Modèles et algorithmes pour la traduction automatique

Gérard Veillon

FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

Mathématiques Appliquées

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Centre d'Etudes pour la Traduction Automatique

Première thèse :

MODÈLES ET ALGORITHMES POUR LA TRADUCTION AUTOMATIQUE

Deuxième thèse :

PROPOSITIONS DONNÉES PAR LA FACULTÉ

Thèses présentées pour obtenir

LE GRADE DE DOCTEUR ÈS SCIENCES APPLIQUÉES

par

Gérard VEILLON

Thèses soutenues en AVRIL 1970

FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

Mathématiques Appliquées

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Centre d'Etudes pour la Traduction Automatique

Première thèse :

MODÈLES ET ALGORITHMES POUR LA TRADUCTION AUTOMATIQUE

Deuxième thèse :

PROPOSITIONS DONNÉES PAR LA FACULTÉ

Thèses présentées pour obtenir

LE GRADE DE DOCTEUR ÈS SCIENCES APPLIQUÉES

par

Gérard VEILLON

Thèses soutenues en AVRIL 1970

L I S T E D E S P R O F E S S E U R S

Doyen honoraire : Monsieur M. MORET
Doyen : Monsieur E. BONNIER

PROFESSEURS TITULAIRES

MM.	NEEL Louis	Physique Expérimentale
	KRAVTCHENKO Julien	Mécanique Rationnelle
	CHABAUTY Claude	Calcul différentiel et intégral
	BENOIT Jean	Radioélectricité
	CHENE Marcel	Chimie Papetière
	FELICI Noël	Electrostatique
	KUNTZMANN Jean	Mathématiques Appliquées
	BARBIER Reynold	Géologie Appliquée
	SANTON Lucien	Mécanique des Fluides
	OZENDA Paul	Botanique
	FALLOT Maurice	Physique Industrielle
	KOSZUL Jean-Louis	Mathématiques
	GALVANI Octave	Mathématiques
	MOUSSA André	Chimie Nucléaire
	TRAYNARD Philippe	Chimie Générale
	SOUTIF Michel	Physique Générale
	CRAYA Antoine	Hydrodynamique
	REULOS René	Théorie des Champs
	BESSON Jean	Chimie Minérale
	AYANT Yves	Physique Approfondie
	GALLISSOT François	Mathématiques
Melle.	LUTZ Elisabeth	Mathématiques
MM.	BLAMBERT Maurice	Mathématiques
	BOUCHEZ Robert	Physique Nucléaire
	LLIBOUTRY Louis	Géophysique
	MICHEL Robert	Minéralogie et pétrographie
	BONNIER Etienne	Electrochimie et Electrométallurgie
	DESSAUX Georges	Physiologie animale
	PILLET Emile	Physique Industrielle-Electrotechnique
	YOCCOZ Jean	Physique Nucléaire théorique
	DEBELMAS Jacques	Géologie Générale
	GERBER Robert	Mathématiques
	PAUTHENET René	Electrotechnique
	MALGRANGE Bernard	Mathématiques Pures
	VAUQUOIS Bernard	Calcul Electronique
	BARJON Robert	Physique Nucléaire

MM.	BARBIER Jean-Claude	Physique
	SILBER Robert	Mécanique des Fluides
	BUYLE-BODIN Maurice	Electronique
	DREYFUS Bernard	Thermodynamique
	KLEIN Joseph	Mathématiques
	VAILLANT François	Zoologie et Hydrobiologie
	ARNAUD Paul	Chimie
	SENGEL Philippe	Zoologie
	BARNOUD Fernand	Biosynthèse de la Cellulose
	BRISSONNEAU Pierre	Physique
	GAGNAIRE Didier	Chimie Physique
Mme.	KOFLER Lucie	Botanique
MM.	DEGRANGE Charles	Zoologie
	PEBAY-PEROULA Jean-Claude	Physique
	RASSAT André	Chimie Systématique
	DUCROS Pierre	Cristallographie Physique
	DODU Jacques	Mécanique Appliquée I. U. T.
	ANGLES D'AURIAC Paul	Mécanique des Fluides
	LACAZE Albert	Thermodynamique
	GASTINEL Noël	Analyse numérique
	GIRAUD Pierre	Géologie
	PERRET René	Servo-mécanisme
	PAYAN Jean-Jacques	Mathématiques Pures

PROFESSEURS SANS CHAIRE

MM.	GIDON Paul	Géologie
Mme.	BARBIER Marie-Jeanne	Electrochimie
Mme.	SOUTIF Jeanne	Physique
	COHEN Joseph	Electrotechnique
	DEPASSEL R.	Mécanique des Fluides
	GLENAT René	Chimie
	BARRA Jean	Mathématiques Appliquées
	COUMES André	Electronique
	PERRIAUX Jacques	Géologie et Minéralogie
	ROBERT André	Chimie Papetière
	BIARREZ Jean	Mécanique Physique
	BONNET Georges	Electronique
	CAUQUIS Georges	Chimie Générale
	BONNETAIN Lucien	Chimie Minérale
	DEPOMIER Pierre	Physique Nucléaire-Génie Atomique
	HACQUES Gérard	Calcul numérique
	POLOUJADOFF Michel	Electrotechnique
Mme.	KAHANE Josette	Physique
Mme.	BONNIER Jane	Chimie
MM.	VALENTIN Jacques	Physique
	REBECQ Jacques	Biologie
	DEPORTES Charles	Chimie
	SARROT-REYNAULD Jean	Géologie
	BERTRANDIAS Jean-Paul	Mathématiques Appliquées
	AUBERT Guy	Physique

PROFESSEURS ASSOCIES

MM.	RODRIGUES Alexandre	Mathématiques Pures
	MORITA Susumu	Physique Nucléaire
	RADHAKRISHNA	Thermodynamique

MAITRES DE CONFERENCES

MM.	LANCIA Roland	Physique Atomique
Mme.	BOUCHE Liane	Mathématiques
MM.	KAHANE André	Physique Générale
	DOLIQUE Jean Michel	Electronique
	BRIERE Georges	Physique
	DESRE Georges	Chimie
	LAJZEHOWICZ Joseph	Physique
	LAURENT Pierre	Mathématiques Appliquées
Mme.	BERTRANDIAS Françoise	Mathématiques Pures
MM.	LONGEQUEUE Jean-Pierre	Physique
	SOHM Jean-Claude	Electrochimie
	ZADWORNY François	Electronique
	DURAND Francis	Chimie Physique
	CARLIER Georges	Biologie végétale
	PFISTER Jean-Claude	Physique
	CHIBON Pierre	Biologie animale
	IDELMAN Simon	Physiologie animale
	BLOCH Daniel	Electrotechnique I. P.
	MARTIN-BOUYER Michel	Chimie (C. S. U. Chambéry)
	SIBILLE Robert	Construction mécanique (I. U. T.)
	BRUGEL Lucien	Energétique I. U. T.
	BOUVARD Maurice	HYdrologie
	RICHARD Lucien	Botanique
	PELMONT Jean	Physiologie animale
	BOUSSARD Jean-Claude	Mathématiques Appliquées (I. P. G.)
	MOREAU René	Hydraulique I. P. G.
	ARMAND Yves	Chimie I. U. T.
	BOLLIET Louis	Informatique I. U. T.
	KUHN Gérard	Energétique I. U. T.
	PEFFEN René	Chimie I. U. T.
	GERMAIN Jean-Pierre	Mécanique
	JOLY Jean-René	Mathématiques Pures
Melle.	PIERY Yvette	Biologie animale
	BERNARD Alain	Mathématiques Pures
	MOHSEN Tahsin	Biologie (C. S. U. Chambéry)
	CONTE René	Mesures Physiques I. U. T.
	LE JUNTER Noël	Génie Electrique Electronique I. U. T.
	LE ROY Philippe	Génie Mécanique I. U. T.
	ROMIER Guy	Techniques Statistiques quantitatives I. U. T.
	VIALON Pierre	Géologie
	BENZAKEN Claude	Mathématiques Appliquées
	MAYNARD Roger	Physique

MM.	DUSSAUD René	Mathématiques (C. S. U. Chambéry)
	BELORIZKY Elie	Physique (C. S. U. Chambéry)
Mme.	LAJZEROWICZ Jeannine	Physique (C. S. U. Chambéry)
M.	JULLIEN Pierre	Mathématiques Pures
Mme.	RINAUDO Marguerite	Chimie
MM.	BLIMAN Samuel	E. I. E.
	BEGUIN Claude	Chimie Organique
	NEGRE Robert	I. U. T.

MAITRES DE CONFERENCES ASSOCIES

MM.	YAMADA Osamu	Physique du Solide
	NAGAO Makoto	Mathématiques Appliquées
	MAREZIO Massimo	Physique du Solide
	CHEECKE John	Thermodynamique
	BOUDOURIS Georges	Radioélectricité
	ROZMARIN Georges	Chimie Papetière

J'exprime ma reconnaissance

à Monsieur le Professeur KUNTZMANN
qui a bien voulu me faire l'honneur de présider le jury

à Monsieur le Professeur VAUQUOIS
qui m'a guidé et conseillé pendant mes travaux

à Monsieur BOLLIET
Maitre de Conférences à l'Institut Universitaire de Technologie de Grenoble, qui a bien voulu faire partie du jury

à Monsieur PAIR
Professeur à la Faculté des Sciences de Nancy
qui s'est intéressé à mon travail et dont les critiques bienveillantes m'ont été précieuses.

Cette thèse reflète ma participation aux travaux de l'Equipe de chercheurs et techniciens du Centre d'Etudes pour la Traduction Automatique dont les efforts ont permis de réaliser complètement ce système.

Je tiens à remercier Mesdemoiselles CHARLES et CURINIER
dont la patience a été mise à rude épreuve pendant la réalisation matérielle de ce document.

TABLE DES MATIERES

C H A P I T R E I

ORGANISATION DU SYSTEME DE TRADUCTION AUTOMATIQUE

Dans cette première partie, nous nous proposons de définir les grandes lignes du système du C.E.T.A., en essayant de montrer comment il peut être justifié.

Ceci nécessite une introduction aux différentes méthodes proposées, tant dans le domaine de la traduction automatique que de la mécano-linguistique. Nous devrons évidemment faire appel aux théories linguistiques. Cependant, n'étant pas linguiste, nous ne considérons ces théories que d'une façon purement "utilitaire" c'est-à-dire comme un auxiliaire précieux dans la définition de nos modèles.

1.1. LES METHODES LEXICOGRAPHIQUES

1.1.1. PRINCIPES GENERAUX

Les premières expériences de traduction automatique ont été réalisées à la Georgetown University, en 1952, par Dostert [Do]. Les principes de base de cette méthode, que nous appellerons lexicographique, ont été repris par plusieurs groupes de recherche. Actuellement, la plupart des réalisations pratiques de traduction automatique font appel à des méthodes lexicographiques, dont nous allons donner rapidement quelques principes.

L'utilisation d'un ordinateur à des travaux linguistiques permet de disposer d'un dictionnaire important et facile à consulter. Il est, de ce fait, aisé de réaliser une traduction mot à mot. Bien entendu, le résultat sera illisible. Si l'on cherche à analyser les causes de l'incohérence d'une telle "traduction", on doit faire appel aux grammaires régissant le couple de langues.

On constate qu'il faut procéder à un certain nombre de "retouches" sur cette sortie mot à mot ; en particulier, modifier l'ordre, insérer des prépositions ou accorder des adjectifs avec le nom qui les régit.

L'idée de départ des premiers travaux fut effectivement d'effectuer ces "retouches" sur le mot à mot. Cette tâche peut être confiée à un traducteur humain ("machine aided translation"). On peut encore essayer d'augmenter la taille du dictionnaire en y rangeant des groupes de mots avec leur traduction (1).

---

(1) Une machine spécialisée a été conçue dans ce but par King (1960) dans les laboratoires de I.B.M. avec la collaboration de l'US Air force . Les résultats montrèrent qu'il était nécessaire d'adjoindre au dictionnaire (Disque photoscopique), un ordinateur classique (Ro). Ceci est le seul cas, à notre connaissance, de solution "Hardware" au problème de la traduction automatique.

De telles méthodes ne peuvent être réellement considérées comme des systèmes de traduction automatique : l'intervention de l'ordinateur, dont la fonction essentielle est l'exécution d'un programme, est extrêmement limitée.

Les systèmes de type lexicographique font appel effectivement à un programme. Celui-ci est conçu directement comme une simulation du processus de traduction humaine. Cette simulation est évidemment extrêmement approximative. Cependant les instructions du programme suivent pas à pas les démarches logiques que pourrait faire un traducteur.

On peut distinguer, dans l'évolution des procédés lexicographiques, deux types de méthodes.

1.1.2. LES METHODES PAR ANALYSE PARTIELLE

La démarche suivie est très précisément celle que nous avons esquissée au début. On procède à un certain nombre de 'retouches' permettant de rendre compréhensible le mot à mot.

Ceci nécessite évidemment un examen du voisinage de chaque mot. En fait, le programme cherche à repérer un certain nombre de mots clés dans la phrase : sujet, prédicats, séparateurs de propositions ou compléments de noms par exemple. L'analyse est partielle, car elle ne permet pas de contrôler la cohérence globale de la phrase. Elle se contente de caractériser un certain nombre de groupes de mots. Dans les cas d'ambiguïté, une seule solution est conservée. Ceci peut être la cause d'un grand nombre d'erreurs, car beaucoup d'ambiguïtés "locales", c'est-à-dire au niveau du groupe de mots, peuvent être levées au niveau global de la phrase : le groupe de mots "Pierre et Paul" est évidemment une coordination de deux substantifs propres. Mais dans la phrase : "J'ai des nouvelles de Pierre et Paul n'a pas écrit", cette coordination n'est plus valable. La conjonction "et" peut avoir une portée plus ou moins grande, et il est nécessaire de construire complètement la phrase pour lever l'ambiguïté.

Les groupes de mots caractérisés sont ensuite "réarrangés" selon leurs fonctions respectives : place du sujet par rapport au verbe, du complément de nom,

etc... Enfin, la dernière phase du programme consiste à insérer les mots manquants, tels que les prépositions ou les articles. Ces programmes ont été généralement mis au point sur le couple Russe-Anglais, ce qui justifie cette dernière phase.

Ces premières réalisations ont pu être menées à bien assez rapidement. Les résultats montrèrent leurs insuffisances et la nécessité de faire appel à des méthodes plus élaborées (1).

1.1.3. LES METHODES PAR ANALYSE COMPLETE

A la suite des expériences de la Georgetown University, P. Garvin [Ca 1,2] s'est attaché à développer une méthode faisant appel à une analyse complète de la phrase. Son "Fulcrum Syntactic Analyser" est constitué de plusieurs phases d'analyse. A chaque passage on associe un certain nombre de mots clés à repérer. Toutes les solutions sont envisagées, et l'on ne retient que celles qui conduisent à une cohérence globale.

En effet, l'ensemble du traitement syntaxique peut être repris autant de fois qu'il y a de possibilités. Ce traitement est de type lexicographique, car le codage du programme utilise un recensement des faits linguistiques.

Le texte obtenu dans une telle traduction est, dans certains cas, excellent. Cependant, le texte anglais est encore très proche des tournures russes. Enfin, un grand nombre de phrases, trop complexes, échappent à l'analyse syntaxique. Garvin a proposé l'écriture d'un second système plus complet. Cependant, l'utilisation de méthodes lexicographiques nécessite de reprendre complètement l'écriture des programmes. Ces derniers sont extrêmement complexes ; chaque procédé d'analyse linguistique est directement codé sous forme d'instructions.

Cette caractéristique des méthodes lexicographiques nous semble rédhibitoire. Les systèmes sont en effet très difficilement améliorables. Or l'examen des phénomènes linguistiques montrent qu'un recensement exhaustif a priori est pratiquement impossible, surtout lorsque l'on analyse des textes de provenances diverses, qui ne sont pas toujours conformes aux grammaires traditionnelles !

---

(1) Les programmes les plus caractéristiques de cette première méthode sont ceux de la Georgetown University et de la Computer Concepts Inc

En outre, il ne semble pas qu'une méthode lexicographique permette d'atteindre les niveaux d'analyse suffisants pour permettre de fournir une traduction utilisable dans tous les cas. Tous les principes doivent être remis en cause si l'on désire changer l'une des deux langues du couple.

1.2. L'UTILISATION DES MODELES

1.2.1. PRINCIPES GENERAUX

Les résultats des premières expériences montrèrent qu'il était difficile de considérer la traduction automatique comme un algorithme simplement déduit des procédés humains de traduction. En particulier, l'utilisation d'ordinateurs, machines mathématiques, nécessite une formalisation rigoureuse des données linguistiques.

Indépendamment des travaux orientés vers la traduction automatique, un certain nombre de théories formelles ont été élaborées par les linguistes et les mathématiciens.

D'autre part, le traitement des langages artificiels (compilation des langages de programmation) a permis de définir et d'appliquer un grand nombre d'algorithmes d'analyse syntaxique [Gp]. De ces divers travaux, l'on peut dégager les principes généraux de l'utilisation des modèles formels pour le traitement automatique des langues.

On doit chercher à représenter les phénomènes linguistiques que l'on se propose de traiter au moyen d'un modèle mathématique. Ce modèle ne constitue qu'une description approximative des données. La qualité du modèle dépendra de la finesse de cette approximation. La définition d'un tel modèle est effectuée en trois étapes :

a) Choix du type de modèle

Ce choix se fait en fonction du problème à résoudre (par exemple :

l'analyse syntaxique), et en tenant compte des algorithmes possibles. Lorsque le type de modèle est choisi, il est possible de définir complètement l'algorithme de traitement associé.

b) Ecriture du modèle proprement dit

Il s'agit alors de donner une description formelle des phénomènes linguistiques, au moyen d'une grammaire et de données lexicographiques. La grammaire se présente sous la forme de règles décrivant globalement les phénomènes linguistiques. Les langues naturelles étant caractérisées par un grand nombre d'unités lexicales (les mots du dictionnaire), il n'est pas possible d'écrire autant de règles qu'il y a de mots. On définira donc une partition des unités en classes d'équivalence. Deux mots sont dans une même classe si ils ont un même comportement dans le modèle donné. Le travail lexicographique consiste à associer à chaque mot la classe d'équivalence à laquelle il appartient.

Cette séparation entre grammaire et lexicographie permet évidemment l'enrichissement du vocabulaire du modèle.

c) Ecriture et mise au point de l'algorithme

Cette étape est complètement indépendante de la précédente ; cette indépendance caractérise les méthodes fondées sur le principe d'utilisation de modèles, en opposition avec les méthodes lexicographiques. En aucun cas, les informations linguistiques n'interviennent dans la rédaction des programmes. Ces derniers peuvent être utilisés avec d'autres modèles du même type, par exemple dans le traitement de langues différentes. Parallèlement, le modèle linguistique n'est pas lié à une technologie donnée.

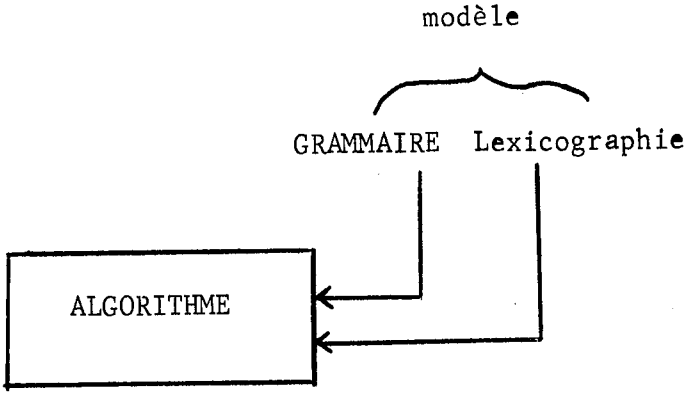


Figure 1

Le modèle se présente comme un véritable paramètre de l'algorithme.

1.2.2. LES MODELES SYNTAXIQUES

De tous les modèles linguistiques, ce sont les modèles syntaxiques qui ont été les plus développés, ce sont les seuls à avoir donné naissance à des algorithmes efficaces.

En effet, l'analyse syntaxique a été la première difficulté importante rencontrée dans les problèmes de reconnaissance des langues. La théorie des systèmes combinatoires [Da] a fourni les outils mathématiques nécessaires à la définition d'une syntaxe [Gi]. Le développement de la théorie des automates et des théories linguistiques [CH 1] ont permis de définir les applications.

Un modèle syntaxique a pour but d'associer une description structurale à toute phrase d'une langue donnée. Bien entendu, nous ne tenons compte que de cet aspect, sans chercher la motivation plus profonde des théories linguistiques : notre propos est seulement de les utiliser dans un système de traduction automatique et non dans leur finalité propre, qui est de présenter un modèle général de la langue.

Cette dernière hypothèse caractérise ce travail par rapport à la plupart des expériences de traitement automatique des langues réalisées actuellement aux Etats Unis (1). En effet ces derniers sont plus orientés vers une investigation linguistique (par exemple les contrôles des modèles proposés) que vers une application concrète.

Nous envisageons ici les deux hypothèses principales de description syntaxique : les descriptions de type syntagmatique et les descriptions de type relationnel.

---

(1) Citons en particulier les travaux réalisés par Hays et Kay (Rand Corporation) Kuno (Haward), Yngve (MIT Université de Michigan) et les travaux réalisés à l'Université de Pensylvanie (Harris).

1) Les descriptions de type syntagmatique ("structures et grammaires de constituants")

Ce sont les descriptions associées aux modèles définis par Harris [Ha] et Chomsky [CH1, CH2, CH3], dans les grammaires génératives. Une telle grammaire se présente comme un ensemble de règles définissant l'ensemble de phrases de la langue. Etant donnée une phrase, elle peut, si elle appartient à la langue, être obtenue par application des règles de la grammaire. A cette "dérivation" d'une phrase de la langue, on associe une structure reflétant le procédé de construction.

Sans aller plus loin dans l'examen de cette théorie, nous ne considérerons ici que l'aspect purement descriptif de la structure syntaxique (1). Celle-ci est caractérisée par la notion de "syntagme", ou groupe de mots ayant des propriétés syntaxiques données.

Le graphe associé à la structure que nous appelerons "graphe de constituants" représente en fait une relation d'appartenance. A chaque sommet correspond un mot (syntagme élémentaire) ou un groupe de mots (syntagme) appartenant lui-même à un autre syntagme de niveau plus élevé. Dans le cas où le modèle utilisé est de type hors contexte ("context free") la structure est équivalente à un parenthésage de la phrase, comme le montre l'exemple suivant :

La phrase "Mon vieil ami chante cette fort jolie chanson" est représentée par la structure suivante :

---

(1) C'est volontairement que nous n'étudierons pas ici l'aspect "modèle génératif" qui est le principe fondamental de l'école de Chomsky. Ce n'est en effet qu'en "utilisateur" que nous considérons ces modèles.

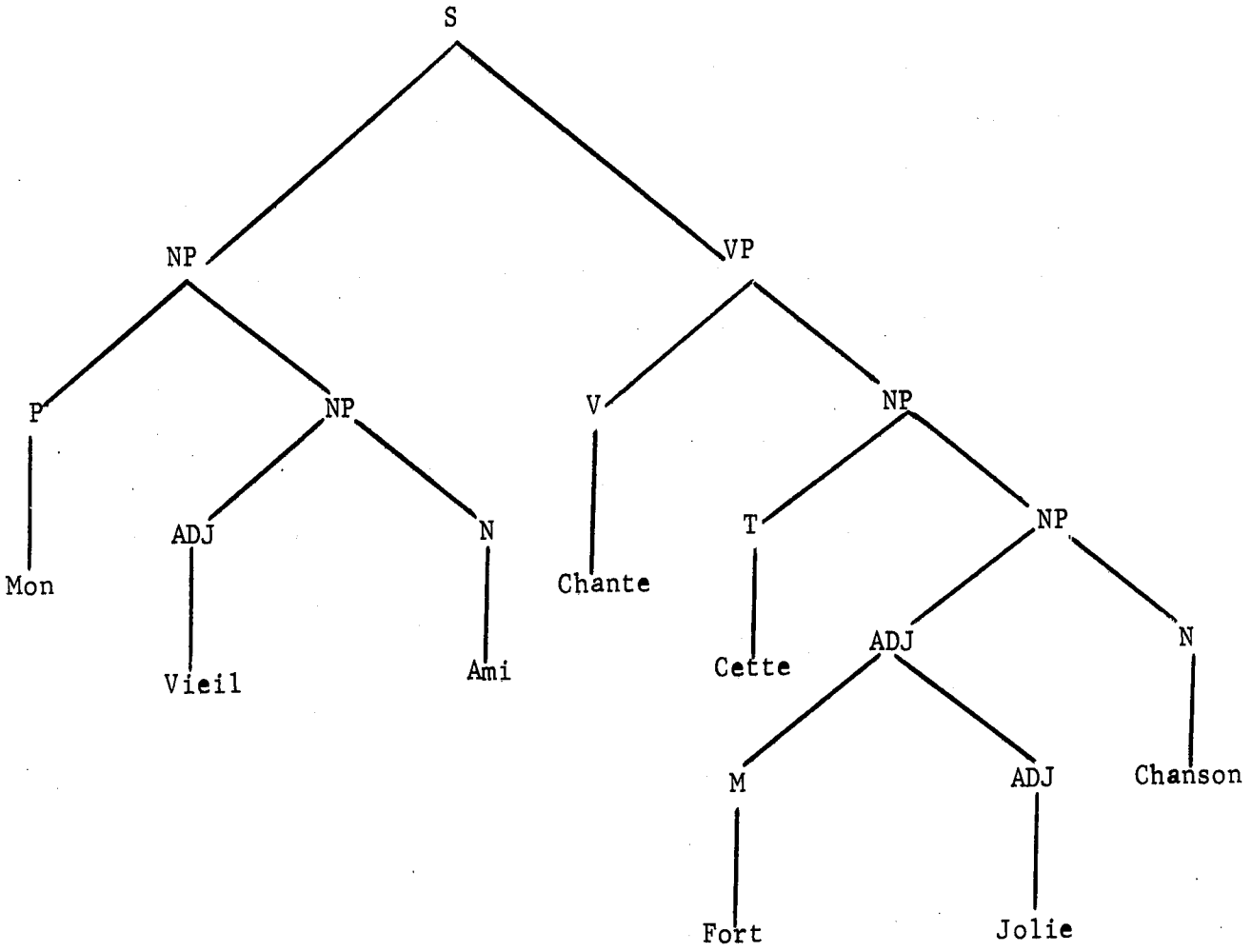


Figure 2

Les symboles apparaissant aux sommets "non terminaux" : S, NP, VP, V... correspondent aux divers syntagmes ("sentence", "nominal phrase", "verbal phrase", "verb" ...).

Le parenthésage associé serait le suivant :

((mon (vieil ami)) (chante (cette ((fort jolie) chanson)))

La structure ainsi obtenue est, comme on le voit, extrêmement liée à la forme de la langue. L'ordre des mots, par exemple, intervient dans la construction des diverses hiérarchies.

Cependant, une telle description structurale présente l'avantage de pouvoir être obtenue aisément à l'aide d'un programme.

2) Les descriptions de type relationnel

Introduites tout d'abord par Tesnières [Te], les descriptions de type relationnel ont été ensuite formalisées par Hays [Hy1], Gaifman [Gf], Lecerf [Le]. Hays a défini notamment la notion de "grammaire de dépendances", véritable grammaire générative permettant d'associer formellement une structure de type relationnel ("structure de dépendances") à une phrase appartenant au langage du modèle.

Dans une telle description, chaque sommet du graphe correspond effectivement à un mot de la phrase. Les arcs du graphe représentent alors les relations entre les divers mots de la phrase. Le graphe est arborescent, c'est-à-dire que chaque mot ne peut avoir qu'une seule fonction (et un seul gouverneur) dans la phrase. Plus généralement, étant donnés deux mots m_1 et m_2 tels qu'il y ait un arc (m_1, m_2), nous dirons que m_1 est "dépendant" de m_2 et que m_2 est "gouverneur" de m_1 .

L'exemple de la figure 2 se représentera par le graphe suivant :

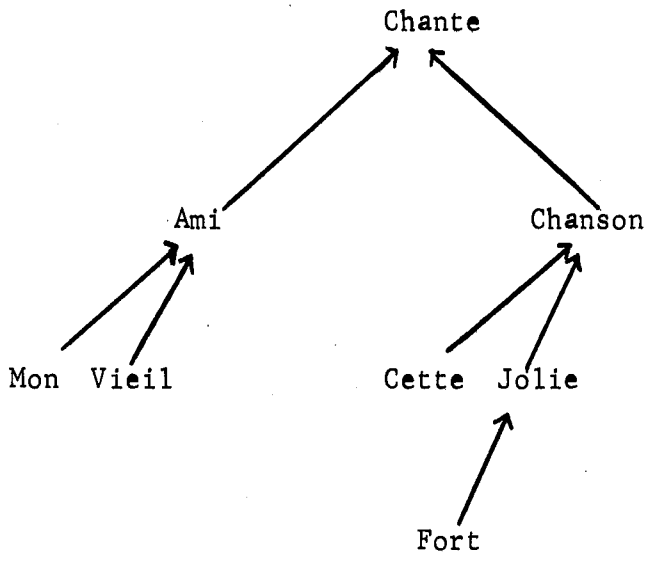


Figure 3

La nature des relations doit évidemment être précisée. Gaifman a montré l'équivalence entre grammaire de constituants (Chomsky) et grammaire de dépendances (Hays). Il est bien évident qu'il n'y a pas de contradiction entre les deux modèles.

Les représentations en graphe de dépendance semblent néanmoins être plus indépendants de la syntaxe propre de la langue : des graphes de dépendance analogues se rencontreront dans plusieurs langues naturelles, conformément à l'idée initiale de syntaxe générale, plus ou moins universelle, de Tesnières.

Par exemple l'ordre relatif des mots, qui est caractéristique d'une langue donnée, n'apparaît pas dans les hiérarchies du graphe de dépendances (1)

1.2.3. LES MODELES SEMANTIQUES

Sans faire une étude précise et exhaustive des modèles sémantiques nous examinerons ici rapidement quelques approches permettant d'associer à des grammaires purement formelles (2) des modèles permettant de faire intervenir la sémantique.

Cet aspect est évidemment fondamental en traduction automatique. En effet, la forme de l'expression, si elle peut être intéressante en linguistique, ne concerne le traducteur que dans la mesure où elle contient une information. Les objectifs de la traduction automatique n'étant encore que de traduire des textes scientifiques ou techniques, nous ne chercherons à transmettre que le contenu et non la forme du texte.

