concaténation de ses plus hautes branches des arborescences constituant la ramification initiale.
Elle est donnée par la fonction phram : $A \rightarrow S$

$$\text{phram}(r+s\times a) = \text{phram}(r)+\text{phb}(s\times a) \quad \text{phram}(\Lambda) = \Lambda$$

Soit la ramification rigide u = r+s×a
Si, pour une substitution σ , σ o u est définie, on a

$$\sigma \text{ o } u = \sigma \text{ o } r + \text{phb}(s\times a) \ \Gamma \ \lambda(\sigma, \text{bb}(s\times a))$$

d'où $$\text{phram}(\sigma \text{ o } u) = \text{phram}(\sigma \text{ o } r) + \text{phb}(\text{phb}(s\times a) \ \Gamma \ (\lambda(\sigma, \text{bb}(s\times a))))$$

$$= \text{phram}(\sigma \text{ o } r) + \text{phb}(s\times a)$$

La propriété est donc vérifiée.

Définissant la fonction *lbb* qui donne la liste des basses branches d'une ramification modèle

$$lbb(r+s\times a) = lbb(r).\text{bb}(s\times a) \quad lbb(\Lambda) = \Omega$$

On conclut que $\forall M \in A$, $\forall Q \in Q$, $\forall \in \Phi$

$$\sigma \text{ o } M = Q \Rightarrow \exists D \in D \text{ tel que } D \equiv Q$$

$$\text{et} \begin{cases} \text{longueur}(M) = \text{longueur}(D) \\ \text{phram}(M) = \text{phram}(Q) \\ \lambda(\sigma, lbb\ (M)) = lbb(Q) \end{cases}$$

2.2.3.2 - Propriété 2

Soient une substitution $\sigma \in \Phi$, une ramification modèle flexible $M = s \times a$ telles que

$$\rho(\sigma,a) \neq \omega \qquad \text{et} \qquad \sigma \circ M = Q \qquad Q \in \mathcal{Q}$$

alors le nombre de points de rattachements de la plus haute branche de M reste constant par application de σ .
En effet, on a alors

$$\sigma \circ M = \rho(\sigma,a) \; \Gamma \; \lambda(\sigma, bb(s \times a))$$

$$\Rightarrow \; nbpa(\rho(\sigma,a)) = taille\,(\lambda(\sigma,bb(s \times a)) = taille(bb(s \times a))$$

$$= nbpa(phb(s \times a))$$

2.3 - ALGORITHME D'UNIFICATION ADMETTANT LA REGLE VIDE

Le premier algorithme exprimé devient caduc lorsqu'on introduit la règle vide. On a vu qu'il était inutile d'examiner exhaustivement les décompositions possibles d'un arbre déterminé. Aussi cherche-t-on maintenant à formuler un nouvel algorithme admettant la règle vide et opérant directement sur une ramification déterminée.

Le problème est donc d'éliminer les décompositions de la ramification déterminées ne conduisant certainement pas à un résultat.

La réponse est indiquée par deux éléments :

- la définition et les propriétés de o, qui entraînent des contraintes sur ces décompositions.

- le modèle lui-même qui sert de guide pour l'élaboration des décompositions.

Deux cas sont possibles pour un tel couple de ramification (modèle-déterminé)

1°) La ramification modèle est rigide.

On sait alors que la plus haute ramification reste inchangée (propriété 1 de o) par application d'une substitution. Les seules décompositions de la ramification déterminée susceptibles de conduire à un résultat sont donc celles possédant une plus haute ramification identique. Et dans ce cas, on peut "simplifier" cette ramification pour étudier successivement chacune des basses branches.

2°) La ramification modèle est flexible.

On doit alors construire une règle de substitution. Si cette règle n'est pas la règle vide les décompositions intéressantes sont celles qui sont cohérentes avec la propriété 2 de o.

On est donc amené à formuler deux algorithmes, l'un pour la réduction des ramifications rigides éliminant les termes communs dans les différentes décompositions envisagées et appelé algorithme d'élagage. L'autre, pour construire les règles de substitution associées à chaque décomposition.

La construction des différentes substitutions se fait alors comme suit. Partant d'un couple donné on applique l'algorithme d'élagage qui amène à un ensemble de couples correspondant aux différentes décompositions.

Pour l'un de ces couples, flexible, on construit les différentes règles et l'on applique immédiatement ces règles. Cette application produit de nouveaux couples, dont le premier élément est rigide et on est ramené au cas précédent.

La production des substitutions consiste donc à appliquer alternativement ces deux algorithmes en exprimant à chaque fois les décompositions et les choix de règles possibles par autant d'arcs dans un graphe des solutions, que l'on construit dynamiquement. On constitue ainsi un espace de recherche [Nils] qu'il ne reste plus qu'à parcourir pour trouver les chemins menant à une solution, chaque arc donnant un couple de la substitution.

Le procédé est similaire à celui utilisé dans un algorithme d'unification en calcul de prédicats, lorsqu'on construit un "matching tree" [Huet 1] pour la construction des diverses substitutions possibles.

Les différentes parties du procédé sont maintenant examinées successivement dans le détail, puis l'algorithme est formulé totalement.

On constante sur un exemple que la règle vide provoque une augmentation de la combinatoire de cet algorithme, pour obtenir des résultats inintéressants. On introduit pour y remédier la notion de conditionnelle, associée à un arbre modèle, et imposant des contraintes supplémentaires sur les possibilités de choix des règles. Un exemple complet est montré, dans le cadre d'un système de traduction de programmes.

2.3.1 - Elagage

2.3.1.1 - Définition

Le but de cette phase est, partant d'un ensemble de couples de ramification de se ramener à un ensemble de couples tel que pour l'un d'eux au moins il est nécessaire de construire une règle de la substitution recherchée, le premier élément de ce couple n'étant plus "réductible". La seconde ramification de ce couple est alors équivalente à celle qui doit être obtenue par application à la première de la substitution recherchée. On doit donc construire à partir d'une liste de couples une nouvelle liste dont un élément au moins est un doublet constitué d'une sous ramification flexible et d'une sous ramification rigide.

A cet effet, on choisit un couple dans la liste initiale. Si ce couple est déjà flexible-déterminé le but est atteint et le traitement s'arrête.
Si ce couple est rigide-déterminé, on le réduit en restant cohérent avec la définition de l'opération o.
Il se peut que ceci soit impossible et l'algorithme donne alors le résultat fail.
Il se peut aussi que plusieurs solutions soient envisageables, en fonction de la décomposition des éléments déterminés en éléments quasi déterminés équivalents, On produit alors autant de nouvelles listes que de possibilités, exprimant là

les différentes substitutions possibles à ce niveau.
Enfin, si les deux éléments du doublet sont identiques on supprime ce doublet et on itère l'algorithme sur un nouveau couple.

Le procédé d'élagage est mis en oeuvre par la fonction élag qui admet donc en entrée une liste de couples de ramification et qui produit une ou plusieurs listes s'il y a une ou plusieurs solutions, ou la liste vide, ou la valeur fail.

Grossièrement, l'algorithme d'élagage s'écrit :

```
fonction élag (listedecouples)
        listeresultat < Ω
        boucle <fail sortie>
          co on boucle sur chacun des couples jusqu'à ce que l'on ait un couple
             flexible-déterminé, ou la liste vide
          co
          si listedecouples  = Ω alors sortie
          sinon
            co on examine le premier couple de la liste <RM RD> ou RM est une
               ramification modèle et RD une ramification déterminée
            co
            <RM RD>:= tête(listedecouple)
            si flexible(RM) alors
                  listeresultat := <listedecouples>
                  sortie
            ssi RF' = RD alors
                  listedecouples := reste (listedecouples)
            sinon
                  a) produire la liste des décompositions possibles pour la plus
                     haute ramification de RM.
                  b) pour chacune de ces décompositions exécuter l'algorithme
                     d'élagage ce qui produit à chaque fois une nouvelle liste
                  c) pour chacune de ces listes remplacer le couple initial
                    de listedecouple par cette liste
            fsi
          fsi
        répéter
          sortie :   listerésultat
          fail   :   fail
        finboucle
```

2.3.1.2. - Algorithme

Cet algorithme nécessite la création des décompositions possibles de RD ayant la même plus haute ramification que RM(a).

On utilise à cet effet une fonction donnant pour un arbre modèle et un arbre déterminé la liste des basses branches possibles celles que la plus haute branche des décompositions soit identiques. Cette fonction est la fonction mêmephb.

L'obtention de la liste des basses branches se réalise par une boucle sur chacune des arborescences de la ramification et un produit cartésien avec le résultat obtenu précédemment à chaque itération.

Le produit cartésien est donné par la fonction cross.

```
fonction cross (liste1 , liste2)
        résultat := Ω
        si liste1 = Ω ou liste2 = Ω alors
           résultat := liste1.liste2
        sinon
           boucle <exit>
               liste3 := liste2
               si liste1 = Ω alors exit
               sinon
                   boucle <sortie>
                      si liste3 = Ω alors sortie
                      sinon
                         résultat := (tête(liste1).tete(liste3)) || résultat
                      fsi
                      liste3 := reste(liste3)
                   répéter
                      sortie : liste1 := reste (liste1)
                   finboucle
               fsi
           répéter
               exit : → résultat
        fsi
finfonction
```

```
fonction mêmephb (squelette,déterminé)
   si squelette = Λ alors
         si déterminé = Λ alors → <Ω>
         sinon → fail
         fsi
   ssi prac(squelette) ≠ * alors
         si prac (squelette) = prac (déterminé) alors
            rp1 := mêmephb (sousram(squelette),sousram(déterminé))
            rp2 := mêmephb(suite(squelette),suite(déterminé))
            si rp1 ≠ fail et rp2 ≠ fail alors
                  → cross(rp1,rp2)
            sinon
                  → fail
            fsi
   sinon
         résultat := Ω
         K := 0
         si déterminé = Λ alors résultat := <ω>  fsi
         boucle <sortie>
            si K = longueur(déterminé) alors sortie
            sinon
                  rp := mêmephb(suite(squelette),nfirst(K,déterminé))
                  si rp ≠ fail alors
                      u := nlast(K,déterminé)
                      si u = Λ alors u := ω fsi
                      résultat := cross(<u>,rp)||résultat
                  fsi
            fsi
            K := K+1
         répéter
            sortie : si résultat = Ω alors → fail
                     sinon → résultat
         finboucle
   fsi
finfonction
```

nfirst et nlast sont les fonctions ainsi définies

```
fonction nfirst(K,ram)
   si K = 0 alors → ram
   sinon→nfirst(K-1,suite(ram))
   fsi
finfonction

fonction nlast(K,ram)
   si K = 0 alors → Λ
   sinon → nlast(K-1,ram)+dernier(ram)
   fsi
finfonction
```

D'ou la fonction même^phram

```
fonction même^phram(rigide,déterminé)
   listeresult := Ω
   boucle <exit échec>
         si rigide = Λ et déterminé = Λ alors
                        éxit
         ssi rigide = Λ ou déterminé = Λ alors
            co longueur (rigide) ≠ longueur(déterminé) co
               échec
         sinon
            L := même^phb(phb(dernier(rigide)),déterminé)
            si L = fail alors échec
            sinon
               listeresult := cross(L,listeresult)
            fsi
         fsi
            rigide := suite(rigide)
            déterminé := suite(déterminé)
   répeter
         exit : → listeresult
         échec: → fail
   finboucle
finfonction
```

Ceci permet d'exprimer la fonction élag

```
fonction élag(listedecouples)
  listerésultat :=  < Ω >
  boucle <échec sortie>
        si listedecouple = Λ alors sortie fsi
           <RM RD>:= tête(listedecouples)
        ssi flexible(RM) alors
            listerésultat := <listedecouples>
            sortie
        ssi RM = RD alors
            listedecouples := reste(listedecouples)
        sinon
            ldécomp := mêmephram(RM,RD)
            si ldécomp = fail alors échec
            sinon
                 L2 := Ω
               boucle <exit>
                 si ldécomp = Ω alors exit
                 sinon
                      L := élag(tête(ldécomp))
                     si L ≠ fail alors

                      boucle <fin>
                        si L = Ω alors fin
                        sinon
                           L2 := (tête(L).reste(listedecouples))‖L2
                        fsi
                        L = reste(L)
                      répéter
                        fin :
                      finboucle
                     fsi
                 fsi
                 ldécomp := reste(ldécomp)
               répéter
                 exit : si L2 = Ω alors échec
                        sinon
                              listederésultat := L2
                        fsi
               finboucle
            fsi
        fsi
   répéter
        sortie : → listerésultat
        échec  : → fail
   finboucle
finfonction
```

2.3.1.3. - Exemples

élag(< <a b c ,a b c> >)= < Ω >

élag(< <(a b [X]) , (a b (c d))> >) = < < <X , (c d)> > >

élag(< <(X [b] [Z]) , a b c> >) = < <<(X [b] [Z]) , a b c> > >

élag(< <(b [X] [Y]) (c [U] e [V]) , (b c d) (c d e e f)> >)

= <
< <X , ω> <Y , c d> <U , d> <V , e f> >

< <X , ω> <Y , c d> <U , d e> <V , f> >

< <X , c> <Y , d> <U , d> <V , e f> >

< <X , c> <Y , d> <U , d e> <V , f> >

< <X , c d> <Y , ω> <U , d> <V , e f> >

< <X , c d> <Y , ω> <U , d e> <V , f> >

>

2.3.1.4 - Etude de l'algorithme

Etant donné une substitution σ, soit P la propriété d'un ensemble E de couples de ramification.

P(E) : pour tout couple <RM,RD> ∈ E, E $\subseteq A \times D$
il existe RQ tel que σ o RM = RQ et RQ ≡ RD.

Soit LE la liste représentant l'ensemble de couple à l'entrée de la procédure d'élagage et LR la liste de listes construite par cette procédure.

2.3.1.4.1 - Echec

Si le résultat de la fonction ELAG est fail, il n'existe aucune substitution vérifiant P pour LE.

En effet, le résultat de la fonction ELAG ne peut être fail que si RM est rigide et qu'il n'existe pas de ramification quasi déterminée équivalente à RD ayant même plus haute ramification que RM. Il ne peut alors exister de substitution vérifiant P, en vertu de la propriété I de o.

2.3.1.4.2 - Succès

La procédure d'élagage conserve P.

Montrons que si P(LE) alors P est vraie pour chacune des listes de LR.

Soit <RM RD> le premier couple de LE.

i) si RM est flexible

LR = < LE >

ii) si RM est rigide

. si RM = RD on supprime le couple de LE ce qui ne modifie pas P(LE)

. si RM ≠ RD

On construit les décompositions de RD en tous les RQ vérifiant

phram(RM) = phram(RQ)

RQ ≡ RD

et on doit avoir, si σ o RM = RQ

$\lambda(\sigma, lbb(RM)) = lbb(RQ)$

Comme on ne fait que remplacer dans LE le couple <RM RD> considéré par la liste de couples obtenue par élagage de ces basses branches la propriété est conservée.

En résumé l'algorithme d'élagage produit à partir d'un ensemble de couples, la valeur fail si P ne peut pas être vérifiée pour cet ensemble et, dans le cas contraire un nouvel ensemble conservant P et tel que pour l'un au moins de ces couples il est nécessaire d'établir une règle de substitution.

2.3.2 - Critères de construction de règles

On examine maintenant le processus de construction d'une règle de substitution à partir d'un couple dont le premier élément est flexible. Il faut pour la variable concernée construire les règles possibles en fonction des différentes décompositions possibles conservant P.

Soit <RM,RD> le couple considéré avec

$RM = y \times t$ et $t \in V$

$RD = r+s\times a$

Si P est vraie, on a σ o $RM = rq + sq \times aq \equiv r+s\times a$

1) si $y = \Lambda$ alors on doit avoir

σ o $(y\times t) = \rho(\sigma,t) = RQ$

Quelque soit alors RQ (RQ ≡ RD) choisi cela ne changera rien au résultat final. On choisit donc le RQ le plus simple, à savoir RD, et on construit la règle <t RD>

2) si $y \neq \Lambda$ alors

i) si $y \neq *$

σ o $(y\times t) = \rho(\sigma,t)\ \Gamma\ \lambda(\sigma,bb(y\times t))$

qui est de la forme $sq \times aq$ ce qui implique :

. $r = \Lambda$

. $nbpa(sq \times aq) = taille\ (\lambda(\sigma, bb(y \times t)))$
$= taille\ (bb(y \times t))$
$= nbpa(y \times t)$

. $\rho(\sigma, t) = phb(sq \times aq)$.

Les seules règles possibles à ce niveau sont celles qui correspondent à une décomposition de RD = s×a possédant K = nbpa(y×t) points de rattachements et on construit autant de règles que de telles décompositions,

<t phb(sq×aq)>

ii) si y = *
En plus des règles correspondant au cas précédent, on peut construire une règle vide.

3°) Si il existe déjà une règle de substitution pour la variable concernée, on vérifie qu'elle est cohérente.

2.4 - ALGORITHME D'UNIFICATION ADMETTANT LA REGLE VIDE

2.4.1 - Définition

On cherche à construire les substitutions σ pour un arbre modèle RM et un arbre déterminé RD telles que {<RM,RD>} vérifie P.

Initialement on constitue la liste contenant le seul couple <RM RD> et supposant P vraie, on applique l'algorithme d'élagage.

i) Si le résultat est une liste de listes, pour chacune de ces listes on sait que P a été conservée et que pour un des couples il faut construire une règle en utilisant les critères de construction. Pour chaque choix possible de règle, on construit dans le graphe des solutions (qui est en fait un arbre), un arc partant du sommet courant et étiqueté par la règle choisie. Puis on applique cette règle au couple considéré. Ceci produit une nouvelle liste de couples qui constitue un nouveau sommet. On obtient ainsi autant de nouvelles listes que de solutions choisies, toutes préservant P. On itère donc le processus sur ces listes.
Si pour aucune des listes résultat de l'élagage, il n'est possible de construire une règle, on construit un arc vers un sommet 'fail'.

ii) Si le résultat est la liste vide, ayant préservé P depuis l'origine et n'ayant plus de règles à déterminer, la succession d'arcs construits jusqu'à ce noeud déterminé une substitution. On construit le sommet 'goal'

iii) Le résultat est fail. Il ne peut exister de substitution unifiant ce couple. On construit le sommet 'fail'.

A l'issue de cet algorithme tous les chemins menant à une feuille 'goal' déterminent une substitution possible.

L'algorithme d'unification utilise les fonctions auxiliaires.

applique (règle,listedecouples) qui applique pour la règle concernée
l'opération o sur le premier couple de la liste.

construirearc(règle,sommet) qui construit un arc étiqueté par la règle concernée
vers un sommet.

construirenoeud(listedecouples) qui construit un sommet de l'arbre avec son
paramètre.

construirefils(sommet,listearcs) qui attribue sommet comme origine de tous les
arcs de listearcs.

L'algorithme s'écrit alors

```
fonction unification (modèle,déterminé)
        racine := elag (< modèle déterminé> )
        si racine = fail alors → construirenoeud(fail)
        sinon → unif (construirenoeud(racine))
        fsi
finfonction
```

```
fonction unif(noeud)
        listerègle := choixrègle(noeud)
        si listerègle = fail alors → construirenoeud(fail)
        sinon
             listefils := Ω
             boucle <fin>
                  si listerègle = Ω alors fin
                  sinon
                       nouvnoeuds := élag(appelique(tête(listerègle)noeud))
                       si noeuvnoeuds = fail alors
                            listerègle := reste(listerègles)
                       ssi nouvnoeuds = <Ω> alors
                            fils := construinoeud (goal)
                            listefils := construirearc(tête(listerègle),fils)
                                         | listefils
                            listerègle := reste(listerègle)
                       sinon
                         boucle  <sortie>
                            si  nouvnoeuds = Ω alors sortie
                            sinon
                                fils := unif(construirenoeud(tête(nouvnoeuds)))
                                listefils := construirearc(tête(listerègle),fils)
                                             | listefils
                                nouvnoeuds := reste(nouvnoeuds)
                            fsi
                         répéter
                            sortie : listerègle := reste(listerègle
                         finboucle
                       fsi
                  fsi
             répéter
                  fin : si listefils = Ω alors → construirenoeud(fail)
                        sinon
                            → construirefils(noeud,listefils)
                        fsi
             finboucle
        fsi
finfonction
```

2.4.2 - Exemples

2.4.2.1 - Exemple 1

Unification de M = (MES[(entre (et [POS1][POS2]))])

et D = (angle (entre (et (tireur le) (cible la)))).

Tout d'abord est formé le couple <M D> dont le premier élément est flexible et qui reste inchangé par élagage.
Pour la variable MES, on a alors les choix possibles

1 <MES ω>
2 <MES (angle *)>
3 <MES (angle (entre *))>
4 <MES (angle (entre (et *)))>
5 <MES (angle (entre (et (tireur le) *)))>
6 <MES (angle (entre (et (tireur le) (cible *))))>
7 <MES (angle (entre (et (tireur *) (cible la))))>
8 <MES (angle (entre (et (* (cible la))))>.

. L'application de la règle 1 donne le couple

<(entre(et [POS1][POS2])) (angle(entre(et(tireur le) (cible la))))>

et l'algorithme d'élagage fail

. L'application de la règle 3 donne le couple

<(angle(entre [(entre(et [POS1][POS2]))]),
(angle(entre(et(tireur le) (cible la)>

qui produit après un premier élagage

<(entre(et [POS1][POS2])), (et(tireur le) (cible la))>

puis fail au second appel de élag

. Il en va de même pour les règles 4 et 8.

. La seule règle acceptable est la règle 2 qui donne

<(angle [(entre (et [POS1][POS2]))]),
(angle (entre (et (tireur le) (cible la))))>

soit après élagage

< < <POS1 , ω> <POS2 , (tireur le) cible la)> >
< <POS1 , (tireur le)> <POS2 , (cible la)> >
< <POS1 , (tireur le) (cible la)> <POS2 , ω> >
>

Il n'y a plus alors de nouveaux choix possibles et on obtient finalement trois substitutions

MES → (angle *)
POS1 → ω
MOS2 → (tireur le) (cible la)

MES → (angle *)
POS1 → (tireur le)
POS2 → (cible la)

MES → (angle *)
POS1 → (tireur le) (cible la)
POS2 → ω

2.4.2.2 - Exemple 2

Obtention des substitutions pour

F = (VERBE [(SUJ[MOD])][OBJ])
D = (saute (cheval le)).

La encore on a le premier couple est flexible et on a le choix pour la règle VERBE entre

1 <VERBE (saute * *)>
2 <VERBE (saute (cheval *) *)>
3 <VERBE (saute (cheval * *))>
4 <VERBE (saute * (cheval *))>
5 <VERBE (saute * * (cheval le))>
6 <VERBE (saute * (cheval le) *)>
7 <VERBE (saute (cheval le) * *)>

Examinons successivement chaque règle

1 Elle même à la construction de

<(saute [(SUJ[MOD])][OBJ]), (saute (cheval le))>

qui après élagage produit

< < <(SUJ [MOD]) , ω> <OBJ , (cheval le)> >
< <(SUJ [MOD]) , (cheval le)> <OBJ , ω> >
>

1.1
Pour la première de ces listes on ne peut construire que
SUJ → ω
ce qui amène
< < <MOD , ω> <OBJ , (cheval le)> > >
et donc successivement
<MOD ω> et <OBJ (cheval le)>

1.2

On a le choix entre

<SUJ ω>

<SUJ (cheval *)>

<SUJ (cheval le *)>

qui impliquent

1.2.1 <MOD (cheval le)><OBJ ω>

1.2.2 <MOD le><OBJ ω>

1.2.3 <MOD ω><OBJ ω>

Les différentes substitutions sont données par l'arbre de la figure 1.