Le rôle d'un modèle sémantique est de permettre d'associer une notation de la signification à toute phrase d'une langue donnée.

---

(1) Nous reprendrons une analyse plus détaillée des correspondances entre graphes de constituants et graphes de dépendances dans le chapitre 2.

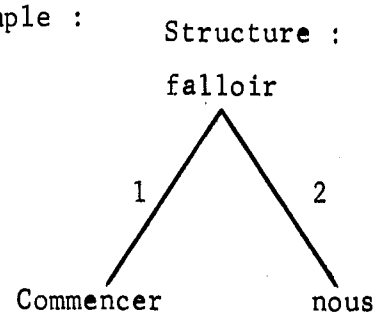
(2) Par grammaire formelle nous entendons grammaire décrivant la <u>forme</u> de la langue

Deux tendances doivent être distinguées. La première est fondée sur l'hypothèse d'existence d'une formulation du sens indépendante de la langue d'expression. En pratique, les modèles correspondants, ou modèles de type "sens-texte", présentés en particulier par Melchouk et Jolkowsky [Me] ne sont guère formulables directement. La seule approche proposée consiste à définir le sens comme le critère permettant de dire que deux phrases f et f' dans une langue donnée sont équivalentes.

Le modèle proposé par Melchouk et Jolkowsky est en fait un modèle permettant de construire, à partir d'une phrase donnée, toutes les phrases équivalentes. Le modèle reste de type syntaxique : à chaque phrase est associée une "structure lexicosyntaxique" (SLS). Il existe un système de règles de transformations permettant de passer d'une "SLS" à une "SLS" équivalente, c'est-à-dire de même signification.

Les structures sont de type relationnel, très proches des structures proposées par Tesnières [Te]. Les transformations peuvent affecter la structure elle-même ou le lexique.

Exemple :



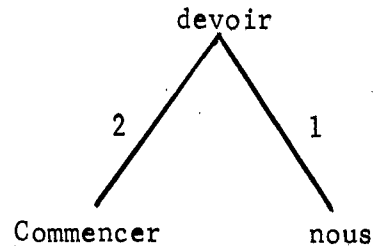
Expression

"Il nous faut commencer"

Figure 4

1 et 2 correspondent aux notions d'actants du verbe définies par Tesnières.

Une première transformation, notée "CONV" permet d'inverser les deux actants : CONV [falloir] = devoir



"nous devons commencer"

Figure 5

Une transformation plus complexe nous conduira à "Nous avons le devoir de commencer".

Parmi les structures lexicosyntaxiques équivalentes, on choisit une structure de base (BSLS), représentant de la famille.

Toutes les structures lexicosyntaxiques ne sont pas exprimables au niveau de la forme.

La structure de base est en quelque sorte une forme canonique à laquelle on associe toutes les structures équivalentes et, réciproquement, peuvent être obtenues à partir de ces structures.

Si φ est la notation sémantique idéale commune aux phrases de même signification, le schéma de transformation proposé par Melchouk et Jolkowsky en traduction automatique serait le suivant :

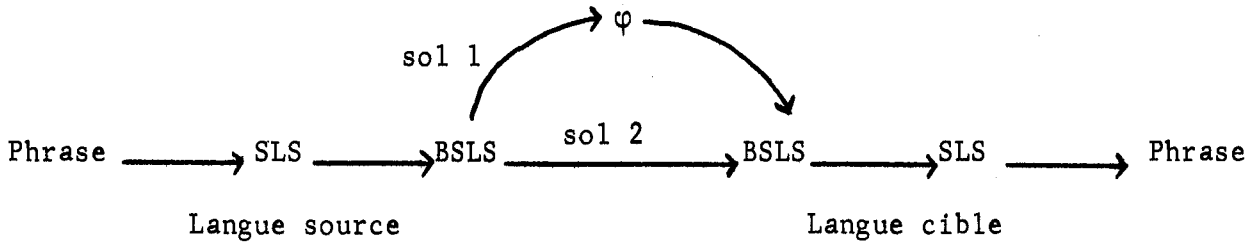


Figure 6

La solution 1 étant actuellement inaccessible, la solution 2 est proposée, c'est-à-dire une transformation directe de structure syntaxique à structure syntaxique.

Chomsky, suivant une tendance opposée au principe d'un système supposant l'existence d'une formulation sémantique unique, propose un modèle de type syntaxique : ce modèle suppose deux niveaux de description structurale : d'une part la structure de "surface", qui est directement interprétable sous la forme d'expression dans la langue, et, d'autre part, une structure "profonde", directement interprétable au niveau sémantique. Le modèle génératif associé comprend

alors une "base", grammaire de constituants, qui décrit l'ensemble des structures profondes, et une composante "transformationnelle" [CH 3] qui permet d'atteindre les structures de surface :

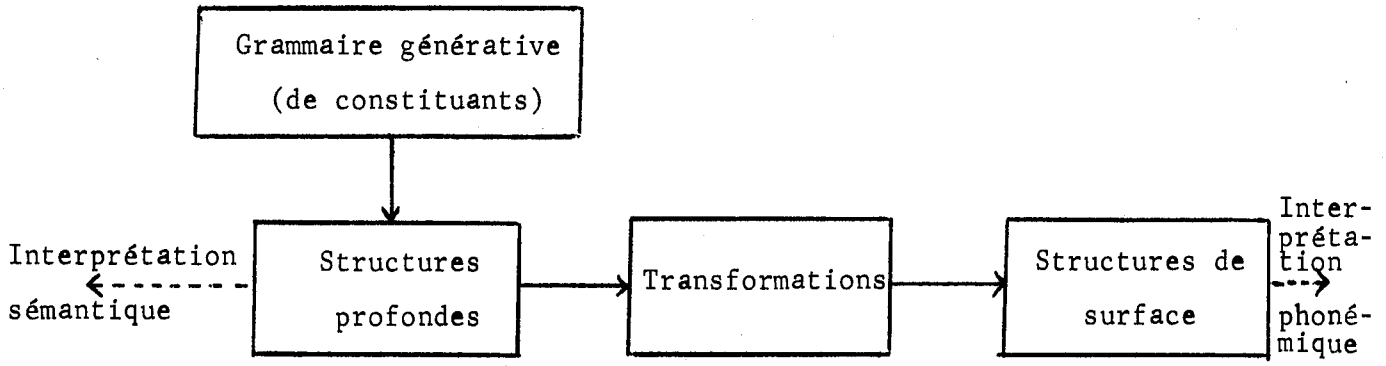


Figure 7

Le processus de génération est alors illustré par la figure

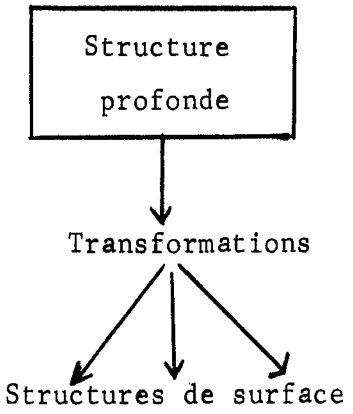


Figure 8

Le modèle est essentiellement interne à une langue donnée, et les diverses structures restent de type syntagmatique.

Il est remarquable de constater que ces deux approches de la sémantique utilisent en fait un procédé analogue en pratique : un système de transformations de structures syntaxiques permettent de mettre en évidence une structure

commune à toutes les phrases équivalentes. Il s'agit en fait d'un système de paraphrasage.

Parallèlement, plusieurs travaux ont été menés afin de déterminer une formulation de la sémantique. Actuellement, il n'y a pas, à notre connaissance, de théorie conduisant à un modèle utilisable dans des procédés automatiques [Ra]

1.2.4. LA STRATIFICATION

Les deux processus de recherche d'équivalence des structures de phrases mettent en évidence les divergences entre deux types de modèles. D'une part les modèles généraux du type de Chomsky, dans lesquels il n'y a en fait qu'une seule grammaire, les aspects sémantiques et phonologiques n'étant que purement interprétatifs. Dans de tels modèles, les descriptions restent de type syntaxique (la structure profonde est elle-même une structure syntaxique).

D'autre part, les modèles faisant appel à plusieurs types de description fondamentalement différents, en particulier la description syntaxique et la description sémantique.

Dans une telle optique, on fait appel réellement à plusieurs modèles, pour définir les divers aspects de la langue. Cette méthode a été particulièrement développée par Lamb [La] sous le nom de stratification.

Selon Lamb, la description de la langue peut se faire en quatre niveaux ou "strata" différents : "sémémique", "lexémique", "morphémique" et "phonémique".

A chaque niveau correspondent des unités de base (éléments) et un ensemble de règles de structuration de ces éléments en une unité complexe. Chaque description dans un niveau donné a une représentation dans le niveau immédiatement inférieur.

Ainsi, au niveau morphémique, les unités de base sont les morphèmes, structurés en chaine pour former le mot, unité complexe. Le mot est la représentation

du lexème (unité du niveau lexémique). Le morphème peut se représenter au niveau phonémique sous la forme de syllabes.

Les divers niveaux sont représentés dans le tableau suivant :

Niveau	Unité de base	Unité complexe	Type de structure
Sémémique	Sémème	Texte	Graphe général (?)
Lexémique	Lexème	Phrase	Arborescence
Morphémique	Morphème	Mot	Chaîne
Phonémique	Phonème	Syllabe	Chaîne

Figure 9

Pour chaque niveau, il est nécessaire de définir, d'une part, les règles d'association permettant de construire le graphe représentant l'unité complexe, et, d'autre part, les règles de représentation de la description dans le niveau immédiatement inférieur (c'est-à-dire la grammaire).

En pratique, dans un système de traduction de textes écrits, nous serons concernés par un niveau graphémique et non phonémique.

La stratification postule la séparation totale entre les diverses

grammaires. En particulier, le niveau sémémique peut être indépendant des niveaux lexémiques (c'est-à-dire syntaxiques) et morphémiques qui, seuls sont propres à une langue donnée. En d'autres termes, le niveau sémémique correspond bien à un modèle sémantique, commun à plusieurs langues.

1.2.5. APPLICATIONS

L'utilisation de modèles a été proposée pour des programmes de traduction automatique, notamment aux Etats Unis [Hy 2] et en U.R.S.S.. Les seules applications pratiques complètes, en dehors de la notre, ont été menées à l'Université du Texas (Austin) et à l'Université de KYOTO. Le principe de ces programmes fait appel à une analyse syntaxique et à un transfert des structures, utilisant des lois de transformation propres au couple de langues, en l'occurrence le couple allemand-anglais.

Cependant de nombreux programmes ont été décrits à partir des modèles, sans chercher une application en traduction. En particulier, les programmes d'analyse syntaxique et de génération ont été particulièrement développés aux Etats-Unis dans une optique de recherche linguistique.

1.3. LE SYSTEME DE TRADUCTION AUTOMATIQUE

1.3.1. PRINCIPES GENERAUX

La définition et la réalisation d'un système de traduction automatique fait intervenir trois disciplines différentes : les mathématiques, la linguistique, et l'informatique.

Les contraintes liées à la fiabilité d'un tel système nécessitent de respecter les principes suivants :

1) Les travaux linguistiques doivent être indépendants de la mise en oeuvre sur ordinateur. Une altération de l'une de ces deux parties ne doit pas mettre l'autre en cause. Ceci rejette pratiquement l'utilisation d'une méthode lexicographique. Il est, en outre, nécessaire de fournir au linguiste des langages de description des grammaires ne nécessitant que la connaissance du type de modèle.

2) L'ensemble des opérations exécutées au cours d'un processus de traduction est trop complexe pour que l'on puisse le considérer, comme un seul traitement. La détection des erreurs exige un fractionnement en plusieurs étapes. Il y correspondra un fractionnement en plusieurs modèles. Ceci permet de simplifier l'enrichissement du système et la correction des erreurs : une correction locale, tant dans la grammaire que dans la lexicographie, ne doit pas conduire à une reprise totale du système.

Ces considérations pratiques confirment l'intérêt théorique d'une méthode utilisant le principe des modèles et une stratification de ces modèles en niveaux successifs.

Le système sera donc conçu comme un enchaînement de phases, chacune d'entre elles ne communiquant avec la suivante que par l'intermédiaire de ses résultats. Ainsi, les modèles comme les programmes sont peu dépendants les uns des autres.

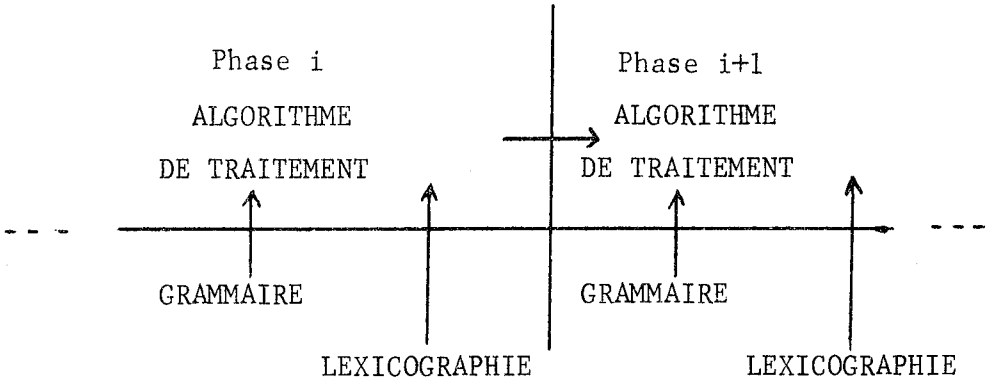


Figure 10

Ce déroulement est schématisé dans la figure 10. Les flèches verticales représentent en fait un ensemble de procédures de traduction des données linguistiques en données utilisables par la machine.

Ces diverses procédures constituent en pratique la plus volumineuse partie du travail de programmation.

1.3.2. TRADUCTION ET LANGAGE INTERMEDIAIRE

La traduction d'un texte écrit dans la langue source L vers un texte en langue cible L' comporte comme seule contrainte la préservation du sens du texte au cours de la transformation.

Nous avons vu qu'il existait des modèles qui définissent des transformations syntaxiques permettant de conserver la signification. Une traduction peut alors se considérer comme unetransformation des structures syntaxiques dans L vers les structures syntaxiques dans L'. La première condition nécessaire pour qu'une telle transformation soit possible, est qu'une telle signification existe. Nous entendons par là que l'on ne peut parler de traduction, ou même de paraphrasage, sans admettre que celle-ci est possible, c'est-à-dire qu'il peut y avoir plusieurs phrases ayant même signification.

Cette condition implique évidemment que l'on accepte une certaine approximation, qui s'exprimera par des infidélités dans la traduction, sans lesquelles aucune traduction n'est possible. Ceci étant admis, soient L et L' les ensembles des phrases $\{\mathfrak{P}_i\}$ et $\{\mathfrak{P}'_j\}$ de deux langues naturelles. Soit $\emptyset = \{\varphi_k\}$ l'ensemble des "formulations sémantiques", ou plus simplement l'ensemble des significations.

Une phrase $\ell_i \in L$ aura au moins une signification, que nous noterons φ_i . Cette signification peut alors se représenter par d'autres phrases de L, soit $L_i \subset L$, et par une famille de phrases L'_i de L'.

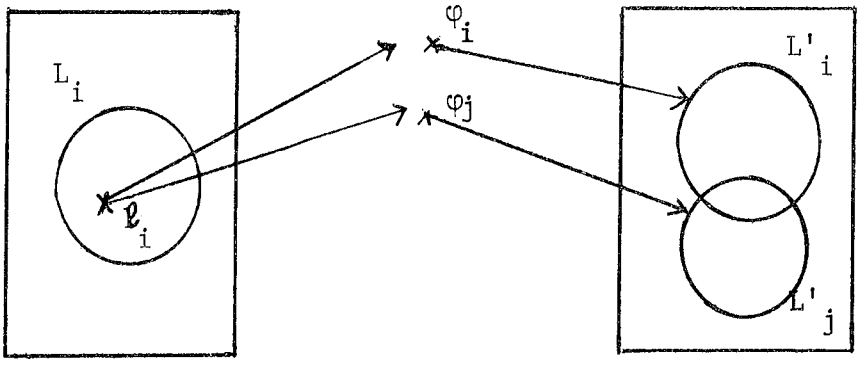


Figure 11

Cependant, la phrase ℓ_i peut avoir une autre signification φ_j . Plusieurs cas sont alors possibles :

1) Les ensembles L'_i et L'_j sont égaux, $L'_i = L'_j$. Dans ce cas, les significations φ_i et φ_j ne doivent pas être nécessairement distinguées, si l'on ne s'intéresse qu'à la traduction.

2) Les ensembles L'_i et L'_j sont différents, mais $L'_i \cap L'_j \neq \emptyset$. Il est possible de ne pas distinguer φ_i et φ_j, si l'on admet que l'on peut définir un processus ne fournissant que les traductions $\ell'_k \in L'_i \cap L'_j$

3) $L'_i \cap L'_j = \emptyset$, et il y a bien deux traductions pour la phrase L_i .

Les points 1 et 2 montrent clairement que, dans un processus de traduction, l'on doit moins chercher à déterminer le sens précis d'une phrase en langue source, (φ_i ou bien φ_j), qu'à conserver toutes les ambiguïtés.

En pratique, il existe peu de phrases intrinsèquement ambiguës, si l'on ne cherche que les ambiguïtés qui doivent nécessairement être levées (cas n° 3). Ce problème est bien illustré par la traduction des mots utilisés d'une façon imagée. La phrase "j'ai un tuyau" peut évidemment avoir deux significations. Il ne

faudra analyser et rechercher la bonne signification que si le mot tuyau n'est pas porteur de la même ambiguïté dans l'autre langue. Ce cas est particulièrement fréquent dans le vocabulaire scientifique et technique, par exemple avec le mot "dérivation" qui n'aura pas le même sens en analyse, en logique formelle ou en électricité.

Nous devrons donc conserver, plutôt que la vraie signification, l'ensemble des significations possibles.

Deux méthodes sont possibles pour réaliser un système de traduction : soit l'écriture d'un ensemble de règles de transformations associant les structures syntaxiques des deux langues, soit la recherche d'un langage intermédiaire, approximation d'une formulation sémantique.

La deuxième méthode peut paraître plus complexe, car elle nécessite un processus en deux étapes, au lieu d'une. Cependant, si l'on tient compte du fait que les règles de transformations syntaxiques doivent tenir compte des propriétés grammaticales des deux langues, l'économie n'est qu'illusoire.

En effet, un tel système doit tenir compte des propriétés grammaticales comme des propriétés lexicales. Soit une configuration syntaxique s_1 de L, et s'_1, s'_2 deux configurations de L' pouvant exprimer s_1.

Par exemple, ces configurations pourraient être les expressions nominales et verbales d'un objet de verbe :

s'_1 "Je désire ta venue"

s'_2 "Je désire que tu viennes"

Dans l'exemple choisi, s'_1 et s'_2 sont exprimables, le mot "venir" existant sous forme verbale et nominale. Ce ne sera pas toujours le cas, par exemple avec la négation :

"Je désire que tu ne viennes pas" et non

"Je désire ta non venue"

De même "Je désire ton argent" ne peut s'exprimer en s'_1. Nous devrons donc prévoir les deux expressions de s_1, le choix étant déterminé par les contraintes lexicales.

En admettant que l'on ait les mêmes expressions s_1 et s_2 dans L, nous pouvons tracer le graphe :

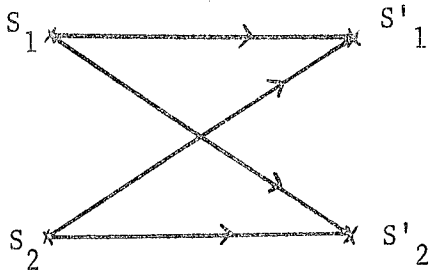


Figure 12

Chaque flèche représente une règle de transformation, qui n'est applicable qu'en fonction des possibilités syntaxiques.

Si l'on admet une formulation unique φ dans un langage intermédiaire, le graphe serait le suivant :

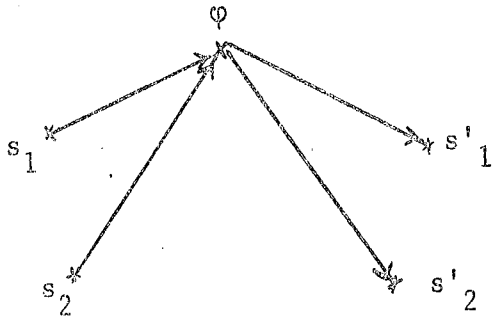


Figure 13

Dans ce cas, le nombre de règles est le même ; dans le cas général d'un sous graphe total à 2n sommets, nous aurons respectivement n^2 règles et 2n règles. De plus, il convient de noter que les règles de s vers φ sont essentiellement des règles d'analyse de s, et les règles φ vers s' sont des règles de production vers s'. Analyse et production sont aussi contenues dans les règles de transformation directe.

L'utilisation d'un langage intermédiaire présente l'avantage de rendre indépendants les processus d'analyse et de synthèse.

Ce langage peut être commun à plusieurs langues, puisqu'il ne dépend pas des expressions syntaxiques.

Cette méthode a été préconisée par Andreev [An]; nous l'avons retenue en recherchant un "langage-pivot" susceptible d'être utilisé pour des langues relativement voisines (langues indo-européennes). La définition en a été faite en examinant plus particulièrement le couple Russe-Français. Bien que ce langage soit indépendant des contraintes d'expression propre aux deux langues, il reste fortement lié aux propriétés communes de leurs structures respectives. Nous devons le considérer plutôt comme une syntaxe générale que comme un modèle sémantique.

1.3.3. LE LANGAGE PIVOT [V_1 , V_{a5}]

1) Principe général

Le langage pivot se présente sous la forme de descriptions structurales de type syntaxique. Il permet de séparer les divers énoncés élémentaires de la phrase. Ainsi la phrase simple "Pierre voit Paul", est constituée par un seul énoncé élémentaire composé d'un prédicat "voir", muni des deux actants "Pierre" et "Paul". Dans ce cas, la structure en langage pivot sera identique à la structure syntaxique. Nous marquerons les trois éléments "voit" "Pierre" et "Paul" par trois sommets d'un graphe arborescent représentant les relations liant ces trois éléments. Ces relations peuvent s'énoncer "Pierre est 1er actant", "Paul est 2ème actant", ce que nous noterons :

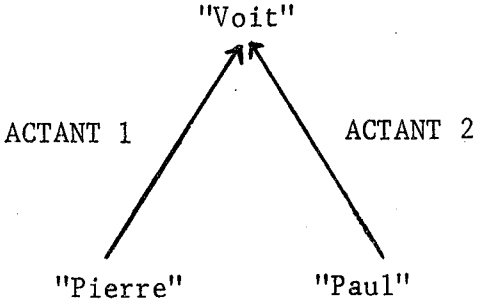


Figure 14

Si l'on interprète effectivement cet énoncé comme un prédicat, nous pourrions utiliser la notation : "Voit (Pierre, Paul)". Nous considérons aussi les relations "ACTANT 1" comme des métaprédicats (prédicats définis dans le modèle lui-même et non dans le lexique), et notons : ACTANT 1 (Pierre, voit (x, Paul)).

Dans ce cas particulier, la relation "ACTANT 1" du langage pivot correspond à la notion de sujet syntaxique, et peut s'interpréter comme "AGENT". Le sujet syntaxique n'est pas la seule expression de la relation "ACTANT 1", par exemple dans le cas de voix passive. D'autre part, l'interprétation "AGENT" nous a paru trop spécifique ; elle ne s'applique qu'aux prédicats exprimant une action.

Dans la description donnée figure 14, nous avons écrit la forme fléchie "voit" empruntée au français. En fait la description étant indépendante de la langue, nous ne pouvons utiliser de telles formes, qui ne sont que des "réalisations possibles de la notion". Chaque sommet sera donc caractérisé par une référence à un lexique, susceptible de s'exprimer par plusieurs formes dans chaque langue. De ce fait, la description donnée figure 14 peut aussi bien s'exprimer par "vision de Pierre par Paul", "Paul est vu par Pierre", "Pierre verra Paul"... etc. Ces différentes expressions sont choisies en fonction d'informations de deux types : d'une part les variables d'actualisation, qui font partie effectivement du langage pivot (par exemple le temps), d'autre part les contraintes d'expression de la langue, indépendantes du langage pivot (on ne peut utiliser le substantif verbal "vision", parce qu'il existe, ce qui n'est pas toujours le cas, par exemple pour le verbe "manger").

Un premier type de composition des énoncés peut être obtenu par l'utilisation d'un énoncé complet comme actant d'un prédicat : "Pierre voit que Paul voit Jean", que l'on peut écrire :

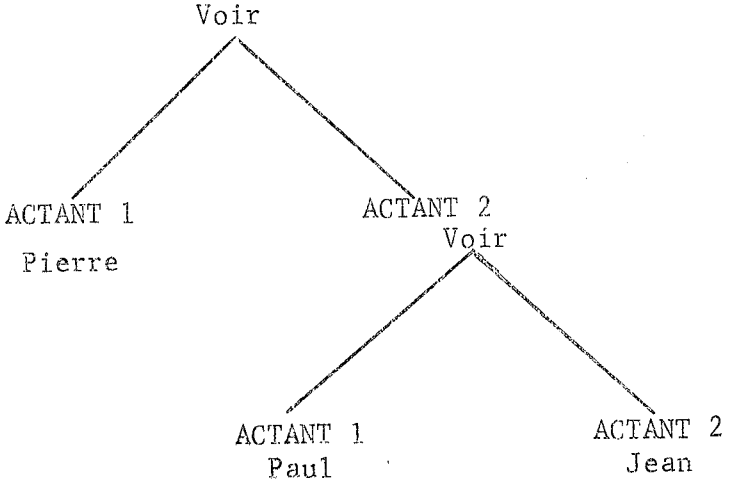


Figure 15

Une telle construction suggère l'interprétation suivante :

Le prédicat est une fonction qui associe à plusieurs énoncés un nouvel énoncé. L'analogie avec des algèbres connues est évidente, et la structure de la figure 15 est l'image de : "a + (b + c)" représentée par

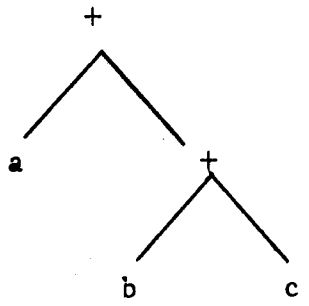


Figure 16

Malheureusement, les autres compositions des énoncés ne permettent pas d'étendre plus loin l'analogie, et nous ne possédons pas encore les lois de composition de ces énoncés.

2) Substitution

La structure arborescente ne permet pas de tenir compte des cas où le même mot a plusieurs fonctions dans la phrase. Ainsi : dans la phrase "Pierre dit à Paul de venir", Paul est à la fois le troisième actant du prédicat dire, et le premier actant de "venir". Ceci pourrait s'exprimer par la structure suivante :

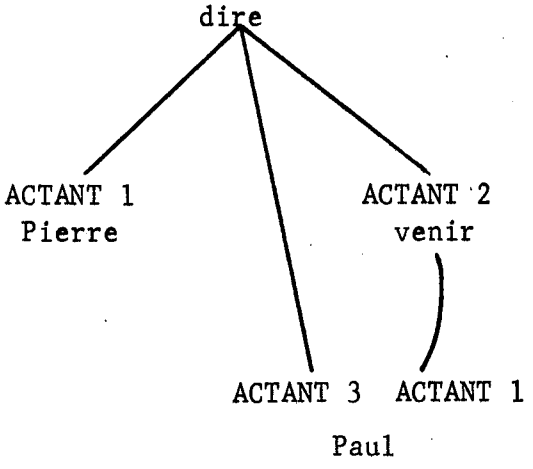


Figure 17

Nous préférons conserver la structure arborescente, et définir une nouvelle relation entre les sommets, que nous noterons "substitution"

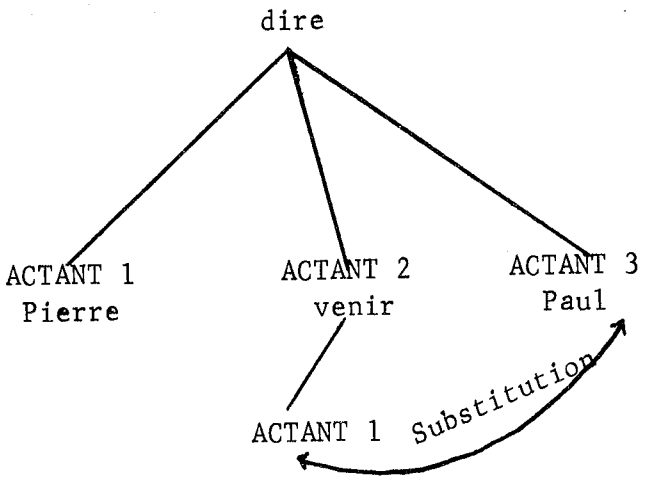
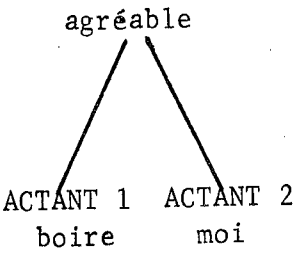


Figure 18

3) Mots prédicatifs et descripteurs

Parmi les mots de la langue , nous distinguerons essentiellement deux classes : les éléments prédicatifs qui peuvent s'exprimer par un verbe, et les éléments non prédicatifs (descripteurs), qui ne peuvent pas s'exprimer par un verbe. Les adjectifs sont considérés comme prédicatifs, et peuvent prendre un ou plusieurs actants.

Exemple 1 : "Il m'est agréable de boire"



(par analogie avec
"boire me plaît",
"boire est agréable pour moi")

Figure 19

4) Généralisation de la notion de métaprédicat

Considérons le groupe "homme blessé". L'adjectif 'blessé' est un mot prédicatif, et l'on peut noter blessé (homme), ou, plus précisément blessé (x, homme) x, premier actant étant indéterminé.