2 - Cette règle amène à l'élagage de

< <(saute (cheval [(SUJ[MOD])]) [OBJ])) , (saute(cheval le))> >

soit à < < <(SUJ[MOD]) , le> <OBJ , ω> > >

D'où la seule possibilité

<SUJ ω><MOD le><OBJ ω>

3 On est amené à la même solution que pour 2

on a <MOD ω><OBJ le>

4 L'application de la règle donne

<(saute [(SUJ[MOD]) (cheval [OBJ])]), (saute (cheval le))>

soit après élagage

< < <(SUJ[MOD]) , ω> <OBJ , le> > >

d'où <SUJ ω><MOD ω><OBJ le>

5 Les solutions 5,6,7 pour la règle verbe mènent à

<SUJ ω><MOD ω><OBJ ω>

On a donc au total 10 solutions exprimées par l'arbre des figures 1,2,3.

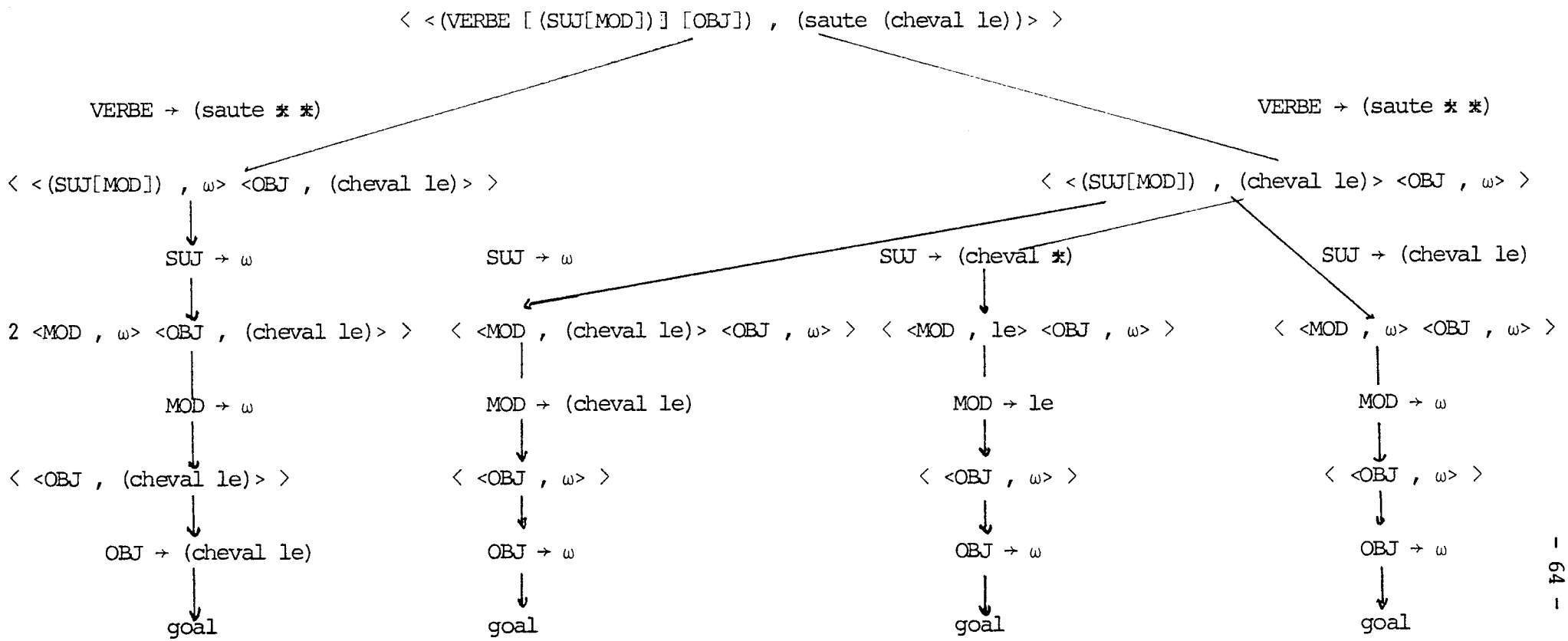


Figure 1 : substitutions obtenues avec la règle 1.

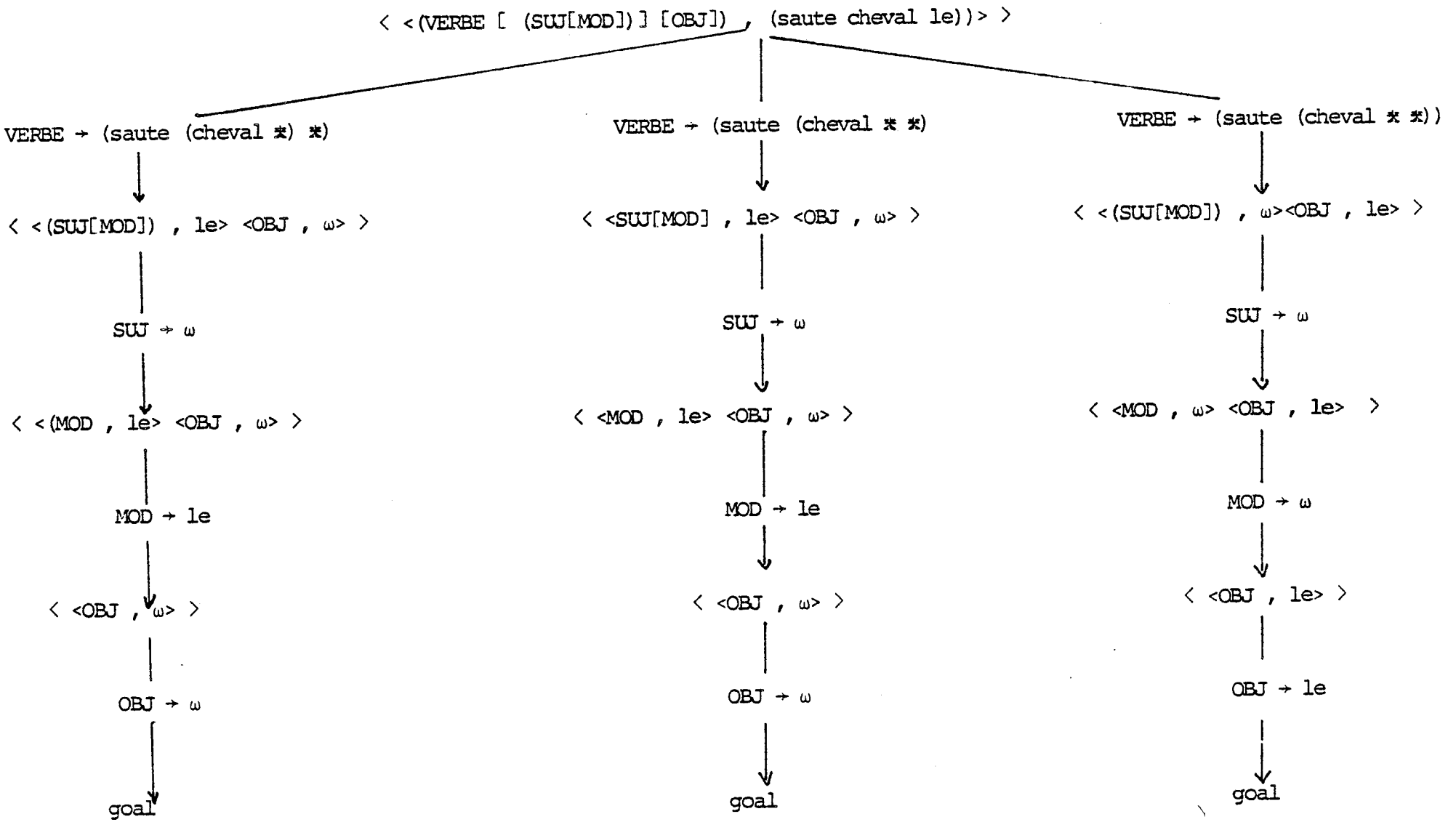


Figure 2 : substitutions obtenues avec les règles 2 et 3.

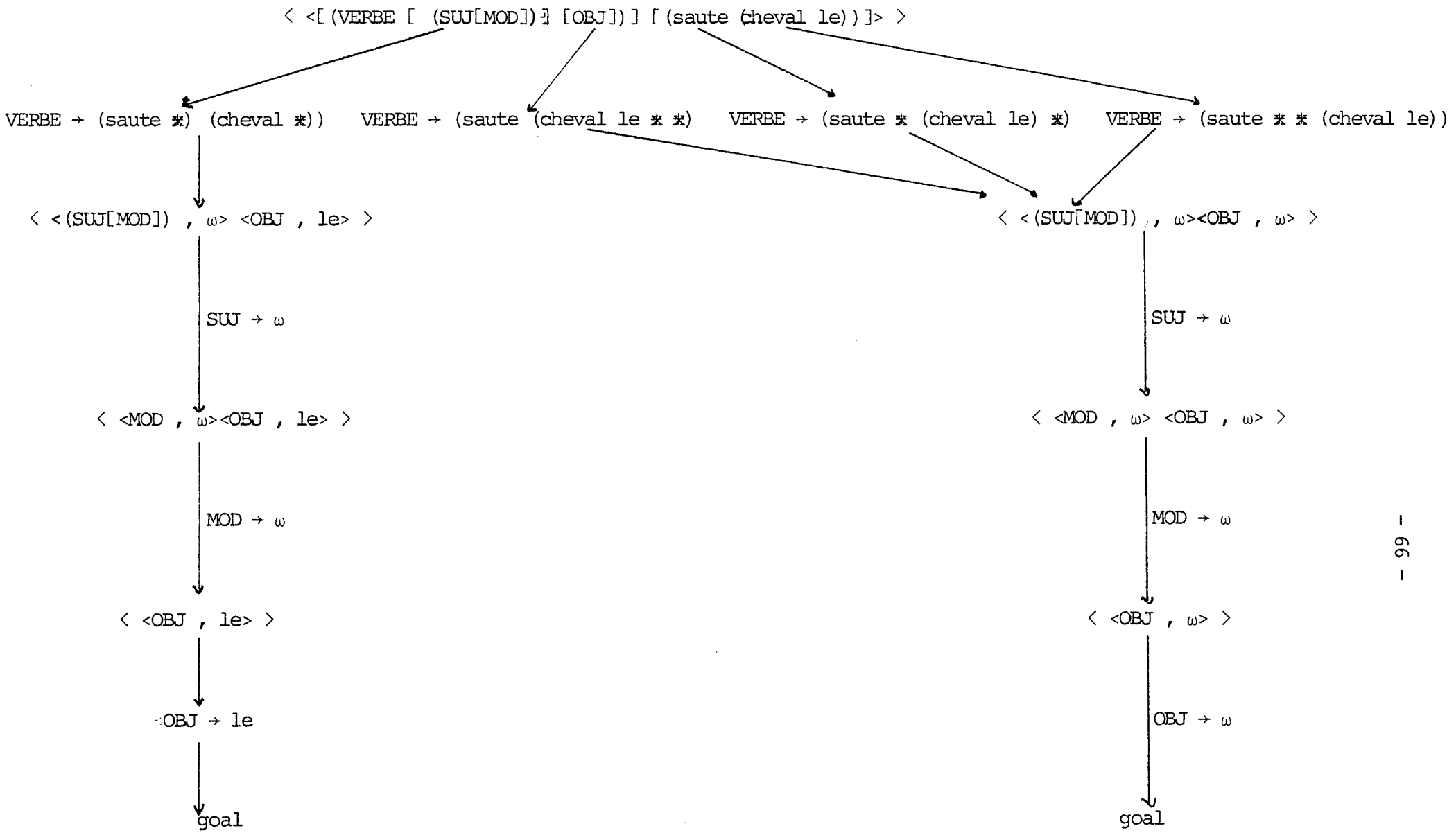


Figure 3 : Substitutions obtenues avec les règles 4, 5, 6, 7.

2.5 - CONDITIONNELLE ASSOCIEE

On a pu constater sur l'exemple, pourtant simple, que l'introduction de la règle vide provoque une augmentation non négligeable de la combinatoire et donc du coût de l'algorithme d'unification, ceci pour obtenir des solutions inintéressantes, au vu de l'application concernée.

On est amené à restreindre les choix de règles possibles aux seuls cas intéressants, ce qui est l'objet de la conditionnelle associée.

La conditionnelle associée est simplement un ensemble de prédicats associé à un arbre modèle, chaque prédicat étant relatif à une variable de cet arbre. Un prédicat lié à une variable V porte sur la ramification associée à V au sein de la substitution. Il est noté V(X) où X représente la partie droite de la règle de substitution <V X>. Il est introduit comme condition supplémentaire au niveau des critères de choix, et on ne construit donc pour une variable donnée que les règles satisfaisant le prédicat relatif à cette variable.

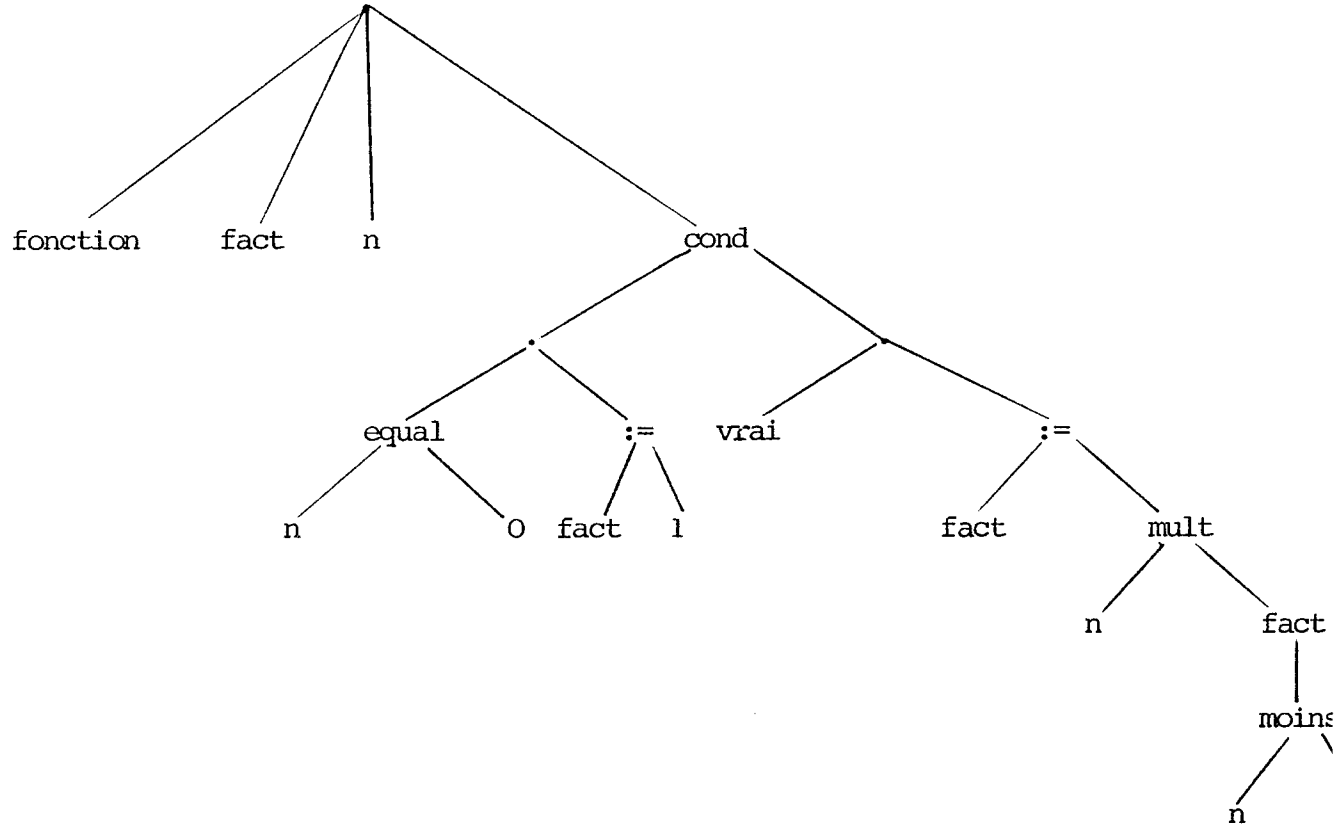
2.6 - EXEMPLE : Traduction de programmes

On donne ici un exemple extrait d'un système de traduction de schéma de programmes en FORTRAN[JOLO] . L'analyse syntaxique d'un schéma produit une ramification déterminée, l'ensemble C des constantes étant alors l'ensemble des mots du langage des schémas.

Le schéma

```
fonction fact(n)
   si n=0 alors fact := 1
   sinon fact := n × fact(n-1)
   fsi
finfonction
```

se présente après analyse sous forme de la ramification



Cette ramification n'étant pas traductible dans le langage cible on doit effectuer une transformation de la ramification. A cet effet on dispose d'une bibliothèque de schémas dont chaque élément est un doublet <MR MI> et où MR est une ramification correspondant à un certain type de schéma récursif d'où MI est la ramification correspondant au type de schéma itératif équivalent. Le principe est alors le suivant : on recherche pour le schéma à traduire quel est le modèle auquel il répond. Ceci revient à construire la substitution unifiant le modèle et le schéma, si elle existe. Cette substitution permet alors de construire la ramification correspondant au schéma itératif équivalent par simple application au modèle MI.

Examinons le fonctionnement sur l'exemple donné. On recherche donc d'abord les substitutions entre MR (figure 1) et le schéma donné

MR :

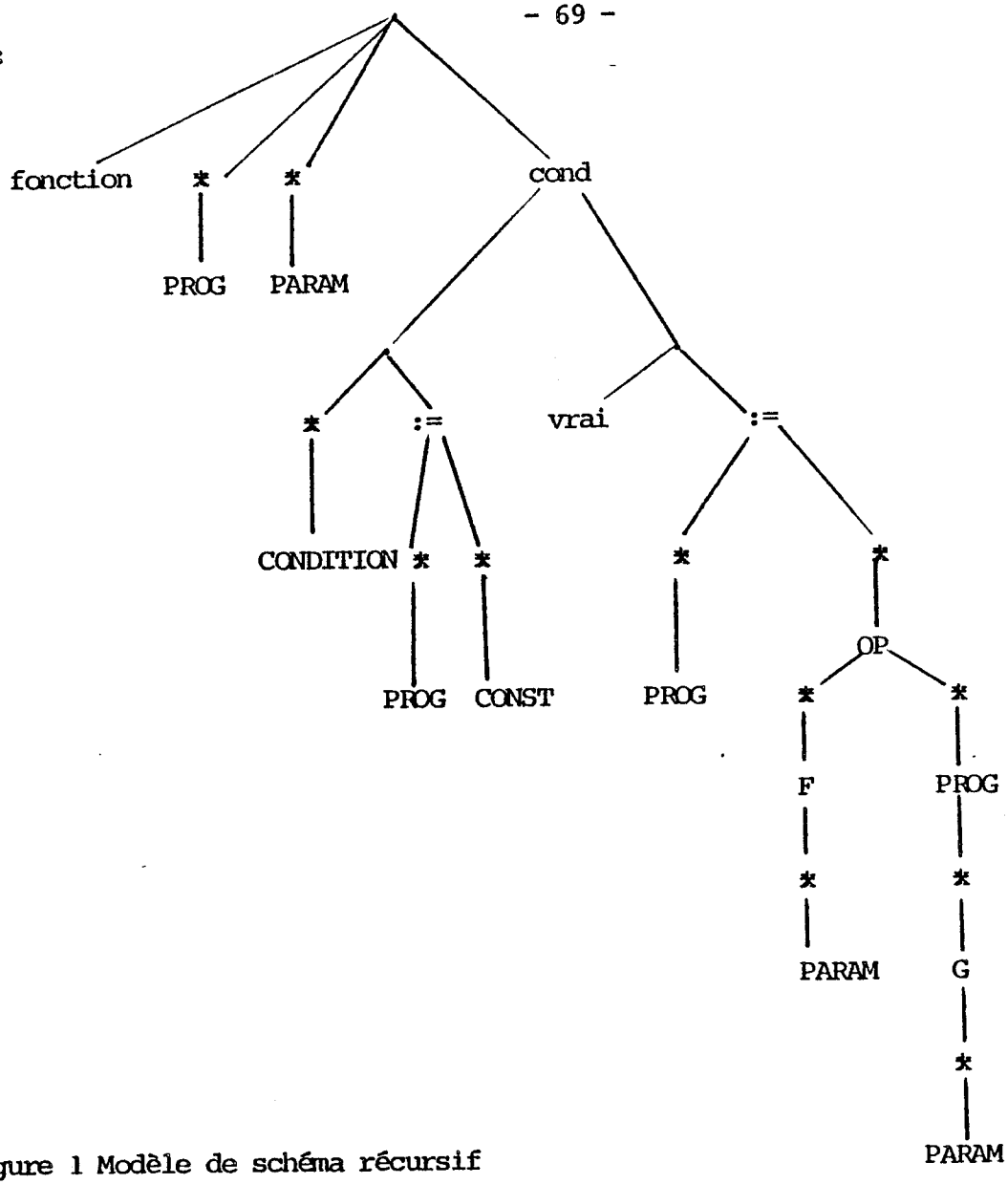


Figure 1 Modèle de schéma récursif

avec la conditionnelle associée

prédicat PROG(X)
identificateur (x)
finprédicat
prédicat OP(X)
étoiles (sousram(X)) et associatif (prac(X))
finprédicat

Le prédicat associé à OP impose que la ramification se substituant OP soit de la forme (I * *) où I est identificateur représentant une application associative [Da-Bu].