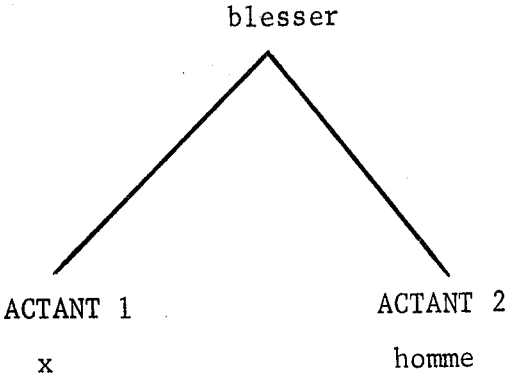


Figure 20

Cependant, il existe une autre relation entre "homme" et "blessé", en effet, l'épithète détermine le substantif. Soit p le prédicat de détermination, nous pouvons noter

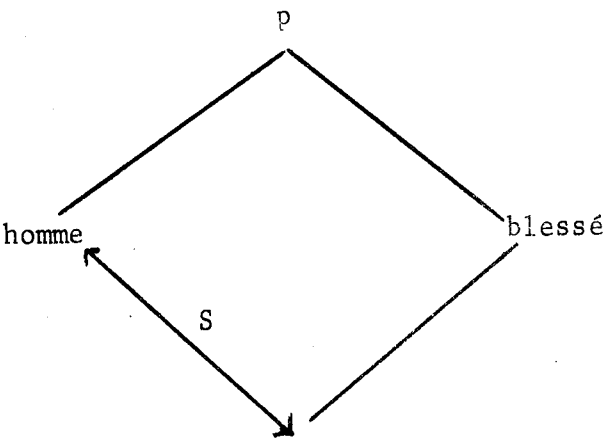


Figure 21

en marquant par une substitution l'identité. La valeur sémantique de p est imprécise, et peut être une spécification ou, dans d'autre cas, une implication logique (les fonctions dérivables sont continues), une succession dans le temps, etc...

Afin de rester plus proche des structures syntaxiques et de marquer la position de dépendance de cet énoncé par rapport à un descripteur donné, nous définirons un métaprédicat 'EPITHE'.

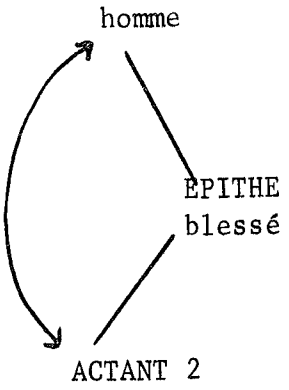


Figure 22

Dans cette même classe, nous notons aussi les relatives. Cependant, nous noterons par un métaprédicat "SPECIF" (spécification) la présence du relatif.

Exemple : "L'homme qui est blessé"

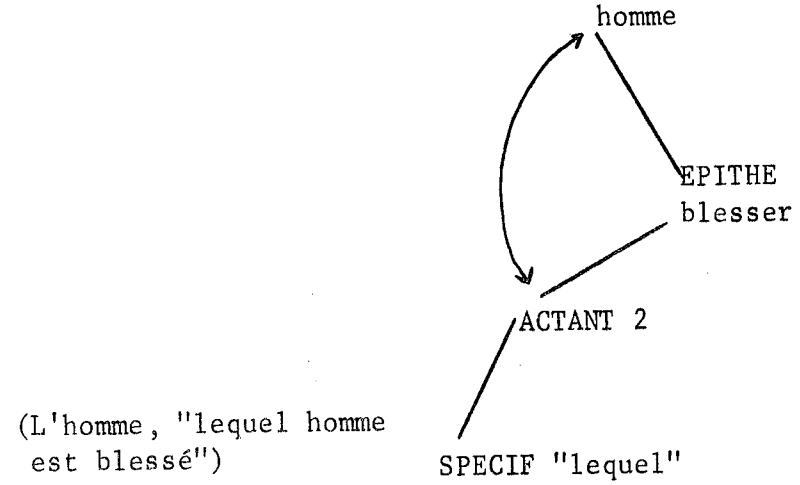


Figure 23

Remarquons que dans cet exemple, nous avons noté "ACTANT 2" la relation liant "homme" à "blesser". Ce faisant, nous confondons les deux significations de blessé, adjectif marquant un état ou voix passive de blesser. Plus généralement, nous noterons "ACTANT 2" l'unique actant d'un adjectif, ce qui permet ici pour distinguer "blessant" et "qui blesse" (ACTANT 1) de "blessé" et "qui est blessé" (ACTANT 2).

. Nous considérons également les conjonctions et prépositions comme des prédicats, lorsqu'elles ne marquent pas un actant donné (comme "à" dans "dire quelque chose "à" quelqu'un"). Bien entendu, il est nécessaire de faire un choix arbitraire, dans plusieurs cas. Certaines prépositions ou conjonctions peuvent être exprimées par un verbe ("causer", "à cause de", "parce que"...).

Nous pouvons donner deux types de structures aux groupes prépositionnels et propositions subordonnées :

soit en considérant la conjonction (préposition) comme prédicat principal : "Je suis triste parce que tu pars"

ou "Je suis triste à cause de ton départ" (figure 24)

Soit en plaçant le groupe en position d'épithète (figure 25)

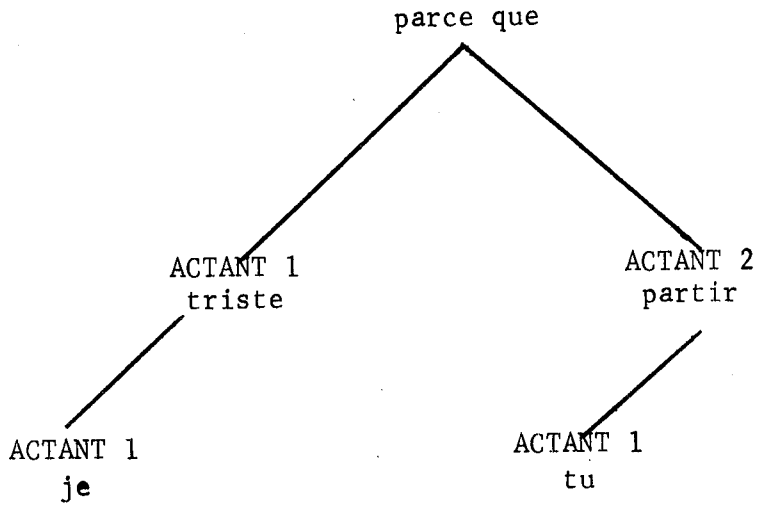


Figure 24

Soit

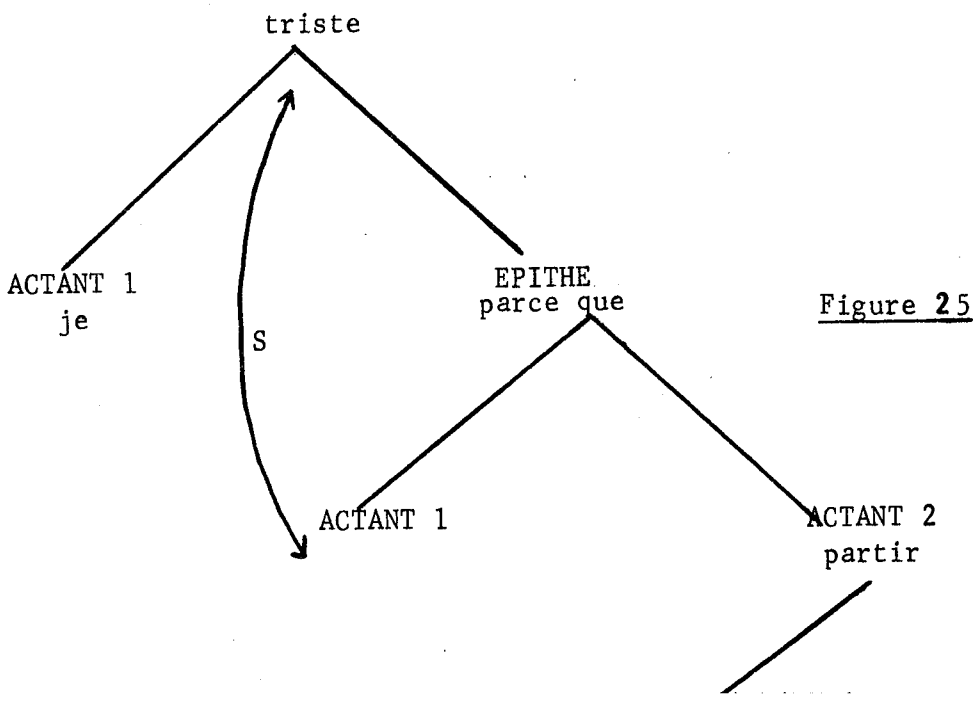


Figure 25

Dans le cas général, nous choisirons la seconde solution qui correspond plus à la structure syntaxique commune. Ce ne sera cependant pas le cas pour les conjonctions de coordination :

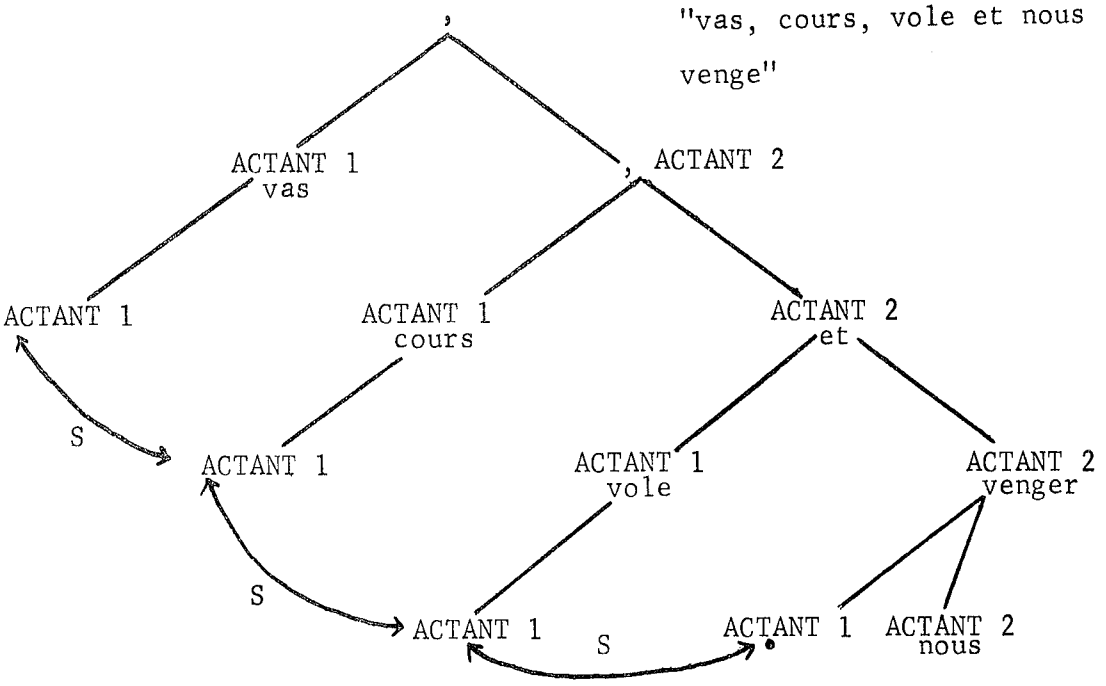


Figure 26

Le prédicat de coordination, marqué par la virgule et "et", exprime ici une succession dans le temps. Remarquons que le premier actant de l'impératif est marqué, mais ne correspond à aucune réalisation en français (actant virtuel).

En pratique, nous utiliserons les métaprédicats "CIRGEN" (circonstant général) pour marquer la relation de type "EPITHE" dans le cas des prépositions et conjonctions, elle implique la substitution du premier actant ; d'autre part, nous noterons "COOR 1", "COOR 2" les deux actants de la coordination.

Nous avons été amené à définir des métaprédicats spécifiques représentant à la fois une relation d'épithète et un prédicat de type prépositionnel. Ceci concerne essentiellement le génitif (préposition "de") dont les significations nombreuses ne sont pas toujours aisées à déterminer. Ainsi dans "fil de fer", "cheval d'arçon" et "relation d'équivalence", le prédicat associé à "de" varie fondamentalement. Cependant, en tenant compte des buts que nous recherchons, et des difficultés d'analyse, nous remplacerons ces trois prédicats p_1 , p_2 , p_3 par le métaprédicat ambigu "DETERM" qui contient aussi une relation de type EPITHE.

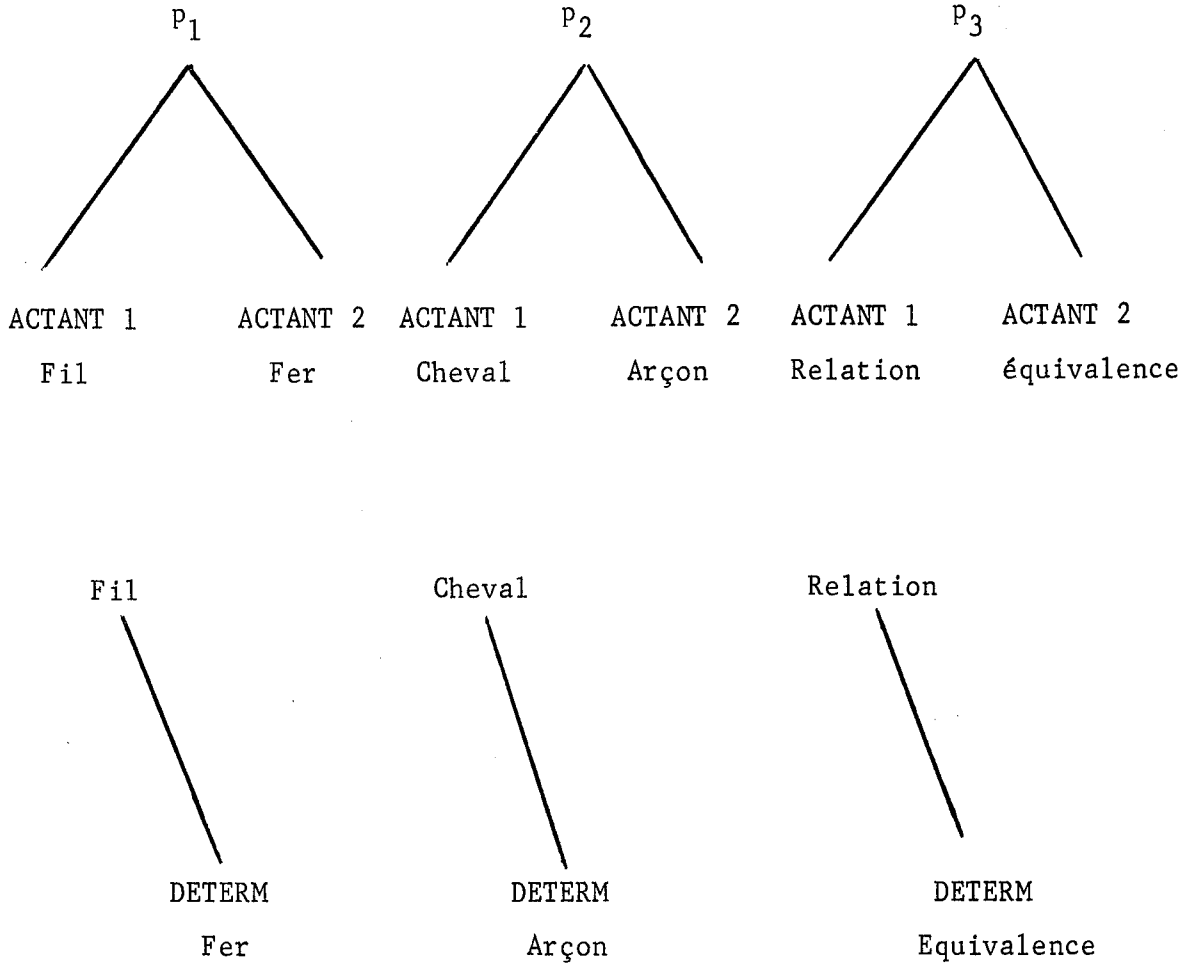

P1
P2
P3
ACTANT 1
Fil
ACTANT 2
Fer
ACTANT 1
Cheval
ACTANT 2
Arçon
ACTANT 1
Relation
ACTANT 2
équivalence
Fil
Cheval
Relation
DETERM
Fer
DETERM
Arçon
DETERM
Equivalence

Figure 27

Cependant, tous les génitifs ne seront pas analysés aussi sommairement, certains étant reconnus comme marquant de "EPITHE", d'autres comme marquants d'actants : "Prise de la Bastille"

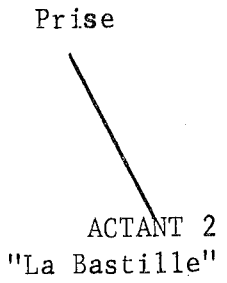


Figure 28

5) Substitution et élision

La substitution marque une équivalence qui n'est pas limitée à un sommet, mais à toute la sous arborescence dont il est racine, ce qui est particulièrement illustré par la figure 26. Le premier actant de "parce que" n'est pas "triste" mais l'énoncé "Je suis triste".

Dans certains cas, nous sommes conduits à considérer une équivalence limitée au sommet lui-même, ou plus exactement, aux mots qui peuvent l'exprimer. Nous noterons "Elision" une telle relation :

Exemple : "Le cheval de mon père et celui de mon oncle"

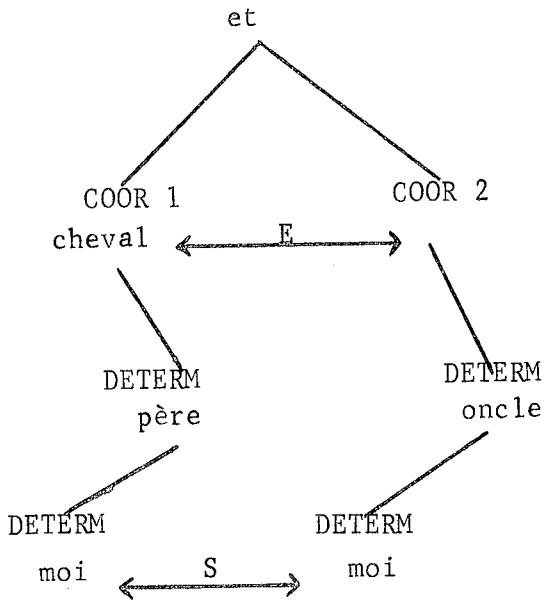


Figure 29

L'élision peut correspondre à un élément prédicatif (ellipse) :

"Collision de x avec y et de z avec t"

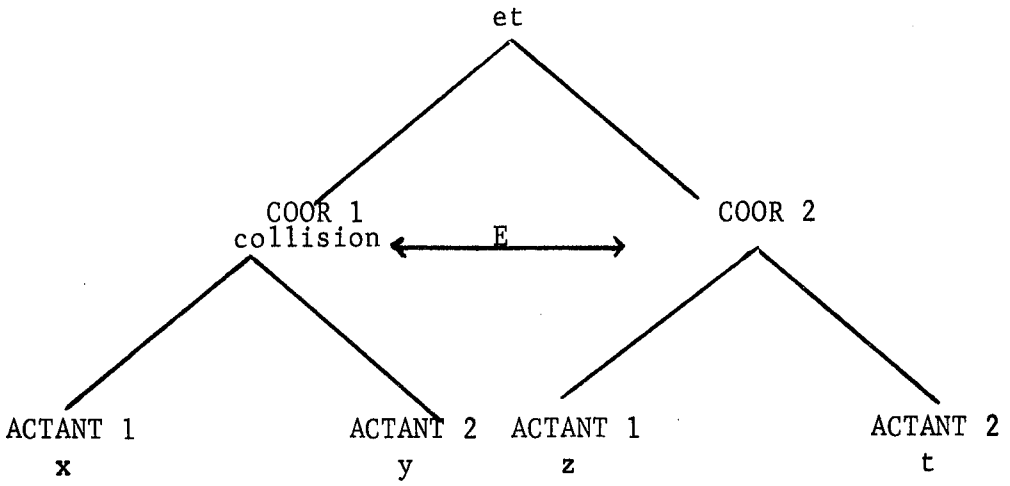


Figure 31

Cet exemple montre clairement que l'élision concerne le sommet seul, et non la sous arborescence associée (il ne s'agit pas de la même collision) ; l'élision concerne le type de prédicat et non l'énoncé.

6) Occurrence, variable et lexique

A chaque sommet, nous avons associé jusqu'ici un mot et un nom de relation. En fait, les informations liées au mot sont plus complexes.

Nous distinguons tout d'abord la notion d'élément du lexique et la notion d'occurrence. L'occurrence est affectée d'un numéro d'ordre.

A chaque sommet de l'arborescence correspond une occurrence unique. Cette information (donnée de l'occurrence) est complétée par un certain nombre de variables dites "persistantes" qui définissent l'actualisation de l'énoncé. Ces variables constituent un ensemble d'informations indépendantes des langues et du lexique. Elles comportent essentiellement le temps, l'aspect, le nombre et le sexe. Lorsque deux sommets sont en relation de substitution, occurrence et variables sont identiques. Dans le cas d'une élision, l'occurrence est la même, mais certaines variables peuvent différer : "Les chevaux de mon père et celui de mon oncle"

7) Le lexique

Nous ne pouvons spécifier les éléments du lexique autrement que par l'intermédiaire de leurs expressions en langue naturelle sous la forme de mots. Cependant, pour chaque unité du lexique, plusieurs expressions peuvent être utilisées dans chaque langue. Dans une langue donnée, on appellera paradigme l'ensemble de ces expressions.

Il n'est guère possible de donner une définition formelle de cette notion.

En effet, si l'on considère un niveau purement morphologique, c'est-à-dire en ne s'intéressant qu'aux règles de formation des mots en tant que suites de caractères, une étude formelle, utilisant un modèle d'états finis, permet effectivement de définir une classification et de construire un ensemble de paradigmes. L'analyse du mot en plusieurs composants (morphèmes) tels que préfixe, base, suffixe et désinence met en évidence les ensembles de 'formes' possibles, chaque paradigme étant caractérisé par l'invariance de la base.

Par exemple, les mots UTILE, UTILITE, INUTILITES, etc... constitueront le paradigme associé à la base 'UTIL'. Malheureusement, une telle classification ne nous permet pas de définir un lexique intéressant au niveau du langage pivot.

En effet, le critère de définition des paradigmes est relatif à la signification et non à la forme de l'expression. Ces deux critères ne sont pas cohérents, comme le montrent un grand nombre d'exemples : un chapelier fabrique des chapeaux, mais un chevalier ne fabrique pas de chevaux ! Même si l'on cherche à limiter les paradigmes, par exemple en ne tenant compte que des variables élémentaires, la signification peut changer : 'dérivé', 'dérivée' etc...

Il est donc nécessaire de renoncer aux critères formels et de définir des équivalences sémantiques.

La définition du lexique devra se faire en deux parties :

1) Au niveau de la langue, il convient de définir les paradigmes. Pour cela, l'on doit considérer, d'une part toutes les 'flections' de chaque mot en fonction de variables : "cheval", "chevaux", acteur, actrice, etc.... D'autre part, il est nécessaire de regrouper dans un même paradigme les divers "translations" possibles, c'est-à-dire les diverses classes syntaxiques que l'on peut exprimer pour une même notion. Un critère de choix de ces translations pourrait utiliser les paraphrasages possibles.

La méthode qui a été choisie ici consiste en la donnée a priori d'un certain nombre de types d'expressions possibles (il y en a 12). Le tableau correspondant, figure 32, n'est jamais rempli complètement. Nous appellerons "unité sémantique" l'ensemble des formes appartenant à un même paradigme.

2) Au niveau d'un ensemble de langues, nous devons définir des unités tenant compte des paradigmes respectifs de celles-ci. Il importe donc de donner une correspondance entre toutes les unités sémantiques des deux langues. De plus, cette correspondance doit tenir compte des contraintes du langage pivot : la permanence du nombre d'actants pour un prédicat, par exemple.

Paradigme - Chaque position peut exister en forme positive ou négative

N°	Nom de la position		Catégorie Syntaxique	Exemple 1	Exemple 2	Exemple 3	Forme analytique (Si la position n'existe pas)
1 p	Procès	positif	verbe	"agir"	Ø	"blesser"	Ø
1 n	Procès	négatif	verbe	Ø	Ø	Ø	ne pas + 1 p
2 p	Chose	p	substantif	"action"	Ø	Ø	"le fait de" + 1 p
2 n	Chose	n	substantif	"inaction"	Ø	Ø	"le fait de + 1 n" ou "non" + 1 p
3 p	Qualité active	p	adjectif	"agissant"	Ø	"blessant"	"qui" + 1 p
3 n	Qualité active	n	adjectif	Ø	Ø	Ø	"qui" + 1 n
4 p	Qualité passive	p	adjectif	Ø	"utile"	"blessé"	"qui" + 1 p passif
4 n	Qualité passive	n	adjectif	Ø	"inutile"	Ø	"qui" + 1 n passif
5 p	Caractère	p	substantif	Ø	"utilité"	Ø	"caractère" + 4 p
5 n	Caractère	n	substantif	Ø	"inutilité"	Ø	"caractère" + 4 n
6 p	façon	p	adverbe	Ø	"utilement"	Ø	"de façon" + 4 p
6 n	façon	n	adverbe	Ø	"inutilement"	Ø	"de façon" + 4 n

Figure 31

Soient L et L' deux langues Σ et Σ' les ensembles d'unités sémantiques correspondantes : $\Sigma = \{\sigma\}$, $\Sigma' = \{\sigma'\}$

Pour qu'un langage pivot soit possible, il faut donner une relation entre les éléments de Σ et ceux de Σ'. Soit $T \subset \Sigma \times \Sigma'$, qui peut se représenter comme un graphe simple entre Σ et Σ'. Ce graphe doit être tel qu'à tout sommet σ ou σ' il corresponde au moins une arête. De ce fait, il n'est pas possible de définir a priori les ensembles Σ et Σ'. En effet, certaines unités sémantiques d'une langue L peuvent ne pas avoir d'équivalent dans une autre langue. Il faudra donc définir des unités sémantiques nouvelles, dont les expressions se feront par périphrases : ainsi le verbe russe СТОЛКНУТЬСЯ se traduit en français par ENTRER EN COLLISION ce qui impose une nouvelle unité en français. Les contraintes de permanence du nombre d'actants peuvent aussi intervenir pour modifier les unités sémantiques : le verbe intransitif russe ВЫЛЕТАТЬ (mot à mot "sortir en volant") doit se traduire en français par "être émis". Comme il n'y a qu'un seul actant sur ce verbe russe, et deux actants pour le verbe français 'émettre' nous devons le séparer du paradigme de 'émettre' et définir une nouvelle unité sémantique.

Le lexique utilisé au niveau du langage pivot, exprimable dans deux langues, est en fait à l'ensemble $T \subset \Sigma \times \Sigma'$. Le cardinal de T est supérieur à ceux de Σ et Σ'. En effet, certaines unités d'une langue ne sont ambiguës qu'au regard de leurs traductions possibles dans d'autres langues. Ainsi, le mot français "FIL" n'est ambigu que si l'on envisage sa traduction en anglais.

Thread (Fil textile) ou wire (fil métallique)

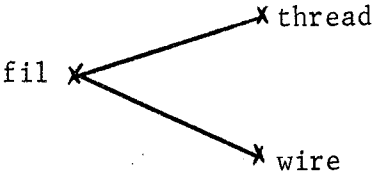


Figure 32

Si l'on considère un ensemble de langues L, L', L"..., le lexique du langage pivot sera une partie de $\Sigma \times \Sigma' \times \Sigma''$...

Si l'on se limite, et c'est notre cas, à envisager la traduction vers une seule langue cible Lc, deux simplifications apparaissent : 1) il n'est pas nécessaire de chercher à recouvrir tous les paradigmes de Lc. On se définira Σ_c uniquement en fonction des unités sémantiques des autres langues. Autrement dit, on peut réduire le vocabulaire de la langue cible.

2) L'on se définira autant de lexiques en langage pivot qu'il y a de langues sources : ce seront donc des ensembles du type $T \subset \Sigma_s \times \Sigma_c$.

Ces simplifications ne suppriment pas l'indépendance du langage, celui-ci n'est lié qu'à l'ensemble Σ_c.

Finalement, une expression du langage pivot est un graphe arborescent, définissant l'articulation des énoncés. Chaque sommet est affecté d'un symbole ou "Etiquette", nom d'un métaprédicat. Deux relations d'équivalence E (élision) et S (substitution) définissent un certain nombre de classes de sommets. A chaque classe est associée au plus une occurrence et une seule, caractérisée par une référence au lexique et un ensemble de valeurs de variables persistantes.

Exemple : "Le cheval de mon père et celui de mon oncle galopent dans le pré qui a été fauché".