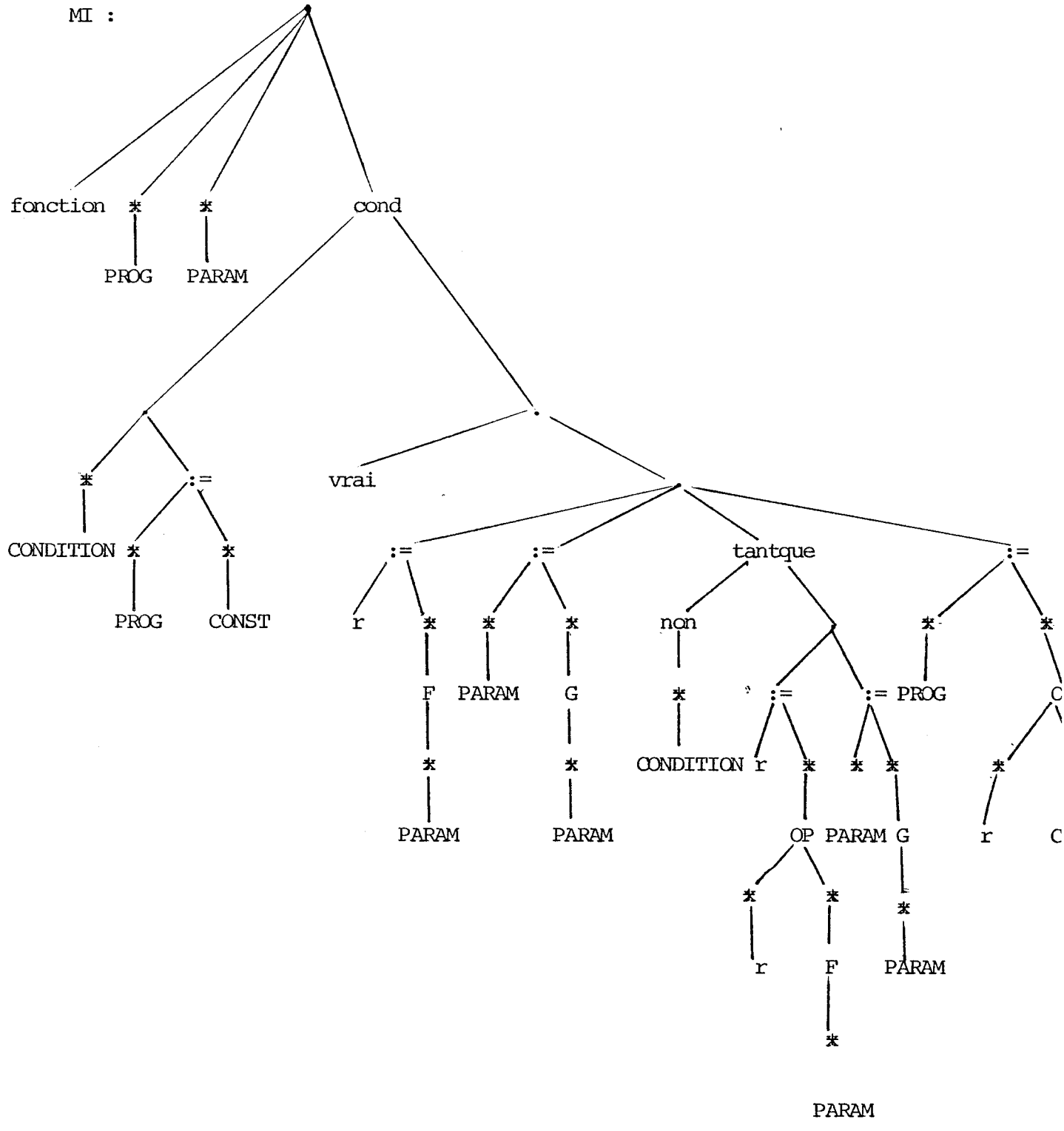


Figure 2 : Modèle de schéma itératif équivalent

Ecrivant les sous-ensembles de couples

C_1 = {<PROG , fact n><PARAM , ω>}

C_2 = {<PROG , fact><PARAM , n>}

C_3 = {<PROG , ω><PARAM , fact n>}

C_4 = {<CONDITION (equal n , 0)>}

C_5 = {<PROG , fact 1><CONST , ω>}

C_6 = {<PROG , fact> <CONST , 1>}

C_7 = {<PROG ,ω> <CONST , fact 1>}

C_8 = {<PROG , fact (mult n (fact (moins n 1)))>
<(OP [(F[PARAM])] [(PROG [(G[PARAM])]) , ω>}

C_9 = {<PROG , fact>
<(OP [(F[PARAM])] [(PROG [(G[PARAM])])]) ,
(mult n (fact moins n 1)))>}

C_{10} = {<PROG , ω>
<(OP [(F[PARAM])] [(PROG [(G[PARAM])])]) ,
fact (mult n (fact (moins n 1)))> } .

On obtient après élagage les ensembles de couple

(1) $C_1 \cup C_4 \cup C_5 \cup C_8$	(10) $C_2 \cup C_4 \cup C_5 \cup C_8$	(19) $C_3 \cup C_4 \cup C_5 \cup C_8$
(2) $C_1 \cup C_4 \cup C_5 \cup C_9$	(11) $C_2 \cup C_4 \cup C_5 \cup C_9$	(20) $C_3 \cup C_4 \cup C_5 \cup C_9$
(3) $C_1 \cup C_4 \cup C_5 \cup C_{10}$	(12) $C_2 \cup C_4 \cup C_5 \cup C_{10}$	(21) $C_3 \cup C_4 \cup C_5 \cup C_9$
(4) $C_1 \cup C_4 \cup C_6 \cup C_8$	(13) $C_2 \cup C_4 \cup C_6 \cup C_8$	(22) $C_3 \cup C_4 \cup C_6 \cup C_8$
(5) $C_1 \cup C_4 \cup C_6 \cup C_9$	(14) $C_2 \cup C_4 \cup C_6 \cup C_9$	(23) $C_3 \cup C_4 \cup C_6 \cup C_9$
(6) $C_1 \cup C_4 \cup C_6 \cup C_{10}$	(15) $C_2 \cup C_4 \cup C_6 \cup C_{10}$	(24) $C_3 \cup C_4 \cup C_6 \cup C_{10}$
(7) $C_1 \cup C_4 \cup C_7 \cup C_8$	(16) $C_2 \cup C_4 \cup C_7 \cup C_8$	(25) $C_3 \cup C_4 \cup C_7 \cup C_8$
(8) $C_1 \cup C_4 \cup C_7 \cup C_9$	(17) $C_2 \cup C_4 \cup C_7 \cup C_9$	(26) $C_3 \cup C_4 \cup C_2 \cup C_9$
(9) $C_1 \cup C_4 \cup C_7 \cup C_{10}$	(18) $C_2 \cup C_4 \cup C_7 \cup C_{10}$	(27) $C_3 \cup C_4 \cup C_2 \cup C_{10}$

On détermine alors les règles possibles pour la variable PROG.
La conditionnelle associée qui impose que PROG soit associée à un identificateur amène alors l'élimination de toutes les solutions (1) à (9) et (19) à (27).

On ne construit donc, à partir du couple initial, que des arcs étiquetés par la règle <PROG fact>.
Pour des raisons de cohérence, cette règle étant unique, toutes les possibilités contenant C_5 ou C_7 ou C_8 ou C_{10} vont mener à un échec.
La seule solution ne menant pas à un échec est donc la décomposition $C_2 \cup C_4 \cup C_6 \cup C_9$, qui après construction des règles

<CONDITION (equal n 0)>
<PARAM n>
<CONST 1>

amène à la construction d'une règle pour

<(OP[(F[PARAM])][(PROG[(G[PARAM])])])
(mult n (fact (moins n 1)))>

Le prédicat associé à OP fait que la seule règle possible est

<OP (mult * *)>

qui mène après élagage à

< (28) < <(F [PARAM]) , n (fact (moins n 1)) >

<(PROG [(G [PARAM])]) , ω> >

(29) < <(F [PARAM]) , n> <(PROG [(G [PARAM])]) , (fact (moins n 1)) > >

(30) < <(F [PARAM] , ω> <(PROG [(G [PARAM])]) , n (fact (moins n 1)) > >
>

Pour des raisons d'incompatibilité avec les règles PARAM et PROG déjà établies (28) et (30) mènent à l'échec.

A partir de (29) il n'est possible que de construire

<F ω> ou <F n>

La seconde impliquant une incompatibilité (<PARAM n> et <PARAM ω>) seule la première conduit à

<(G[PARAM]), (moins n 1)>

Et la encore seule la règle <G (moins * 1)> conduit au succès.

Il n'existe donc qu'une seule substitution satisfaisant la conditionnelle dans le cas présent, à savoir

σ = <PROG fact>
<PARAM n>
<CONDITION (equal n 0)>
<CONST 1>
<OP (mult * *)>
<F ω>
<G (moins * 1)>

Il ne reste plus qu'à appliquer cette substitution au modèle MI (figure 2) pour obtenir la ramification :

P A R T I E - I I

EVALUATION SEMANTIQUE D'ENONCES

EN LANGUE NATURELLE

CHAPITRE - III

ANALYSE DE QUELQUES MODELES ET TECHNIQUES

Il convient de distinguer dans le traitement des langues les modèles et les systèmes. [Boi]. Les modèles reflètent une théorie, tentent d'exprimer un comportement linguistique et relèvent de la logique. Les systèmes sont les réalisations pratiques visant un traitement linguistique particulier et relèvent de l'algorithmique.

Les premiers n'impliquent pas les seconds et réciproquement. On envisage difficilement cependant l'élaboration d'un système ne reposant pas sur un modèle, ce qui implique un souci de "réalisabilité" pendant la définition du modèle.

Afin de dégager clairement ces deux notions on cherchera à situer les problèmes afférant à l'un ou l'autre dans deux parties distinctes.
On dégagera de cet examen le support d'un nouveau modèle, basé sur une grammaire de cas et une évaluation conceptuelle procédurale. Dans la réalisation ce modèle apparaît comme un paramètre de l'outil réalisé, ce qui conserve la philosophie adoptée dans le système PIAF.

Le système proposé ici opère sur des structures arborescentes et utilise naturellement les techniques décrites précédemment.

3.1 - QUELQUES MODELES LINGUISTIQUES

3.1.1. - Le modèle transformationnel

Les premières expériences de traitement des langues se sont déroulées à partir des années 50, simultanément à l'essort de la théorie transformationnelle, se reflétant essentiellement par les travaux de Harris [Har] et les célèbres "structures syntaxiques" [Ch1] puis "aspects de la théorie de la syntaxe" de Chomsky [ch2], qu'elles ont adopté.

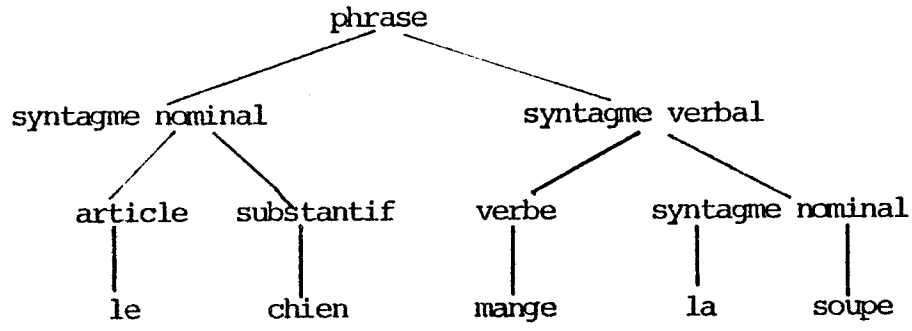
A l'origine les grammaires transformationnelles sont destinées à engendrer les phrases grammaticalement correctes pour le langage concerné. On peut les caractériser par le fait qu'elles assignent à chaque phrase qu'elles produisent à la fois une "structure profonde" et une "structure de surface" en reliant systématiquement ces deux analyses.

Elles comportent donc deux sortes de règles, les règles syntagmatiques qui permettent la construction d'une structure profonde par production à partir d'un axiome, et les règles transformationnelles qui fournissent un moyen de passage de la structure profonde à la structure de surface.

Les règles syntagmatiques

phrase → syntagme nominal + syntagme verbal
syntagme nominal → article + substantif
syntagme verbal → verbe + syntagme nominal
article → le, la
substantif → chien , soupe
verbe → mange

permettent la construction de



Les règles transformationnelles permettent le passage de cette même structure profonde à

la soupe est mangée par le chien.

Les grammaires transformationnelles permettent la distinction entre deux analyses possibles par des moyens purement syntaxiques.
Si l'on considère par exemple la phrase

le boucher sale la tranche

La première interprétation correspond à une structure verbe + syntagme nominal comme dans "le cuisinier sale la côte", la seconde résulte de la transformation de "le boucher sale tranche la côte" par pronominalisation de "la côte".

Bien que les grammaires transformationnelles aient été conçues pour structurer un énoncé et ce, de façon générative, certains [Zw] ont été tentés de les exploiter en mode analytique dans des systèmes informatiques. Comme l'a souligné R. Kaplan [Kap] ils se sont heurtés à des problèmes complexes dans l'élaboration des règles et le contrôle des systèmes, grossissant sans converger. Ceci d'autant plus qu'il est difficile pour une grammaire transformationnelle d'assurer de manière intrinsèque, sans aucune référence à l'univers réels, la génération des phrases du langage visé et <u>seulement</u> de ce langage.

3.1.2 - Introduction de la sémantique

L'insuffisance des techniques purement grammaticales, sans référence externes, a amené Katz et Fodor à introduire une théorie sémantique en 1963 [Ka-Fo]. Tout en restant dans un cadre transformationnaliste, leur théorie a pour but d'interdire l'utilisation de règles de la grammaire si les éléments de cette règle ne présentent pas une certaine cohérence sémantique. A une règle de la grammaire est liée une règle sur son application ("projection rule"). La sémantique est introduite au niveau du dictionnaire et "véhiculée" par ces règles. Le dictionnaire contient autant d'entrées que de différents sens d'un même mot. Pour chaque entrée on trouve :

- la catégorie syntaxique du mot (nom concret, verbe transitif, etc...)
- une suite de marqueurs sémantiques indiquant les concepts auxquels se réfère l'entrée considérée quand elle est utilisée dans le sens correspondant
- un 'distingueur' ("distinguisher") exprimant une caractéristique de cette entrée
- une règle de sélection qui est un prédicat sur sa relation avec d'autres éléments.

Une entrée est de la forme

clé : catégorie → marqueur 1 → ... → marqueur i → [distingueur] → <restriction>

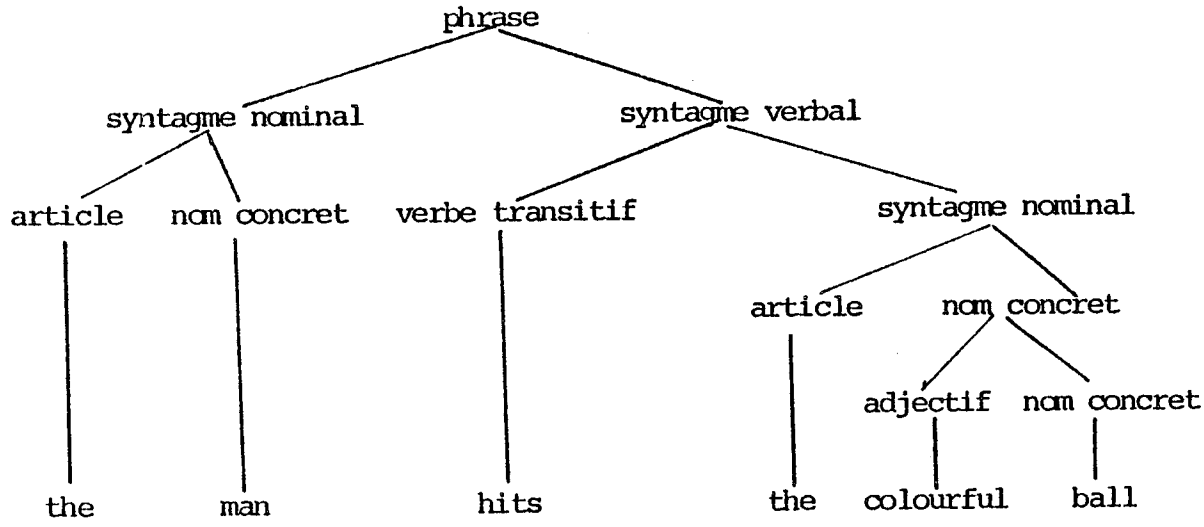
exemple

ball : nom concret → objet physique → [forme arrondie] (≃ balle)
ball : nom concret → activité sociale → assemblée → [danse] (≃ bal)
ball : nom concret → objet physique [lancé par un engin de guerre] (≃ obus)
colourful : adjectif → coloré → [variété de couleurs vives]
→ <objet physique ou activité sociale> (≃ coloré)
colourful : adjectif → evaluatif → [ayant un caractère vif, animé]
→ <objet esthétique ou activité sociale>

Le principe est alors le suivant : étant donné une règle de la grammaire (par exemple nom concret → adjectif + nom concret) dont l'un est dit dominant (nom concret ici) on ne peut appliquer cette règle que si l'un des marqueurs de l'élément vérifie la règle de sélection de l'autre.

Sur l'exemple on interdit donc de construire les structures conduisant à interpréter "colourful ball" comme un "obus pittoresque" ou une "balle pittoresque".

La règle de sélection imposée par le verbe "to hit" conduit à construire pour "The man hits the colourful ball" une seule structure correspondant à l'interprétation de"L'homme frappe la balle colorée".



Ce type de modèle linguistique a été utilisé par Stanley PETRICK [Pet] pour la réalisation du système REQUEST. REQUEST [Pla] est un système d'interrogation en langue naturelle de la base de données Fortune 500 contenant des données essentiellement numériques sur les 500 plus grandes entreprises américaines.

On peut noter cependant que Fodor et Katz ont avancé alors qu'une théorie sémantique devait relever seulement de la connaissance linguistique et non de la connaissance de l'univers. Ainsi ils ont prétendu que l'ambiguité de :

"We sell horse shoes" (≃ on vend des fers à cheval)
et "We sell alligator shoes" (≃ on vend des chaussures en crocodile)

ne relevait pas de leur théorie. Mais ils n'ont pas su définir la frontière si elle existe entre connaissance linguistique et connaissance de l'univers. On comprend d'ailleurs assez mal pourquoi les différents sens de "ball" (dont une "activité sociale" se référant au monde réel) relèvent de leur théorie et pas ceux de "shoe". On peut donc considérer que Katz et Fodor ont introduit malgré eux la notion de représentation de la connaissance, que Winograd utilisera plus tard en se réclamant d'eux.

3.1.3 - Les grammaires de cas

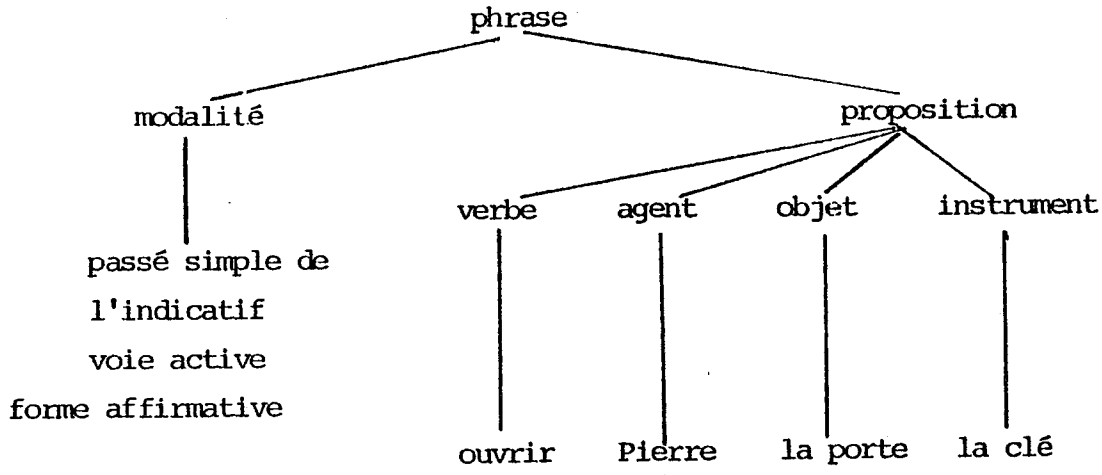
Un peu plus tard, une approche plus conceptuelle et plus fonctionnelle était proposée par FILLMORE. Dans son "Case for Case"[Fil]Fillmore montre que la structure sous jacente à une phrase repose sur une notion fonctionnelle des éléments qui la constitue, en quelque sorte qu'il existe une structure plus profonde que la structure profonde des grammaires transformationnelles. La fonction d'un élément dans la phrase est précisément exprimée par un cas profond ou relation de cas.

Il montre sur l'exemple

Pierre brise la vitre
Le marteau brise la vitre

que les catégories sujet-objet traditionnelles ne sont que les manifestations de surface de fonctions profondes : agent, objet, instrument.
Il propose donc une décomposition de la phrase en modalité et proposition.
La modalité exprime le mode, la voie, le temps, etc...
La proposition est un vecteur verbe + {cas}*

Exemple :



est la représentation de "Pierre ouvrit la porte avec la clé".

Fillmore a proposé 6 relations de cas agent, objet, datif, instrument, locatif, factitif, en précisant que cette liste n'était pas exhaustive.

Remarquons que Fillmore n'a fait qu'officialiser (en langue anglaise) un courant de pensée suggérant une analyse fonctionnaliste de la phrase déjà envisagée par Tesnière [Tes] avec la notion de prédicat-actant et par les chercheurs soviétiques [Mel-Zho].

Le terme de grammaire est d'ailleurs impropre ici, les "grammaires" de cas ne pouvant ni engendrer une phrase du langage ni accepter l'appartenance d'une phrase au langage.
Elles ne sont qu'un moyen de représentation coïncidant à la fois avec le monde réel et avec la linguistique, les cas flectionnels de certaines langues (grec ancien) n'étant que la manifestation de surface de cas profonds. Ce qui explique la large utilisation qui en a été faite par les chercheurs en Intelligence Artificielle, y compris dans le modèle proposé ici, à travers diverses réalisations que nous nous proposons d'examiner ensuite.

3.1.4 - Le modèle conceptuel de Schank

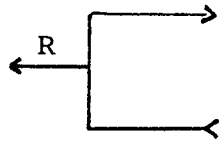
Ce modèle construit sous forme de réseau sémantique une représentation conceptuelle d'un énoncé. Cette représentation est obtenue en utilisant un jeu de primitives exprimant une dépendance conceptuelle entre les éléments :

Ces primitives représentent à l'origine des cas conceptuels chaque cas étant noté par un symbolisme particulier [SCH1] [SCH2] [SCH3].

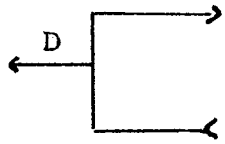
agent : <=>

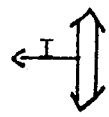
objet : ♀

récipient : cas indiquant la transition de possession et noté



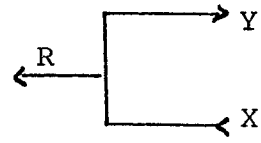
la direction indiquant sous la même forme que pour le récipient la transition d'un lieu vers un autre.



L'instrument noté 

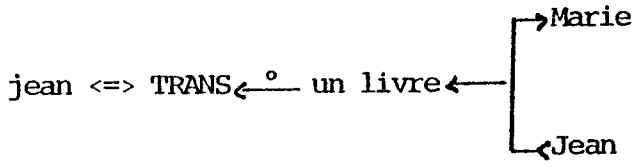
l'attributif noté 

Ces primitives servent à exprimer des "conceptualisations" qui sont des schémas conceptuels avec lesquels les phrases sont interprétées.

Z <=> TRANS ←°— objet physique 

est un schéma conceptuel correspondant au verbe "donner" si X = Z et au verbe "prendre" si Z = Y qui sont des verbes exprimant un transfert (TRANS).