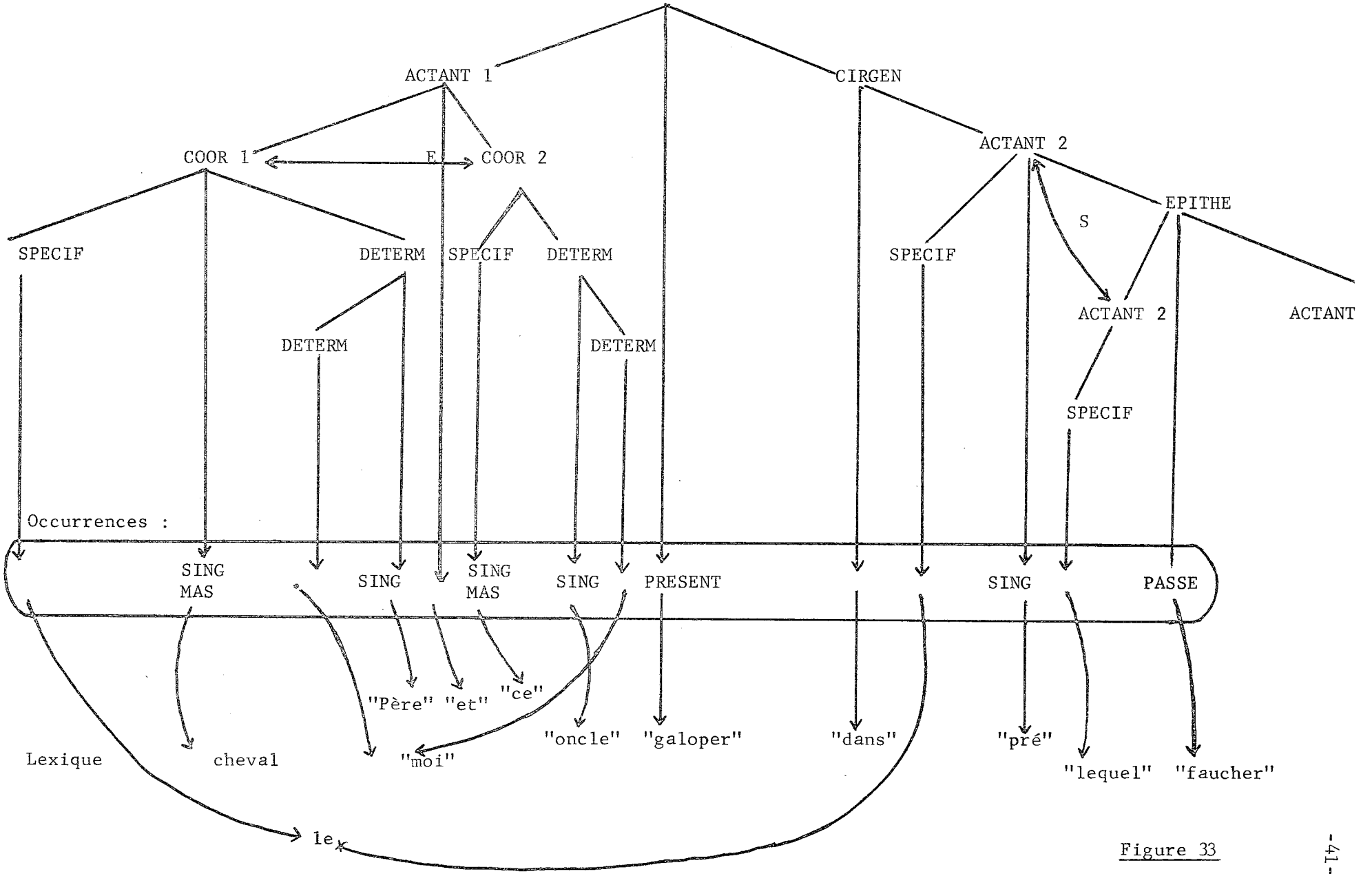

ACTANT 1
CIRGEN
COOR 1
E
COOR 2
ACTANT 2
EPITHE
S
SPECIF
DETERM
SPECIF
DETERM
SPECIF
ACTANT 2
ACTANT
DETERM
DETERM
SPECIF
Occurrences :
SING
MAS
SING
SING
MAS
SING
PRESENT
SING
PASSE
"Père"
"et"
"ce"
"oncle"
"galoper"
"dans"
"pré"
"lequel"
"faucher"
"moi"
Lexique
cheval
le

Figure 33

La figure 34 montre un exemple d'application et les insuffisances du modèle. D'une part, les métaprédicats CIRGEN, DETERM remplacent le métaprédicat EPITHE dans certains cas, d'autre part, le métaprédicat "SPECIF" auquel nous associons une occurrence et une référence au lexique devrait être remplacé dans un modèle de la détermination. La donnée de l'article défini, des démonstratifs et des relatifs, ne devrait pas être lexicale. Ceci se traduit, en particulier, par une impossibilité dans le choix des articles si l'on traduit du russe, qui n'en possède pas, en français.

La première personne du singulier est une unité du lexique. Dans le cas ou un actant est indéterminé (1er actant de "faucher") nous l'indiquons sans référence (actant virtuel) ("que l'on a fauché"). Enfin, les variables, dont nous n'avons marqué que les plus significatives, sont liées aux occurrences. Ceci ne permet pas de tenir compte des variations possibles dans les cas d'élision (les chevaux de mon père et celui de mon oncle). L'information est alors notée sur l'occurrence du démonstratif "celui" (singulier).

D - ENCHAINEMENT DES MODELES [Va_{1} , 2 , 3]

En utilisant une stratification, proche de celle définie par Lamb (La) nous définissons une cascade de modèles successifs. L'ensemble de ces modèles est destiné d'une part à déterminer l'expression en langage pivot correspondant à une phrase de langue source (analyse), d'autre part à produire une phrase en langue cible à partir de cette expression. A chaque modèle correspondent un algorithme, une grammaire et une lexicographie. Matériellement, chaque modèle correspond à une phase du programme, composée d'un ou plusieurs chargements.

La succession effective de ces phases est représentée par la figure 35. Chaque phase comporte deux paramètres, la lexicographie et la grammaire. L'entrée et la sortie sont en fait des pseudotextes codés et structurés. Nous distinguerons grossièrement deux types de structures : les structures en chaîne (cas du texte lui-même par exemple) et les structures arborescentes (structures syntaxiques et langage pivot).

En ne considérant que cet aspect, nous rencontrerons des modèles de type "chaîne → chaîne" (M_1 et M'_1), "chaîne → arbre" (M_2), "arbre → arbre" (M_3) et "arbre → chaîne".

Du point de vue de la réalisation sur ordinateur, les travaux à réaliser pour chaque modèle peuvent se décomposer ainsi : définition et compilation du métalangage de description des grammaires, gestion et mise à jour des fichiers lexicographiques (dictionnaires), réalisation de l'algorithme d'exploitation proprement dit.

La deuxième partie de cet exposé, est consacrée à l'étude du modèle de type "chaîne → arbre" (modèle M_2 , analyse syntaxique) des algorithmes correspondants. Cette phase a pour effet d'associer à toute chaîne codée issue du modèle M_2 (morphologie) une structure syntaxique de dépendance.

La troisième partie étudie les langages et les algorithmes de transformations de type "arbre → arbre" et "arbre → chaîne", appliqués essentiellement aux modèles M_3 et M'_2 .

Les modèles de types chaîne-chaîne, et plus généralement l'organisation générale du système sont décrits en particulier dans [$V_{2,3}$ Co]

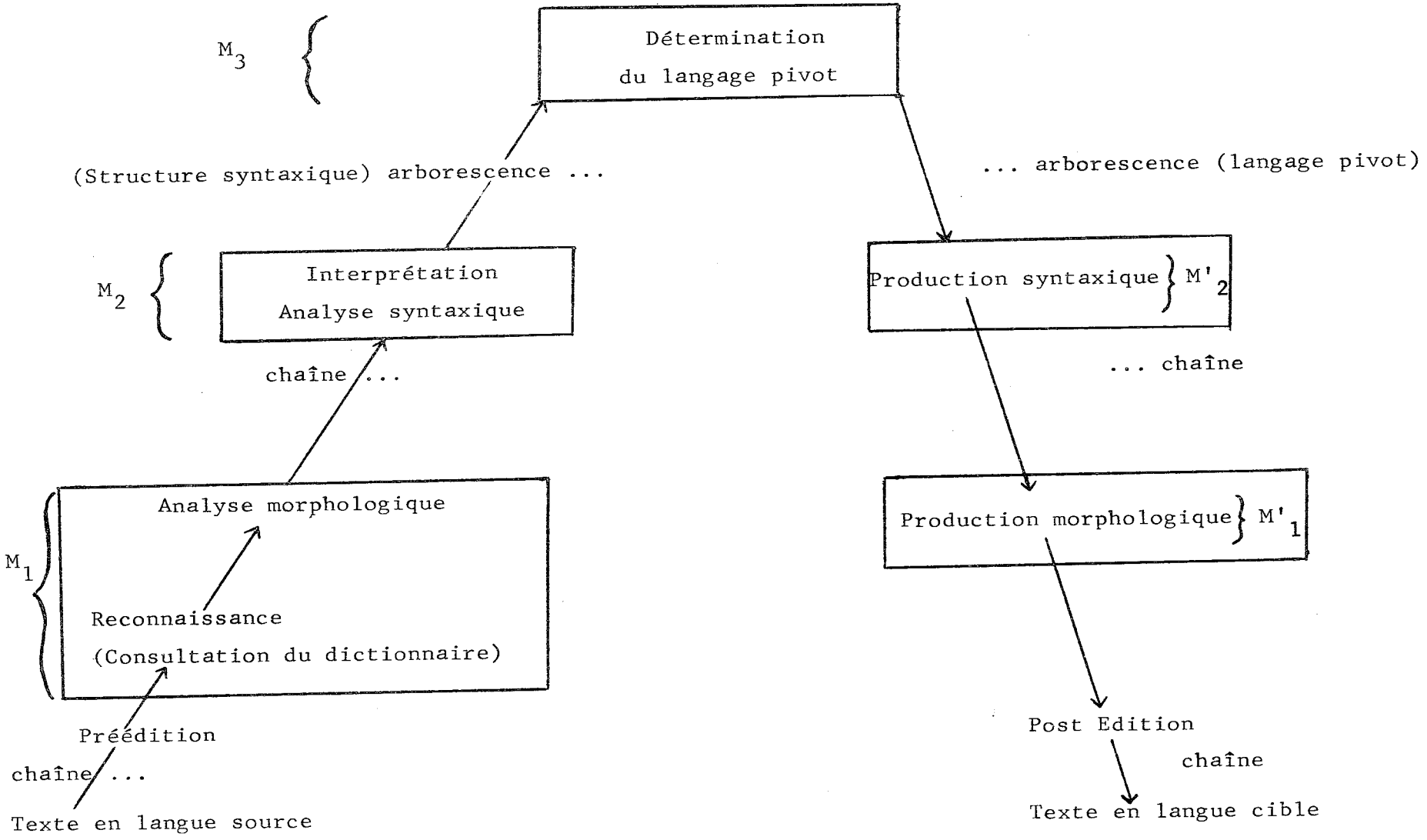

M3
Détermination
du langage pivot
(Structure syntaxique) arborescence ...
... arborescence (langage pivot)
M2
Interprétation
Analyse syntaxique
Production syntaxique
M'2
chaîne ...
... chaîne
Analyse morphologique
M1
Production morphologique
M'1
Reconnaissance
(Consultation du dictionnaire)
Post Edition
Préédition
chaîne ...
chaîne
Texte en langue source
Texte en langue cible

Figure 34

Dans ce chapitre nous définissons la terminologie et les conventions d'écriture que nous utiliserons dans la suite de cet exposé.

1.4.1. TERMINOLOGIE PROPRE AU TRAITEMENT DES LANGUES

1.4.1.1. Corpus

Nous nous proposons de construire un ensemble de modèles de langues naturelles pour établir un processus de traduction des textes d'une langue source vers les textes équivalents en langue cible. Ces modèles ne peuvent être que des approximations et, en particulier, ne prétendent représenter qu'un sous-ensemble de la langue source. Ce sous-ensemble est défini par les textes que nous nous proposons de traiter, qui constituent un "corpus". Cette notion n'existe d'ailleurs qu'au niveau des langues sources (entrée du système), le corpus en langue cible étant défini par les traductions, donc par la langue source.

La mise au point linguistique du modèle est réalisée par l'enrichissement du corpus qui peut provoquer des modifications des grammaires ou des dictionnaires.

1.4.1.2. Forme et occurrence

Etant donné une chaîne de symboles constituant un texte, nous pouvons en extraire un certain nombre de "formes". Chaque forme est une suite maximale de caractères alphabétiques qui peut apparaître dans le corpus. En d'autres termes, une forme est une suite de caractères que l'on peut rencontrer comprise entre deux 'blancs' ou, plus généralement, deux séparateurs.

Nous appelons occurrence d'une forme, ou occurrence toute apparition de la forme dans un texte.

Par extension, toute chaîne de symboles non alphabétique du texte constitue une occurrence (signe de ponctuation, figure, formule mathématique).

1.4.1.3. Informations grammaticales

A chaque occurrence le modèle M1 associe un ou plusieurs groupes d'informations ou "syntagme élémentaire".

Les informations qui constituent ce syntagme sont de deux classes :

a) Les variables, définies par un nom de variable et un ensemble de noms de valeurs.
Un syntagme élémentaire peut comporter plusieurs valeurs de la même variable.
Chaque variable peut être représentée par un vecteur booléen dont chaque élément est nommé par une valeur.

Exemples : variable : genre, nom GNR, valeurs (MAS, FEM, NEU)
variable : nombre, nom NBR, valeurs (SIN, PLUS)

A une occurrence de "FOIS", nous pourrons associer la combinaison :
GNR = (FEM), NBR = (SIN, PLU).

b) Les "types", qui peuvent être définis soit par un nom de type et une liste de valeurs finie, soit par un nom de type et un ensemble infini.
Un syntagme donné ne contient qu'une valeur et une seule de chaque type.

Exemples : type : catégorie, nom CAT, valeurs (SUBC, VERB, ADJ ...)
type : numéro d'unité lexicale, nom UL, valeurs : entiers positifs
type : numéro d'occurrence, nom OC, valeurs : entiers positifs

c) Nature du lexique

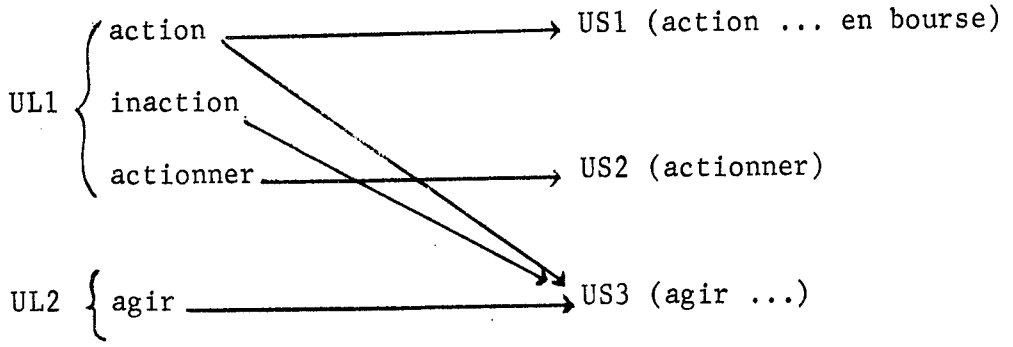
Au niveau du modèle M1 (ou M'1), le lexique est constitué d'un ensemble de racines, dont on peut dériver un certain nombre de formes. L'ensemble des formes constituant un paradigme complet (c'est-à-dire permettant de parcourir toutes les valeurs de variables et de types admises) correspond à une unité lexicale (type UL).

Au niveau du modèle M3 ou M'2, nous faisons appel à un paradigme plus étendu (voir figure 32), qui caractérise l'"unité sémantique" (type US).

L'unité lexicale n'est définie qu'en fonction de considérations formelles ou grammaticales. L'unité sémantique fait intervenir la signification.

Exemple :

Action, inaction, actionner peuvent appartenir à la même unité lexicale. Au niveau des unités sémantiques, nous distinguerons trois unités correspondant à [agir, action, inaction] , [action (en bourse)] , [actionner]



La distinction entre les divers US associés à un UL est marquée par des informations grammaticales, notamment le type "dérivation" Δ qui caractérise les passages formels tels que "action" → "inaction", "action" → "actionner". Dans le cas ou cette distinction n'est pas marquée ("action en bourse" ou "agir"), nous avons une polysémie qui peut être distinguée au niveau M3, en utilisant les "codes d'étiquetage" propres aux unités US1 et US3. Bien entendu, cette ambiguïté n'est pas toujours résolue.

Pour une langue source donnée, le lexique est finalement fractionné en trois parties distinctes : le dictionnaire morphologique, le dictionnaire syntaxique, le dictionnaire d'étiquetage. La référence commune à ces trois dictionnaires est constituée par le numéro d'unité lexicale et l'ensemble des informations grammaticales.

Le dictionnaire morphologique permet de coder (modèle M1) les syntagmes élémentaires de chaque forme.

Le dictionnaire syntaxique fournit des informations complémentaires propres au comportement syntaxique.

Le dictionnaire d'étiquetage contient la liste des codes d'étiquetage et des US associés à chaque UL.

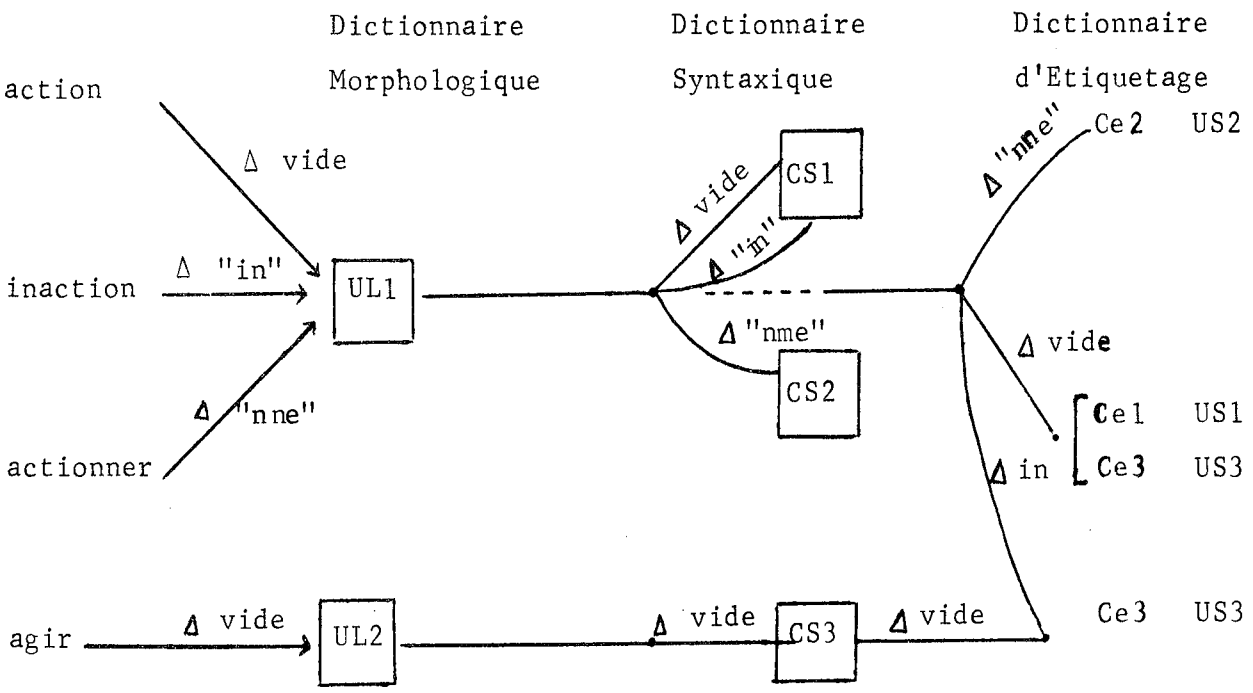


Figure 35

CS1 étant, par exemple, un code vide (substantif courant)

CS2 caractérisant les verbes transitifs

CS3 caractérisant les verbes gouvernant la préposition "sur"

d) Syntagme élémentaire

Un syntagme élémentaire (sortie de M1, entrée de M'1), est formé par une concaténation de valeurs de types et de variables comportant :

le numéro d'occurrence (OCC) par exemple : 63

le nom de catégorie (CAT) par exemple : SUBC

le nom de dérivation (Δ) par exemple : in

un ensemble de valeurs de variables, par exemple : { FEM, SIN }

le numéro d'unité lexicale, par exemple : 582

Plusieurs syntagmes élémentaires peuvent porter le même numéro d'occurrence, ils se distinguent alors par au moins une valeur de type ou de variables ("homographie"). Ainsi, la forme "BOIS" correspondra à au moins deux syntagmes, l'un contenant SUBC et (MAS, SIN, PLU), l'autre contenant VERBE et

[IND, PRES, UN, DEUX], les numéros d'unités lexicales étant évidemment différents.

1.4.2. NOTATION DES ARBORESCENCES

1.4.2.1. Arborescence [Be]

Une arborescence A est un graphe fini (X,U), X ensemble des sommets fini non vide, $U \subset X \times X$ tel que :

a) Le graphe est connexe

b) Pour tout sommet $x \in X$ il existe au plus un sommet y tel que $(x,y) \in U$

c) Il existe un sommet unique x_0 tel qu'il n'existe aucun sommet $y \in X$ tel que $(x,y) \in U$

Nous appellerons racine le sommet x_0.

Nous appellerons arc tout élément $u = (x,y) \in U$.

Si $(x,y) \in U$, nous noterons aussi $x\ U\ y$.

Nous noterons $(X,\hat{U})$ le graphe de fermeture transitive de (X,U) ; un tel graphe est demi-latticiel, selon la relation de préordre $\hat{U}$. A tout sous-ensemble $X' \subset X$ nous associerons une borne $b \in X$.

Nous appellerons sommets terminaux ou feuilles les sommets $x \in X$ tels qu'il n'existe pas d'arc $(y,x) \in U$.

Nous noterons U^- la relation inverse de U, $\hat{U}^-$ sa fermeture transitive.

1.4.2.2. Sous-arborescence

- Etant donnée une arborescence A = (X,U), nous appellerons sous-arborescence de A toute partie connexe de A.

- Sous-arborescence complète

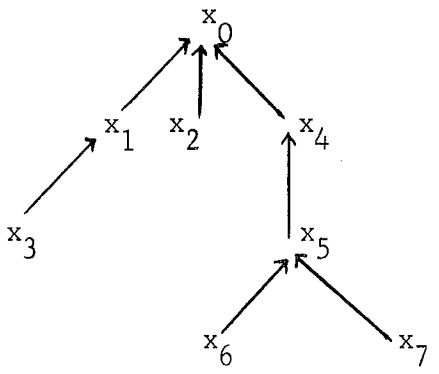
Etant donnée une arborescence A = (X,U) de racine x_0, une sous-arborescence complète A' = (X',U') de A est une sous-arborescence de A telle que pour tout $x \in X'$, $\{y \mid x\, U'^{-1}\, y\} = \{y \mid x\, U^{-1}\, y\}$

- Sous-arborescence propre

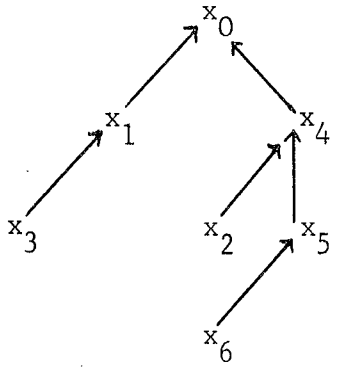
Etant donnée une sous-arborescence A = (X, U) de racine x_0, nous appellerons sous-arborescence propre toute sous-arborescence complète de A de racine x'_0 telle que $x'_0\, U\, x_0$.

Exemple :

Arborescence :



Sous-arborescence :



Sous-arborescence complète :

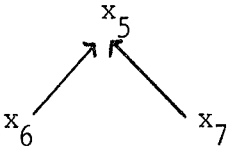


Figure 36

Sous-arborescences propres :

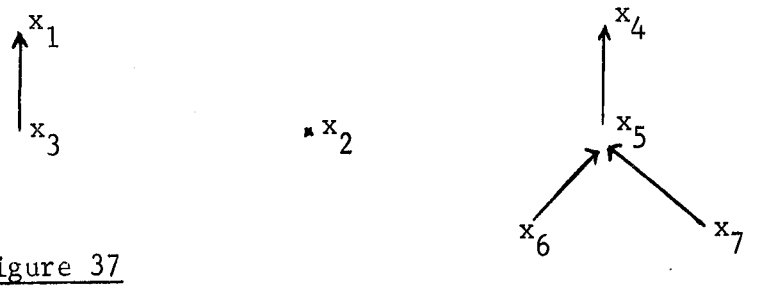


Figure 37

- A tout sommet y ∈ X, nous associerons l'ensemble des sommets de la sous-arborescence complète dont y est racine, que nous noterons Xy ⊂ X si A = (X,U).

1.4.2.3. Arborescence étiquetée

Nous appellerons arborescence étiquetée une arborescence dont chaque sommet x est affecté d'un nom n(x) ∈ V, V étant un vocabulaire fini et n une application de X sur une partie de V.

1.4.2.4. Borne

La fermeture transitive de la relation U définit une relation d'ordre, et la structure associée à l'arborescence est une structure de demi treillis.
A tout sous-ensemble de sommets nous pouvons associer une borne unique dans cette structure. Par extension, nous appellerons borne d'un sous-ensemble S de sommets d'une arborescence la borne de S dans le demi treillis associé.

1.4.2.5. Arborescence orientée

Nous appellerons arborescence orientée une arborescence (X, U) munie d'une relation d'ordre partiel O telle que les restrictions de O aux ensembles E_x $\{y \mid (y , x) \in U\}$ sont des ordres totaux, et que les éléments y, z, $y \in E_x$, $z \notin E_x$, ne sont pas comparables par O.

1.4.2.6. Nous appellerons pseudo-arborescence sur V une arborescence étiquetée sur V et orientée. Nous noterons $\mathcal{A}$ (V) l'ensemble des pseudo-arborescences sur V.

1.4.3. LANGAGE

Nous nous limiterons à donner ici les principales définitions et les notations que nous utiliserons sans donner toutes les propriétés et leurs démonstrations, qui sont bien connues ($G_{i\ 1}$).

1.4.3.1. Etant donné un ensemble V, ou vocabulaire, et une opération associative sur V, notée (concaténation) telle que : a. b = ab , a ϵ V, b ϵ V, et qu'il existe un élément ξ unique, tel que ξ. a = a. ξ = a (élément neutre), nous noterons V^* le monoïde libre sur V engendré par cette opération.

Nous appellerons mot sur V tout élément de V^* , soit :

$m = v_1\ v_2 \cdots v_p$, $v_i \epsilon V$

Nous appellerons v_i occurrence de v ϵ V dans m. Un langage L est une partie de V^*. Par définition, la longueur d'un mot m, soit ℓ (m) est le nombre d'occurrences de m, ℓ (m) = p

1.4.3.2. Grammaire

Une grammaire générative G est un 4 - uple. (V, V_N, S, P) tel que V et V_N sont des ensembles finis, $V \cap V_N = \emptyset$, S ϵ V_N , P est un ensemble de couples (ϕ, ψ) notés aussi $\phi \rightarrow \psi$ tels que $\phi \epsilon V^*_N$, $\psi \epsilon (V_N \cup V)^*$.

1.4.3.3. Dérivation

Etant données deux chaines φ et ψ sur $(V \cup V_N)^*$ et une grammaire G, nous noterons $\varphi \overset{G}{\Rightarrow} \psi$ si $\varphi = \psi$ ou si il existe χ_1 et χ_2 telles que $\varphi = \chi_1 \varphi' \chi_2$ $\psi = \chi_1 \varphi' \chi_2$ et si $(\varphi' , \psi') \epsilon P$.

Nous noterons $\varphi \overset{G*}{\Rightarrow} \psi$ la fermeture transitive de cette relation, c'est-à-dire si il existe $\varphi_1 , \varphi_2 \cdots \varphi_p$ telles que

$$\varphi_i \overset{G}{\Rightarrow} \varphi_{i+1} \quad \text{et } \varphi_1 = \varphi , \varphi_p = \psi$$

Le langage L associé à la grammaire G est

$$L_G = \{\varphi \mid S \overset{G*}{\Rightarrow} \varphi\} \cap V^*$$

1.4.3.4. Une grammaire hors contexte est une grammaire G dans laquelle les productions de P sont du type $A \rightarrow \theta$, $A \in V_N$.

A chaque dérivation $S \overset{*}{\Rightarrow} \varphi$ d'une telle grammaire, on associe une pseudo-arborescence sur $V_N \cup V \cdot [G_i , P_a]$

Exemple : Soit G (V, V_N, S, P) telle que :

$V \{a, s, b\}$, $V_N = \{A, S, B\}$

P = S → ASB
S → s
A → a
B → b

et la dérivation S ⇒ ASB ⇒ aSB ⇒ ... ⇒ aasbb

La pseudo-arborescence associée s'écrit :

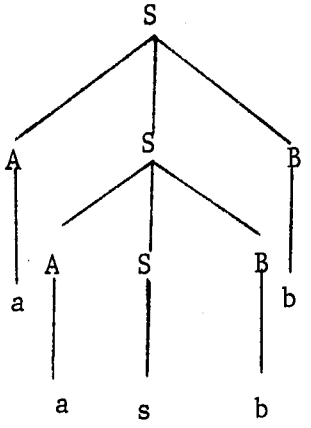


Figure 38

1.4.4. AUTOMATE A PILE

Un automate à pile est un 7-uplet (Q, V, V_A, δ , q_0 , Z_0, F) dans lequel :

Q est un ensemble fini d'états
V est un alphabet fini (alphabet d'entrée)
V_A est un alphabet auxiliaire (alphabet de pile)
$q_0 \in$ Q est l'état initial
$Z_0 \in V_A$ est un symbole spécial du vocabulaire V_A (symbole initial de la pile)
F $\subseteq$ Q est l'ensemble des états finaux
δ est une application de Q x (V $\cup$ {ε} x V_A dans les parties finies

Le langage accepté par un automate à pile est défini de la façon suivante : Soient $a \in V \cup \{\varepsilon\}$, γ et $\beta \in V_A^*$, $Z \in V_A$, si $(p, \beta) \in \delta(q, a, Z)$ nous noterons

$$(a, q, Z\gamma) \underset{m}{\vdash} (p, \beta\gamma)$$

et par transitivité, pour $W \in V^*$,

$$(W, q_1, \gamma_1) \overset{*}{\underset{m}{\vdash}} (q_{n+1}, \gamma_{n+1})$$

Si $W = a_1 \ldots a_n$, et si il existe $q_1, q_2 \cdots q_{n+1}$ et $\gamma_1, \gamma_2 \cdots \gamma_{n+1}$ tels que :

$$(a_i, q_i, \gamma_i) \underset{m}{\vdash} (q_{i+1}, \gamma_{i+1})$$

Le langage accepté est :

Soit $T(m) = \{W \mid (q_0, Z_0) \overset{*}{\underset{m}{\vdash}} (q, \gamma)$, $q \in V\}$ (accepté par état final)

Soit $N(m) = \{W \mid (q_0, Z_0) \overset{*}{\underset{m}{\vdash}} (q, \varepsilon)$, $q \in Q\}$ (accepté par pile vide)

1.4.5. TRANSDUCTEUR A PILE

Un transducteur à pile associé à un automate à pile est un 7-uplet $T = (Q, V, V_A, V_S, \mu, Z_0, Q_0)$ dans lequel : Q, V, V_A, Z_0, Q_0 ont la même signification que pour un automate à pile.

V_S est un vocabulaire (sorties)

μ est une application de $Q \times (V \cup \{\varepsilon\}) \times \delta$ dans les parties finies de $Q \times V_A^* \times V_S^*$

C H A P I T R E II

MODELES D'ANALYSE (MODELES CHAINE-ARBRE)

Dans ce chapitre, nous nous proposons de montrer comment nous pouvons associer à une phrase du texte une structure interprétable, c'est-à-dire une structure à partir de laquelle nous pourrons calculer une expression du langage pivot. Nous chercherons, bien entendu, à nous rapprocher le plus possible de ce langage . C'est pourquoi la structure que nous obtenons est, formellement, de même nature que l'expression en langage pivot. Cependant, les étiquettes seront des noms de fonctions syntaxiques en langue source, les occurrences (et le lexique), sont ceux de la phrase en langue source, et il n'y aura pas de relation d'élision ou de substitution. De plus, les variables sont elles aussi des variables de surface de la langue source : par exemple le genre de 'table' serait féminin, le temps de "a" dans "il a composé" serait 'présent' etc...

Une telle structure est alors équivalente à une structure de dépendances, étiquetée, ce que nous nous proposons de rechercher ici.

2.1. GRAMMAIRE DE CONSTITUANTS ET GRAMMAIRE DE DEPENDANCES

2.1.1. STRUCTURES

2.1.1.1. Structures de constituants

Définition : Etant donnée une phrase $\varphi = x_1 \ldots x_n$, nous appellerons structure de constituants sur φ un graphe arborescent (Z, U) tel que :

1° $Z = X \cup Y$, $X = \{x_i\}$ ensemble des occurrences de φ , $Y \cap X = \emptyset$

2° La relation U est telle que X soit l'ensemble des sommets terminaux de l'arborescence.

2.1.1.2. Structure de constituants continue

Nous dirons qu'une structure de constituants est continue si, pour tout triplet (x_i, x_j, x_k) tel que $\min(i, j) < k < \max(i, j)$, l'on a x_i U z et x_j U z implique x_k U z.

Exemple :

Structure continue

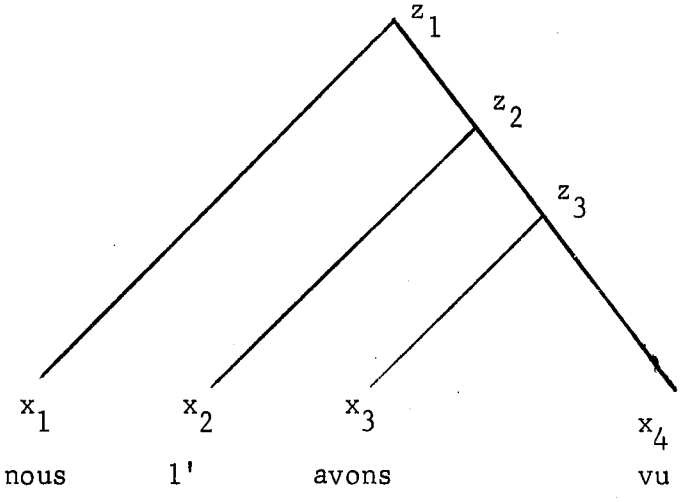


Figure 39

Structure discontinue :

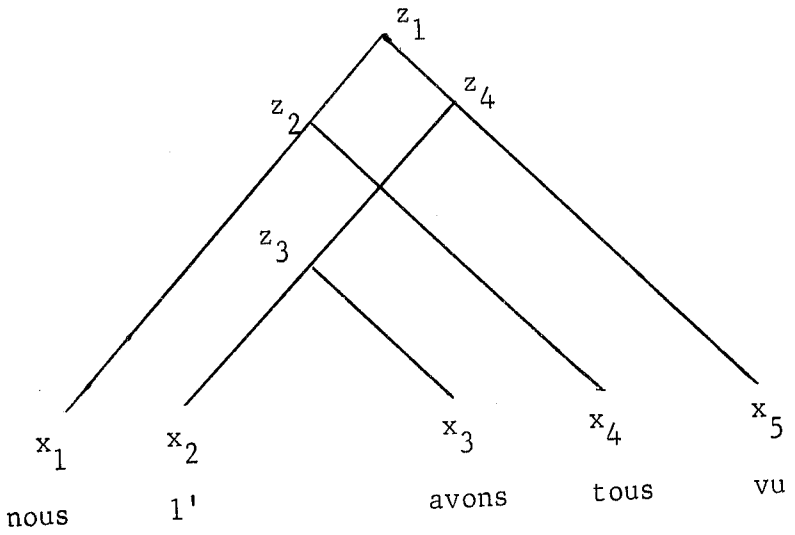


Figure 40

2.1.1.3. Structure de dépendances

Une structure de dépendance sur une phrase $\varphi = x_1\ x_2 \cdots\ x_n$, est un graphe arborescent (X, D), $X = \{x_i\}$ ensemble des occurrences de φ

2.1.1.4. Structure de dépendances projectives (M_a)

Une structure de dépendance est projective si, pour tout triplet (x_i, x_j, x_k) , tel que min (i, j) < k < max (i, j), $x_i\ D\ x_j$ implique $x_k\ \hat{D}\ x_j$

Exemples :

Structure projective

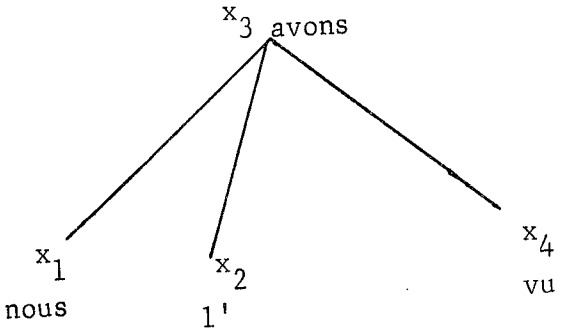


Figure 41

Structure non projective :

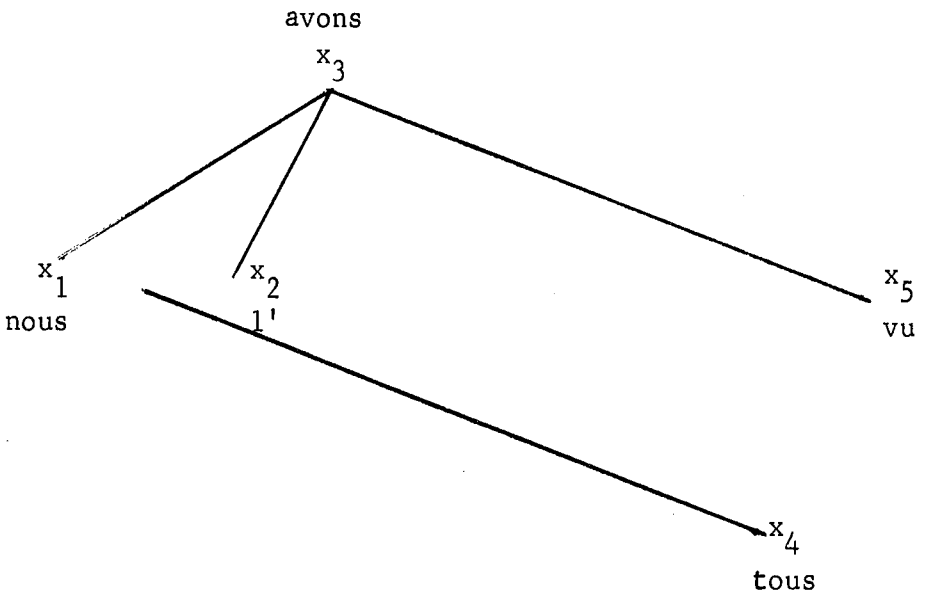


Figure 42

2.1.1.5. Dans une structure de dépendance (X, D), si x D y nous dirons que x est dépendant de y et que y est gouverneur de x

2.1.2. STRUCTURES DE CONSTITUANTS ASSOCIEES A UNE GRAMMAIRE HORS CONTEXTE

Etant donnée une grammaire hors contexte G, et une dérivation $S \overset{*G}{\Rightarrow} \varphi$ on peut associer à cette dérivation un "arbre de dérivation", qui est une structure de constituants continue. (1.4.3.4.)

2.1.3. GRAMMAIRES DE DEPENDANCES [Hy_1 , Gf]

2.1.3.1. Une grammaire de dépendances sur un vocabulaire V est constituée par la donnée d'une famille de parties C_i de V, telles que $\bigcup C_i = V$, et d'un ensemble de règles des deux formes suivantes :

1) $* (X)$

2) $X (X_1 X_2 \ldots X_i * X_{i+1} \ldots X_n)$

Chaque partie constitue une 'catégorie syntaxique'. Elle est caractérisée par son nom de catégorie. Nous noterons par une lettre majuscule ces catégories Ainsi les symboles X_i se représentent des noms de catégories syntaxiques. Par convention nous utiliserons le même symbole pour noter la catégorie ou son nom.

2.1.3.2. Langage produit par une grammaire de dépendances

A toute grammaire de dépendances G_d, nous associons une grammaire de constituants $G = (V, V_N, S, P)$ définie ainsi :

Soit $G = (V, V_N, S, P)$

$V_N = \{C_i\}$ ensemble des catégories

A toute règle de dépendance, on associera une production

à $*(X)$ nous associerons $S \rightarrow X$

et à $X(X_1 X_2 \ldots X_{i-1} * X_{i+1} \ldots X_n)$ $\quad X \rightarrow X_1 X_2 \ldots X_{i-1} x X_{i+1} \ldots X_n$

pour tout $x \in X$

Par définition, nous dirons que le langage $L_{G_d} = L_G$

2.1.3.3. Structure

A une dérivation de la grammaire hors contexte ainsi définie, nous pouvons associer une structure de dépendances. Soit (Z, U), $Z = X \cup Y$ la structure de constituants associée, nous lui ferons correspondre la structure de dépendances (X, D) telle que $x_i \; D \; x_j$ si et seulement si il existe y_i et y_j tels que $(x_i \; U \; y_i)$ $(y_i \; U \; y_j)$ et si $x_j \; U \; y_j$.

La structure de constituants est telle que pour tout sommet non terminal y, il existe un sommet terminal x unique tel que x U y. Pour tout sommet x_i , nous

trouverons un sommet x_j unique satisfaisant la condition, sauf pour la racine.

exemple :

Soit la grammaire de dépendances :

$V = \{a, b\}$ $\quad c = \{A, B\}$ $\quad A = \{a\}$ $\quad B = \{b\}$

$\ast$ (A)

A ($\ast$ B)

B (A $\ast$)

B ($\ast$)

Nous lui associerons la grammaire hors-contexte suivante :

$V = \{a, b\}$ $\quad V_N = \{S, A, B\}$

$S \longrightarrow A$

$A \longrightarrow a\ B$

$B \longrightarrow A\ b$

$B \longrightarrow b$

Soit la dérivation $S \overset{*}{\Longrightarrow}$ a a a b b b, nous aurons la structure de constituants :

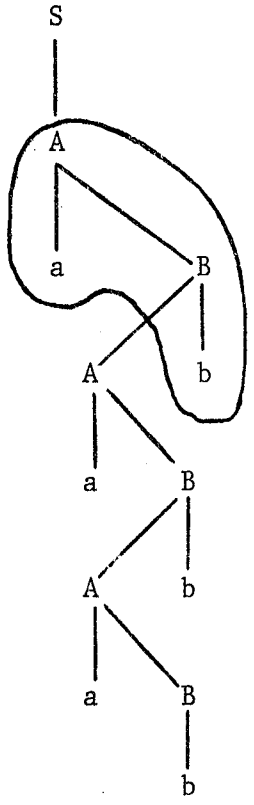


Figure 43

et la structure de dépendances

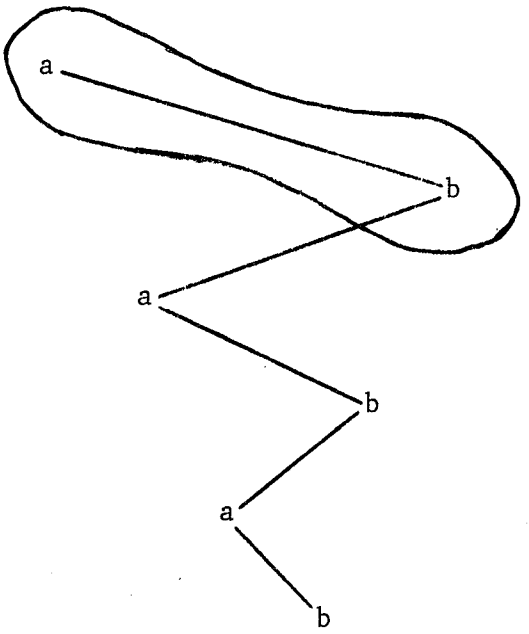


Figure 43 bis

2.1.3.4. La structure de dépendances associée est projective

Ceci se déduit de la continuité de la structure de constituants. En effet, x_i D x_j correspond à x_i U y_j et x_j U y_j. Soit x_k, tel que $\min(i, j) < k < \max(i, j)$, la continuité implique que x_k U y_j , dont x_k D x_j.

2.1.4. COMPARAISONS ENTRE GRAMMAIRES DE CONSTITUANTS ET GRAMMAIRES DE DEPENDANCES

Gaifman a étudié [Gf] les équivalences entre grammaires de constituants et de dépendances. Cependant, les comparaisons restent purement théoriques et ne concernent que la puissance des deux types de grammaires.

2.1.4.1. Equivalences au sens de Gaifman et de Hays

En théorie des langages, on dit que deux grammaires sont équivalentes si elles décrivent le même langage. Elles sont fortement équivalentes si les structures associées aux dérivations sont les mêmes.

Pour définir une équivalence forte entre grammaire de constituants et grammaires de dépendances, il est nécessaire de définir des structures de même type.

L'étude de Gaifman considère tout d'abord les structures telles que nous les avons définies en 2.1.3.2. et 2.1.3.3. Ceci restreint considérablement les grammaires hors contexte, puisque chaque règle de production doit comporter un élément du vocabulaire terminal unique dans sa partie droite.

Un deuxième type de correspondance a été proposé par Hays et analysé par Gaifman.

a) Une arborescence de type C = (X $\cup$ Y, U) et une arborescence de type D = (X', D) correspondent si X = X', c'est-à-dire si l'ensemble des sommets terminaux de C est identique à l'ensemble des sommets de D.

b) Nous dirons qu'une structure de dépendances D et une structure de constituants C sont en correspondance si à toute sous arborescence complète (1.3.2) de D correspond une sous arborescence de C et si à toute sous arborescence de C correspond une sous arborescence de D

Exemple :

Structure C_1

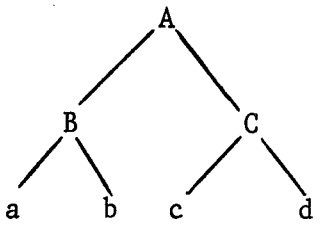


Figure 44

Structure D_1 en correspondance

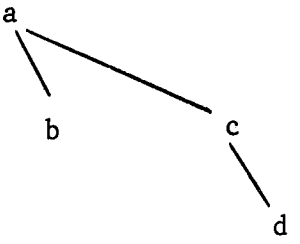
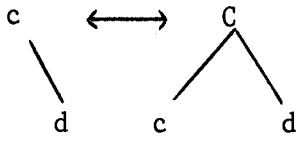
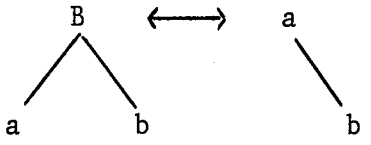


Figure 44 bis

car :



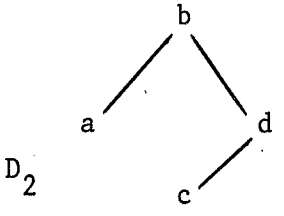
et



... etc

Plusieurs structures D peuvent correspondre à une même structure C :

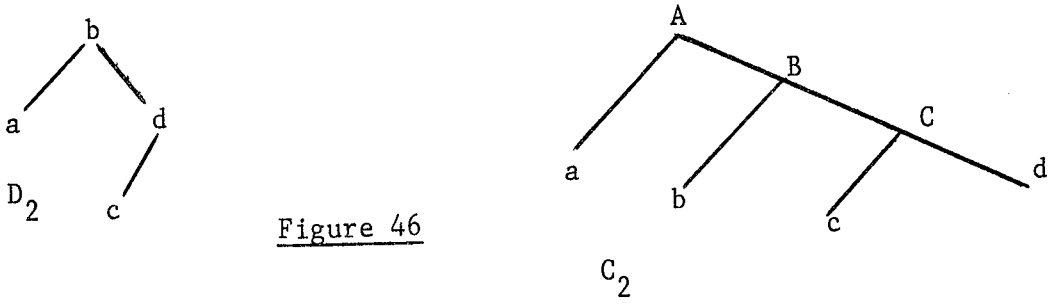
par exemple D_2 correspond aussi à C_1.



D_2

Figure 45

Inversement, la structure de dépendances D_2 est aussi en correspondance avec C_2



D_2 C_2

Figure 46

Nous appellerons degré d'une arborescence la longueur du plus court chemin allant d'une feuille à la racine. Le degré d'une grammaire est la borne supérieure du degré de ses arbres de dérivation. Dans le cas des correspondances entre grammaire de constituants et grammaire de dépendances, Gaifman a montré que seules les grammaires hors contexte de degré fini pouvaient correspondre à des grammaires de dépendances

Soit la grammaire hors contexte :

$V_N = \{S, A, B\}$ $V = \{a, b\}$ P :

$S \longrightarrow SS$

$S \longrightarrow AB$

$A \longrightarrow a$

$B \longrightarrow b$

de degré infini

Les dérivations sont de la forme :

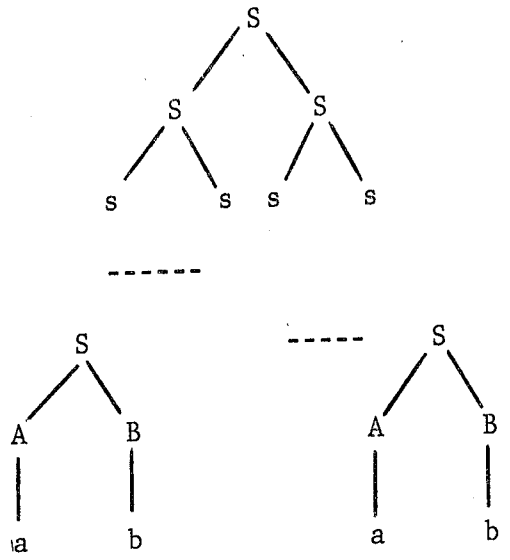


Figure 47

Aucune longueur de chemin n'est bornée dans le graphe correspondant, et les structures de dépendance en correspondance, telles que :

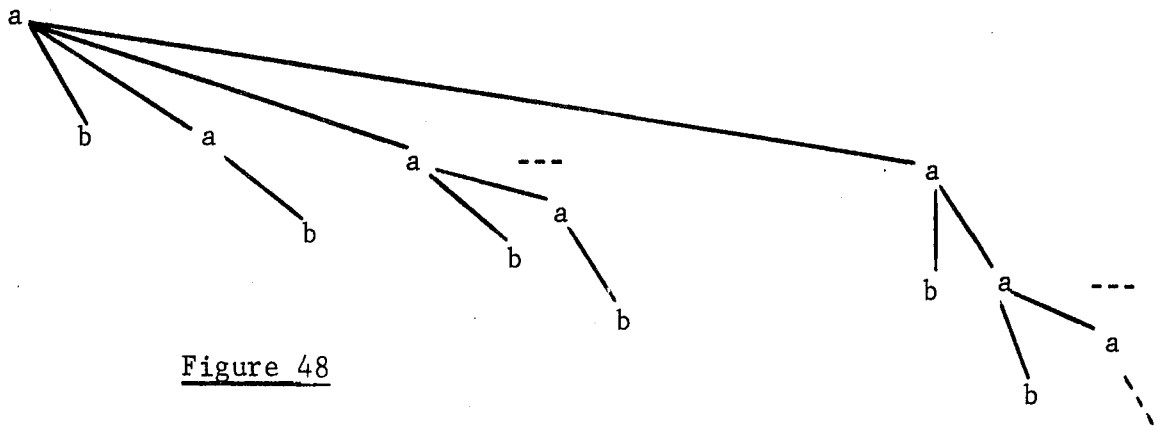


Figure 48

nécessitent une grammaire contenant des règles :

a (* b)

a (* ba)

a (* baa)

---

a (* ba... a) en nombre non fini.

Cette contrainte n'est pas une contrainte de puissance, toute grammaire hors contexte est faiblement équivalente à une grammaire hors contexte de degré fini (il existe un chemin borné dans l'arborescence).

2.1.4.2. Comparaison entre les deux types de grammaires

Notre but étant d'obtenir une structure de dépendances, nous pourrions chercher à utiliser directement une grammaire de ce type. Mais les restrictions données en 4.1. sont en fait beaucoup plus contraignantes. En effet, la grammaire de dépendance doit contenir autant de règles qu'il y a de configurations possibles de dépendants sous un gouverneur donné.

Si, par exemple, nous voulions construire l'ensemble des groupes nominaux français, munis ou non d'un article et de un, deux, ou trois adjectifs, tels que 'le beau petit chien noir', la grammaire de dépendances serait :

$X = \{chien\}$ $X_1 = \{le\}$, $X_2 = \{beau\}$, $X_3 = \{petit\}$ $X_4 = \{noir\}$

et

X (*)	'chien'
X (X_1 *)	'le chien'
X (X_2 *)	'beau chien'
X (X_1 X_2 *)	'Le beau chien'
X (X_3 *)	'petit chien'
X (X_1 X_3 *)	'le petit chien'

X (X_1 X_2 X_3 * X_4)	'le beau petit chien noir'

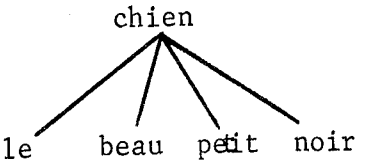


Figure 49

Une grammaire hors contexte de même puissance serait

1 $X \longrightarrow x_1\ X^1$

2 $X^1 \longrightarrow x_2\ X^2$

3 $X \longrightarrow x_2\ X^2$

4 $X^1 \longrightarrow x_3\ X^3$

5 $X^2 \longrightarrow x_3\ X^3$

6 $X \longrightarrow x_3\ X^3$

7 $X \longrightarrow x\ x_4$

8 $X^1 \longrightarrow x\ x_4$

9 $X^2 \longrightarrow x\ x_4$

10 $X^3 \longrightarrow x\ x_4$

11 $X \longrightarrow x$

12 $X^1 \longrightarrow x$

13 $X^2 \longrightarrow x$

14 $X^3 \longrightarrow x$

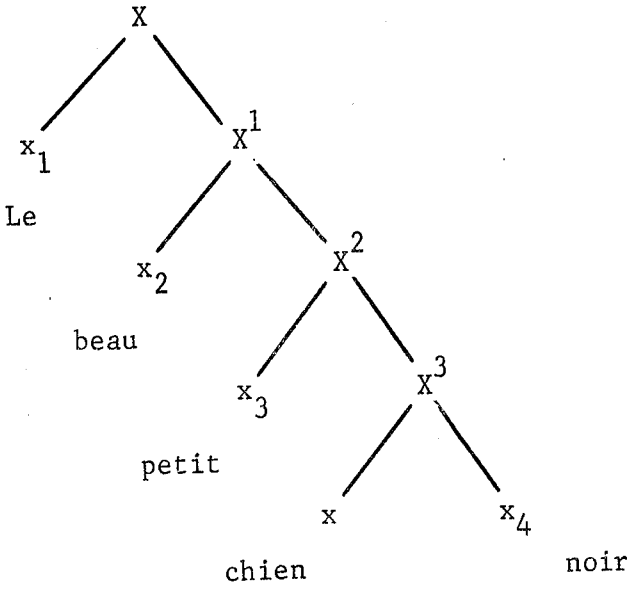


Figure 50

Nous avons 2^n règles de dépendances, et seulement $\frac{n\ (n+3)}{2}$ règles hors contexte.

De plus, bien que cela ne soit peut être pas significatif en linguistique, il peut être intéressant d'admettre un nombre de constituants non borné. Ce serait par exemple le cas des compléments circonstanciels.

Ces dernières considérations montrent qu'il est moins intéressant d'utiliser une grammaire de dépendance pour la mise au point d'un modèle réel. Nous pourrions étudier une telle grammaire, mais nous devrions pour cela effectuer un dénombrement complet de toutes les configurations. Notre but n'est pas de décrire une langue, mais simplement d'associer une structure au phrases. De ce fait, le modèle peut représenter plus de phrases que n'en comprend la langue. L'utilisation d'une grammaire hors contexte permet de réaliser une telle stratégie, en permettant de ne pas borner le nombre des dépendants, et en n'effectuant pas tous les contrôles.

Une telle méthode est utile si l'on désire construire un modèle génératif, contrôlant parfaitement toutes les constructions permises. Ici, nous désirons seulement associer à chaque phrase d'un texte une structure interprétable linguistiquement.

Il n'est donc pas nécessaire de décrire exactement la langue, mais plutôt d'accepter toutes les phrases qu'elle utilise.

Une grammaire hors contexte permet de réaliser au mieux une telle stratégie. Le langage décrit peut être plus vaste que la langue naturelle, ce qui n'est guère possible avec une grammaire de dépendances.

2.2. TRANSFORMATION DU GRAPHE DE CONSTITUANTS EN GRAPHE DE DEPENDANCES

2.2.1. PRINCIPE GENERAL

Nous avons montré la plus grande efficacité d'un modèle hors contexte pour l'écriture d'une grammaire de reconnaissance. Cependant, nous recherchons une structure de dépendances. C'est pourquoi, sans considérer l'équivalence des grammaires, nous nous bornerons à extraire une structure de dépendances à partir de la structure de constituants. Il suffit, pour cela, de définir une correspondance entre les deux structures. Cette correspondance sera la suivante :

Etant donnée une structure de constituants ($X \cup Y$, U) nous lui ferons correspondre une structure de dépendances (X, D) par une application f de $X \cup Y$ sur une partie de X telle que :

1° pour tout $y \in Y$, f (y) = x tel que (f^{-1} (x), U) soit connexe

2° la relation D est telle que : x U y implique f (x) = f (y) ou f (x) D f (y)

En d'autres termes, nous définissons une partition P de ($X \cup Y$) et la relation D est définie comme la relation quotient U/P

Exemple :

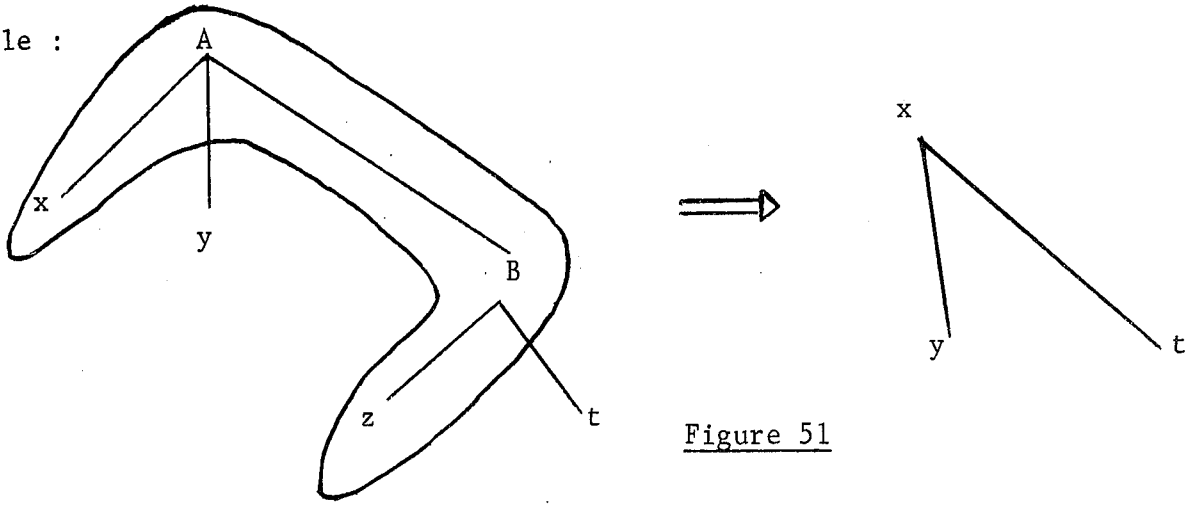


Figure 51

Une telle structure de dépendances ne tient pas compte de toutes les occurrences de la phrase. Ceci nous permet de définir des occurrences complexes (ainsi x-z dans l'exemple de la figure 51) que nous rencontrons en particulier dans le cas desparticules séparables en allemand. Les transformations de ce type nous permettraient sans doute d'accéder à une formule proche de celles du langage pivot.

En pratique, dans le cadre du système actuel, nous avons préféré limiter l'application f de la façon suivante :

f est une application de (X ∪ Y) sur X telle que :

1° pour tout x ∈ X, f(x) = x

2° pour tout y ∈ Y, f(y) = x tel que ($f^{-1}(x)$, U) soit un ensemble totalement ordonné par U.

3° La relation D est telle que : XUY implique f(x)=f(y) ou f(x)Df(y).

La condition 2 impose que f^{-1} (x) soit un chemin de la structure de constituants (XUY, U).

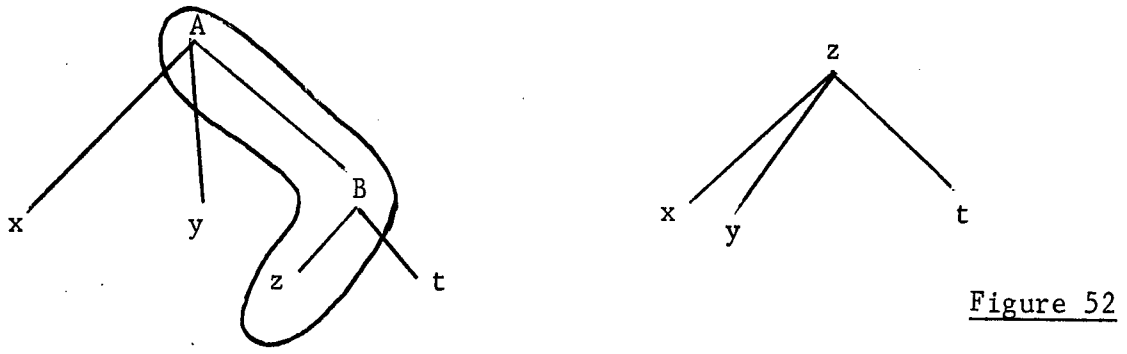


Figure 52

2.2.2. PROPRIETES DE LA CORRESPONDANCE

2.2.2.1. La correspondance obtenue en limitant la condition 2 aux chemins est la correspondance de Gaifman et Hays (2.1.4.1.)

En effet, soient C et D les deux structures. Soit une sous arborescence complète de racine z dans C. D'après la condition 3, si il existe un chemin de z' à z, soit z' $\hat{U}$ z, il existe un chemin de f (z') à f (z), soit f (z') $\hat{D}$ f (z).
Si x_0 = f (z), nous aurons x $\hat{D}$ x_0 pour tout sommet x de l'arborescence de racine z, donc une sous arborescence de D de racine x_0.