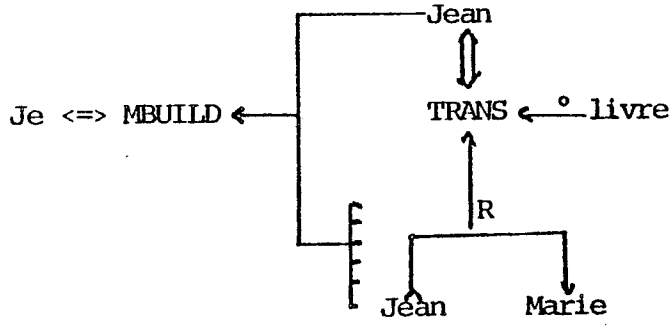
On interpretera "Jean donne un livre à Marie" sous la forme



Ces schémas peuvent s'imbriquer dans d'autres schémas pour la représentation de phrases plus complexes :

"Je conclus que Jean a donné un livre à Marie"

se représente :



ou MBUILD exprime une construction mentale.

Ce modèle d'interprétation a donné lieu à de nombreux développements aux USA, en particulier pour des systèmes de question réponse ou d'apprentissage [SCH4] [SCH5] [LE1] [CUL]. Mais il faut noter que tous ces auteurs sont remarquablement discrets au sujet de la mise en oeuvre de leur modèle et des moyens utilisés en particulier pour l'obtention du réseau.

3.2 - REALISATIONS INFORMATIQUES

3.2.1 - Le robot de Winograd

Ce robot manipulateur de blocs est commandé par des ordres exprimés en langue naturelle. Ces ordres sont analysés par une "grammaire systémique", sorte de grammaire générative utilisant une hiérarchie définie sur le vocabulaire pour fonctionner en analyse. Ce type d'analyse n'amenait rien sur le plan linguistique mais a été le premier à introduire les techniques d'Intelligence Artificielle dans le domaine linguistique, et en particulier la notion de représentation de la connaissance. La représentation de la connaissance est ici une somme de données sur l'univers de blocs dans lequel le robot évolue [WIN1]. La grande originalité de ces travaux a été de réaliser cette représentation sous une forme procédurale. Ceci entraîne un mécanisme "calculatoire" amenant simultanément la compréhension de l'énoncé et la manipulation du robot de façon dynamique. Il autorise également un certain parallélisme, la vérification sémantique s'effectuant simultanément à l'analyse grammaticale.

exemple : procédure (dans le langage PLANNER utilisé) spécifiant la sémantique du cube (bloc dont les arêtes sont d'égale dimension)

```
cube : (THPROG(X)
           (THGOAL(X) (# IS $ ? X  #BLOCK))
           (#EQDIM  $?X))
```

Ce système prometteur n'a pourtant pas eu de successeurs et n'a pas pu non plus être enrichi, la technique d'analyse linguistique utilisée étant un obstacle certain. Ce système était davantage une proposition méthodologique vers la résolution de problèmes que vers le traitement des langues.

3.2.2 - SIMMONS

SIMMONS a sans doute été le premier [SIM2] [SIM3] à élaborer un système utilisant une grammaire de cas, comprenant les cas suivants :

- actant causal, thème, source, but,

Ces cas ne sont pas définis formellement mais seulement à l'aide d'exemples. Il semble que la distinction principale avec Fillmore réside dans le fait qu'un syntagme peut remplir simultanément plusieurs relations de cas.
Dans : John court

John est à la fois l'actant causal et le locus

Une phrase peut comporter plusieurs actants causal (agent, datif).
Dans un premier temps, SIMMONS utilise un réseau de transition augmenté [§ 3.2.5] pour effectuer l'analyse de la modalité.
A partir de la phrase

A merry widow had been dancin a jig

le reseau de transition amène la construction de

C_1 tok : merry	C_2 : tok : widow	C_3 : tok : dance	C_4 : nbr	3pp
	nbr : sin	modal C_4	tense	past
	det : indef		aspect	perfect
	mod : C1		mood	indic
			voice	active

C_5 tok jig
nbr sin
det indef

A ce stade la consultation du dictionnaire amène :

dance = case : (actant causal locus 1) thème locus2 ;
subject : (actant causal locus1) - object : thème - prep : locus2 ;
locus : animate - thème : dance - locus2 : animale

qui implique qu'au verbe danser peuvent être associés les cas actants et locus qui est alors le sujet du verbe, thème qui est l'objet et locus 2 qui est un groupe prépositionnel. A chacun de ces cas est assigné un marqueur sémantique précis. Le dictionnaire est donc ici très explicite et détermine quasi totalement la structure possible, ne permettant l'interprétation que d'une classe limitée d'énoncés.

Cette consultation du dictionnaire va autoriser ou non la construction d'un réseau sémantique [SIM1]. Chaque réseau construit est placé dans une base de données, pour constituer un système de question-réponse [SIM3].

3.2.3. - WILKS

Le système de WILKS est plus complexe et plus complet que celui de SIMMONS.
Examinons en le fonctionnement sur l'exemple [Wil 2]

"The big policeman interrogates the crook"

Dans un premier stade l'analyse syntaxique segmente le texte en

(the big policeman) (interrogates) (the crook)

Puis la consultation d'un dictionnaire amène à la construction de "formules" qui elles-mêmes amènent un patron ("template") qui est un vecteur de marqueurs. "crook" étant ambigu, on obtient ici deux patrons

MAN FORCE MAN et MAN FORCE THING

On consulte alors une grammaire des patrons, dans laquelle on trouve

```
        * PO FORCE *EN
et      *PO → MAN | FOLK | BEAST | PLANT ...
        *EN → MAN | FOLK | GRAIN | THING ...
```

ce qui rend possible les deux interprétations.

On examine alors les formules qui ont été construites dont

interrogate : ((MAN SUBJ) (MAN OBJE) (TELL FORCE))

qui conduit à la sélection de la solution MAN FORCE MAN.

Mais ces formules ne sont pas impératives, c'est-à-dire qu'elles n'interdisent pas une autre interprétation si aucune ne vérifie les formules. C'est que que WILKS nomme sémantique préférentielle [Wil3].
Par exemple la définition dans le dictionnaire du verbe

boire : agent 'animé'
objet ('écoulement' 'substance')
contenant 'animé'
direction ('passage' 'ouverture') spécifique de 'animé'
cause 'existence'

exprime que le verbe boire est impliqué de préférence dans une phrase ou l'agent est animé et l'objet un liquide s'écoulant à travers une ouverture de l'être animé.
Mais elle n'empêche pas la construction d'une structure pour

"Cette voiture boit beaucoup d'essence"

On voit que le système de WILKS, destiné à la traduction automatique à l'origine repose également sur une grammaire de cas reposant sur une notion conceptuelle. Ce système est plus souple que celui de SIMMONS et plus puissant sans doute puisqu'il permet l'inférence [WIL1] mais il manque de dynamisme dans la manipulation des concepts.

3.2.4 - COLBY

Le but du système PARRY est de reproduire un comportement paranoiaque dans le discours, en vue d'études psychiatriques. L'auteur [Col1][Col2] part du principe qu'un être humain n'utilise certainement pas une grammaire transformationnelle, voire une grammaire tout court pour comprendre ce qu'on lui dit, mais reconnait plutôt des schémas mémorisés. Son système n'utilise donc pas de techniques linguistiques sophistiquées mais essaie de reconnaître à travers un énoncé un "patron". Ensuite on recherche ce patron dans une base de données et on génère une réponse influencée par le modèle psychiatrique.

Exemple : "What do you do for a living ?"

devient le patron WHAT BE YOU JOB

pour lequel le système produit une réponse.

Colby s'est astreint à ce que le système ne soit pas fragile, c'est-à-dire ne s'arrête jamais même s'il ne reconnait pas un énoncé.
Ce système original est donc à la fois le plus volumineux et celui qui utilise le moins d'informations linguistiques. Par contre, l'exigence de reconnaître quasiment toute la langue courante du dialogue a amené les constatations suivantes :

- l'importance fondamentale d'une morphologie complète et rapide pour le traitement d'un gros corpus, reconnaissant les locutions et les idiomes (heure de pointe, casse pipe , etc...

- l'intérêt de la multiplicité des modèles, éventuellement redondants, pour la mise en oeuvre, la détection d'erreurs et le parallélisme.

Ce système a l'avantage d'accepter tous les énoncés qu'on lui donne et d'être vraiment "naturel". Le principal reproche qu'on peut lui faire est qu'il n'analyse pas l'énoncé puisqu'il s'agit d'une simple recherche de patrons pour trouver la réponse et que Colby a beau jeu d'accepter tous les énoncés en simulant un comportement paranoiaque. (En cas d'échec de la recherche d'un patron, le modèle paranoiaque engendre une réponse du style -En quelle saison sommes-nous ? - j'en ai assez de cette interview !).

3.2.5 - Les réseaux de transitions augmentés

Au vu de l'inadéquation des grammaires transformationnelles pour l'analyse des langues dans un cadre informatique, WOODS a suggéré une méthode analytique basée sur les grammaires d'état fini. Ces dernières sont faciles à mettre en oeuvre et analysent une chaîne de la gauche vers la droite. Elles se révèlent donc particulièrement adaptées.

Rappelons qu'une grammaire d'état fini est un ensemble fini de sommets appelés états connectés par des arcs orientés étiquetés appelés transitions. L'étiquette associée à un arc indique le symbole qui doit être présent dans la chaîne d'entrée pour effectuer la transition. Une chaîne d'entrée est acceptée par la grammaire si la succession de transitions conduit à un état particulier appelé état final.

Ces grammaires étant néanmoins trop faibles pour l'analyse linguistique WOODS [WO1] a présenté un système appelé réseau de transition augmenté introduisant un mécanisme de contrôle récursif basé sur le principe suivant :

- les états de la grammaire sont étiquetés et peuvent figurer comme étiquette d'un arc.

- A l'occurrence d'un nom d'état en tant qu'étiquette d'un arc, l'état courant est empilé ("arc push") et l'analyse reprend à partir de l'état désigné par l'étiquette.

- Lorsqu'un état final est atteint, on relance l'analyse à partir de l'état qui est retiré du sommet de la pile ("arc pop").

- Une chaîne est acceptée par la grammaire si l'on atteint simultanément la fin de cette chaîne et un état final, et que la pile est vide.

Ceci constitue un réseau de transition, qui peut analyser un langage à contexte libre, mais qui est encore insuffisant pour le traitement d'une langue, car il ne permet pas la transduction d'informations.

La puissance nécessaire est obtenue par les actions et les conditions.

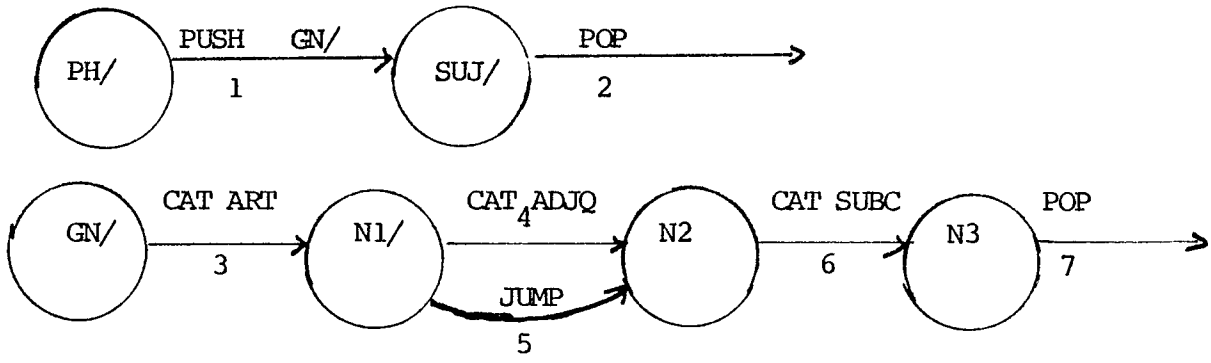
- Une action est une fonction opérant sur des registres. Un registre est assimilable à une variable d'un langage de programmation.
  Les actions ont pour effet d'affecter à ces registres une valeur calculée (l'affectation est notée ←).

. Une condition est un prédicat sur le contenu de ces registres. Une transition ne peut être effectuée que si le prédicat est vrai à ce moment. On exécute alors l'action associée.

Pour assurer le fonctionnement global, on utilise un registre spécial appelé *. Ce registre contient normalement le symbole d'entrée courant. Pour un arc push il contient la structure construite au préalable et que l'on empile. Un arc pop a pour effet de placer le contenu du sommet de la pile dans ce registre.

Un réseau de transition muni de conditions, d'actions et du registre * est dit réseau de transition augmenté (abrévié en A.T.N. pour augmented transition networ

Exemple : Analyse du groupe nominal "La belle femme" par le réseau



Une transition CAT cs impose que le symbole d'entrée appartienne à la catégorie syntaxique cs.
Une transition JUMP s'effectue sans faire avancer la chaîne d'entrée.
Les conditions et actions associées à chaque arc sont données ci-après ainsi que le résultat obtenu sur l'exemple.
Les actions utilisent les registres cat, def, gnr, nbr, det, mod, qui contiennent respectivement un indicateur de catégorie, de définition, de genre, de nombre, de déterminants de modulateur.

La fonction auxiliaire construire construit au fur et à mesure les sommets de la structure produite.

La fonction get (indicateur) recherche la valeur associée à l'indicateur concerné pour le symbole se trouvant dans le registre *.

Arc	condition	action	résultat
1	vrai		
2	vrai		
3	vrai	def ← get(determ) nbr ← get(nombre) gnr ← get(genre) cat ← get(catsynt) construire (S1, (*,cat,gnr,nbr,det))	S1 la cat = art nbr = sin gnr = fem def = def
4	(gnr = get(genre)) et (nbr = get(nombre))	construire (S2, (*,cat,gnr,nbr))	S2 belle cat = adjq nbr = sin gnr = fem
5	vrai		
6	(gnr = get(genre) et (nbr = get(nombre))	det ← S1 mod ← S2 construire (S3, (*,cat,gnr,nbr,del,mod))	S3 femme cat = subc gnr = mas nbr = sin det = S1 mod = S2
7	vrai		

Les A.T.N. utilisés à l'origine par WOODS pour le système LUNAR [WO2] ont été repris ensuite par d'autres chercheurs aux USA essentiellement [Bob1] [SIM3] [KAP] .

3.1.3 - CONCLUSIONS

Sur le plan linguistique, quelles que soient les stratégies adoptées l'analyse d'un texte repose finalement toujours sur les mêmes éléments

- Une structure permettant d'exprimer un énoncé de façon syntaxique ou conceptuelle (structure profonde , structure casuelle, conceptualisation etc...).

- Un lien entre le texte et cette structure "les éléments de sens exigeant pour leur réalisation dans la connaissance un nombre précis d'autres éléments d'un type donné et ayant avec eux des relations déterminées" [IINOF]; c'est-à-dire un moyen d'exprimer comment un élément du texte peut s'inscrire dans un schéma structurel et ce qu'il représente alors (case frames, formules,...) [Wil4] . Ce lien s'effectue en général au moyen d'un dictionnaire que ce soit pour y rechercher une structure de cas, ou une formule, ou un "patron" ou une condition d'application de règles.

- Une représentation de la connaissance de l'univers sous jacent au corpus, reposant quasi systématiquement sur un ensemble de marqueurs sémantiques liés par des relations et permettant d'exprimer à la fois la cohérence de cette structure et ce qu'elle représente.

Sur le plan technique, l'analyse conceptuelle repose sur un premier découpage de l'énoncé en unités et une analyse des modalités du verbe (mode, temps), éventuellement une analyse syntaxique. Mais ces analyses peuvent se réaliser avec des modèles indépendants dont la multiplicité est intéressante:

- Elle autorise le parallélisme donc une accélération du processus, du simple fait du parallélisme mais également parce que l'un des modèles peut détecter une erreur et stopper les autres.

- Les différents modèles parallèles peuvent être redondants ce qui facilite la détection et la correction des erreurs [Co2].

- Elle se traduit sur une implémentation modulaire.

- Mais essentiellement on peut espérer atteindre par là un modèle complet par recoupement de modèles partiels que l'on peut compléter aisément de façon incrémentielle.

CHAPITRE - IV

UN NOUVEAU MODELE D'EVALUATION

ET SA REALISATION

Dans un cadre d'Intelligence Artificielle, le but est de "faire comprendre" à la machine un énoncé en langue naturelle.
On pourrait donc penser que la représentation interne des énoncés doit être une représentation "conceptuelle", c'est-à-dire une représentation en quelque sorte indépendante de l'énoncé et permettant d'effectuer des inférences, des jugements etc...

C'est probablement cette démarche qui a amené Schank et d'autres auteurs particulièrement aux Etats-Unis [Rig] [Le2] [Hen] [Qui] à formuler des modèles où l'énoncé disparait dans une représentation qui se veut une interprétation totale de l'énoncé. Cette attitude nous parait dangereuse :

- Elle souscrit au mythe de la forme canonique [WO2] et à la croyance en l'existence d'une interprétation totale de l'énoncé, c'est-à-dire complète (regroupant toutes les interprétations de tous les lecteurs éventuels) et invariable avec le temps.

- Elle élimine toute possibilité de "retour arrière", c'est-à-dire que lorsque la représentation est constituée elle ne permet plus le retour au texte original, ce qui peut être gênant (Dans une application documentaire comme celle qui envisagée ici par la suite, l'utilisateur peut désirer savoir si "Jean a donné un livre à Marie" ou si "Marie a pris un livre à Jean" parce que cette information est vitale pour lui.

Pour éviter ces travers la représentation visée ici est une simple représentation de l'énoncé, où le texte est conservé à travers une structure reflétant une dépendance conceptuelle. Cette représentation correspond à une grammaire de cas. L'interprétation sémantique "se réalise par des références des éléments de la structure à des "concepts".

Il doit être clair ici que cette représentation se veut partielle et partiale. On cherche ici à interpréter un énoncé exclusivement en fonction de l'application concernée. Il s'agit en quelque sorte d'une interprétation guidée par le but, c'est-à-dire effectuée selon des critères définis par l'application visée.

Il apparaît donc que les composants du modèle doivent être des paramètres linguistiques du système, modifiables à chaque utilisation de façon interactive.

Après avoir formulé la représentation adoptée, on examine les problèmes posés par sa construction, d'où on dégage les caractéristiques du modèle présenté. On y fera apparaître en particulier la notion d'évaluation dynamique des concepts à partir d'une hiérarchie de marqueurs qui en fait l'originalité. Ayant défini les constituants de ce modèle on verra qu'il opère à partir de structures syntaxiques, et utilise naturellement l'algorithme défini dans la partie I.

Dans la phase de réalisation abordée ensuite, ces structures sont produites par le système PIAF qui assure toute la partie morphologique et syntaxique. Après un bref rappel des caractéristiques de ce système on montrera comment se réalise l'intégration des différents composants dans un même module.

IV-I - LE MODELE

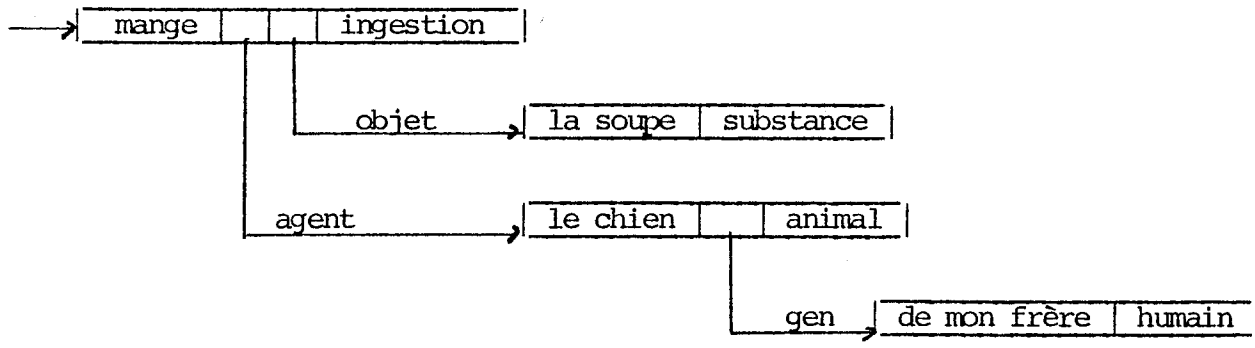
4.1.1 - REPRESENTATION DE L'ENONCE

L'aboutissement de l'analyse effectuée ici est une représentation de l'énoncé sous forme d'une structure arborescente similaire aux réseaux de SIMMONS [SIM1]. La structure produite est un ensemble de noeuds et d'arcs étiquetés.

Chaque arc correspond à une relation de cas dont la nature est spécifiée par l'étiquette.

Un noeud est caractérisé par un littéral (un élément de l'énoncé), les cas profonds qu'il régit, et une référence à un marqueur sémantique. Un noeud est l'extrémité d'un arc.

La phrase "Le chien de mon frère mange la soupe" sera représentée par :



que l'on écrira aussi :

```
(mange
      (agent ( le chien
                        (gen   (de mon frère)
                               humain))
               animal))
      (objet (la soupe)
             substance))
 ingestion)
```

Comme on l'a déjà souligné, le choix de cette représentation est significatif

4.1.2 - LA GRAMMAIRE DE CAS

On a placé diverses définitions derrière le terme grammaire de cas [Bru] et on essaie maintenant de dégager ce que représentent les relations de cas et comment on peut les construire. On utilisera à cet effet et à titre d'illustrations, un système casuel inspiré de celui défini par M. POTTIER [Pot] qui comporte les cas suivants :

- agent (AGT) : groupe nominal instigateur de l'évènement suggéré par le verbe.

- objet (OBJ) : syntagme nominal correspondant à l'interprétation du verbe.

- datif (DAT) : entité désignée par l'évènement.

- bénéfactif (BEN) : entité tirant profit de l'évènement.

- instrument (INS) : objet physique servant à la réalisation de l'évènement.

- situatif (SIT) situe l'évènement. Le situatif peut être locatif, temporel, ou notionnel .

- sociatif (SOC) : associé à l'évènement.

- causal (CAUS) : cause de l'évènement.

- final (FIN) : but de l'évènement.

Exemples :

Le petit cheval blanc galope dans les près
AGT SIT

Annie ferme la porte avec précaution
AGT OBJ SOC

Les soldats chargent quand le clairon sonne
AGT SIT

Il s'est blessé en tombant
OBJ AGT SIT

Le prévenu est condamné par erreur
OBJ CAU

Je travaille pour un salaire de misère
AGT FIN

Mais il faut noter que ce système casuel n'est donné que pour illustrer les exemples qui vont suivre. On ne prétend pas l'imposer comme solution universelle (non plus que son auteur d'ailleurs) et on verra au contraire que ce système est à redéfinir à chaque application.

Le problème des grammaires de cas est qu'elles expriment de façon satisfaisante la représentation d'un énoncé, mais qu'elles n'induisent en rien une méthode analytique permettant de construire cette représentation.

On cherche donc maintenant à dégager les caractéristiques de ces relatio de cas, ainsi que leur lien avec l'interprétation sémantique de l'énoncé dans lequel elles s'insèrent, afin d'aboutir à une méthode constructive de la représentation visée.

4.1.2.1. - Notion de paradigme

Les auteurs [Tes] [Fil] [Io-Me] ont déjà mis en évidence le fait qu'à un élément lexical est lié une structure paradigmatique, c'est-à-dire qu'ils sont toujours utilisés dans une structure possédant les mêmes cas profonds spécifiques, ou en d'autres termes, que cet élément est un prédicat sur des actants dont la fonction est déterminée. (Par exemple, pour le verbe donner, on distingue celui qui donne, ce qu'il donne et à qui il le donne). Il est évident que la réalisation de surface n'est pas invariable, certains actants pouvant par exemple ne pas être utilisés. Le problème est donc de reconnaître ces actants pour une structure donnée, de savoir "qui est quoi".