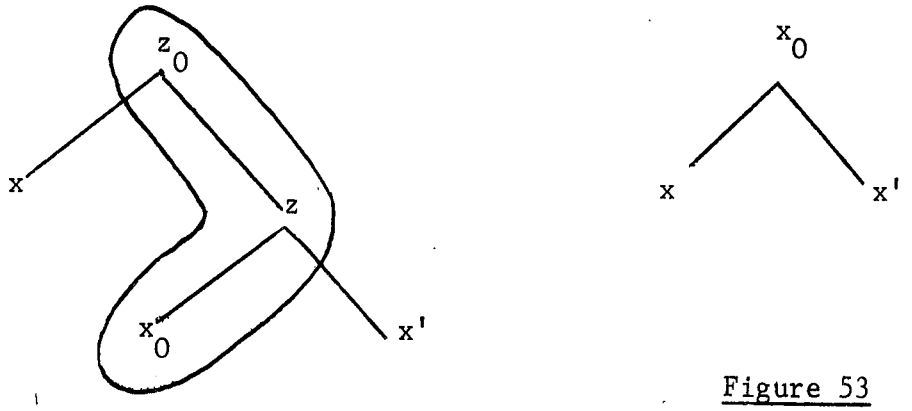
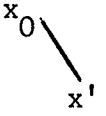


Figure 53

La sous arborescence  correspond à z.

Soit maintenant une sous arborescence complète de C, de racine x_0. $f^{-1}(x_0)$ constitue un chemin d'extrémité z_0.

Soit x élément de la sous arborescence de racine x_0. Nous avons $x \hat{D} x_0$. La relation x D x' signifie qu'il existe $z \in f^{-1}(x)$ et $z' \in f^{-1}(x')$ tels que z U z'; z étant sur un chemin d'origine x, $x \hat{U} z$. Nous aurons donc $x \hat{U} z_0$ si $x \hat{D} x_0$, et x appartient à la sous arborescence z_0.

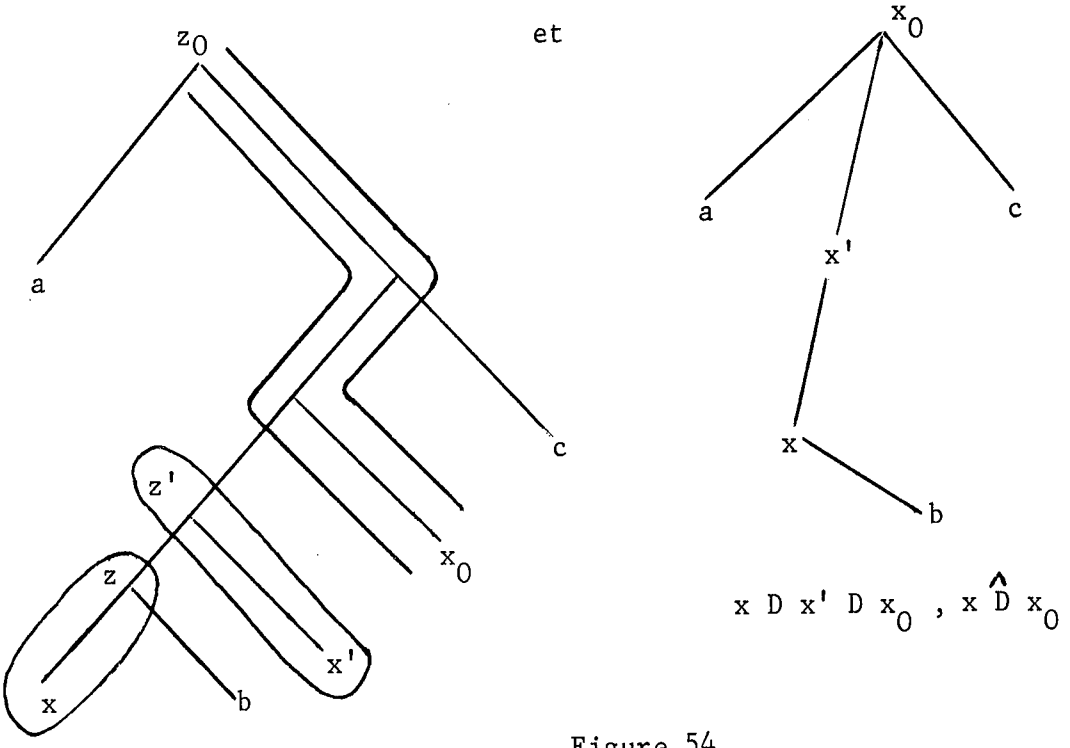


Figure 54

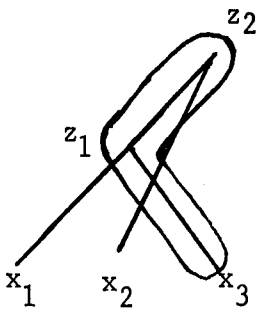
2.2.2.2.Une structure non projective ne peut être obtenue qu'à partir d'une structure non continue.

Soit en effet une structure non projective. Il existe donc un triplet (i, j, k) tel que $\min(i,j) < k < \max(i, j)$, $x_i \, D \, x_j$ et non $x_k \hat{D} x_j$.

A x_i et x_j correspondent des sous arborescences de racines z_i et z_j dans le graphe des constituants. De plus, si $x_i \, D \, x_j$, l'arborescence z_j contient l'arborescence z_i et $z_i \hat{U} z_j$. z_j est donc borne de x_i et x_j. Or l'on n'a pas $x_k \hat{D} x_j$, donc il n'existe pas de chemin $x_k \ldots z_j$ dans C. La condition de continuité n'est pas satisfaite.

Remarquons que la réciproque n'est pas vraie, et qu'il peut exister des structures projectives correspondant à des structures de constituants non continues :

Exemple :



et

$f^{-1}(x_3) = \{z_1, z_2, x_3\}$

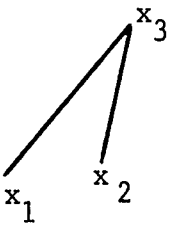


Figure 55

2.2.3. INTERPRETATION DES RELATIONS [Annexe III]

2.2.3.1. Notion de fonction syntaxique

Nous reviendrons ici sur la différence fondamentale entre l'aspect syntagmatique et l'aspect relationnel. La grammaire de constituants met en évidence les syntagmes caractéristiques. Ainsi le groupe 'le chien' constitue un syntagme nominal : son rôle dans la phrase, que nous appellerons fonction syntaxique, diffère essentiellement selon la structure : sujet, complément d'objet... etc

Ces "fonctions" syntaxiques, qui sont différentes des relations du langage pivot, sont effectivement ce que nous cherchons à représenter dans un graphe de dépendance.

Elles ne sont pas mises en évidence dans la grammaire hors contexte et dans la dérivation associée.

Ainsi les règles $S \longrightarrow N\ V$ et $V \longrightarrow V\ N$ montrent le schéma de production du groupe 'Nom' 'Verbe' 'Nom', sans spécifier les fonctions syntaxiques.

Celles-ci sont en fait caractéristiques des règles elles-mêmes, la première représentant la construction du sujet et la seconde celle du complément

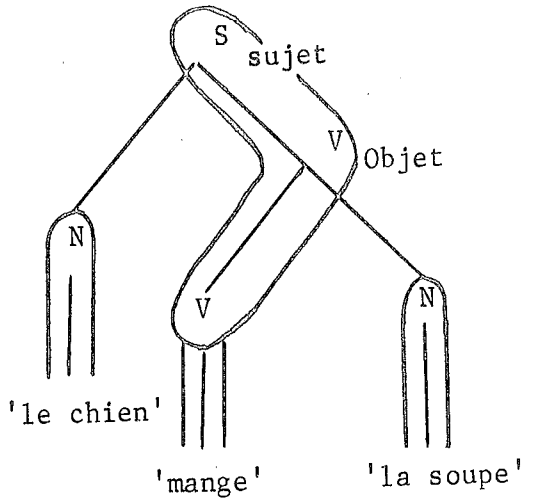


Figure 56

La structure de dépendance correspondante serait la suivante :

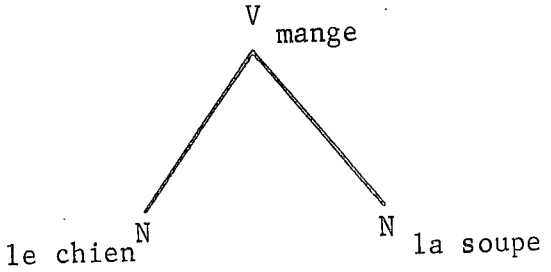


Figure 57

Le chemin f^{-1} (V) associé au verbe, correspond aux applications successives des règles de sujet et de complément. La relation D liant 'le chien' et 'mange' est la fonction syntaxique sujet. Dans la structure de dépendances, chaque occurrence x est caractérisée par une fonction unique correspondant à l'arc (x,y) le liant au gouverneur.

D'autre part, ces fonctions sont marquées dans la grammaire hors contexte par les règles de production elles-mêmes.

2.2.3.2. Attribution des fonctions

Nous pouvons donc associer à l'application f définissant la structure de dépendance une application g attribuant un nom de fonction syntaxique à chaque sommet. Nous pouvons noter (figure 58) sur chaque sommet z du graphe de constituants le nom n (z) de la règle de production p qui l'a développé. Ce nom p sera interprété comme une fonction syntaxique dans le graphe de dépendance.

Pour tout x non racine de l'arborescence D, il existe une arborescence

de racine z dans C correspondant à la sous arborescence x . g (x) = p (z'), z' étant le sommet unique tel que z U z'

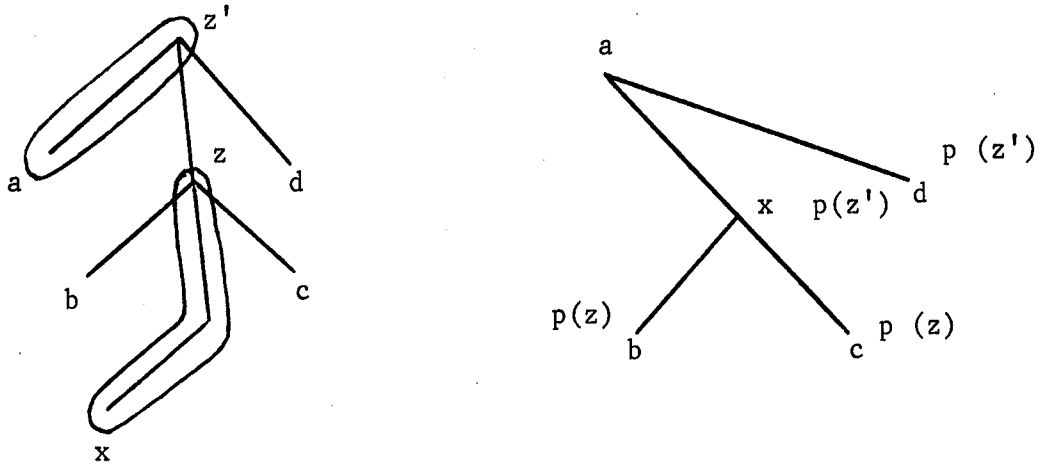


Figure 58

2.2.3.3. Forme de la grammaire hors contexte

En appliquant la correspondance avec les restrictions que nous avons proposées, chaque production de la grammaire hors contexte est représentée sur le graphe par autant de chemins qu'il y a d'éléments dans la partie droite :

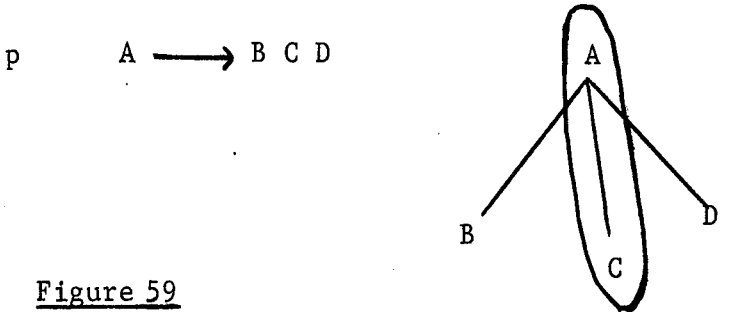


Figure 59

Comme $f^{-1}(x)$ définit un chemin unique, tous les sommets sauf un seront affectés du même nom de fonction g (x). Ainsi, dans la figure 58, nous aurons g (x) = g (d) = p (z')

Si l'on tient compte de l'ordre des occurrences, l'on pourra peut être différencier le rôle du sommet x et du sommet d qui portent le même nom de fonction syntaxique.

Nous préférons affecter à chaque sommet un nom de fonction syntaxique qui le caractérise complètement Ceci est rendu possible par l'utilisation d'une grammaire mise sous forme normale, c'est-à-dire une grammaire dont toutes les

productions sont du type :

$A \longrightarrow BC$ $\qquad A, B, C \in V_N$

ou $\quad A \longrightarrow a \qquad a \in V$

Ainsi chaque règle de production marque une relation et une seule entre deux éléments.

2.2.4. METHODE GENERALE

La recherche de la structure associée à une phrase sera menée en deux étapes successives. D'une part l'application des règles de production d'une grammaire mise sous forme normale. Il s'agit en fait de résoudre le problème de décision suivant : "la phrase proposée appartient-elle au langage ?" que l'on résoud en vérifiant si cette phrase est dérivable ou non.

Dans le cas d'une réponse favorable, nous interpréterons le processus de dérivation formelle en analysant la structure de constituants qui lui est associée. Pour cela, nous déterminerons la structure de dépendances et nous calculerons les fonctions syntaxiques.

Dans ce but, il faut associer à chaque règle de production p deux informations : d'une part le nom p lui-même , d'autre part l'indication I (p) = 'gauche' ou 'droite'. Si I (p) = 'gauche' , le chemin définissant l'application f devra contenir l'arc correspondant au premier élément de la règle binaire, si I (p) = 'droite', c'est le second arc qui interviendra. Cette information est fournie par le linguiste ; c'est l'interprétation de la règle formelle.

Exemple : p : $A \longrightarrow BC$ I (p) = 'gauche'

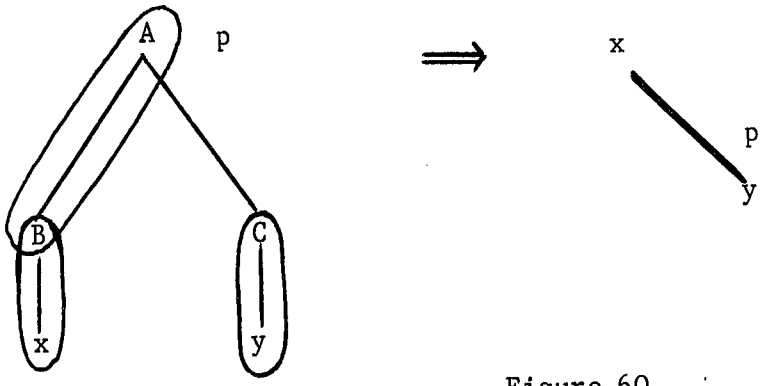
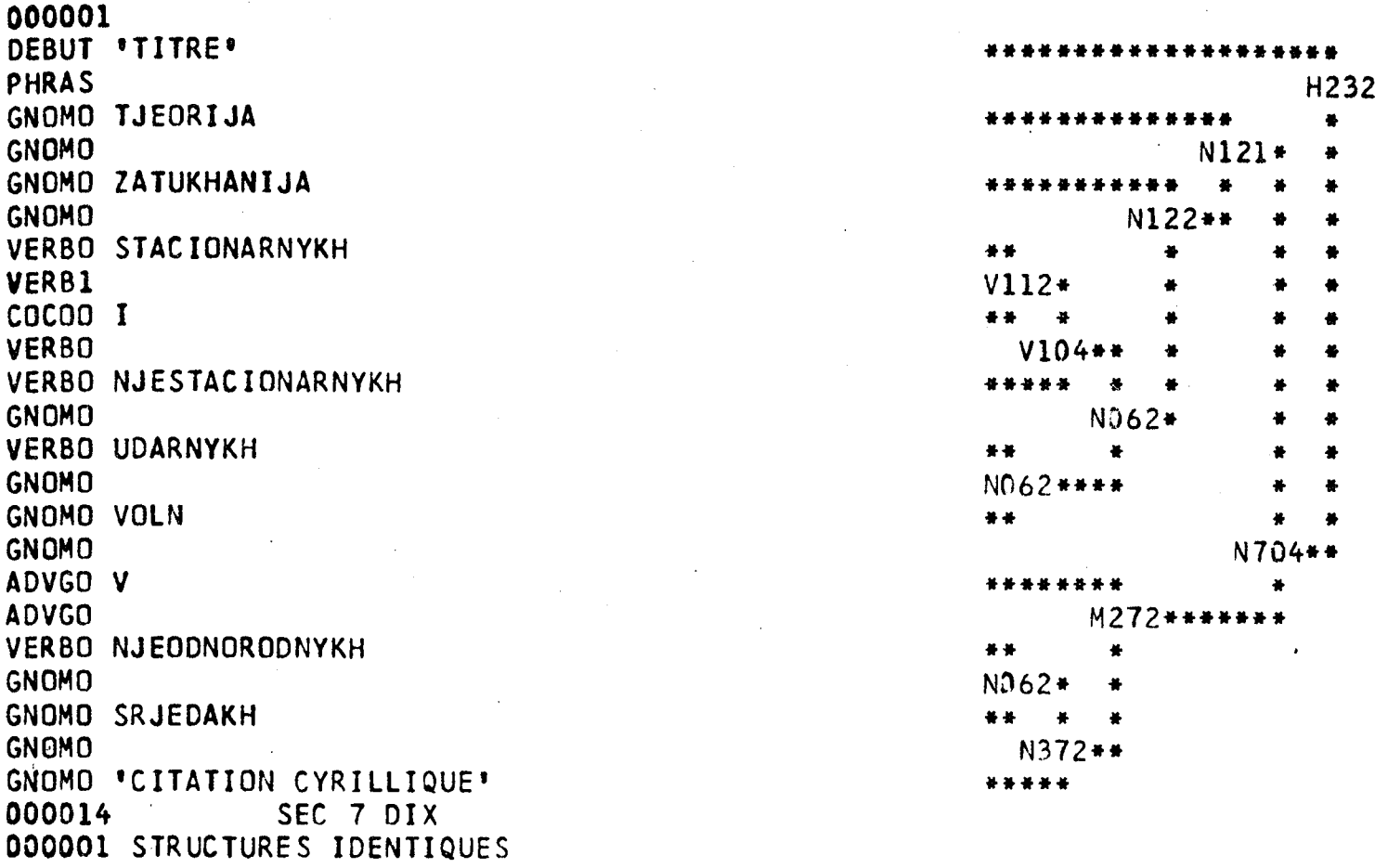


Figure 60

Finalement, le rôle de la grammaire hors contexte, et de l'algorithme d'analyse syntaxique, est de permettre la détermination d'un graphe arborescent compte tenu des contraintes de projectivité.

Les figures suivantes montrent un exemple complet des structures syntaxiques de constituants et de dépendance.

STRUCTURE DE CONSTITUANTS D'UNE PHRASE RUSSE



(H232, N704 .... sont des numéros de règle)

STRUCTURE DE DEPENDANCE ASSOCIEE A LA STRUCTURE

PRECEDENTE : I (H232) = gauche
I (N121) = gauche
I (N704) = droite ...

```
000001
0000***'TITRE'
      *
      *
      H232***TJEORIJA
            *
            *
            N121***ZATUKHANIJA
            *     *
            *     *
            *     *           V112***STACIONARNYKH
            *     *           *
            *     *           *
            *     *     N062***I
            *     *     *     *
            *     *     *     *
            *     *     *     V104***NJESTACIONARNYKH
            *     *     *
            *     *     *
            *     *     N062***UDARNYKH
            *     *     *
            *     *     *
            *     N122***VOLN
            *
            *
            N704***V
                  *
                  *
                  *     N062***NJEODNORODNYKH
                  *     *
                  *     *
                  M272***SRJEDAKH
                        *
                        *
                        N372***'CITATION CYRILLIQUE'
```

2. 3. EXTENSIONS DU MODELE HORS CONTEXTE

2.3.1. STRUCTURES NON PROJECTIVES ET GRAMMAIRES HORS CONTEXTE

Nous avons montré [2.2.2] qu'une structure non projective ne pouvait être obtenue à partir d'une grammaire hors contexte. En effet, une telle structure correspond à une structure de constituants discontinue.

La théorie des langages [CH2] définit une hiérarchie des types de grammaire ; dans cette hiérarchie, le niveau supérieur au niveau des grammaires hors contexte ('context free') correspond aux grammaires sous contexte ('context sensitive').

Nous étudierons ici la possibilité d'utiliser un tel modèle pour l'analyse de langages dont les structures sont non projectives.

Les modèles de la théorie des langages sont définis en fonction de leur puissance générative, et non en fonction de la nature des structures associées.

En particulier, une configuration donnée n'échappe à un type de modèle que dans la mesure où elle peut se présenter en nombre infini. Nous illustrerons ceci par l'exemple suivant :

Soit la phrase : 'nous l'avons tous vu' dont la structure de dépendance serait :

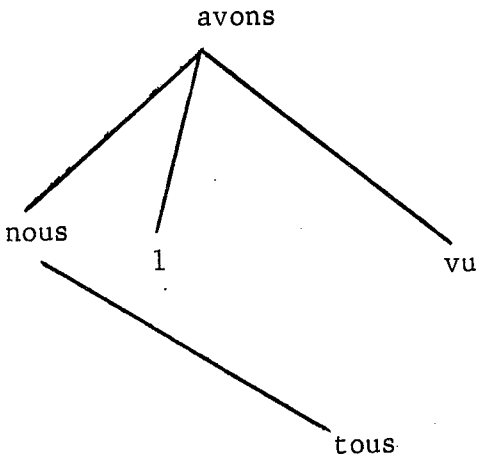


Figure 61

et la structure de constituants :

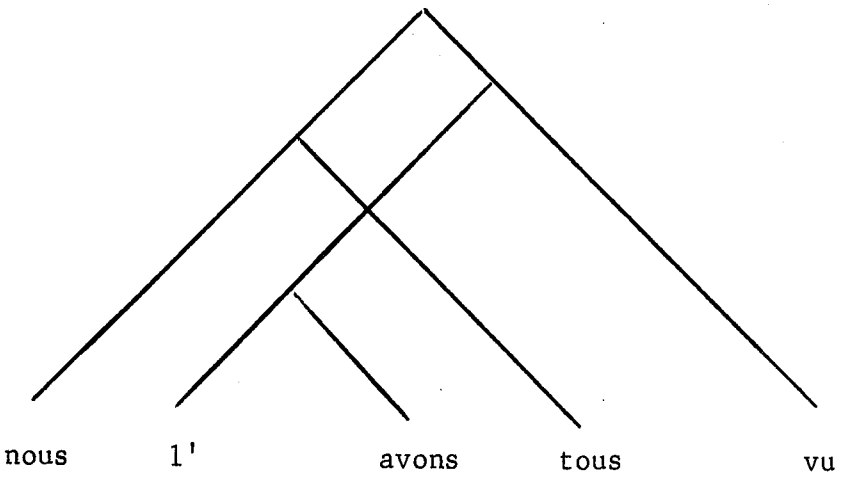


Figure 62

Une telle structure ne peut correspondre à une dérivation dans une grammaire hors contexte. Par contre, la discontinuité étant liée à l'apparition du seul mot 'tous', il est facile d'écrire une grammaire hors contexte générant cette phrase. La structure sera évidemment continue, par exemple :

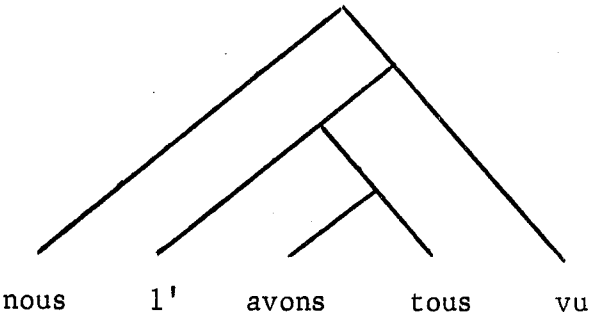


Figure 63

Cependant, la règle de production du groupe "avons tous" n'est pas interprétable, et la structure est incorrecte.

Si l'on considère maintenant une phrase du type "x, y, z, ... sont respectivement x', y', z'..." , la structure est discontinue, et la grammaire associée doit être de niveau supérieur, car le nombre d'éléments est non borné.

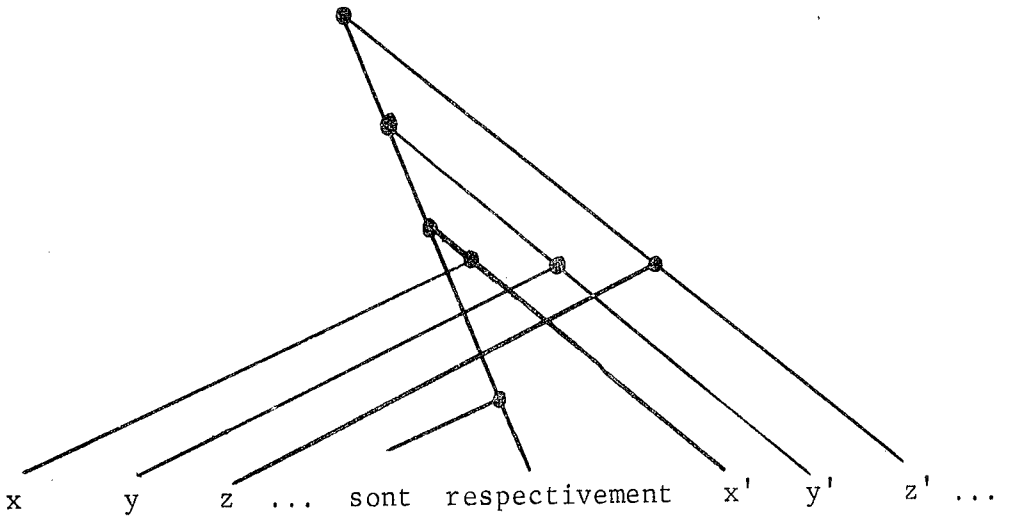


Figure 64

le langage formé de telles phrases est du type"sous-contexte".

On sait que les grammaires sous-contextes (context sensitive grammars) sont caractérisées par des règles de production du type $\varphi \longrightarrow \psi$ dans lesquelles $\ell(\varphi) \leqslant \ell(\psi)$. De ce fait, la structure associée à la dérivation n'est plus arborescente.

Ainsi, soit $AB \longrightarrow CD$ une production

Le sous graphe associé serait

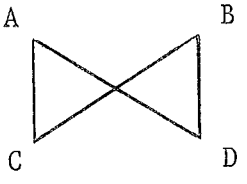


Figure 65

D'une façon générale, ces types de grammaires ne permettent pas de construire des dérivations interprétables. De plus, les algorithmes d'analyse syntaxique sont peu efficaces.

2.3.2. LES EXTENSIONS POSSIBLES DU MODELE HORS CONTEXTE

Compte tenu des difficultés dues aux modèles sous contexte, plusieurs travaux ont été consacrés aux extensions possibles des modèles hors contexte. Il s'agit de définir des langages de niveau supérieur, en utilisant une grammaire hors contexte munie de certaines contraintes. En ce qui nous concerne, nous recherchons des modèles définissant des structures arborescentes interprétables, et permettant d'utiliser des algorithmes relativement efficaces.

Les extensions qui ont été proposées reposent pour la plupart sur les mêmes principes : étant donnée une grammaire hors contexte $G = (V, V_T, S, P)$, on limitera les dérivations possibles en utilisant des contraintes sur l'ordre d'application des productions de P. Ces contraintes seront exprimées d'une façon globale sur la dérivation complète (Control Set Grammars Gi - 2) soit localement au cours de la génération (grammaires matricielles)[Ab] , "scattered grammar" [Gr] indexed grammars [Ah]

2.3.3. L'UTILISATION DES VARIABLES DE TRANSFERT (VARIABLES VEHICULAIRES)

2.3.3.1. Principe général

Cette notion a été introduite dans le but de permettre l'analyse des discontinuités dans les cas finis, en fournissant une structure interprétable (figure 62). Il est possible de la généraliser à des cas non finis, et d'en déduire un algorithme d'analyse de certains langages sous contexte.

La définition que nous en donnerons peut être apparentée à celle des grammaires indexées [Ah] ; nous associerons à certains symboles non terminaux une chaine sur un vocabulaire auxiliaire notée V_v.