Par ailleurs, cette notion de paradigme a été l'objet d'études montrant le regroupement d'un ensemble de mots possible derrière un même comportement linguistique. M. Gross a ainsi classifié environ 4000 verbes de la langue française [Gro1].

On partira donc ici de cette notion de paradigme pour la détermination des relations de cas.

4.1.2.2. - Distinction par l'information conceptuelle

Une relation de cas conceptuelle est évidemment liée au signifiant du syntagme qu'elle étiquette.

La distinction entre

Je donne des bonbons à mon frère
OBJ DAT

Je donne des bonbons à la menthe
OBJ

repose sur la reconnaissance du concept 'être humain' associé à "mon frère" dépendant du verbe donner, et du concept 'parfum' qualifiant le bonbon.

On ne propose pas par là une classification spécifique des marqueurs, ni une méthodologie. On veut simplement indiquer que dans les paradigmes du verbe donner la reconnaissance d'un actant humain en fait un candidat possible pour un cas datif, ce qui n'est pas le cas de "menthe".

4.1.2.3 - Cas et concepts - Leurs liens réciproques

Une relation de cas n'est pas significative par elle-même pour évaluer un concept.

Les deux phrases le lapin ronge la carotte
AGT OBJ

l'acide ronge le fer
AGT OBJ

ont exactement la même structure de cas, et pourtant elles reflètent deux concepts différents. Ces deux conceptualisations sont dues aux différents sémèmes se rattachant au verbe ronger. Selon la nature sémantique de ses dépendants, le verbe peut référer à différents concepts.

On fait apparaître ici la notion d'évaluation, c'est-à-dire de "calcul" nécessaire pour le rattachement d'un énoncé à un concept et le fait que les relations de cas sont les vecteurs de ce calcul.

Mais cette notion va plus loin encore, les concepts eux-mêmes pouvant être résultat d'une évaluation.

Considérons les phrases

- le gouvernement étend le bénéfice de l'assurance maladie aux prêtres
- le bénéfice de l'assurance maladie est étendu aux prêtres à partir du 1er janvier 1978
- le projet gouvernemental étend le bénéfice de l'assurance maladie

Le concept qui se dégage de ces trois énoncés est la "généralisation de la sécurité sociale". Il est clair que l'on se place ici dans le cadre d'une application donnée (§ 5.7) pour laquelle c'est ce concept qui résulte de l'interprétation.

Deux constations se dégagent de ces exemples.

Une relation de cas peut n'avoir aucune influence sur les concepts : Ici, quelque soit la nature de l'agent et qu'il soit présent ou non le concept résultant est le même.

Le concept globalisant un énoncé, ou un fragment d'énoncé, résulte d'une combinaison conceptuelle amenée par les constituants de cet énoncé. Ici le verbe "étendre" amène la notion de 'généralisation' de l'actant "assurance maladie", qui fait référence au concept 'sécurité sociale', "bénéfice" étant un mot transparent.

On peut donc dire que l'interprétation d'un énoncé est le résultat d'une évaluation "procédurale" dont les paramètres sont les relations de cas.

Il résulte de ces considérations que

⊕ les concepts associés à un énoncé résultent de la combinaison de deux types d'information, l'une liée directement à l'élément lexical (frère, bonbon), l'autre liée au paradigme dans lequel cet élément est imbriqué. D'une façon générale les concepts associés à un énoncé résultent d'une évaluation dans un certain "contexte". On trouvera donc dans le dictionnaire, associé un élément lexical, une indication sur les paradigmes dans lesquels il peut se réaliser, et une fonction sémantique exprimant le référent de cet élément selon la construction dans laquelle il est employé.
ronger (agent, objet) - si l'agent est un animal et que l'objet est comestible s'il existe alors le référent est 'ingestion'
- si l'agent et l'objet sont deux corps naturels le référent est 'action chimique'.

⊕ Une relation de cas ne fait qu'assigner une notion fonctionnelle à un actant mais n'induit rien de par sa nature sur le plan conceptuel. Elle est par contre "véhiculaire" d'une information conceptuelle et fournit précisément un élément du "contexte" d'évaluation de la fonction sémantique.

4.1.2.4. - Importance de la morphologie et de la syntaxe

Nous sommes d'accord avec Mel'chuck [Io-Me] pour penser que l'analyse syntaxique est un premier pas nécessaire vers une analyse plus conceptuelle de l'énoncé. La syntaxe, en plus de la distinction sujet-objet, permet essentiellement de structurer l'énoncé en groupe prépositionnels, chacun de ces groupes vérifiant une relation de cas, qu'il ne reste plus qu'à "étiqueter" par le modèle sémantique. Le modèle syntaxique peut aussi détecter des incohérences d'énoncé ou un découpage particulier d'énoncé par des règles grammaticales que peut difficilement, si ce n'est pas du tout, détecter un modèle sémantique ("la maison de l'oncle que nous avons vu"). Ces deux modèles sont complémentaires, même si leur intersection n'est pas vide.

Au niveau du système, l'outil syntaxique, par réduction des constructions verbales, formation des groupes nominaux, etc... opère un dégrossissage facilitant le travail du modèle sémantique.

Une analyse morphologique sophistiquée est également un atout non négligeable à la fois pour le contrôle syntaxique (détection des conjugaisons, analyse des locutions du langage) et au niveau sémantique. Un repérage particulier des prépositions en particulier est une heuristique très forte pour la détermination d'une relation de cas. Par exemple la préposition "parce que" introduit toujours une relation de finalité. D'autres sont ambigues mais permettent de restreindre le champ des possibles.

4.1.2.5 - Conclusion

Les expériences menées jusqu'à maintenant ont construit les relations de cas de façon déterministe. Le dictionnaire de SIMMONS impose totalement la structure de cas liée à un élément, les formules de WILKS, bien que préférentielles, aussi. Il n'y a pas de "matching" d'une structure donnée contre une possible. C'est ce qu'on se propose de faire ici : rechercher à partir d'un paradigme possible, une coïncidence avec l'énoncé à analyser.

Cette recherche s'effectue non pas directement sur l'énoncé mais à partir d'une structure syntaxique.

Et pour déterminer le paradigme on utilisera aussi bien des informations de nature lexicale que syntaxique ou sémantique. Une relation de cas apparaît donc comme un prédicat sur une structure syntaxique et c'est sous cette forme qu'elle existe dans le système.

La détermination de ces relations de cas permet de constituer un contexte d'évaluation pour une fonction sémantique qui fournit l'"interprétation" de cette structure, offrant par là une notion dynamique d'évaluation conceptuelle.

Ceci constitue le principe du modèle présenté.

4.1.2.6. - Définition de relations de cas sur les substantifs

Habituellement les relations de cas s'appliquent aux constructions verbales. Afin de pouvoir traiter uniformément les énoncés, c'est-à-dire les relations entre les groupes nominaux ainsi que les phrases nominales qui apparaissent presque exclusivement dans les applications traitées, la grammaire de cas utilisée ici s'étend aux substantifs. Un travail similaire a déjà été accompli au C.E.T.A. [Ve2] et même plus complet, puisqu'il comprenait le traitement d'adjectifs particuliers que l'on a laissé de côté.

On distingue essentiellement deux types de relations selon la nature du substantif : substantif primaire ou secondaire [Pot].

Les substantifs primaires sont ceux qui désignent un concept primitif : chien, maison, fleur, etc...

Les substantifs secondaires sont ceux qui dérivent d'un verbe par nominalisation d'un verbe : distribution, vente, morsure, etc...

Les substantifs primaires admettent trois relations de cas, génitif, situatif, attributif.

Le génitif exprime le "complément de nom"

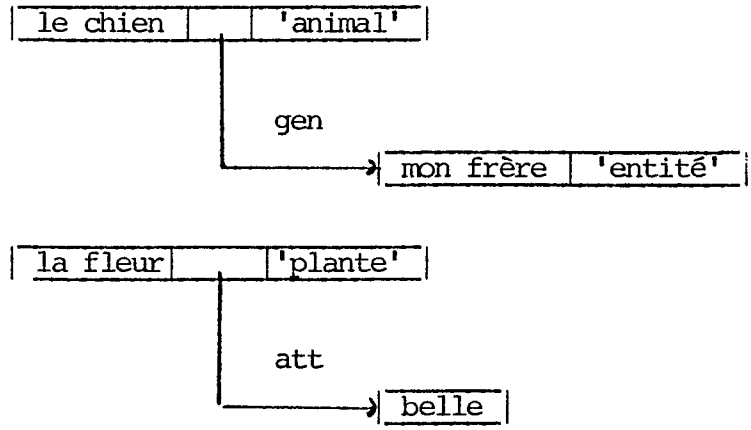
la maison de pierre, le banc du jardin, le chien de mon frère

L'attributif qualifie le groupe nominal "le beau petit chien"

Le situatif, comme pour les verbes, situe le nom dans le temps ou l'espace

"le chien dans la niche", "la vitesse avant le choc"

Le génitif et le situatif nécessitent une évaluation qui n'est pas nécessaire pour l'attributif. On aura par exemple :



Les substantifs secondaires admettent la même structure de cas que le verbe dont ils dérivent avec une construction syntaxique différente.

4.1.3. - LE DICTIONNAIRE

Chaque entrée du dictionnaire comporte deux parties l'une utilisée pendant l'analyse morphologique et syntaxique, sur laquelle nous reviendrons, et l'autre à la construction de la représentation sémantique qui nous concerne maintenant.

Cette deuxième partie se divise elle-même en deux, l'une assez traditionnelle qui exprime le lien entre le texte et les relations de cas, nommé modèle casuel (case frame) ou modèle paradigmatique ; l'autre est la fonction sémantique.

4.1.3.1 - Les modèles casuels

C'est ce qui définit les paradigmes possibles pour l'item concerné (une entrée dans le dictionnaire).

A un item donné est associé un certain nombre de relations de cas dépendantes possibles ou nécessaires pour la réalisation de cet item. C'est ce que Fillmore note par exemple :

ouvrir : + [- O(I) (A)] pour indiquer que ce verbe possède un objet et éventuellement un instrument et un agent.

On a vu que l'on pouvait définir un ensemble d'éléments lexicaux ayant le même comportement, c'est-à-dire le même jeu d'éventualités de relations de cas. Ce jeu d'éventualités est ce que l'on appelle un modèle casuel qui est ici étiqueté

exemple : VERBT = (? agt & objt & inst)

Le préfixe & indique devant un symbole ou une liste de symboles que ce qui suit est facultatif.

Le préfixe ? indique la nécessité.

On trouve ensuite dans le dictionnaire pour un item un ou plusieurs modèles auxquels il fait référence :

ronger : VERBT

Il va de soi que l'établissement de ces modèles n'est pas un problème trivial ; les travaux des linguistes à ce sujet étant assez récents.

Mais nous pensons que la découverte de ces modèles peut se réaliser par des méthodes d'apprentissage qui permettraient en plus de conserver la caractéristique incrémentielle nécessaire pour un système.

4.1.3.2. - Les fonctions sémantiques

Ces fonctions expriment, de façon similaire au robot de Winograd, à la fois une connaissance linguistique et une connaissance de l'univers concerné par le corpus traités. Chacune exprime le référent de l'élément du dictionnaire auquel elle est attachée. Cette fonction est exprimée dans un langage symbolique similaire à un langage de programmation et pour lequel on dispose à la fois d'un éditeur et d'un interpréteur.

Une entrée dans le dictionnaire comporte donc deux champs sémantiques et on la note :

/ entrée / modèle casuel / fonction sémantique .

Une fonction sémantique s'écrit :

```
fsem (liste des éléments servant à l'évaluation) ;
     expression du langage symbolique
```

Exemple :

```
/ ronger / VERBT /
     fsem  (agent, objet)
       si agent = 'animal' et (si existe (objet)
                                  alors 'ingestion
                                  sinon
                                        vrai
                               fsi)
          alors
                'ingestion'
          ssi agent = 'substance' et objet = 'substance'
          alors 'action chimique'
          sinon
                'nul'
       fsi
```

Cette fonction sémantique peut être invariante :

4.1.4. - LES DESCRIPTEURS SEMANTIQUES

L'ensemble des descripteurs est l'ensemble des concepts fondamentaux exprimés par le corpus. C'est la grille à travers laquelle s'effectue l'interprétation. Ces descripteurs doivent donc être déterminés en fonction du corpus et de l'application visée. Ils peuvent être liés entre eux par des relations et le groupement descripteurs-relations constitue en fait un thésaurus.
La définition de ce thésaurus peut être très simple pour un certain corpus (un corpus scientifique par exemple au concepts restreints et définis) ou au contraire un problème pour lequel il n'exite pas de solution satisfaisante à l'heure actuelle (base documentraire hétéroclite par les sujets traités).

IV-2 - REALISATION : LE SYSTEME

On examine comment se fait la mise en oeuvre du modèle, c'est-à-dire comment on obtient pratiquement la représentation de l'énoncé. Dans un premier temps on utilise une version adaptée du système PIAF qui construit une structure arborescente sur laquelle opère le modèle d'interprétation dont on rappelle brièvement les caractéristiques. Puis on revient à l'interprétation sémantique.

4.2.1. - PRODUCTION DE STRUCTURES SYNTAXIQUES

Elle se fait en deux temps, une analyse morphologique et une analyse de dépendance.

4.2.1.1. - Analyse morphologique

Elle repose sur l'utilisation d'un transducteur général d'état fini qui permet l'analyse d'une langue naturelle à l'aide d'un dictionnaire, d'un ensemble de modèles et d'une grammaire de validation [Gral] [Col] [Co3].

- Le dictionnaire contient des bases, préfixes, suffixes, désinences et des formes comme "au sujet de", "au fur et à mesure", le caractère espace (" ") n'étant pas un séparateur dans ce système. Chaque entrée du dictionnaire fait référence à un modèle.

- Chaque modèle décrit le comportement linguistique des éléments qui lui font référence, et lui même valide ou sature l'application des règles de la grammaire.

C'est à ce niveau qu'on a introduit la notion de trait sémantique en distinguant des catégories syntaxiques différentes qui n'ont pas lieu d'exister sur un plan purement grammatical.
On constitue par exemple les quatre modèles homme, chien, baton, espoir, qui tous les quatre font référence aux mêmes règles de la grammaire, celles qui permettent d'identifier un substantif masculin prennant la terminaison "s" au pluriel. Mais ceci permet immédiatement la distinction conceptuelle entre humain, animal, objet concret, nom abstrait.

De même les prépositions ou locutions sont scindées en plusieurs groupes dont chacun indique un concept. "de" "à propos de", "au sujet de" sont indexées sur le modèle ABOUT. "Avec", "au moyen de" qui ont le même comportement grammatical sont indexés sur WITH.

- La grammaire de validation autorise ou interdit la concaténation des différents éléments identifiés par le dictionnaire, et la transduction des catégories syntaxiques et des variables.

A l'issue de cette analyse on dispose donc d'une fragmentation de l'énoncé en morphèmes, les ambiguités étant énumérées.
L'analyse de "Le boucher sale la tranche" produit

le : { artd mas sin
     { popl mas sin acc

boucher : subp mas sin

sale { verb uno|tre sin ind pres
     { verb dos sin imp
     { verb uno|tre sin sub pres
     { adjq mas fem sin

la { artd fem sin
   { popl fem sin acc

tranche { verb uno|tre sin ind pres
        { verb dos sin imp
        { verb uno|tre sin sub pres
        { subo fem sin

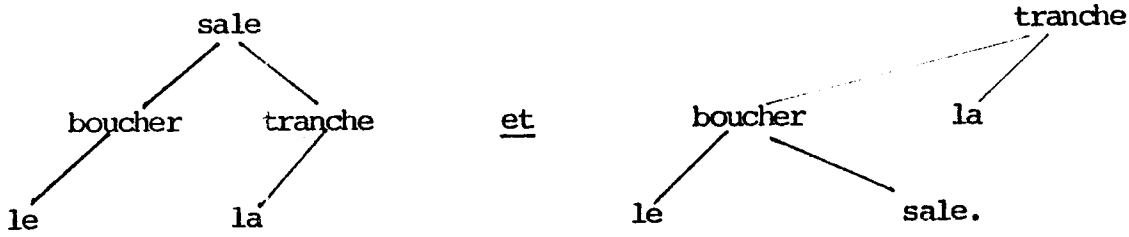
4.2.1.2. - Analyse de dépendance

Elle construit des structures de dépendance syntaxique uniquement en fonction des catégories fournies par l'analyse morphologique et des relations de dépendance établies par le linguiste. Pour construire des structures on n'utilise donc pas de grammaire, mais seulement le graphe des relations entre catégories, et des données topologiques, la structure obtenue devant être projective.

Les relations :

PHRA * VERB := 1 ; VERB * SUBC := -20, 20 ; VERB * POPL := -10 ;
SUBC * ARTD := -30 ; SUBC * ADJQ := -20, -10, 10 ;

on construit pour "le boucher sale la tranche "


sale
boucher
tranche
le
la
et
tranche
boucher
la
le
sale.

4.2.1.3. - Un modèle syntaxique

Un modèle supplémentaire effectuant le regroupement des groupes nominaux la détection des temps composés, des verbes composés (tenir compte) et un contrôle syntaxique fait l'objet des recherches actuelles et des travaux de M. LOPEZ [LOP], bien que le système PIAF dispose déjà d'un modèle syntaxique opérationnel mais insuffisant.

Ce modèle peut se réaliser en une phase parallèle à la morphologie utilisant un transducteur identique au premier mais fonctionnant sur une grammaire différente. Par la suite, nous supposerons acquis certains de ces résultats, déjà obtenus dans d'autres contextes avec les réseaux de transitions (également insuffisants d'ailleurs pour certains traitements).

4.2.2. - EVALUATION SEMANTIQUE

On en détaille le fonctionnement sur quelques exemples.

4.2.2.1 - Fonctionnement général

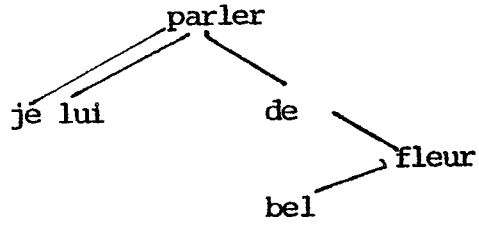
On construit le réseau en opérant sur les structures de dépendance produites. Si il existe plusieurs structures pour un énoncé, on essaie de construire un réseau pour chacune. Si aucun réseau ne peut être construit la phrase est rejetée sinon chaque réseau correspond à une interprétation de la phrase.

L'interprétation d'une structure se fait toujours en appelant la fonction analyse qui possède donc comme paramètre une structure de dépendance et toute l'information associée.

Le rôle de la procédure analyse est de sélectionner un traitement particulier en fonction de la nature linguistique de la structure qui lui est présentée. Ces traitements ont toujours pour effet de rassembler les éléments nécessaires à l'évaluation des fonctions sémantiques et subséquemment à la construction du réseau.

Exemple : "Je lui parle de ses belles fleurs".

On obtient pour cette phrase une seule structure de dépendance



avec l'information associée

parler : VERB, PERS = UN, TPS = PRES, MODE = IND, FORM = AFF,
je : PPS, PERS = UN ;
lui : PPD ;
fleur : SUBO , GNR = FEM, NBR = PLU, DET = POSS ;
bel : ADJQ, GNR = FEM, NBR = PLU ;

En présence d'une structure de ce type la fonction analyse appelle la fonction trouvecas.

Le rôle de la fonction trouvecas est de construire les relations de cas existant dans la structure qu'on lui donne.
A cet effet, elle examine l'information extraite du dictionnaire à ce sujet, à savoir une référence au modèle casuel et construit avec ce modèle une structure.

Ayant extrait pour l'exemple le modèle

(? agani & (about objet) & datif)

La procédure trouvecas appelle à son tour la fonction matchframe avec deux paramètres, la structure à interpréter et le modèle qui doit guider l'interprétation, ici :

(parler je lui (de (fleur bel)))
(parler ?agani &(about objet) &datif)

C'est ici que se produit le lien avec l'algorithme exprimé dans la première partie. En effet, on est en présence à ce stade de deux structures arborescentes. La première, en considérant l'ensemble du vocabulaire français comme l'ensemble des constantes C de la première partie est un arbre déterminé. La seconde est un arbre modèle où les variables sont marquées par un préfixe, variables parmi lesquelles figurent les relations de cas. La production d'une substitution va permettre la mise en correspondance d'une variable cas et d'une sous-structure. On a vu que la définition d'une relation de cas était un prédicat sur la structure qu'elle représente. Ce prédicat vient s'intégrer ici comme contrainte sur le choix de la règle de substitution associée à la variable qu'il représente.

Si l'algorithme de reconnaissance réussit à produire une substitution associant aux variables une sous-structure, on sait que ces sous-structures répondent à la définition des cas conceptuels représentés par ces variables et on est à même de construire la représentation de l'énoncé telle qu'on l'a définie.

Mais la structure donnée en paramètre à la fonction matchframe et correspondant au modèle casuel n'est pas ordonnée. Le rôle de la fonction matchframe se borne donc à construire les arbres modèles possibles et leur conditionnelle associée, et à rappeler la fonction trouvegram.

Sur l'exemple on construit donc l'arbre modèle

(parler [AGANI] [(ABOUT [OBJET])] [DATIF])

et la conditionnelle associée

prédicat agani(x) ; co relation de cas agent animé co
$x \neq \omega$ et concept (x) = 'entité'

prédicat about (x) ;
$x = \omega$ ou catégorie (x) = about

prédicat objet (x)
$x = \omega$ ou concept (x) $\neq$ nul .

prédicat datif (x)
x = ω ou catégorie (x) = popd
ou (catégorie (gouv (x)) = at et concept (det (x)) = entité).

Les prédicats des conditionnelles font appel à un jeu de fonction standard

catégorie (x) qui donne la catégorie syntaxique du paramètre si celui-ci est un mot.

gouv (x) qui donne le mot gouvernant la structure x

dep (x) donne la branche de l'arbre x si celle-ci est unique.

concept (x) s'applique à une structure et donne comme résultat le descripteur sémantique du sommet du réseau représentant cette structure si ce réseau existe.

Ici le premier choix effectué est celui de la variable AGANI, qui doit être non vide et pour laquelle on recherche

concept (x) = 'entité'

La structure n'a pas encore été évaluée. La fonction concept relance alors la fonction analyse avec cette fois le paramètre "je". Cette fois la structure n'a pas de dépendants et il n'est pas nécessaire de déterminer les relations de cas. La fonction analyse procède donc directement à l'évaluation sémantique par la procédure évalsem . Cette fonction a deux paramètres, une fonction sémantique et le contexte dans lequel doit s'évaluer cette fonction. Ici, le contexte est vide, la fonction sémantique associée à je dans le dictionnaire donne 'entité'.

Le résultat de analyse est le noeud du réseau.

| je | 'entité' |

et le prédicat associé à AGANI est vérifié et l'algorithme construit la règle AGANI → je et poursuit la construction de cette substitution avec [(ABOUT [OBJET])] pour lesquels on construit d'abord :

ABOUT → ω
OBJE → ω
} vérifiant le prédicat

Ce qui amène à DATIF → ω et au résultat fail de la fonction d'élagage. On examine alors la construction de DATIF → lui qui vérifie le prédicat catégorie (x) = popd et amène la construction de

| lui | 'entité' |

Mais la procédure d'élagage amène de nouveau fail (la phrase n'est pas finie). On reconsidère alors le choix de ABOUT. Mais on ne peut jamais vérifier catégorie (x) = about et il est impossible de construire cette règle.

La fonction trouvegram échoue donc totalement et redonne le contrôle a matchframe qui reprend la recherche d'une substitution en proposant cette fois l'arbre modèle

(parler [AGANI] [DATIF] [(ABOUT [OBJET])])

Comme dans la première phrase on évalue et construit

AGANI → je et | je | 'entité' |

DATIF → lui | lui | 'entité' |

Puis une première tentative ABOUT → ω amenant fail on construit :

ABOUT → de pour lequel le prédicat est vérifié

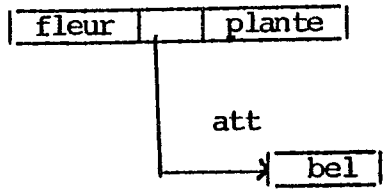
et

OBJET → (fleur bel)

qu'on est amené à évaluer en appelant de nouveau analyse.

En présence d'un substantif, la fonction teste la présence d'adjectif, trouve en l'occurence "bel" et construit le contexte (att bel) dans lequel elle évalue la fonction associée à "fleur" qui est invariant : 'plante'.

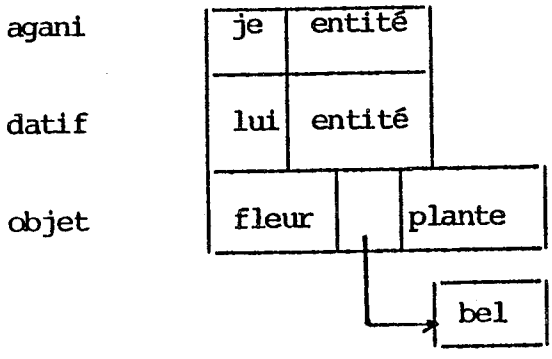
Un nouveau noeud est donc construit



et le prédicat objet est vérifié.

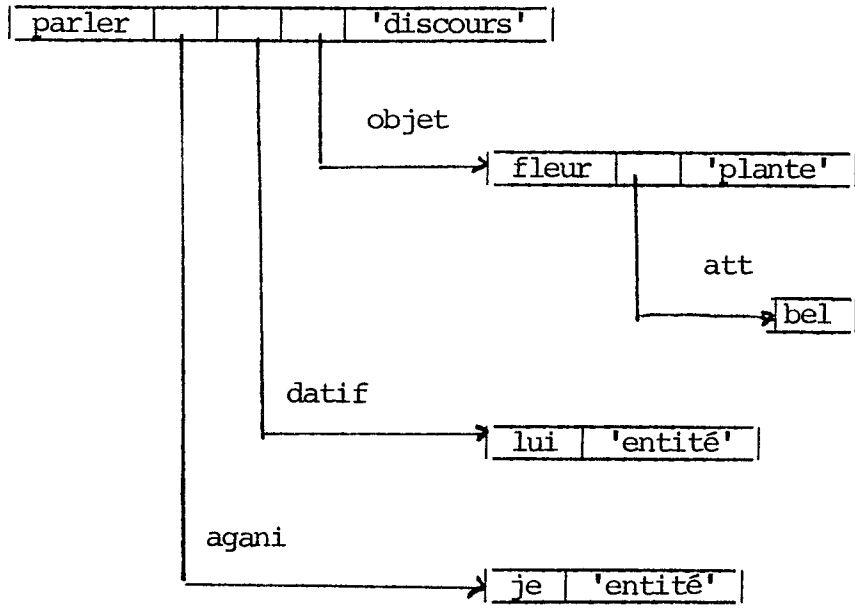
Cette fois la fonction trouvegram a obtenu un résultat qu'elle communique à matchframe qui elle récupère les noeuds construits pour les renvoyer à trouvecas.

Trouvecas à son tour remet à analyse les résultats obtenus, à savoir :



Ceci constitue le contexte dans lequel s'évalue la fonction sémantique de "parler" également invariante et référant à 'discours'.

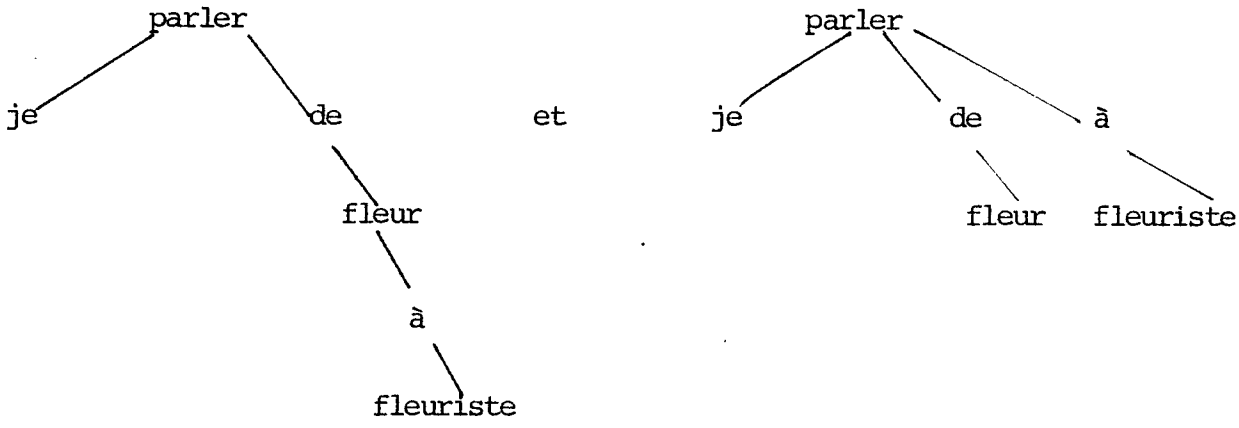
Le résultat final est donc le réseau



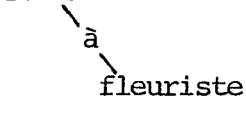
Examinons maintenant l'analyse de :

Je parle de ses fleurs à la fleuriste

On obtient ici deux structures syntaxiques possibles :



La première structure amène la construction de ABOUT → de et l'analyse de



Comme on ne trouve pas de modèle correspondant à cette construction pour fleur dans le dictionnaire, la prodécure analyse ne peut pas construire de réseau correspondant à cette structure. Le prédicat OBJET ne peut pas être vérifié puisque concept (x) = nul et cette voie échoue.

La deuxième possibilité est la construction de

ABOUT → ω, OBJET → ω et
DATIF →(de (fleur (à fleuriste)))

Mais alors on ne peut vérifier ni

catégorie (x) = popd ni catégorie (gouv (x)) = at

et cette construction échoue.

Comme aucune autre construction n'est possible cette structure est rejetée.

On passe alors à l'examen de la seconde structure.
Cette fois les règles ABOUT → de et OBJET → fleur autorisent l'évaluation de

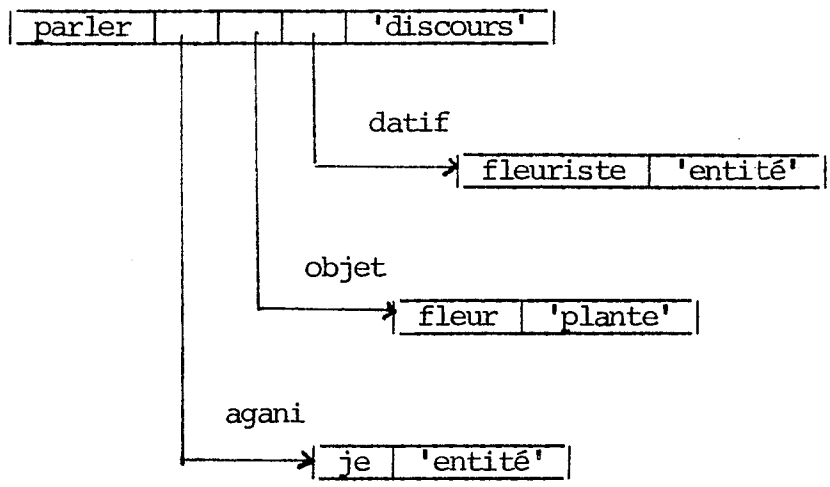
| fleur | plante |

Puis construisant DATIF → (à fleuriste) on vérifie cette fois
catégorie (gouv (x)) = at et on évalue

| fleuriste | entité |

et le prédicat associé à DATIF est vérifié.

On aboutit finalement au résultat :



L'évaluation sémantique a donc levé l'ambiguité syntaxique.

En résumé, les principales fonctions du système sont :

- analyse : Interpréteur de l'énoncé, qui oriente le traitement selon les critères linguistiques et construit le réseau.

- trouvecas : recherche les relations de cas existant dans une structure.

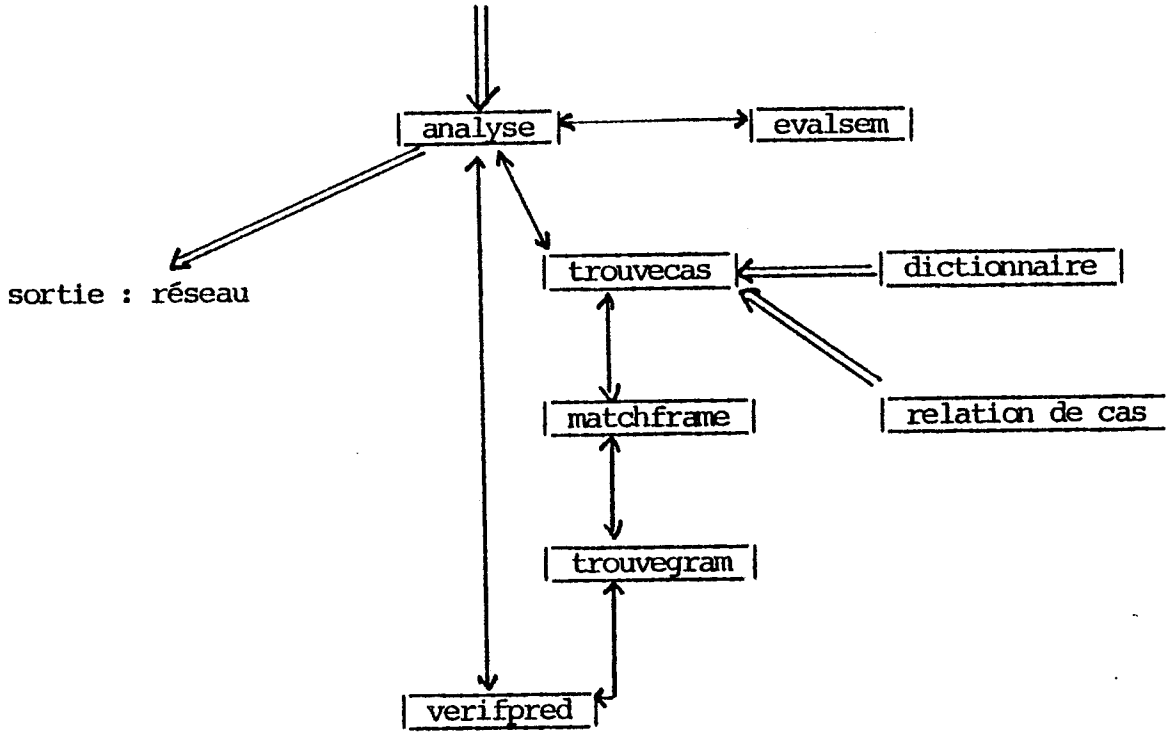
- matchframe : dont le rôle est d'envisager les différentes combinaisons possibles pour construire un arbre modèle et sa conditionnelle associée et qui amène la construction des relations de cas.

- trouvegram : algorithme décrit dans la première partie qui produit les grammaires assurant la validité d'une relation de cas.

- verifpred : qui n'a pas encore été évoquée. C'est la fonction qui est chargée d'assurer la vérification des prédicats associés pendant la construction de la grammaire. C'est cette fonction qui peut provoquer une nouvelle évaluation d'une sous-structure et qui doit donc organiser la gestion des portions de réseaux déjà construites.

- évalsem : qui évalue les fonctions sémantiques dans le contexte qu'on lui donne, qui est donc l'interpréteur du langage symbolique dans lequel ces fonctions sont exprimées.

entrée : structure de dépendance



4. .3 - IMPLEMENTATION

Elle s'est effectuée sur l'ordinateur IBM 360/67 du Centre de Calcul Interuniversitaire de Grenoble.

4. .3.1. - Implémentation de l'algorithme d'unification

Elle apparait comme la réalisation d'une technique de filtrage désormais classique dans les langages d'Intelligence Artificielle [Gre] [Rul] [PLAN].

Un arbre modèle apparait comme un patron dans lequel les variables (de substitution) sont marquées par un préfixe

L'arbre modèle (X [(a b [Y])] [c] [(U [z])])

peut se noter (? x (a b & y) c (↓u z))

Une procédure de décodage adéquate se charge de produire l'arbre modèle correspondant.

Les différents préfixes ont pour effet d'imposer une condition préalable sur cette variable, c'est-à-dire d'ajouter un élément au prédicat de cette variable dans la conditionnelle associée.

Ils peuvent indiquer par exemple qu'à cette variable ne peut être associée la règle vide ou tout autre condition prévue par l'utilisateur. Ces préfixes attribuent en quelque sorte un type aux variables qu'ils désignent [Bob1].

Chaque variable joue ici un double rôle puisqu'elle apparait à la fois comme une ramification auquelle elle se substitue et comme un prédicat associé dans la conditionnelle. Elles correspondent à la notion "d'acteur" (ou Kappa-expression) dans la définition de PLANNER .[PLAN].

Pour assurer le contrôle dans la production des substitutions, une structure a été définie par un mécanisme identique à l'implémentation d'algorithmes non déterministes [Mo-Pa-Tu], la fonction de création des règles apparaissant comme la fonction de choix dans l'algorithme d'unification sous forme non déterministe.

Tous les programmes ont été écrit en LISP [Mac]

4. .3.2. - Réalisation du modèle global PIAF-LISP

Afin de concilier PIAF et LISP et de rester cohérent avec la philosophie générale des modèles parallèles un module global a été défini, autorisant le fonctionnement indépendant de PIAF et LISP en pseudo-parallélisme et l'échange de messages, c'est-à-dire que l'on peut quitter un des systèmes pour relancer l'autre et réciproquement, tout en lui fournissant un message, transitant par un interface.

Le système (R) LISP local [Lux1] a été conçu pour autoriser diverses configurations d'entrée-sortie aux utilisateurs tout en conservant les fonctions standard LISP. A cet effet, on aiguille ces fonctions sur des procédures d'acquisition production qui sont des paramètres du système. Cette méthode s'avère particulièrement adéquate dans le cas présent et pour les problèmes de transport [Lux2].

Cela évite de figer le moyen de communication et surtout de fixer à priori une zone tampon. On résout ainsi les problèmes d'adressage qui sont à la seule responsabilité des procédures d'acquisition production. Bien que cohabitant en mémoire les deux systèmes sont totalement ignorants l'un de l'autre et la modification de l'un n'a pas de répercussion sur l'autre.

Détail du fonctionnement

Deux fonctions ont été ajoutées au système LISP standard [Mac]

- RDPIAF( ) qui joue un rôle analogue à la fonction READ( ) mais aiguille la lecture de la chaîne d'entrée par la fonction RDOBJ( ) vers une procédure d'acquisition qui a été donnée à l'initialisation.

- PIAF( ) qui provoque le vidage du tampon de sortie RLISP par la fonction de production également donnée à l'initialisation, puis rend le contrôle à l'interface.

Le fonctionnement est alors le suivant :

A l'initialisation du système deux modes sont possibles : pas à pas ou continu. En mode pas à pas, on rend le contrôle à l'utilisateur après chaque production d'une structure de dépendance qui peut alors choisir de multiples solutions (appel LISP, sortie, acquisition d'un nouvel énoncé, modification de l'énoncé en cours, etc...). En mode continu on donne systématiquement le contrôle à l'interface après chaque production d'une structure de dépendance en lui signalant l'adresse de cette structure.

L'interface signale alors la procédure d'acquisition cette adresse, sauvegarde l'état courant PIAF et restaure l'état courant LISP. Si cette restauration est impossible, c'est-à-dire si cette exécution de LISP est la première, on effectue une exécution du ramasse miettes et on signale l'adresse des procédures d'acquisition et d'exécution, créant par la même un état courant. On relance alors cet état courant.

Deux solutions sont alors possibles lorsqu'on revient à l'interface par utilisation de PIAF( ).

- La fonction de production a trouvé un tampon de sortie vide. Il n'y a donc pas de message à faire passer. L'interface sauvegarde alors l'état courant LISP, restaure l'état courant PIAF et relance l'exécution.

- La fonction de production actionnée par PIAF( ) a trouvé un message. L'interface force alors PIAF à l'état attente de commande en signalant le message à l'utilisateur qui reste ainsi maître du système.

CHAPITRE - V

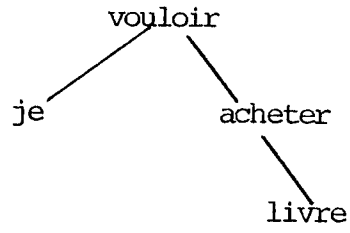
EXEMPLES - APPLICATIONS

5.1 - VERBES DE MODALITES

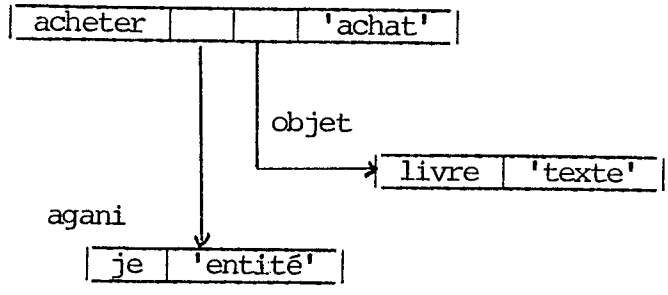
Il s'agit de verbes apportant une modalité à un autre verbe, utilisés dans une construction infinitive (vouloir, pouvoir, désirer, penser, espérer, etc...).

Ex : Je veux acheter un livre.

ou



L'analyse d'une telle structure amène simplement l'évaluation de la branche infinitive comme s'il s'agissait d'une forme active de ce verbe, en plaçant le sujet du berbe de modalité comme sujet de l'infinitif. Tout se déroule donc comme pour l'analyse de : j'achète un livre.
On rajoute simplement une variable correspondant à la modalité.
Ici on obtiendra le réseau



et on trouvera pour le verbe acheter

acheter : VERB PERS = UN MODE = IND TPS = PRES
MODAL = VOULOIR

5.2 - DISTINCTION ENTRE DEUX SENS D'UN MEME MOT

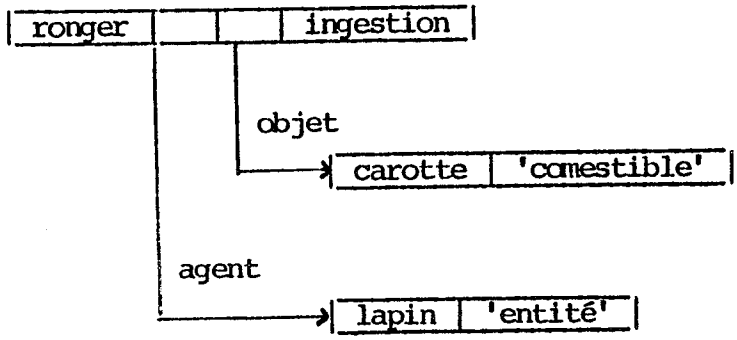
Reprenons l'exemple :

Le lapin ronge la carotte
l'acide ronge le métal

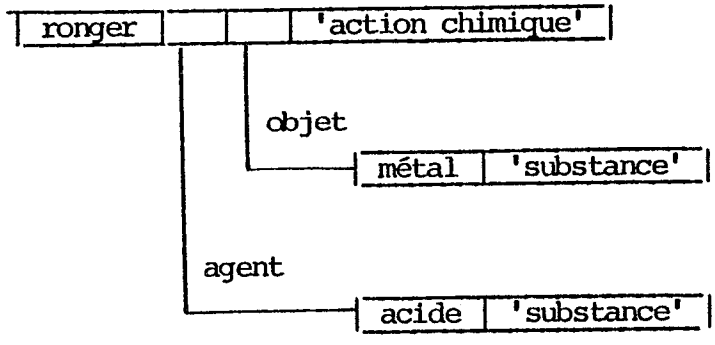
Dans le premier pas on construit les relations de cas

(agent (lapin 'animal'))
et (objet (carotte 'comestible'))

L'évaluation de la fonction sémantique de ronger (cf. § 4.1.3.2.) produit le concept 'ingestion' d'où le réseau :



Dans le deuxième cas, la fonction sémantique produit le concept 'action chimique' d'où un autre réseau



Si l'on avait eu la phrase le métal ronge le lapin, la fonction sémantique n'aurait pas pu s'évaluer et la phrase eût été rejetée.

5.3 - INTEGRATION DE CONCEPTS

Pour une application donnée certains mots n'expriment par eux-mêmes qu'un fragment d'un descripteur plus spécifique précisé par le paradigme dans lequel ils sont utilisés. Dans un corpus documentaire (cf. 5.7) on parlera par exemple du statut des chercheurs, du statut de Paris, du statut du fermage, etc. Le terme statut n'y a de sens que s'il est associé à un autre. C'est le cas typique où un descripteur d'une structure est une composition des autres descripteurs.

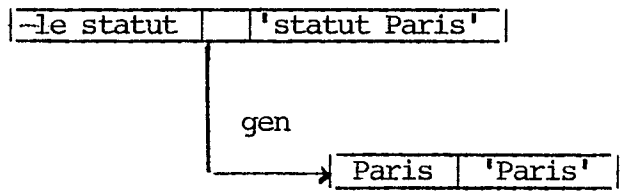
Cette composition est obtenue grâce à la fonction sémantique de statut par composition des descripteurs au moyen de la fonction mkconcept.

ex : statut : fsem (gen) ; mkconcept ('statut', gen).

Pour l'expression le statut de Paris, on évalue dans un premier temps :

gen : | Paris | 'Paris' |

puis la fonction de concept associée à statut d'où



De la même façon le verbe étendre dans ce corpus a toujours le sens de généralisation de l'objet sur lequel porte le verbe.
Et on trouvera dans le dictionnaire

étendre : fsem (objet) ; mkconcept ('généralisation', objet) ;

5.4 - TRANSFERT DE CONCEPT

D'autres mots du corpus sont des mots "outils". Ils ne font référence à aucun concept dans le cadre visé. Ils ne servent alors qu'à transférer les descripteurs de la structure qu'ils gouvernent, lequel transfert s'effectue toujours par le moyen de la fonction sémantique, qui à leur tour peuvent être transférer ou composés.

ex : bénéfice : fsem (objet) ; objet.
vente : fsem (objet) ; objet.