La grammaire est ainsi définie : $G = (V, V_N, S, P, V_v, \lambda)$

V_N, V, S correspondent aux définitions des grammaires hors contexte [1.4.3.4]. V_v est un vocabulaire auxiliaire (véhiculaires), et λ un symbole spécial. Les productions de P sont de quatre types :

1) type P_0 : $A \longrightarrow \varphi$ $\qquad A \in V_N$, $\varphi \in (V_N \cup V)$

2) type P_λ : $A\lambda \longrightarrow \varphi B \lambda \varphi'$ $\qquad B \in V_N$, $\alpha \in V_v$

3) type $P\uparrow$: $A\lambda \longrightarrow \varphi B \alpha \varphi'$

4) type $P\downarrow$: $A\alpha \longrightarrow \varphi B \lambda \varphi'$

A et B étant des éléments de V , α un élément de V_v et φ et φ' des chaines sur $(V_N \cup V)$

La relation de dérivation $\psi \Longrightarrow \psi'$ est définie ainsi : Reprenant la notation de Ah_0, nous appellerons symbole indexé la chaine $A\ \eta \in V_N (V_v)^*$

Soit $\psi = \chi_1 A \eta \chi_2$, χ_1 et $\chi_2 \in (V_N (V_v)^* \cup V)^*$

nous noterons $\psi \Longrightarrow \psi'$ si :

a) $\psi' = \chi_1 \varphi \chi_2$, η est vide et si $A \longrightarrow \varphi$ est une règle de P_0

ou si b) $\psi' = \chi_1 \varphi B \eta' \varphi' \chi_2$

1) $\eta = \eta'$ et $A\lambda \longrightarrow \varphi B \lambda \varphi'$ est dans P_λ

2) $\eta = \eta' \alpha$ et $A\alpha \longrightarrow \varphi B \lambda \varphi'$ est dans $P\downarrow$

3) $\eta = \eta \alpha$ et $A\lambda \longrightarrow \varphi B \alpha \varphi'$ est dans $P\uparrow$

2.3.3.2. Exemple

$V_N = \{ S, R \}$

$V = \{ s, r, a, b, c \}$

$V_v = \{ \alpha, \beta, \gamma \}$

P_0 : $R \longrightarrow r$

P_λ : $S_\lambda \longrightarrow s\ R\ \lambda$

$P\uparrow$: $S_\lambda \longrightarrow a\ S\ \alpha$

$S_\lambda \longrightarrow b\ S\ \beta$

$S_\lambda \longrightarrow c\ S\ \gamma$

$P\downarrow$: $R\alpha \longrightarrow R\ \lambda\ a$

$R\beta \longrightarrow R\ \lambda\ b$

$R\gamma \longrightarrow R\ \lambda\ c$

La figure suivante montre une dérivation dans laquelle chaque symbole indexé correspond à un sommet

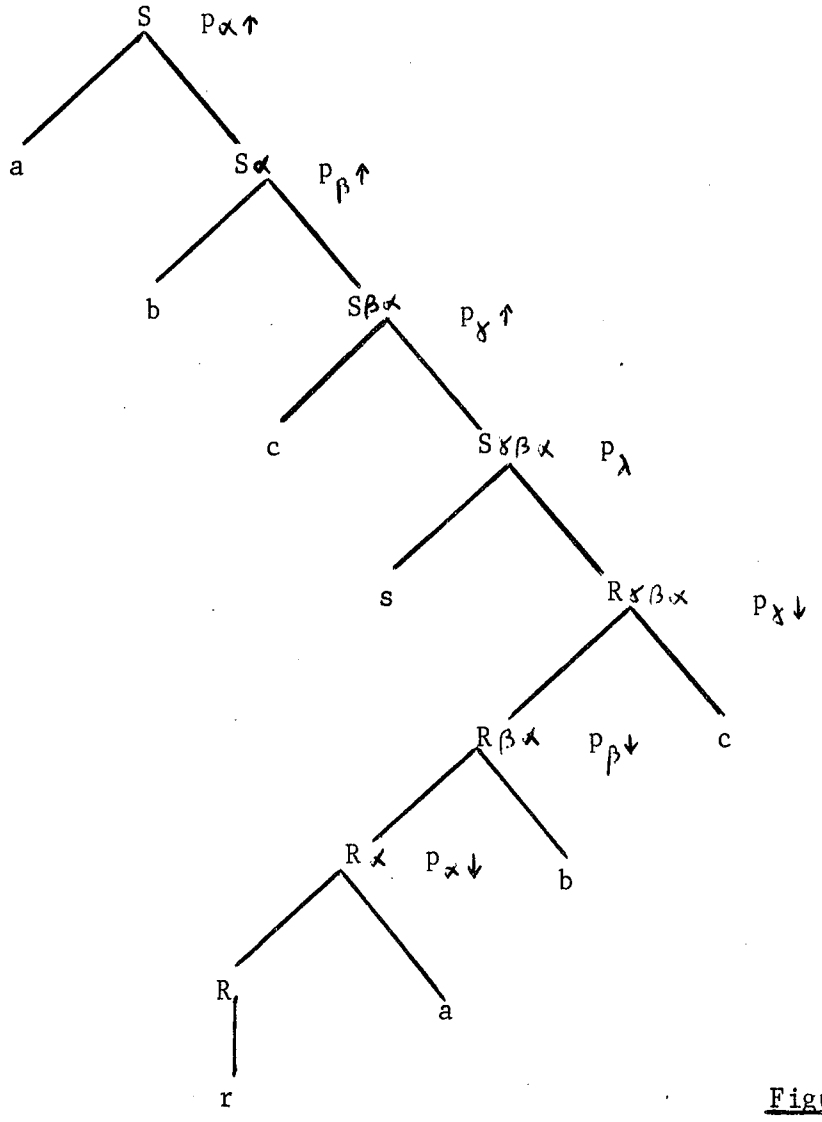


Figure 66

Dérivation de a b c s r a b c

Montrons qu'une telle grammaire définit le langage : x s r x, $x \in \{a, b, c\}^*$. Le langage x s r x est un langage sous contexte.

Nous noterons x^- la chaine inverse de x, telle que $(y\ z)^- = z^-\ y^-$ quelles que soient les chaines y et z. On peut définir une bijection entre $\{a, b, c\}$ et $V_V = \{\alpha, \beta, \gamma\}$ associant a et α, b et β, c et γ. On lui fait correspondre un ismorphisme I entre les phrases de $\{a, b, c\}^*$ et de V_V^*. Par exemple : $I(a\ b\ b\ c) = \alpha\ \beta\ \beta\ \gamma$

Pour montrer que $S \overset{*}{\Longrightarrow} x\ s\ r\ x$, pour tout $x \in \{a, b, c\}^*$, on montre que $S \overset{*}{\Longrightarrow} x\ s\ r\ I(x)^-$. En effet, toute règle $P\uparrow$ correspond à $S \Longrightarrow a\ S\ I(a)^-$ donc $x\ S\ I(x)^- \Longrightarrow x\ a\ S\ I(a)^-\ I(x)^- = x\ a\ S\ I(xa)^-$. Par récurrence, il vient $S \overset{*}{\Longrightarrow} x\ S\ I(x)^-$ et, par application de la règle P_λ : $S \overset{*}{\Longrightarrow} x\ s\ R\ I(x)^-$.

De même, $R\ I(x)^- \overset{*}{\Longrightarrow} R\ x$. En effet, pour tout symbole indexé $R\ \eta$, $\eta \in V_V^*$ il existe une règle $P\downarrow$ telle que $\eta = \mu\eta'$, $R\ \mu\ \eta' \Longrightarrow R\ \eta'\ I^{-1}(\mu)$, et $R\ \mu\eta'\ y \Longrightarrow R\ \eta'\ I^{-1}(\mu)\ y$. Donc, par récurrence :

$$R\ \eta\ \eta' \overset{*}{\Longrightarrow} R\ \eta'\ I^{-1}(\eta)^-$$

Donc $R\ I(x)^- \overset{*}{\Longrightarrow} R\ x \Longrightarrow r\ x$ et $x\ s\ R\ I(x)^- \overset{*}{\Longrightarrow} x\ s\ r\ x$

Théorème :

Les langages hors contexte constituent un sous ensemble propre des langages L_{G_V}.

2.3.3.3. Propriétés des grammaires utilisant les variables véhiculaires

L'exemple précédent (figure A_{31}) montre que les chaines successives sur V_V correspondent aux états d'une pile $[Pa]$

Plus généralement, nous noterons sur le graphe de dérivation d'une phrase, les noms des règles de production avec les conventions suivantes :

p si la règle appliquée est de type p_0

p_λ si la règle appliquée est de type p_λ

$p_{\alpha\uparrow}$ si la règle appliquée est de type $p\uparrow$ et si le symbole de V_V est α

$p_{\alpha\downarrow}$ " $p\downarrow$ "

Il existe sur le graphe des chemins dont les sommets sont affectés de noms de règles différents de p. Ces chemins sont disjoints, et la suite $\eta_0\ \eta_1 \cdots \eta_i \cdots \eta_n$ des phrases sur V_V^* correspondent aux symboles indexés de ces chemins est telle que :

$\eta_0 = \eta_n = \wedge$ chaine vide

$\eta_i = \eta_{i+1}$ si le nom de règle affecté à η_i est p_λ

$\alpha\eta_i = \eta_{i+1}$ si le nom de règle affecté à η_i est $p_{\alpha\uparrow}$

$\eta_i = \alpha\eta_{i+1}$ si le nom de règle affecté à η_i est $p_{\alpha\downarrow}$

Ainsi, sur la figure A_{31}, nous trouvons

$\eta_0 = \wedge$, $\eta_1 = \alpha$, $\eta_2 = \beta\alpha$, $\eta_3 = \gamma\beta\alpha$, $\eta_4 = \beta\alpha$, $\eta_5 = \alpha$, $\eta_6 = \wedge$

Ceci correspond aux états d'une pile, conformément à la définition donnée dans [Pa]

Nous considérons maintenant le langage constitué par les chaines sur $\{p_\lambda, \{p_{\alpha\uparrow}, p_{\alpha\downarrow}\}\}$ correspondant aux chemins non marqués par P maximaux. La relation d'ordre sur la chaine correspond à la relation $\Longrightarrow$ de la dérivation.

Ainsi, dans l'exemple de figure A_{31}, nous aurons un seul chemin maximal et une seule chaine : $p_{\alpha\uparrow}\ p_{\beta\uparrow}\ p_{\gamma\uparrow}\ p_\lambda\ p_{\alpha\downarrow}\ p_{\beta\downarrow}\ p_{\gamma\downarrow}$

Ce langage est, à un homomorphisme près (effacement des p_λ) un langage de Dyck ; il est accepté par pile vide par un automate $M = (\{q_1\}, \{\bigcup_{\mu\in F} (p_{\mu\uparrow}, p_{\mu\downarrow})\} \cup \{p_\lambda\}, \bigcup_{\mu\in F} \{\mu\}, \delta, q_1, Z_0, \emptyset)$

avec $F = \{\alpha, \beta, \gamma\}$ et δ telle que :

$\delta(q_1, p_{\mu\uparrow}, Z_0) = (q_1, \mu Z_0)$ $\mu \in F$

$\delta(q_1, p_{\mu\downarrow}, \mu\gamma) = (q_1, \gamma)$ $\gamma \in \{\alpha, \beta, \gamma\}^*$

$\delta(q_1, p_\lambda, \gamma) = (q_1, \gamma)$

A chaque application d'une règle de type p_λ , $p_{\alpha\uparrow}$, $p_{\alpha\downarrow}$ correspond une opération de l'automate, et la bande de manoeuvre contient effectivement la chaine η correspondante.

De ce fait, on peut associer à toute grammaire G_V une grammaire G_0 contenant toutes les productions sans les symboles de V_V ni λ . Le langage L_{G_0} est tel que $L_{G_V} \subset L_{G_0}$. De plus, étant donnée une phrase φ de L_{G_0} , on peut décider si $\varphi \in L_{G_V}$ en vérifiant si tous les chemins marqués sont acceptés par l'automate.

2.3.3.4. Exemples de langages de type "sous contexte"

a) langage x x. Soient $V = \{a, b\}$, $V_N = \{S, S', S''\}$, $V_V = \{\alpha, \alpha', \beta, \beta'\}$

et P :

$S\lambda \longrightarrow a\ S'\alpha'$

$S\lambda \longrightarrow b\ S'\beta'$

$S'\lambda \longrightarrow a\ S'\alpha$

$S'\lambda \longrightarrow b\ S'\beta$

$S'\lambda \longrightarrow a\ S''\alpha$

$S'\lambda \longrightarrow b\ S''\beta$

$S''\alpha \longrightarrow S''\ \lambda\ a$

$S''\beta \longrightarrow S''\ \lambda\ b$

$S''\alpha' \longrightarrow a$

$S''\beta' \longrightarrow b$

Dérivation de x x, $x \in V^*$

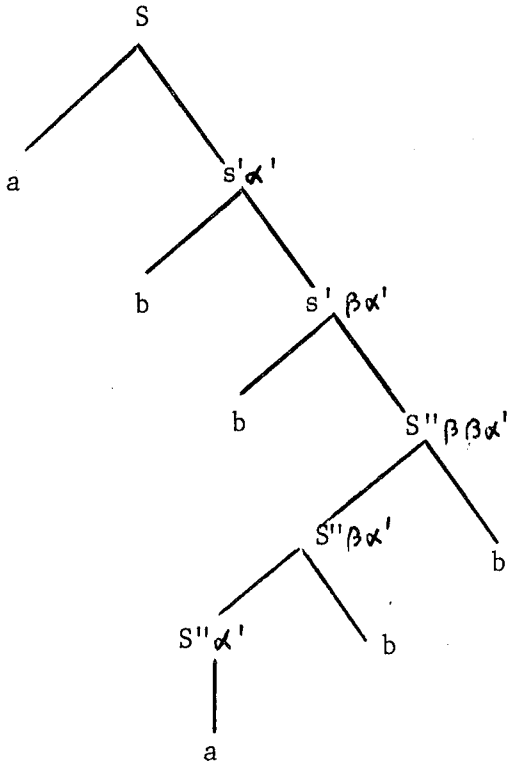


Figure 67

b) langage $a^n\ b^n\ c^n$

$$V = \{a, b, c\} \qquad V_N = \{S, S', S''\} \ , \ V_T = \{\alpha, \alpha'\}$$

$S\ \lambda \longrightarrow a\ S'\ \alpha'\ C$

$S'\lambda \longrightarrow a\ S'\ \alpha\ C$

$S'\lambda \longrightarrow a\ S''\ \alpha\ C$

$S''\alpha \longrightarrow S''\ \lambda\ b$

$S''\alpha' \longrightarrow b$

Dérivation de $a^3\ b^3\ c^3$

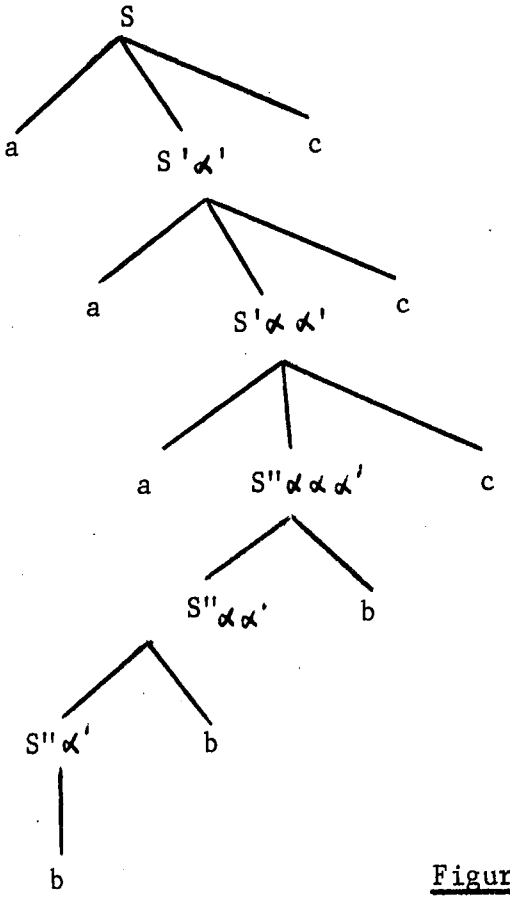


Figure 68

2.3.3.5. Application pratique

La grammaire utilisant les variables véhiculaires permet d'associer aux phrases de certains langages de type sous contexte des structures arborescentes.

De ce fait, nous pourrons lui appliquer les algorithmes d'analyse utilisés pour les grammaires hors contexte classiques.

Nous pouvons disjoindre l'analyse en deux parties : recherche d'une dérivation par G_0, puis contrôle de cette dérivation par l'automate. Cependant, ce dernier étant déterministe, il est possible de l'appliquer à chaque pas de l'analyse, et d'associer à chaque sommet non terminal un état de la pile.

Ainsi, le nombre de pas de l'analyse d'un langage L_{G_V} est égal au nombre de pas de l'analyse d'un langage L_G, chaque pas comportant, en plus des opérations d'analyse habituelle, une opération de l'automate à pile.

2.3.3.6. Construction des structures discontinues

Les structures obtenues par dérivation d'une grammaire G_V n'ont pas d'interprétation linguistique. Pratiquement, l'intervention des variables véhiculaires permet de traiter des structures discontinues, qu'il faut obtenir à partir de la structure continue.

La discontinuité correspond à l'existence d'un constituant regroupant les sous arborescences dont les racines ont pour noms $p_{\alpha\uparrow}$, $p_{\alpha\downarrow}$ reconnus par l'automate à pile. La structure discontinue associée à un tel couple contiendra le même nombre de sommets affectés des mêmes noms. Cependant, la structure sera telle que la sous arborescence de racine $p_{\alpha\uparrow}$ sera en relation avec la sous arborescence de racine $p_{\alpha\downarrow}$. La transformation peut être décrite formellement de la façon suivante :

Soit (Z, U) une structure de constituants, et un couple de sommets de noms $p_{\alpha\uparrow}$, $p_{\alpha\downarrow}$.

a) si z U $p_{\alpha\uparrow}$ et si z n'est pas sur le chemin $p_{\alpha\downarrow}$ $p_{\alpha\uparrow}$, z U' $p_{\alpha\uparrow}$

b) si z U $p_{\alpha\uparrow}$ et si z est sur le chemin $p_{\alpha\downarrow}$ $p_{\alpha\uparrow}$, si $p_{\alpha\uparrow}$ est

racine de (Z, U), z est racine de (Z, U'), si $p_{\alpha\uparrow}$ U z', z U' z'.

c) $p_{\alpha\uparrow}$ U' $p_{\alpha\downarrow}$

d) si z U $p_{\alpha\downarrow}$ et si z $\neq$ z_λ , z U' z $p_{\alpha\uparrow}$, sinon z U' $p_{\alpha\downarrow}$

Une telle transformation ne détruit pas les autres chemins $p_{\alpha\uparrow}$ $p_{\alpha\downarrow}$ de (Z, U). La structure peut être transformée complètement par réduction successive des couples $p_{\alpha\uparrow}$, $p_{\alpha\downarrow}$

La structure ainsi obtenue est une arborescence, car chaque sommet a un demi degré extérieur égal à 1, et le graphe reste connexe.

La figure 69 montre la structure de l'exemple 66

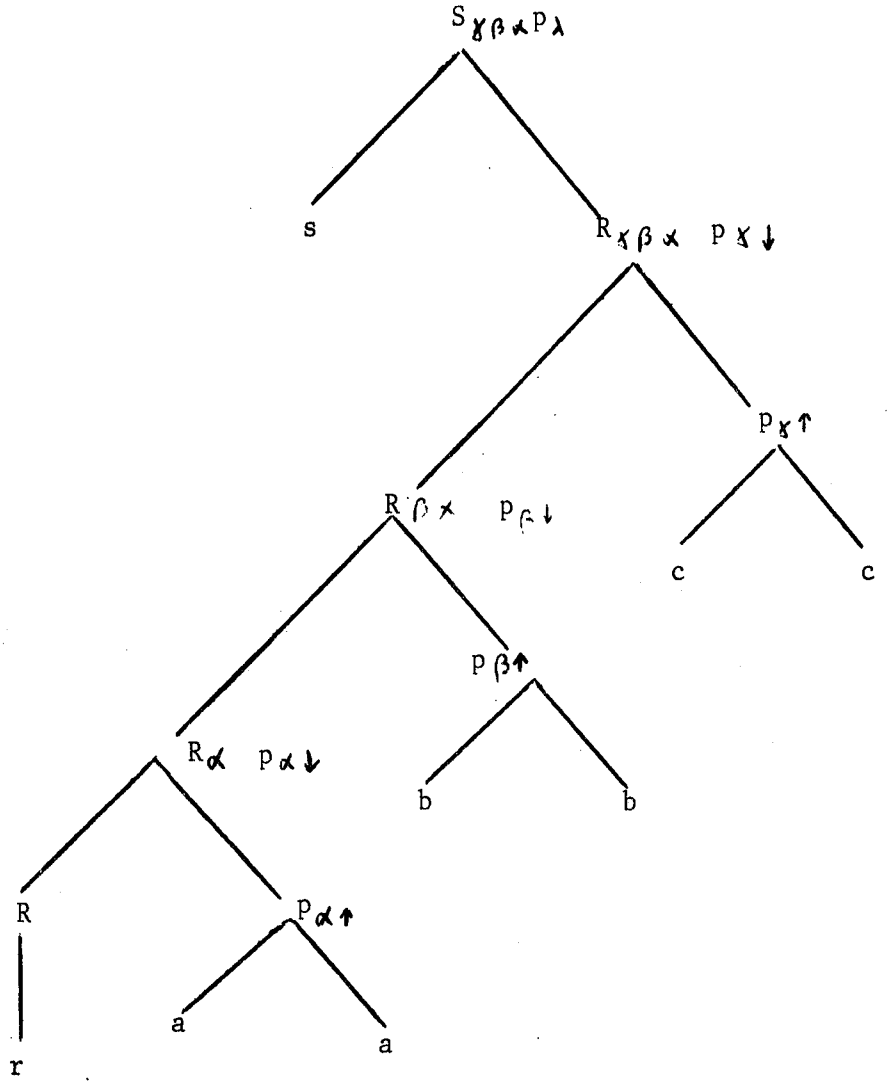
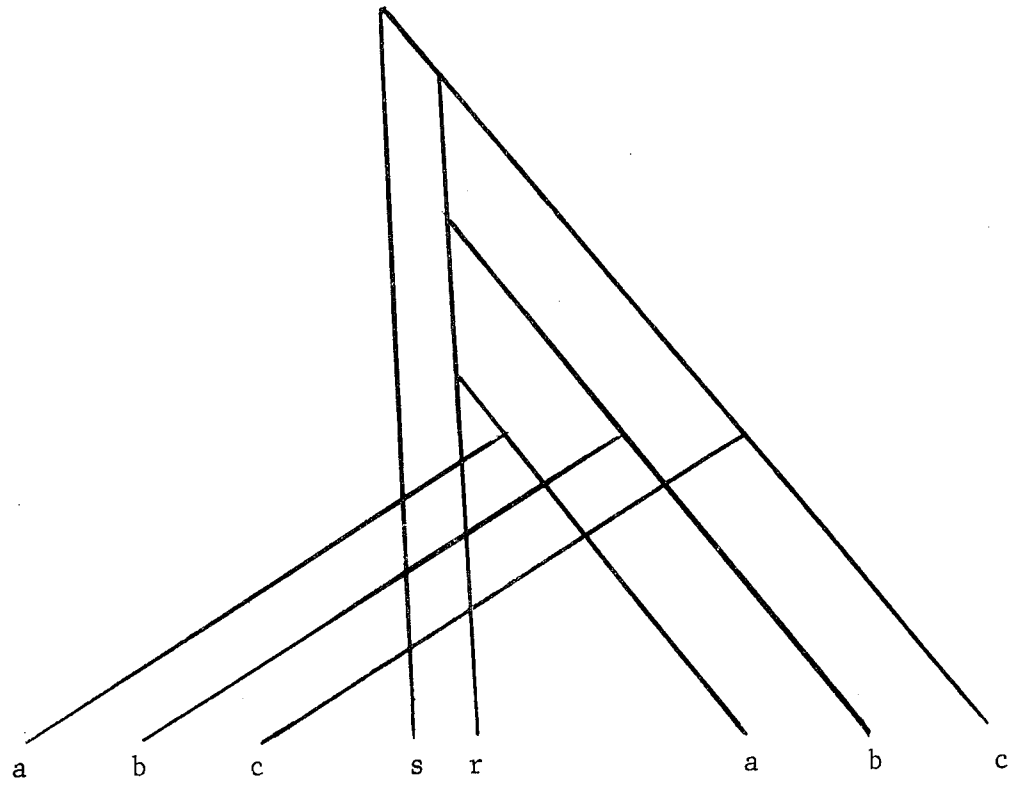


Figure 69



Structure discontinue

Figure 69 bis

2.3.3.7. Interprétation des structures discontinues

Nous pouvons appliquer aux structures discontinues l'interprétation définie en 2.2. Pour cela, il nous faut considérer le cas d'une grammaire sous forme normale, et étudier les quatre possibilités correspondant aux côtés respectifs du gouverneur et des symboles indexés, selon les quatre schémas suivant, issus de la même structure de constituants :

Structure de constituants

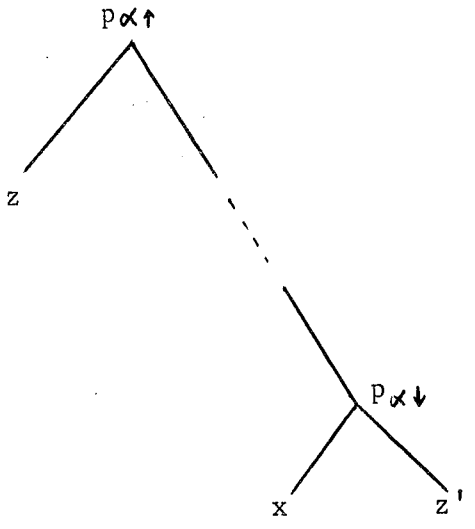
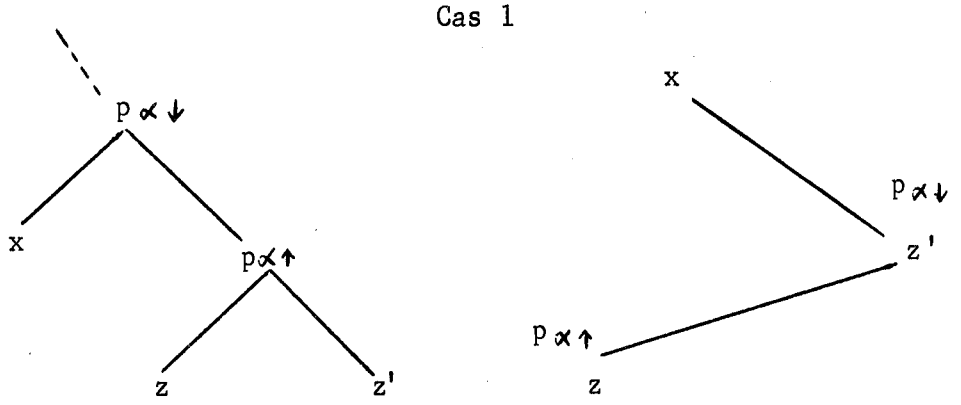


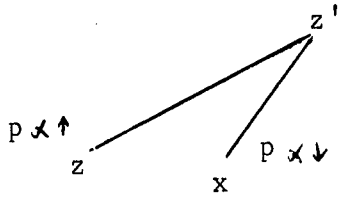
Figure 70

Structure transformée

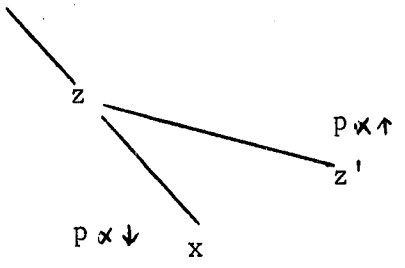
Cas 1



cas 2



Cas 3



Cas 4

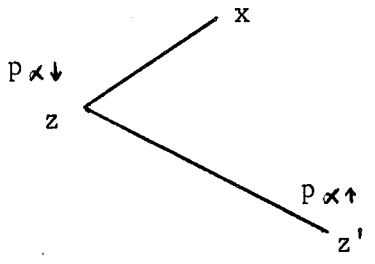
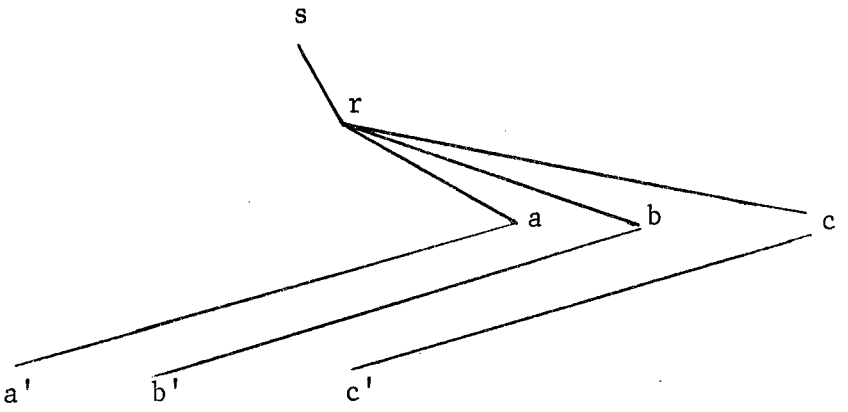


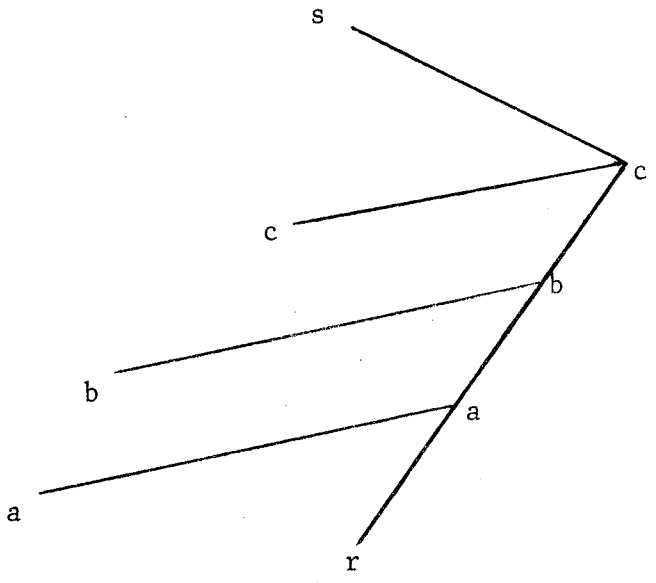
Figure 70 bis

A titre indicatif, les structures associées de l'exemple 69 seront les suivantes :

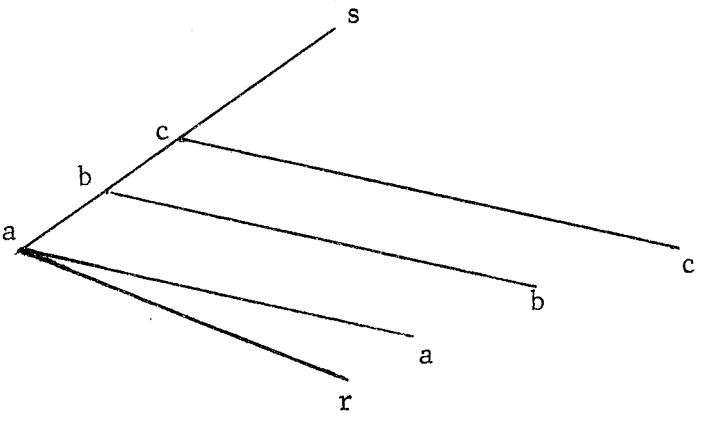
Cas 1



Cas 2



Cas 3



Cas 4

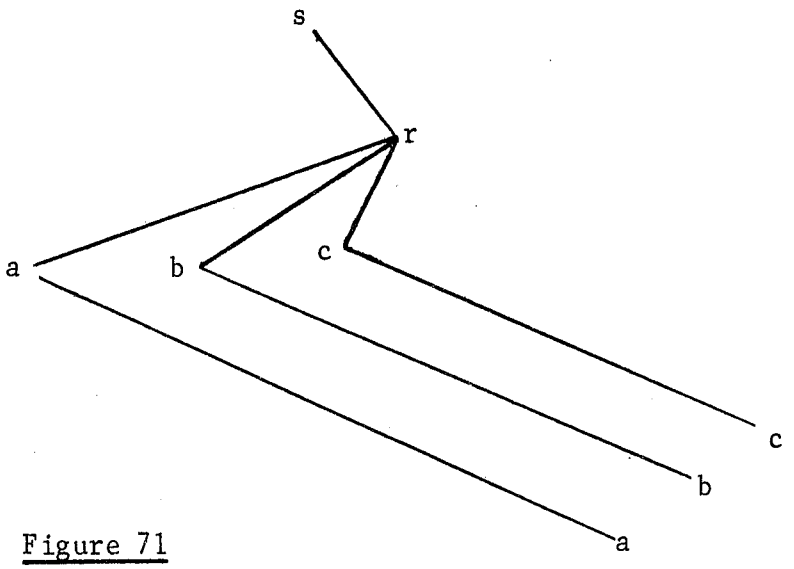


Figure 71

Les quatre structures sont non projectives, ce qui n'apparaitrait pas dans les cas 2 et 3 en se limitant à un seul couple ✗↑, ✗↓

Il existe des structures non projectives qui ne peuvent être obtenues à partir de grammaires utilisant les variables véhiculaires. Ce sera le cas de la structure :

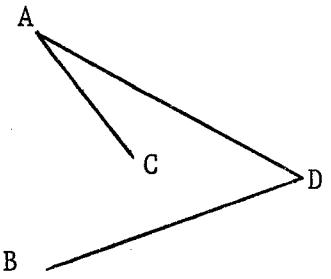
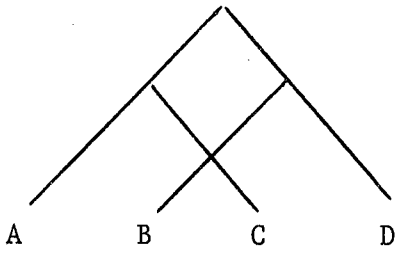


Figure 72

dont la structure de constituants serait :



Ce fait, correspond intuitivement à la remarque suivante : dans cette structure, il existe deux discontinuités (A, C et B, D) , et chaque discontinuité peut être résolue au moyen de deux règles de type $\alpha\downarrow, \alpha\uparrow$. Or nous n'avons ici que quatre terminaux, donc trois règles possibles.