"Le gouvernement étend le bénéfice de l'assurance maladie aux prêtres"

La structure "le bénéfice de l'assurance maladie" s'interprète

```
(objet (bénéfice (objet (assurance maladie) 'sécurité sociale')
       'sécurité sociale')
```

et on obtient pour la phrase complète

```
(étendre
        (agent (le gouvernement) 'entité)
        (objet (le bénéfice
                        (objet (l'assurance maladie)
                        'sécurité sociale')
                'sécurité sociale'))
        (datif (aux prêtres) 'église')
'généralisation sécurité sociale').
```

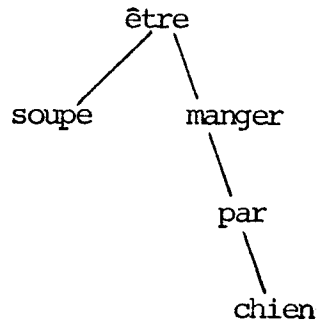
5.5 - FORME PASSIVE

Comme dans une grammaire transformationnelle, les deux formes d'une même phrase se ramènent à la même structure.

En présence d'une structure gouvernée par le verbe être, l'interpréteur teste la présence d'une forme passive ou d'une forme attributive.

Dans le premier cas, on lance la fonction trouvecas sur la branche dépendante droite

ex : La soupe est mangée par le chien



Considérant l'information associée au dictionnaire, trouvecas construit la structure à rechercher et les prédicats associés

(manger &agent &causal)

prédicat agent (x)
  x = ω ou catégorie (gouv (x)) = by
        et trait (dep (x)) = concret
prédicat causal (x)
  x = nul ou catégorie (gouv (x)) = by et trait (dept(x)) = abstrait

Il en résulte ici la création de

(agent → (par chien)
(agent (chien entité))

L'évaluation de la branche gauche donne

objet ⟶ (soupe)
(objet (soupe comestible))

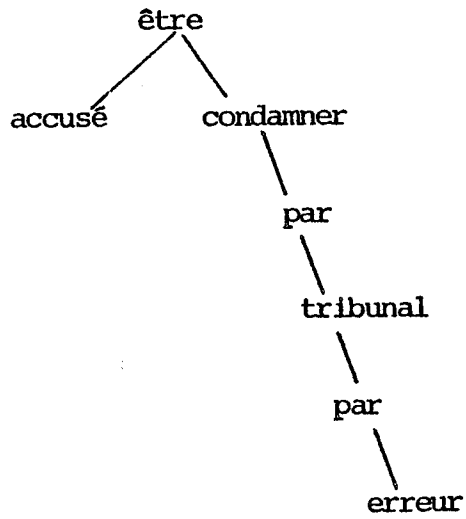
Et on obtient finalement le même réseau que pour la phrase

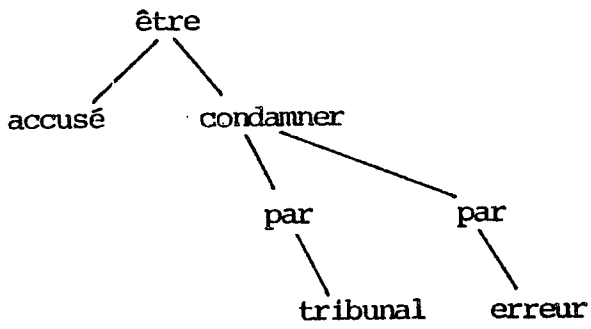
Le chien mange la soupe

Exemple 2

"L'accusé est condamné par le tribunal par erreur".

Il y a 2 structures de dépendance possibles





La première structure est rejetée parce qu'on ne peut pas évaluer

"tribunal par erreur"

Pour la seconde on produit par interprétation de la branche droite

```
(agent (tribunal 'entité'))
(causal (erreur 'erreur))
```

puis après évaluation de la branche gauche et de la fonction sémantique

```
(condamner (agent (tribunal 'entité')
           (causal (erreur 'erreur')
           (objet (accusé 'entité')
'justice')
```

5.6 - UTILISATION DANS LE CADRE D'UN EXPERIENCE D'ENSEIGNEMENT ASSISTE

Cette expérience a été menée en collaboration avec l'équipe d'enseignement assisté par ordinateur de l'Université Scientifique de Grenoble [EAO]. Il s'agit de faire découvrir à des élèves les lois physique gouvernant le phénomène des chocs sans frottement. Dans ce but, ils disposent d'un terminal graphique sur lequel ils peuvent simuler un choc entre deux palets cylindriques dans les conditions expérimentales qu'ils désirent. Ils peuvent ensuite poser des questions, demander des résultats à la machine, en langue naturelle.

Le but est donc ici de produire une représentation des énoncés qui soit directement assimilable par un interpréteur qui donne la réponse.

5.6.1. - Définition du système

5.6.1.1. - Les descripteurs

Ils sont évidemment en nombre limité, le domaine sémantique se limitant aux objets mathématiques manipulés et à leur représentation. On trouve les descripteurs

'force', 'vecteur', 'réel', 'point', 'palet', 'image', etc...

5.6.1.2. Les relations de cas

Les relations de cas dans ces énoncés traités sont également peu nombreuses et limitées aux substantifs, toutes les phrases étant nominales.

On distingue :

- le génitif

  prédicat gen (x) ;
  catégorie (gouv (x)) = of et concept (dep (x)) ≠ nul ;

- le situatif locatif

  prédicat loc (x) ;
  catégorie (gouv (x)) = on et concept (dep (x)) ≠ nul

- le situatif temporel

  prédicat tps (x) ;
  catégorie (gouv (x)) = temp et concept (dep(x)) ≠ nul

- deux relations de cas permettant de régler les problèmes de coordination, coco et cocogen

  prédicat coco(x) ;
  gouv (x) = "et" et concept (depg (x)) = concept (det d(x))

Ce prédicat exige que les expressions coordonnées par un et expriment le même concept.
Pour les structures de la forme "de... et de ..." existe le même prédicat

5.6.1.3. - Les modèles casuels

Comme pour l'analyse morphologique, les modèles sont repérés par une étiquette qui est un représentant de la classe.

trajectoire : (? gen)

angle : (entre ? coco)

somme : (? cocogen)

vitesse : (? gen ? tps)

projection : (? gen ? loc)

choc : nil

Le dernier modèle est celui des mots qui ne peuvent pas avoir de dépendants.

5.6.1.4. - Le dictionnaire

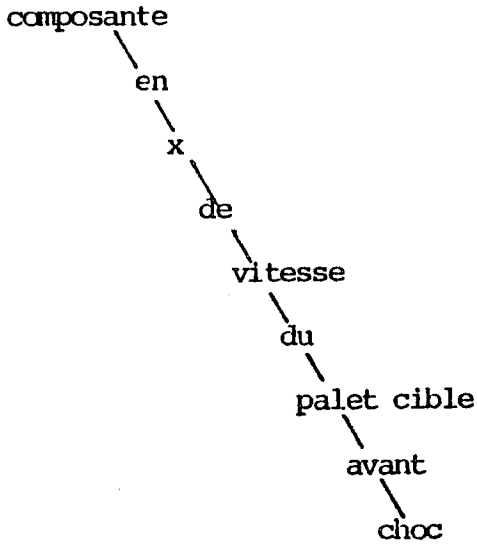
Il comporte au plus une centaine de mots dont

/ angle / distance / fsem (coco) ; si coco = 'vecteur' alors 'réel'.
/ axe Ox / chox / fsem ; 'vecteur' .
/ choc / choc / fsem ; 'force' .
/ composante / projection / fsem (gen) ; si gen = 'vecteur' alors 'réel'.
/ écran / choc / fsem ; 'image'.
/ norme / trajectoire / fsem (gen) ; si gen = 'vecteur' alors 'réel'.
/ palet cible / choc / fsem ; 'palet'.
/ palet lancé / choc / fsem ; 'palet'.
/ somme / somme / fsem (cocogen) ; si cocogen = 'vecteur' alors 'vecteur'
sinon si cocogen = 'réel' alors 'réel'.

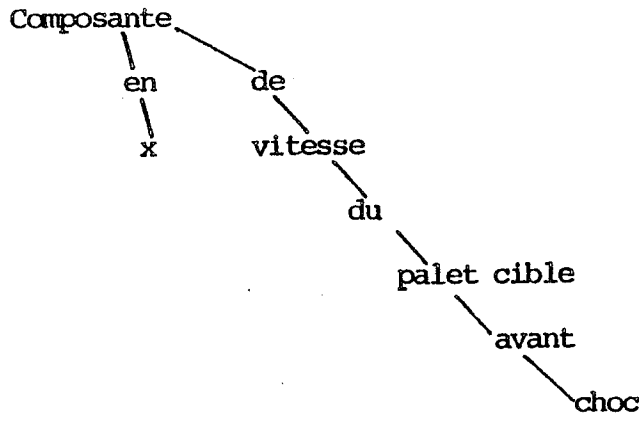
5.6.2. - Exemples

"Composantes en x de la vitesse du palet cible avant le choc".

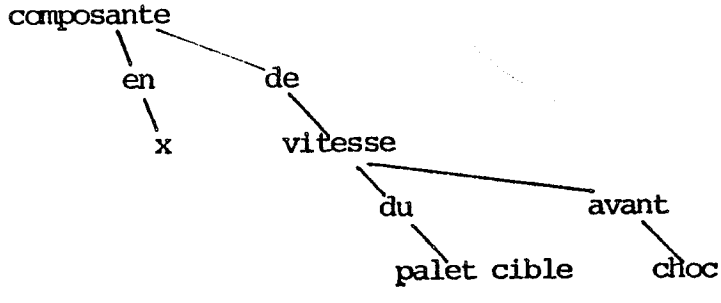
Il existe 6 structures de dépendance possibles



(1)

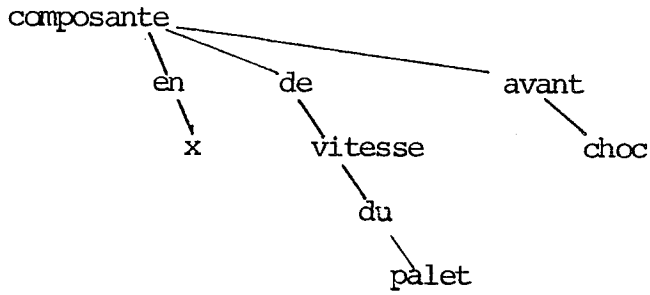


(2)

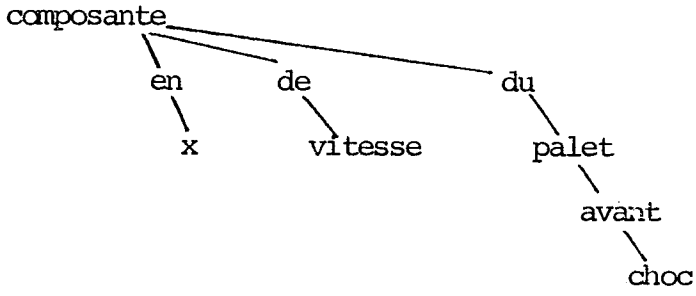


(3)

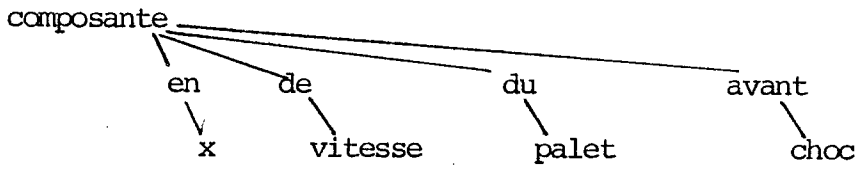
(4)



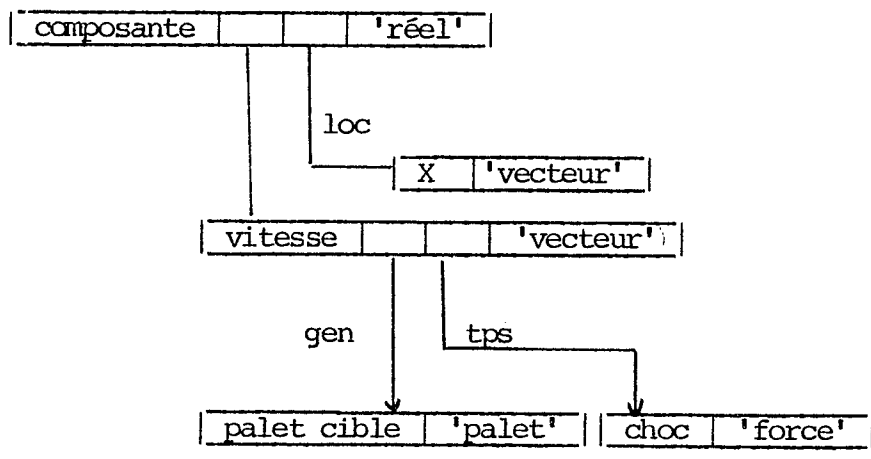
(5)



(6)

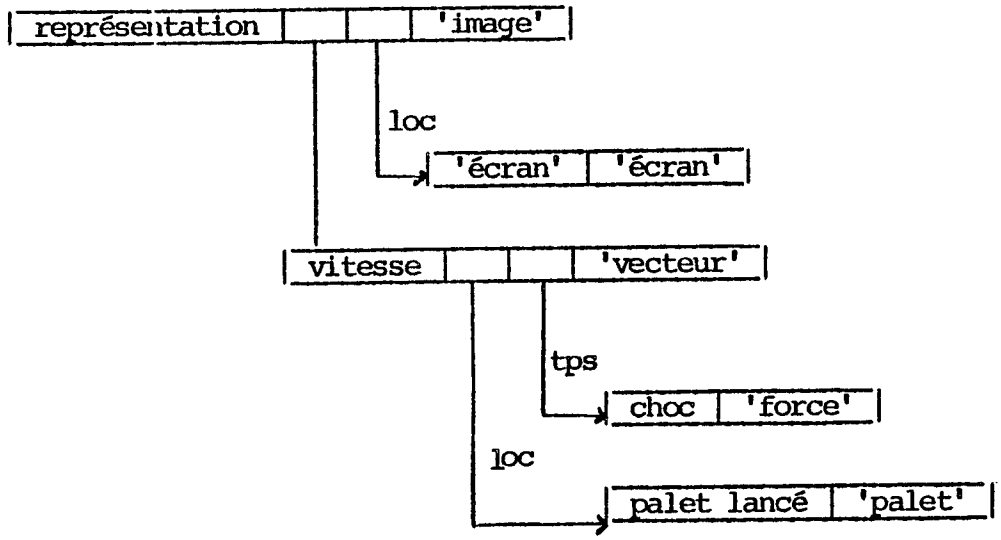


Pour les structures (1) (2) et (5) le modèle de palet ne peut pas être respecté et on ne peut pas interpréter ces structures.
Il en va de même pour (4) et (6) qui ne respectent pas le modèle associé à composantes.
Il n'y a donc que pour (3) que l'on construit un réseau



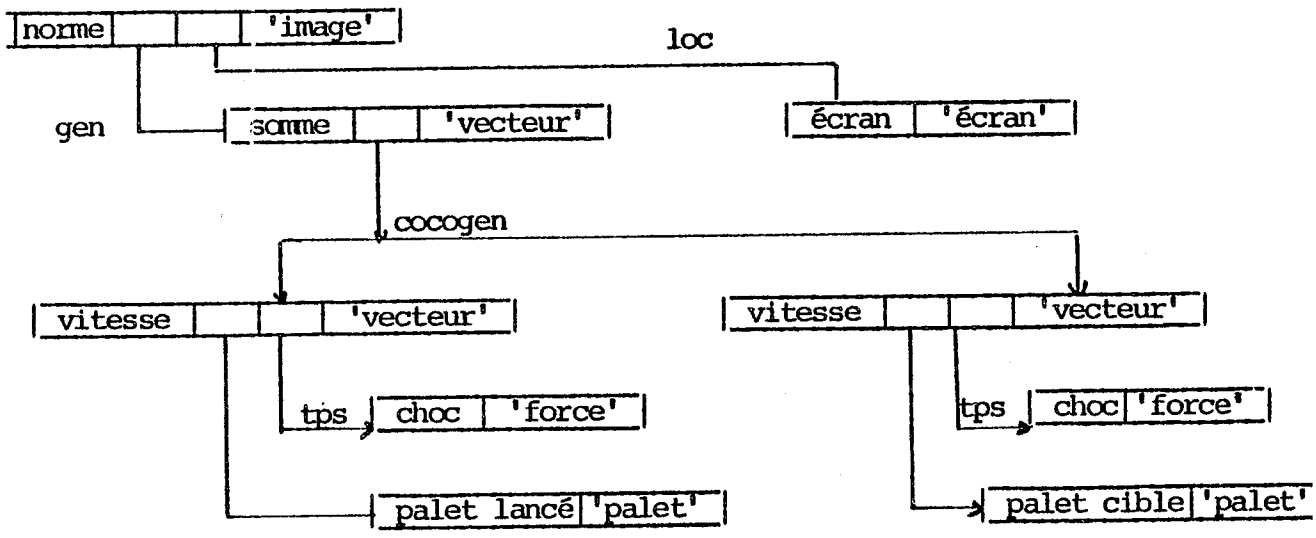
"Représentation sur l'écran de la vitesse du palet lancé avant le choc".

On a cette fois également 6 structures dont une seule est validée



"Norme de la somme de la vitesse du palet lancé avant le choc et de la vitesse du palet cible après le choc".

Il y a alors structures dont une aboutie à



5.7 - UNE APPLICATION A LA DOCUMENTATION AUTOMATIQUE

Les investigations menées à ce sujet se sont effectuées autour du projet PIAFDOC [PFD]. Les expériences ont été réalisées sur une base documentaire du Centre de Documentation Française. La base concernée est une chronologie des évènements politiques qui ont eu lieu en France, rédigée en langue naturelle. Dans le système actuel le public désirant interroger la base communique ses requêtes aux documentalistes qui effectuent la recherche. Les propositions faites ici visent à une interrogation en langue naturelle conversationnelle effectuée directement par l'utilisateur.

5.7.1. - Rappel des notions documentaires

Une base documentaire est un ensemble de documents caractérisé par des champs (numéro, titre, date, etc...). Dans la base qui nous intéresse l'un de ces champs est un texte en langue naturelle. Ce texte est indexé par des mots clés. A la base est donc associée un lexique ou dictionnaire des mots clés et un thésaurus dont on dispose à l'interrogation.

Le thésaurus est un ensemble de descripteurs reliés en général par des relations hiérarchiques, associative, d'équivalence [Mo]. Chaque descripteur exprime un concept on un ensemble de concept auquel est rattaché une classe d'entrées du lexique. La relation hiérarchique est la relation de spécificité (et inversement générique) exprimant une notion d'inclusion de concept "coude" est spécifique de "bras" qui est spécifique de "membre" etc...

La relation associative est une relation de voisinage entre les descripteurs parfois appelée relation "voir - aussi".

exemple : emploi "voir aussi" chomage

La relation d'équivalence indique la synonymie ou l'identité entre deux termes.

L'utilisation du thésaurus permet une amélioration de la qualité des résultats qui se mesurent par :

- le silence : c'est l'ensemble des documents répondant à la question posée mais qui n'ont pas été retirés de la base à l'interrogation. On mesure le taux de silence par un coefficient appelé le rappel.

- le bruit : c'est l'ensemble des documents retirés qui ne sont pas pertinent. Le taux de bruit est mesuré par la précision.

Un certain nombre de systèmes informatiques ont été commercialisés [MIST] [STA] basés sur un traitement très primitif des textes en tant que chaîne de caractères sans aucune information linguistique et utilisant le même principe d'interrogation :

- une première recherche primaire avec les mots clés obtenus à partir de la question, via le thésaurus, ce qui produit un certain nombre de documents par consultation d'un fichier inverse.

- une seconde recherche sur ces documents par des critères d'éloignement (on exige par exemple que deux mots clés soient séparés au plus par n mots).

Mais il s'avère que ces systèmes ne fournissent pas des résultats très satisfaisants (taux de rappel et de précision dépassant rarement 60 %) même en utilisant des méthodes très sophistiquées utilisant des modèles statistiques [Sal]. Et les spécialistes semblent d'accord pour introduire des techniques linguistiques, permettant en outre une réduction des dictionnaires et un contrôle de saisie, des recherches ayant d'ailleurs déjà été accomplies dans cette voie [Gro2] [Bor].

C'est la seconde phase que nous suggérons de remplacer par des critères d'analyse linguistique.

On supposera que la saisie des documents s'est effectuée par un système tel que PIAFDOC qui effectue une indexation automatique des documents en indexant avec une forme standard de mot-clés (ce qui réduit considérablement

le dictionnaire, les systèmes classiques ayant par exemple autant de mots clés dans le lexique que de conjugaisons pour un même verbe ou de formes pour un substantif).

5.7.2. - Propositions pour une nouvelle stratégie d'interrogation

La méthode que nous proposons travaille sur les textes déjà retirés par un critère primaire. Elle n'a donc aucun impact sur le rappel et ne peut qu'influencer la précision. Une augmentation du rappel par une méthode linguistique nécessiterait soit une analyse de toute la base à chaque question, technique impensable sur le plan du coût, soit une analyse à la saisie des documents comme dans les systèmes de question-résonse.

Mais cette méthode ne peut être envisagée dans le cadre de la documentation pour plusieurs raisons :

- on est de toute façon obligé de conserver le texte original, certaines interrogation se faisant sur des termes précis ou des citations. Une stratégie effectuant l'analyse à la saisie entraîne le stockage à la fois du texte et de sa représentation d'où une utilisation de ressources en double emploi.

- Une représentation interne du texte est le fruit d'une interprétation à travers une grille. Cette grille n'est pas forcément celle voulue par l'utilisateur et surtout cette grille évolue avec le temps. Les textes analysés à une certaine date ne possèderont pas la représentation voulue par la suite en raison de la modification ou de la création de certains concepts. Or la seule grille intéressante est évidemment celle valide au moment de la questio

Considérons par exemple le premier document introduit dans la base parlant "d'une voie ferroviaire souterraine reliant l'Angleterre au continent". A cette date le concept "tunnel sous la manche" n'existe pas et on ne peut simplement pas indexer ce texte par un concept connu. Par la suite les média créent l'expression et "tunnel sous la manche" devient un concept connu. Mais jamais on ne retrouvera le premier document introduit en interrogeant sur ce descripteur.

La seule méthode d'augmentation du rappel reste dans le domaine du raffinement des thésaurus [Mo-Gi], aussi se concentrera-t-on ici sur une augmentation de la précision par une méthode en trois temps :

- Analyser la question et extraire à l'aide du thésaurus les mots clés nécessaires de façon conversationnelle avec l'utilisateur.

- Extraire les documents susceptibles d'être intéressant par une méthode classique.

- Rechercher parmi ceux-ci ceux qui correspondent à la question. Pour ne pas rejeter les documents incertains mais finalement pertinents on adopte comme critère le rejet des documents qui ne répondent pas à la question.

5.7.3. - Organisation du système - Exemples

L'interpréteur sert évidemment ici à l'analyse des textes sélectionnés. Pour la réalisation de l'outil

- le graphe des descripteurs est constitué directement par le thésaurus
- le dictionnaire et les relations de cas sont à définir par le documentaliste.

A partir de la question, on obtient, via le thésaurus une expression booléenne de mots clés qui sert à la consultation du fichier inverse, et une structure modèle de descripteurs qui est celle recherchée. Après confirmation de l'utilisateur on recherche parmi les documents retenus ceux qui correspondent à cette structure modèle.

Exemple 1 : Recherche sur un seul descripteur

La question posée est "la crise agricole"

L'analyse de la question produit une interrogation sur:

(crise ou difficulté ou problème)
et (agriculture ou agricole ou agriculteur ou horticulture ou élevage ou lait ou beurre)

et le descripteur 'crise agricole' auquel se limite la recherche.

Document retiré :

M. Chirac espère trouver une solution aux difficultés de l'élevage.

qui est interprétée sous la forme :

```
(trouver
        (agent (M. Chirac 'entité'))
        (objet (solution
                        (objet (difficultés (gen (élevage 'agricole'))
                                'crise agricole')
        'solution'))
'solution')
```

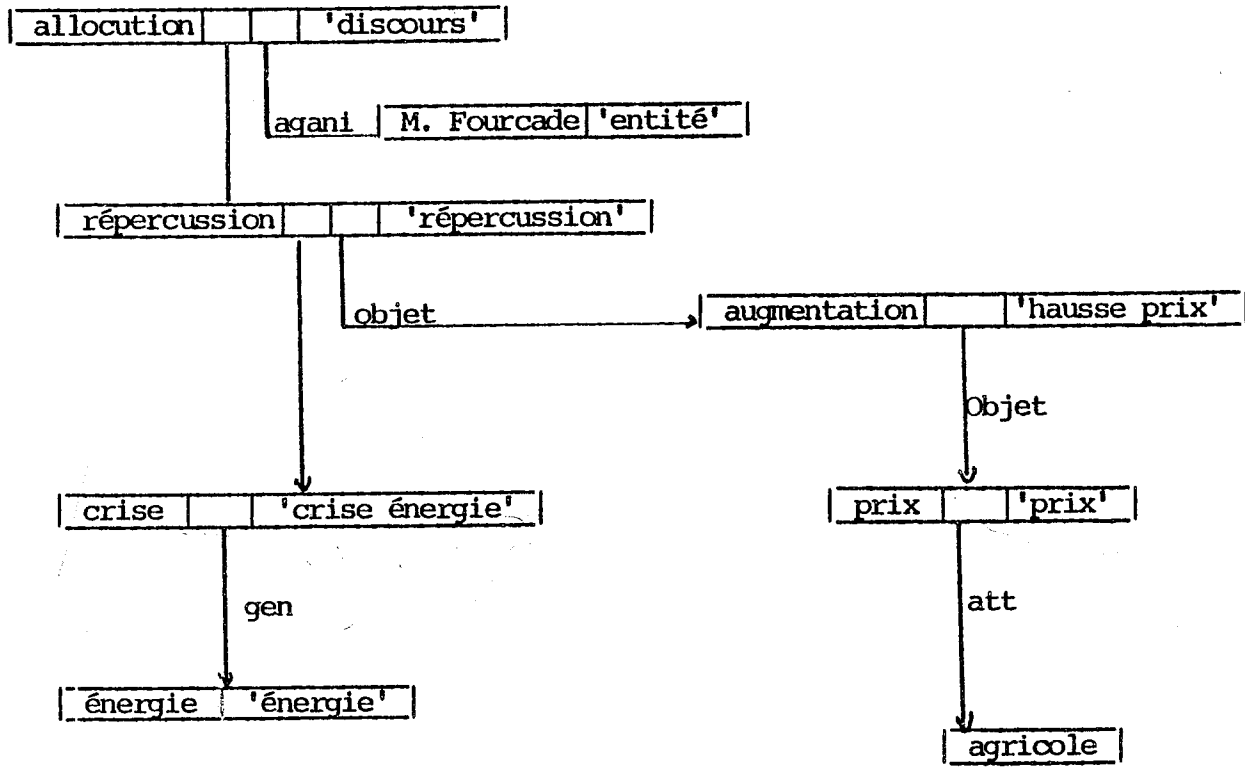
Le descripteur crise agricole est obtenu par composition de descripteurs

Le descripteur recherché est trouvé et le document est conservé

Document retiré

Allocution de M. Fourcade sur les répercussions de la crise de l'énergie sur l'augmentation des prix agricoles.

Le réseau obtenu est cette fois



Le descripteur recherché ne figure plus dans l'interprétation et ce document est rejeté.

Exemple 2

La question "le statut de Paris"
provoque la recherche sur

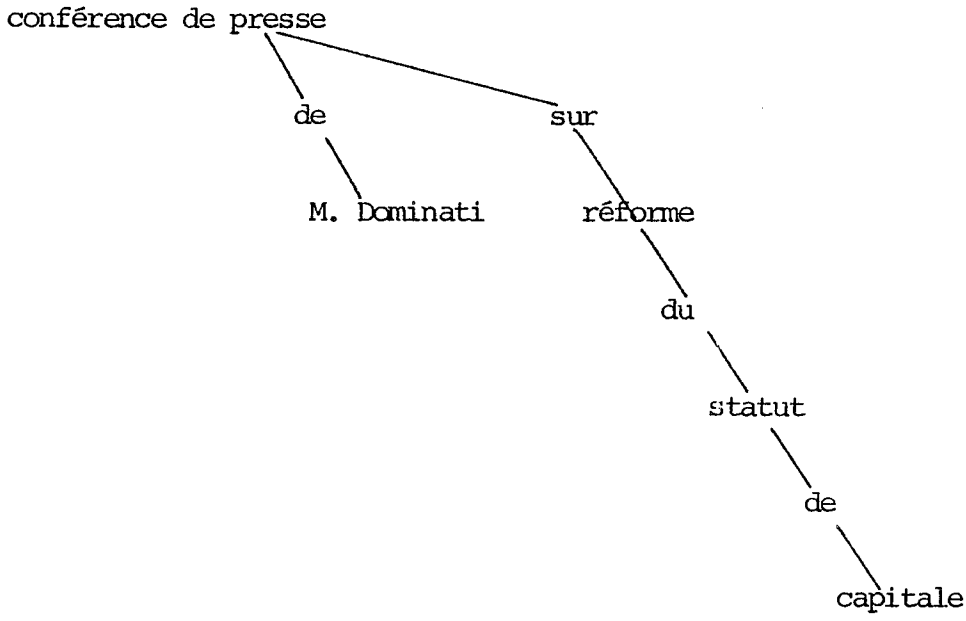
statut et (Paris ou capitale)

et sur le descripteur 'statut Paris'

Document retiré

Conférence de presse de M. Dominati sur la réforme du statut de la

Pour cette phrase on obtient la structure



Dans le dictionnaire on trouve associé à conférence de presse :

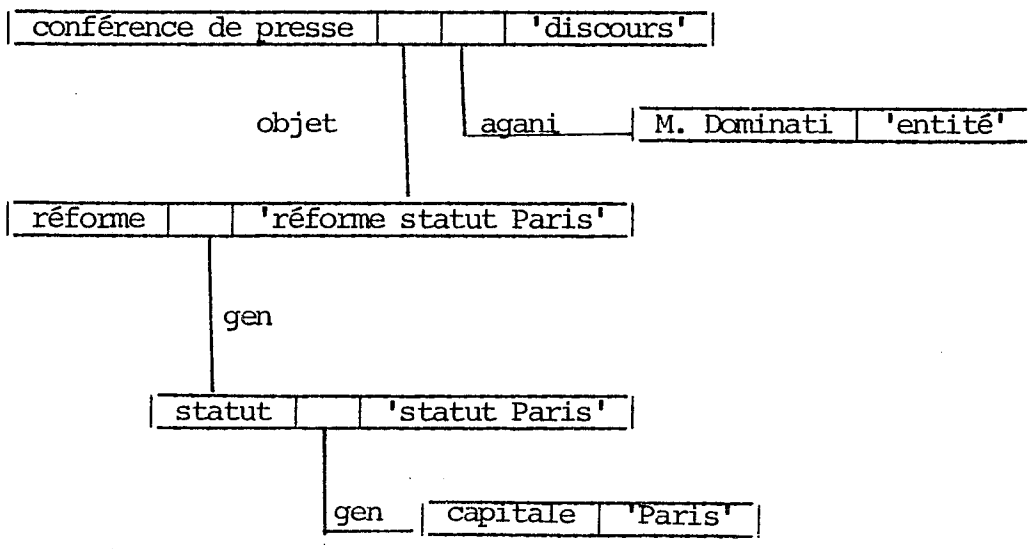
(& (AT SIT) ? (ØF AGANI) ? (ABOUT OBJET))

et

fsem ; 'discours'.

puis pour réforme : (& (ØF OBJET) & (BY AGANI))
fsem (objet) ; mkconcept ('réforme', objet).

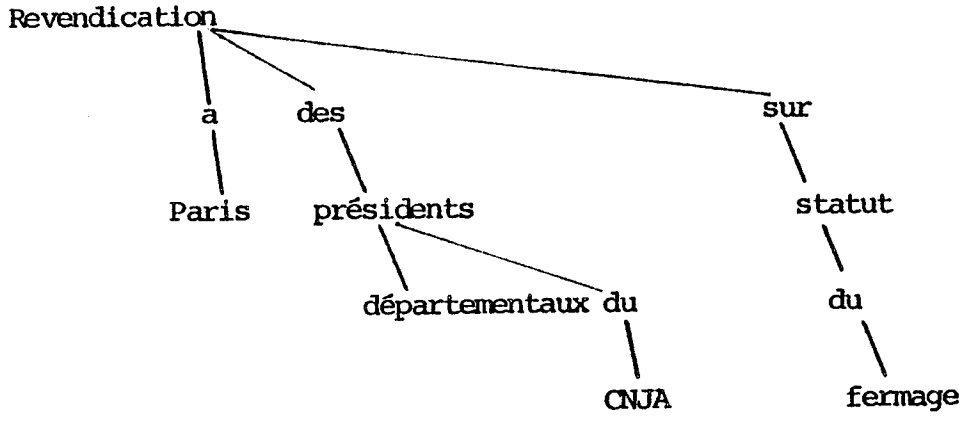
Ce qui aboutit à la construction de



Document retiré

"Revendication à Paris des présidents départementaux du CNJA sur le statut du fermage".

La structure syntaxique



qui est la seule validée ne produit évidemment pas le concept recherché pendant l'évaluation et est rejetée.

C O N C L U S I O N

L'utilisation de cette réalisation dans un cadre de documentation automatique ouvre un nouveau champ d'application au traitement automatique des langues, et permet de situer à la fois les limites et les ouvertures du système. La limite est celle de tout système de ce genre : quel que soient la méthode et l'outil proposés, il est nécessaire d'emmagasiner une grosse somme d'information pour le fonctionnement du système, laquelle information n'est pas toujours maîtrisée par l'utilisateur. Un développement de ces travaux vers un système à grande échelle impliquerait sans doute un contrôle de l'information et un travail de systématisation et de classification qui procède davantage de la linguistique que du champ informatique dans lequel nous nous situons. Mais on peut envisager de mettre à la disposition du linguiste un système d'apprentissage basé sur les techniques de pattern matching présentées ici.

Enfin il serait nécessaire sans doute d'introduire des notions sémantiques plus globales, à savoir des sortes de scénarii indiquant un schéma type d'évolution d'une situation, ainsi que les co-notations habituelles à cette situation.

Dans un cadre différent de la documentation, le système présenté doit pouvoir fournir des résultats opérationnels sans extension majeure. On peut envisager par exemple deux applications immédiates, la commande d'un robot industriel tel que celui défini par les chercheurs milanais avec qui nous avons déjà coopéré, ou l'interrogation en langue naturelle d'une base de données répartie. Cette dernière application semble la plus intéressante à la fois par son intérêt pratique et par sa réalisabilité.

B I B L I O G R A P H I E

Liste des abréviations utilisées :

A.J.C.L. American Journal of Computational Linguistics
C.A.C.M. Communication of the Association for Computing Machinery
I.C.C.L. International Conference for Computational Linguistics
I?J.C.A.I. International Joint Conference on Artificial Intelligence

---

BER BERGE C.
Théorie des graphes et ses applications.
Dunod 1963

Bob1 BOBROW D. & FRASER B.
An augmented state transition network analysis procedure
Proc. IJCAI Washington 1969.

Bob2 BOBROW D.
New programming languages for Artificial Intelligence research.
Artificial Intelligence, vol. 6, n° 3 1974.

Boi BOITET C.
Un essai de réponse à quelques questions théoriques et pratiques liées à la traduction automatique. Définition d'un système prototype.
Thèse d'Etat - Université de Grenoble 1976.

Bor BORILLO A.
Analyse de texte - Analyse linguistique dans Analyse et validation dans l'étude des données textuelle.
Editions du CNRS - 1977

Bru BRUCE B.
Case systems for natural language.
Artificial Intelligence, vol 6, 1975

Chau CHAUCHE J.
Transducteurs et arborescences. Etude et réalisation de systèmes appliqués aux grammaires transformationnelles.
Thèse d'Etat - Université de Grenoble 1974

Cho1 CHOMSKY N.
Syntactic Structures.
Mouton La Hague 1957

Cho2 CHOMSKY N.
Aspects of the theory of syntax
M.I.T. Press - Cambridge - Massachussetts 1965

Col1 COLBY K.M.
Pattern matching rules for the recognition of natural language dialogue expression.
AJCL - Microfiche 5. 1964

Col2 COLBY K.M.
Conversational language comprehension using integrated pattern matching and parsing.
Artificial Intelligence, vol 9, 1977

Co1 COURTIN J.
Un analyseur syntaxique interactif pour la communication homme-machine
Proc. ICCL Pise 1973

Co2 COURTIN J.
Utilisation des redondances pour l'analyse et le contrôle automatique des langues.
Proc. ICCL Ottawa 1976

Co3 COURTIN J.
Algorithmes pour le traitement interactif des langues naturelles.
Thèse d'Etat - Université de Grenoble 1977

Cul CULLINGFORD R.E.
The application of script based world knowledge in an integrated story understander system.
Proc. ICCL Ottawa 1976

Da-Bu DARLINGTON J. - BURSTALL R.
A system which automatically improves program.
Proc. IJCAI Stanford 1973

EAO FAFIOTTE G., FOUCHIER J., GAUCHE J., PAINVIN S.
Simulation sur ordinateur d'un phénomène physique, destinée à une expérimentation pédagogique.
Enseignement et informatique, n° 16, 1978

Fil FILLMORE
The case for case
dans Universals in Linguistics theory - Holt, Rinehart & Wilson, 1968

Har HARRIS Z.S.
Co occurence and transformation in linguistic structure
Language, n° 33, 1957

Hen HENDRIX G.G.
Partitioned networks for the mathematical modeling of natural language
Ph.D. Dissertation - Computer Science Department -
University of Texas - Austin 1975

Fisch FISCHER M.J.
Grammars with macro-like productions
Mathematical linguistics and automatic translation
Aiken Laboratory - Harvard University - report NSF 22 - 1968

Gin GINSBURG S., Partee B
A mathematical model of transformational grammars
Information and Control 15, 1969

Gra1 GRANDJEAN E.
Conception et réalisation d'un dictionnaire pour un analyseur interactif des langues naturelles.
Mémoire CNAM, Université de Grenoble 1975

Gra2 GRANDJEAN E.
Le système PIAFDOC
Troisième congrès européen sur les systèmes et réseaux documentaires
Luxembourg 1977

Gre GREUSSAY P.
Contribution à la définition interprétative et à l'implémentation des lambda-langages.
Thèse d'Etat - Université de Vincennes - 1977

Gro1 GROSS M.
Méthodes en syntaxe
Editions Hermann 1977

Gro2 GROSS M.
Sur quelques problèmes posés par la représentation formalisée des textes.
Analyse et validation dans l'étude des données textuelles.
Editions du CNRS 1977

PLAN HEWITT C.
PLANNER : A language for manipulating models and proving theorems in a robot.
MIT Artificial Intelligence - Mémo n° 168 - 1970

Huet1 HUET G.P.
A unification algorithm for typed lambda-calculus
Theoretical Computer Science 1, 1975

Huet2 HUET G.P.
Résolution d'équations dans les langages d'ordre 1,2,...,ω.
Thèse d'Etat - Université de PARIS VII - 1976

IlKINOV Il'in G.M., LEIKINA B.M., NIKITINA T.N., OTKAPEZKOVA M.I., FITALOV S.I.
Le modèle sémantique du texte et le système dans la sémantique en URSS.
Centre de Linguistique quantitative de l'Université de Paris VI
Document n° 10 - 1971

Io-Me IOMDINE L.L., Mel'CHUK I.A., PERTSAV N.V.
Fragment de modèle de syntaxe russe de surface
dans analyse et validation dans l'étude des données textuelles
Editions du CNRS 1977

Jolo JOLOBOFF V.
Traduction de schémas de programmes en Fortran.
Université de Grenoble R.R. n° 77 1977

Kap KAPLAN R.M.
Augmented transition networks as psychological models of sentence comprehension.
Artificial intelligence, vol 3, 1972

Ka-Fo KATZ J. & FODOV J.A.
The structure of a semantic theory
dans The structure of Language. Prentice Hall Inc 1964

Knuth KNUTH D.E.
Structured programming with go to statement
ACM Computing Surveys, Vol. 6, 1974

Leh1 LEHNERT W.
Question-Answering in a story understanding system
Yale University - Report n° 57 - 1975

Leh2 LEHNERT W.
A conceptual theory of Question-Answering
Proc. IJCAI, MIT, 1977

Lop LOPEZ J.
Transformations d'arborescences et graphes de transition
ICCL Bergen 1978

Lux1 LUX A.
Etude d'un modèle abstrait pour une machine LISP et de son implémentati
Thèse 3ème Cycle - Université de Grenoble.

Lux2 LUX A
Technique d'implantation et structure du noyau
dans Langages et traducteurs. Document IRIA 1977

Mac MAC CARTHY J.
LISP I.5. Programmer's manual
M.I.T. Computer Center & Research Laboratory of Electronics
Cambridge - Massachussetts 1962

Me-Zo MEL'CHUK I.A. & ZHOLKOVSKIJ
Contruction d'un modèle actif de la langue sens → texte
La sémantique en URSS Centre de Linguistique quantitative de l'Université de Paris VI - Document n° 10, 1971

MIST MISTRAL
Manuel d'utilisation
Document CII - 1974

MO-Pa-Tu MONTANGERO G., PACINI G., TURINI F.
Two level structure for nondeterministic programs
CACM, vol 20, 1977

Mo MOUREAU M.
Les aspects linguistiques des stratégies d'interrogation dans la recherche bibliographique sur ordinateur
Documentaliste, Vol 13, 1976

Mo-Gi MOUREAU M. & GIRARD A.
Les possibilités de recherches bibliographiques en conversationnel de l'information scientifique et technique.
Revue de l'Institut Français du Pétrole - mars 1976

Pa-Qu PAIR C. & QUERE A.
Définition et étude des bilangages réguliers
Information and Control, Vol B, 1968

Pet PETRICK S.R.
Semantic Interpretation in the REQUEST system
Proc : ICCL Pise 1973

Pla PLATH W.
REQUEST A natural language question answering system
IBM Journal of Research, vol 20, 1976

Pot POTTIER B.
Linguistique générale - Théorie et description
Editions Klincksieck 1974

QLISP SACERDOTI E., FIKES R., REBOH R., SAGALOWICZ D., WALDINGER R., WILBER B.
QLIST : A language for the interactive development of complex systems.
National Computer Conference 1976

Qui QUILLIAN M.
The teachable language comprehender : a simulation program and theory of language.
CACM, vol 12, 1969

Rig RIEGER C.J.
A mechanism for the interpreation of sentence meaning in context.
Computer Science Department - Univeristy of Maryland.
Report TR 354 - 1975

Ros ROSEN B.K.
Tre manipulating system and Church Rosser theorem

Sim1 SIMMONS R.F. & BRUCE B.
Some relations between predicate calculus and semantic net representation of discourse
Proc. IJCAI - Londres 1971

Sim2 SIMMONS R.F. & SLOCUM J.
Generating English discourse from semantic nets.
CACM, vol 15, 1972

Sim3 SIMMONS R.F.
Semantic networks ; their computation and use for understanding English sentences.
dans Computer models of thought and language
W.H. FREEMAN & Co - San Francisco 1973

STA Storage and Information Retrieval System / Virtual Storage (STAIRS/VS). General Information
IBM System / 370 - PP 5740 XRI - 1974

Tes TESNIERE L.
Eléments de syntaxe structurale
Editions Kleinckseick, 1959

Ve1 VEILLON G.
Les problèmes d'analyse en traduction automatique
ICCL - Pise 1970

Ve2 VEILLON G.
Modèles et algorithmes pour la traduction automatique
Thèse d'Etat - Université de Grenoble - 1970

Wil1 WILKS Y.
Natural language inference
Stanford University -memo AIM 211 - 1973

Wil2 WILKS Y.
An intelligent analyser and understander of English
CACM, vol 18, 1975

Rul RULIFSON J.F., DERKSEN J.A., WALDINGER R.J.
QA4 : A prodecural calculus for intuitive reasining
Stanford Research Institute - Artificial Intelligence Center
Technical note 73-1972

Sch1 SCHANK R.C.
Intention, Memory and computer understanding
Stanford University - Memo AIM 140 - 1971

Sch2 SCHANK R.C. Goldman N., RIEGER C.J., RIESBECK C.K.
Primitive concepts underlying verbs of thought
Stanford University - Memo AIM 162 - 1972

Sch3 SCHANK R.C. RIEGER C.J.
Inference and the computer understanding of natural language
Stanford University - Memo AIM 197 - 1973

sch4 SCHANK R.C. & ABELSON R.
Scripts, plans and knowledge
Proc. IJCAI, Tbilissi, 1975

Sch5 SCHANK R.C.
Research at Yale in natural language processing
Yale University - report n° 84 - 1976

Sch6 SCHANK R.C.
How to learn/What to learn
Proc IJCAI, M.I.T., 1977

Salt SALTON G.
The SMART retrieval system. Experiments in automatic document processing.
Prentice Hall, 1971

Wil3 WILKS Y.
Knowledge structures and language boundaries
Proc. IJCAI MIT 1977

Win WINOGRAD T.
Procedures as a representation for data in a computer program for understanding natural language.
Ph. D. Thesis - MIT - Massachussetts 1971

Wol WOODS W.
Transition network grammars for natural language analysis
CACM, vol 13, 1970

Wo2 WOODS W.
Foundation for semantic networks
dans Representation and understanding studies in cognitive science.
Academic Press Inc - 1975

Wo3 WOODS W., BATES M., BROWN B., BRUCE B., COOK C., KLOVSTAD J., MARHOUL J
NASH-WEBBER B., SCHWARTZ R., WOLF J., ZUE V.
Final report
Bolt Beranek and Newman, report n° 3438, 1976

Zahn ZAHN C.
A control statement for natural top down structured programming
Symposium on Programming language, Paris 1974

Zw ZWICKY A., FRIEDMAN J., HALL B., WALKER D.
The MITRE syntactic analysis procedure for transformationnal grammars
Proc. Fall Joint Computer Conference 1965.

Dernière page d'une thèse

VU

Grenoble, le 12 Juillet 1978

Le Président de la thèse

Vu, et permis d'imprimer,

Grenoble, le 13 juillet 1978

Le Président de l'Université
Scientifique et Médicale