Il serait intéressant de connaître la caractérisation des grammaires sous contexte traduisant cette limite.

Du point de vue de l'analyse syntaxique, la difficulté est évidente : il n'existe aucun couple de terminaux voisins (AB, BC, CD) auquel on puisse associer une règle interprétable de la grammaire.

Les grammaires à variables véhiculaires se présentent comme un cas particulier des grammaires indexées définies par Aho . Une autre généralisation pourrait être associée aux "Control Set Grammars" de Ginsburg. Les "Control Set Grammars" permettent une extension des grammaires hors contexte par un contrôle sur le langage défini par la chaine des productions dans la dérivation la plus à gauche. Ici, nous contrôlons chaque chemin de la dérivation. Or le langage défini sur les chemins des dérivations d'une grammaire hors contexte est un langage d'état fini, car la seule contrainte est une relation binaire entre symboles successifs. Il est facile de voir qu'une contrainte de type d'état fini sur les chemins ne change pas de famille d'une grammaire hors contexte. Les variables véhiculaires introduisent une contrainte de type 'hors contexte' sur certains chemins et permettent de reconnaître des langages de type sous contexte.

2.4. ALGORITHME D'ANALYSE

2.4.1. ALGORITHME

Les algorithmes d'analyse des langages hors contexte ont été particulièrement étudiés dans le cas des langages artificiels [GP, Cn]. Les applications aux langues naturelles sont plus délicates, et, si les mêmes principes peuvent y être utilisés, il convient de tenir compte d'un certain nombre de particularités, liées surtout aux ambiguïtés. Deux techniques d'analyse ont été proposées : l'analyse descendante et l'analyse ascendante. Elles correspondent aux deux modes de parcours des arborescences. La première technique a été particulièrement développée par Kuno et Oettinger [Ku] sous le nom d'analyse prédictive.

La technique ascendante permet de mener en parallèle toutes les dérivations possibles. De ce fait, nous verrons que l'on peut conserver toutes les parties communes des diverses analyses, ce qui n'est pas possible dans le cas des méthodes descendantes. L'algorithme le plus utilisé dans ce cas est l'algorithme de Cocke, appliqué en particulier par J. Robinson [Ro] et Sakai [Sa], et dont nous avons réalisé une variante $[V_4, V_{a_4}]$

2.4.2. PRINCIPE DE L'ALGORITHME

En supposant que nous appliquons une grammaire mise sous forme normale (règles de productions binaires), une dérivation peut être représentée par une arborescence binaire.

Si la phrase analysée est composée de n mots, l'arborescence comportera (n - 1) sommets non terminaux. Chaque sommet est racine d'une sous-arborescence comportant j sommets terminaux. Nous appelerons niveau du sommet le nombre j, et nous noterons σ_{ij} chaque sommet, i étant le rang de l'occurrence du dernier mot de la sous arborescence. Ainsi les mots de la phrase correspondent aux sommets $\sigma_{1,1}$, $\sigma_{2,1}$, ..., $\sigma_{n,1}$ si elle comporte n mots.

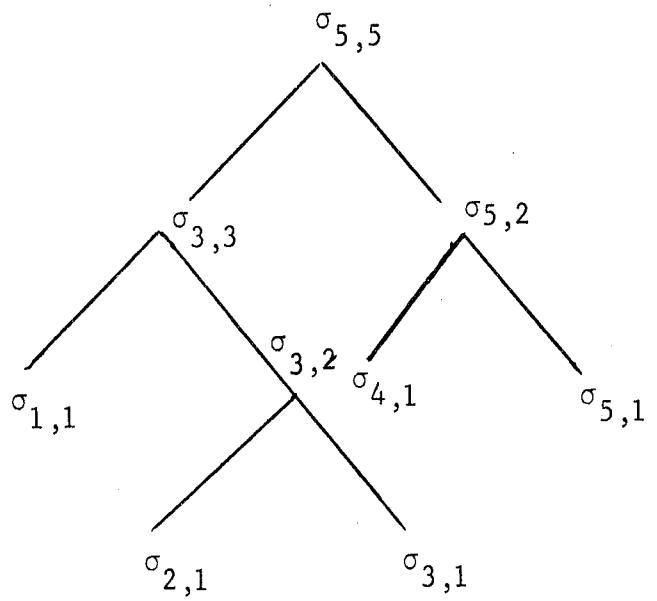


Figure 73

Sans tenir compte des éléments de vocabulaire affectés à chaque sommet, chacun d'entre eux peut être racine de $\frac{1}{2j-1} C^{j}_{2j-1}$ arborescences différentes.

L'analyse ascendante consiste à construire l'arborescence à partir des sommets terminaux. Ainsi dans le cas où il n'y a pas d'ambiguïté, le sommet σ_{ij} ne sera construit qu'après les (j - 1) sommets de sa sous arborescence. Si il peut exister des ambiguïtés, il faudra examiner toutes les sous arborescences possibles, en tenant compte des $\frac{j\,(j-1)}{2}$ sommets qui peuvent être dans une sous arborescence de racine σ_{ij}. Ceci est illustré par le tableau suivant, dans lequel nous avons représenté en coordonnées les rangs et les niveaux.

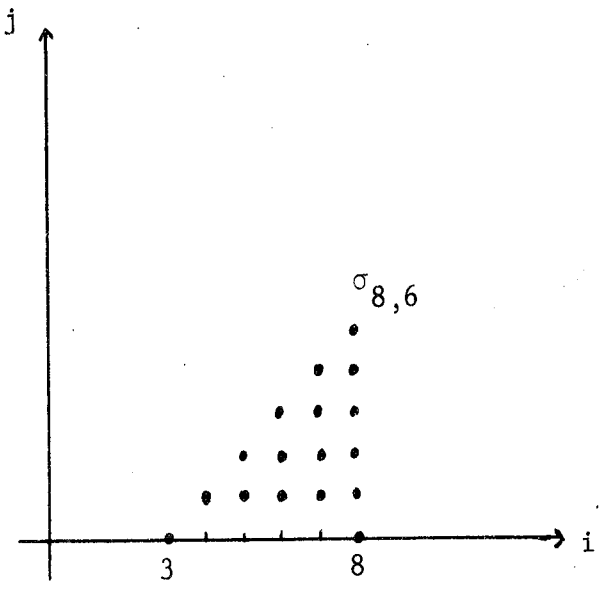


Figure 74

Dans une telle représentation, les sommets possibles dans le cas d'une phrase de n mots s'inscrivent dans un tableau triangulaire, la racine étant $\sigma_{n,n}$.

L'algorithme de Cocke consiste à construire successivement les lignes du tableau. Pour déterminer les sous arborescences associées à un sommet σ_{ij} , nous devrons examiner successivement les combinaisons binaires :

$(\sigma_{i-j,1}, \sigma_{i,j-1})$ $(\sigma_{i-j-1,2}, \sigma_{i,j-2})$ $\cdots$ $(\sigma_{i-1,j-1}, \sigma_{i,1})$.

Ceci correspond à (j-1) combinaisons pour chaque sommet de niveau j. Si il y a n mots dans la phrase, nous aurons (n-j) (j-1) opérations par niveau et $\sum_{j=1}^{n} (n-j)\,(j-1) = \frac{n\,(n^2 - 1)}{6}$ opérations pour la phrase.

En utilisant les mêmes conventions que celles que nous prendrons pour décrire l'algorithme complet [II-2] le balayage selon l'algorithme de Cocke peut s'exprimer par le programme ALGOL 68 suivant :

```
begin
Initialisation :
        int i, j, k ; [1 : n ] ref [    ] ref sommet T ;
        for i to n do  T [i ]: = [1 : i ] ref sommet ;
    ¢ n est le nombre de mots de la phrase,T est un tableau triangulaire,
      sommet représente les informations liées aux éléments non terminaux
      ayant mêmes coordonnées ¢

Balayage :
        for j : = 2 by 1 to n do
            for i : = n - j by 1 to n
                do for k : = 1 by 1 to j - 1
                    do grammaire (T [i-j-k+1 ][k ], T [i][j-k], T [i ][j] )
    ¢ grammaire est une procédure appliquant les règles AB ⟶ C, dont les
      trois paramètres sont les sommets affectés des symboles A et B, et le
      sommet qui sera affecté du symbole C, cf annexe II ¢
end
```

Un des inconvénients présentés par cet algorithme est qu'il suppose connue la longueur n de la phrase dès le début de l'analyse. Il est facile de s'affranchir de cette contrainte en utilisant un parcours différent.

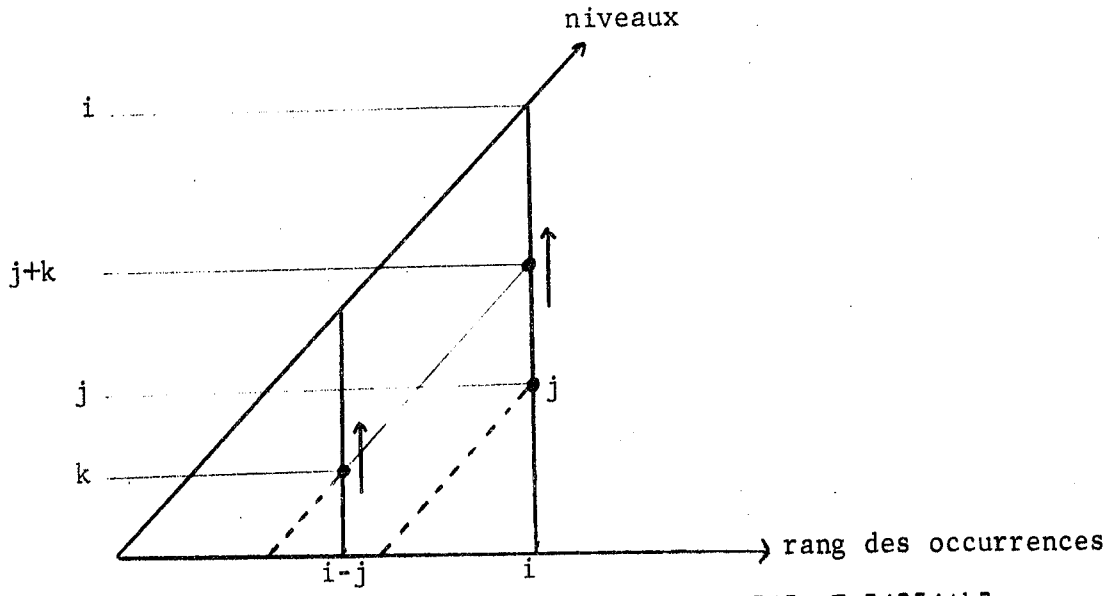
Nous construirons ainsi les structures syntaxiques en effectuant des itérations sur le nombre de mots, et non sur le niveau.

Le programme 'balayage' s'écrirait alors :

```
Balayage : for i to ℓ while cat of synt of T [i ][i] ≠ phrase
               do
                   for j to i - 1 do
                         for k to i - j do
                               grammaire (T [i-j ][k], T [i][j], T [i][j + k]
```

ℓ est alors une borne supérieure de la longueur de phrase. En fait, il est alors possible de ne plus se donner a priori le tableau triangulaire, mais de le construire à chaque pas.

Le mode de parcours est illustré par la figure 75



application de la règle AB → C sur T [i-j], [k], T [i][j] → T [i][j+k]

Figure 75

Bien entendu, le nombre d'opérations de l'algorithme est le même. Cependant, il y a lieu de remarquer que le nombre d'opérations peut être réduit dans certains cas. En effet, les différents appels de la procédure grammaire, utilisent séquentiellement le deuxième sommet de coordonnées i, j.

Ce sommet sert donc de "pivot". Si l'on sait que ce sommet ne peut apparaître comme élément de droite dans une arborescence, il est possible d'éviter les (i-j) appels de la procédure grammaire. Ce cas interviendra si le symbole associé, soit A, est tel qu'il n'existe aucune règle $C \rightarrow BA$, quels que soient C et B dans la grammaire.

De plus, il est clair que tout sommet de type $\sigma[i,j]$ est racine d'une arborescence contenant $\sigma[i, 1]$ comme élément le plus à droite. Un tel sommet ne peut donc être construit que si $\sigma[i, 1]$ peut intervenir au moins une fois comme élément de droite.

Nous éviterons les deux boucles sur j et k pour tout I tel que $\sigma[i, 1]$ ait cette propriété. Le nombre d'itérations gagnées est alors de $\frac{i\,(i-1)}{2}$ Ceci est réalisé par le test "pas droit" de l'algorithme décrit en annexe 2.

Finalement, la "trace" d'une analyse syntaxique (voir annexe I) se présente sous la forme d'un profil en "dents de scie", laissant apparaître des parallélogrammes inexplorés [figure 76].

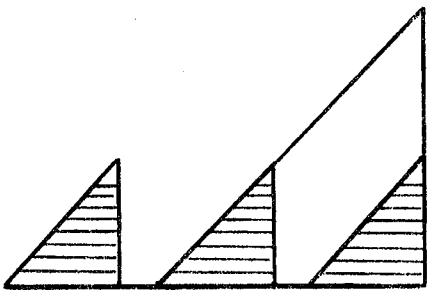


Figure 76

2.4.3. MISE EN OEUVRE DE L'ALGORITHME SUR ORDINATEUR 7044

Le principe de balayage que nous venons d'exposer supposait l'absence d'ambiguïtés. En pratique, chaque sommet σ_{ij} peut être affecté de plusieurs symboles. De plus, plusieurs sous arborescences binaires différentes peuvent correspondre à un tel sommet.

Exemples :

Soit la suite : "Pièce de monnaie ancienne", il lui correspond un sommet σ_{44} affecté d'un seul symbole (substantif féminin singulier), mais deux dérivations peuvent y conduire.

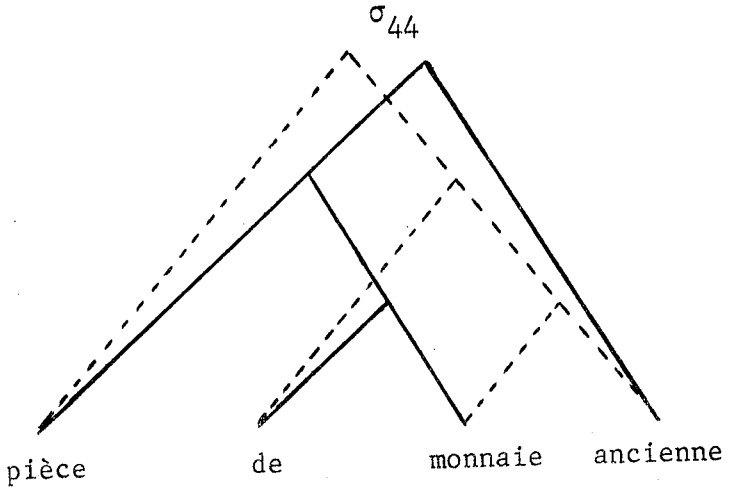


Figure 77

Dans la phrase "le pilote ferme la porte", il existe un sommet σ_{55} affecté du symbole unique 'PHRASE', mais correspondant à deux dérivations différentes.

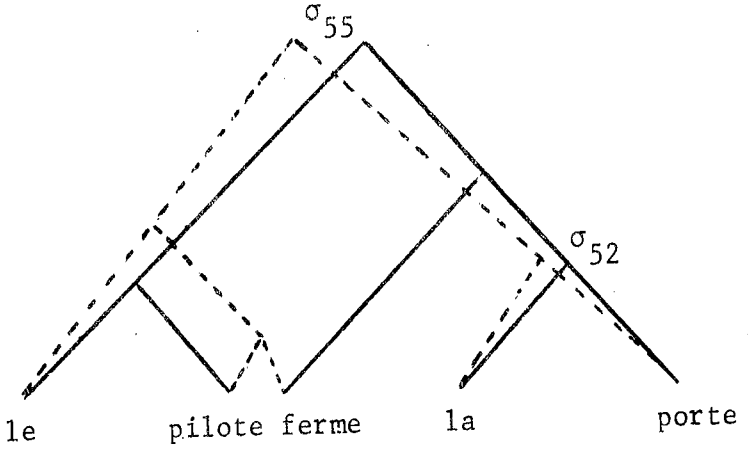


Figure 78

Remarquons que le sommet σ_{52} correspond à deux symboles différents (porte verbe, porte substantif), de même que σ_{41} , σ_{51} ainsi que σ_{31} .

Nous tenons compte de ces deux remarques en considérant indépendamment, dans le programme, le système d'adressage (tableau triangulaire), les symboles et les structures, ces trois types d'informations étant évidemment liées par un ensemble de pointeurs.

Nous utilisons un tableau d'adressage triangulaire de $\frac{n\ (n+1)}{2}$ pointeurs pour noter le repérage des sommets de la structure, conformément à l'algorithme général.

Nous avons utilisé le langage Algol 68 pour décrire la méthode de programmation qui a été effectivement réalisée sur 7044 .[Gd]

2.4.4. ORGANISATION DES STRUCTURES EN MEMOIRE

Chaque sommet d'une arborescence de dérivation est caractérisé par un symbole du vocabulaire non terminal, le numéro de la règle de production qui a permis de le construire, et deux pointeurs caractérisant les sommets de niveau inférieur : soit : R_1 BC $\rightarrowtail$ A (reconnaissance)

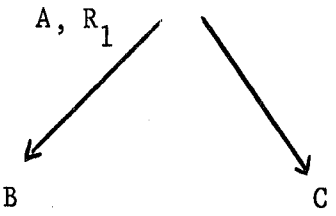


Figure 79

Cependant, dans le cas d'une analyse parallèle, plusieurs dérivations peuvent conduire à une même position σ_{ij} . Deux cas sont alors possibles, ou bien le symbole du vocabulaire terminal est différent, et nous devrons distinguer deux sommets en les plaçant dans une liste, ou bien, le symbole est le même. Dans le dernier cas, plutôt que de considérer plusieurs apparitions du même symbole, ce qui

augmentera d'autant le nombre d'opérations, nous ne distinguerons que les numéros de règles et les pointeurs.

La figure 80 illustre cette disposition. A un sommet σ_{ij} correspond une liste de symboles, liés entre eux par le pointeur 'autre' ; à chaque symbole, correspond une liste de noms de règles et de doublets 'droite', 'gauche'.

Ceci correspond aux déclarations de structure 'sommet' et 'liste' dans le programme (annexe 2).

```
Struct sommet = (pointe autre, ref liste constituants, syntagme synt,
                 bool cote)
     liste = (bool première, ref liste suivante, int règle, pointe
              gauche, pointe droite)
     syntagme = (int cat, bits variables, int occ, int ul)
mode pointe = union (ref sommet, ref som)
```

Le mode union pointe permet de réaliser le nettoyage ('garbage collector') les informations 'cote' et 'première' ne sont pas utilisées à ce niveau.

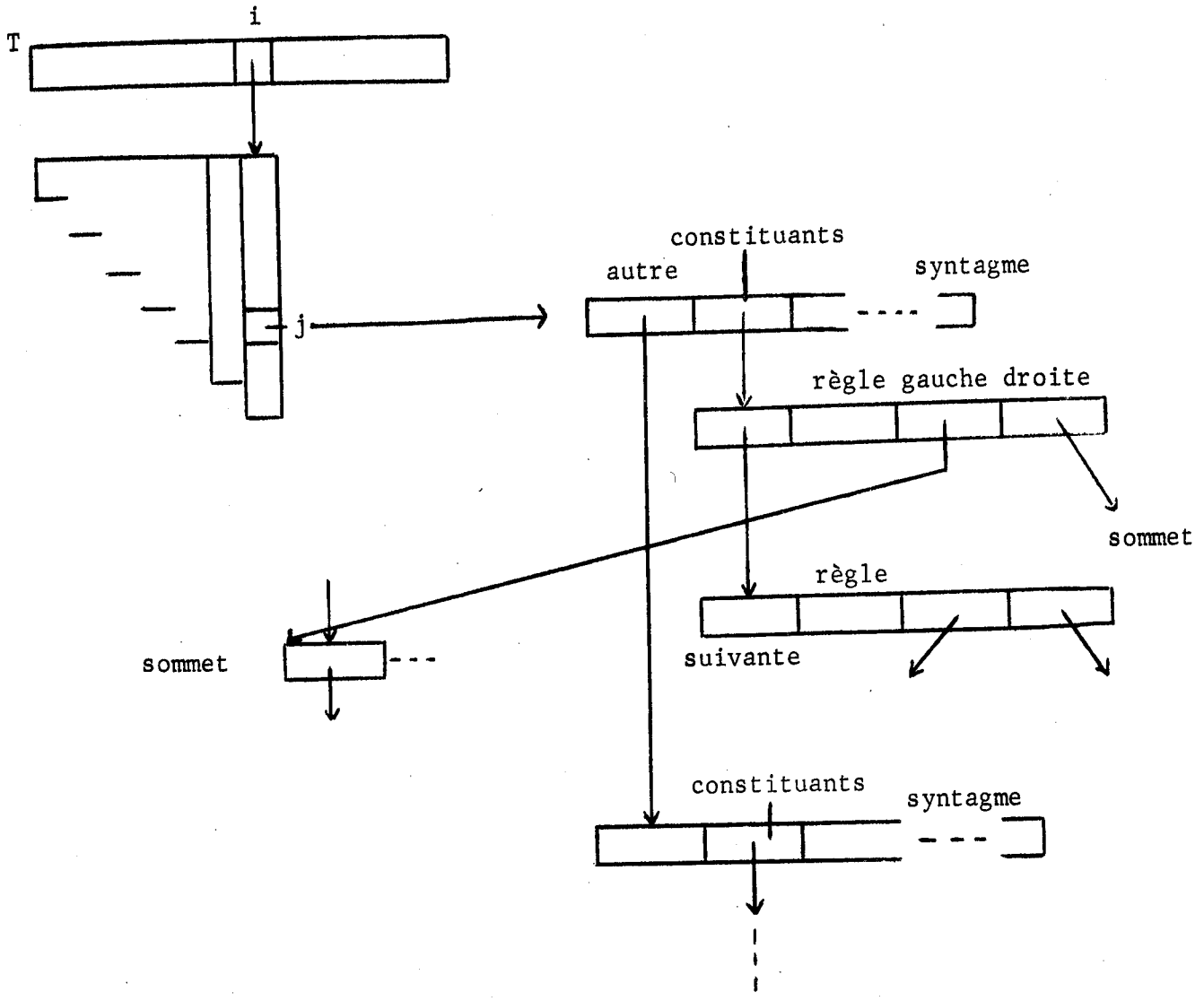


Figure 80

L'utilisation des algorithmes d'analyse ascendante a pour défaut de nécessiter un volume important de mémoire. En effet, il est nécessaire de conserver jusqu'à la fin du traitement toutes les analyses partielles. Cet inconvénient correspond effectivement à l'utilisation d'un traitement parallèle, permettant la mise en commun des sous structures invariantes.

L'expérience nous a montré que, sur l'ensemble des informations conservées sur les dérivations, seulement 10 % était effectivement utilisées dans des structures allant jusqu'à la phrase complète.

Le nombre d'ambiguïtés ne dépend pas directement de la longueur de la phrase, mais plutôt des constructions syntaxiques utilisées. C'est pourquoi nous avons recherché un mode de rangement qui n'impose pas, a priori, une longueur maximale limitée. Les limites imposées par la taille de la mémoire n'interviennent que sur le volume des structures elles-mêmes.

Chaque règle de production est caractérisée par un couple de symboles unique. Les deux pointeurs de structure renvoient donc à un symbole d'une liste et un seul.

Chaque liste de symboles peut être adressée par un pointeur T_{ij} du tableau. La procédure "grammaire" devra donc rechercher tous les couples de symboles $\sigma\,[k,\ i-j-k+1\,]$, $\sigma\,[i-j+1,\ j]$ et construire, ou mettre à jour la liste des symboles et les listes de règles de $\sigma\,[k,\ i-k+1\,]$.

Ces données étant complètement organisées en listes, leur rangement en mémoire peut être quelconque.

Le tableau d'adressage est lui-même variable ; compte tenu de ses propriétés, il peut être formé de n vecteurs colonnes de 1, 2 ... n pointeurs. Une table d'adressage permet de repérer chaque vecteur. Ainsi, la lecture du mot de rang (n + 1) provoque la création d'un vecteur de (n + 1) pointeurs.

La borne supérieure du nombre de mots n'est donc liée qu'à la taille de la table d'adressage du vecteur, qui est fixée par un paramètre (150 mots). L'ensemble de mémoires correspondant à la structure et aux vecteurs d'adressage constitue une seule zone, que l'on garnit progressivement.

Il a été possible ainsi d'analyser certaines phrases russes de 80 mots, alors que des phrases très ambigues de 50 mots suffisaient à saturer la capacité de mémoire.

2.4.5.EFFICACITE DE L'ALGORITHME

Cet algorithme, qui a été programmé en langage assembleur MAP 7044 par E. Grandjean [Gd], analyse les phrases russes à une cadence moyenne de 5 à 10 mots par seconde. La contrainte sur la taille des phrases n'est pas excessive, si on la compare avec celles imposées par les autres algorithmes (72 mots pour l'algorithme de Kuno Oettinger). En pratique, une faible proportion (1 %) des phrases ne peuvent être analysées à cause des restrictions sur la longueur. L'utilisation d'une grammaire mise sous forme normale (règles binaires) peut être, dans certains cas, une contrainte dans l'écriture de la grammaire par le linguiste.

Par contre, cette présentation permet de réduire le nombre de pas de l'algorithme. En effet, la reconnaissance d'une chaine nécessite dans tous les cas une comparaison par symbole. Ainsi, la reconnaissance de la chaine ABC, correspondant à la règle X →ABC ne nécessite pas moins de comparaisons que la reconnaissance de AB, X'C si l'on utilise les deux règles : X' → AB X → X'C.

Dans les cas d'ambiguïtés, plusieurs symboles du type C peuvent apparaître au cours de dérivations, fournissant les chaines ABC', ABC", ABC"' . Dans ce cas, les règles binaires, appliquées indépendamment, permettent d'analyser : AB puis X'C', X'C", X'C"' ; le nombre de comparaisons est alors inférieur à l'application de X → ABC.

Il semble que le gain de temps le plus important soit dû à la possibilité de regrouper plusieurs dérivations conduisant à un seul symbole, (exemple : "pièce de monnaie ancienne").

Dans l'optique d'un algorithme d'analyse syntaxique des langues naturelles, les méthodes ascendantes ont l'avantage de ne pas nécessiter une grammaire générative rigoureuse. En effet, le but de l'analyse est de fournir la ou les structures syntaxiques correctes pour les phrases d'un corpus. La grammaire que l'on utilise devra être capable de fournir une analyse, sans pour cela contrôler strictement la correction de toutes les structures. Par exemple, il ne sera peut

être pas nécessaire de contrôler les places respectives de Ne, Le et Lui dans : "JE NE LE LUI AI PAS DIT", si cela ne conduit pas à de fausses ambiguïtés. Nous pouvons donc tenir compte des redondances du langage pour son analyse. Ceci ne serait pas possible dans un système d'analyse descendante, qui utilise effectivement les productions de la grammaire. De ce fait, cette dernière sera plus difficile à écrire, et comportera un plus grand nombre de règles de production.

2.4.6. ECRITURE DE LA GRAMMAIRE

Il ne peut être envisagé d'écrire directement l'ensemble des règles de production d'une grammaire d'une langue naturelle. C'est pourquoi nous utilisons une notation permettant de simplifier cette écriture.

La notation tient compte des possibilités de l'analyse, et la grammaire ainsi écrite est non réversible : il n'est pas possible de l'utiliser comme grammaire générative.

Chaque symbole non terminal de la grammaire est en fait caractérisé par un nom de catégorie syntaxique, et un ensemble d'informations appelées valeurs de variables. Chaque variable est susceptible de prendre un certain nombre de valeurs : ainsi la variable GENRE prendra les valeurs MASCULIN, FEMININ, NEUTRE. Un "syntagme", correspondant à un élément du vocabulaire non terminal, est caractérisé par un ensemble de valeurs de variables. Les règles de production ne sont applicables que si certaines conditions logiques sur les variables sont satisfaites : la règle combinant substantif et adjectif ne peut s'appliquer que si il y a accord en genre et en nombre.

Le langage de descriptions des grammaires a particulièrement été décrit dans [V_4]. Les règles sont réalisées par des sous programmes, qui sont générés par un compilateur [G_d]. Dans le programme ALGOL 68 (Annexe 2) , la grammaire est donnée sous la forme simplifiée d'un tableau de procédures.

2.5. ALGORITHME DE TRANSFORMATION

2.5.1. ECRITURE DE LA STRUCTURE DE DEPENDANCE

Nous pouvons noter une structure de dépendance sous la forme d'une chaine parenthésée, en utilisant la structure de constituants fortement équivalente.
Exemple :

La structure :

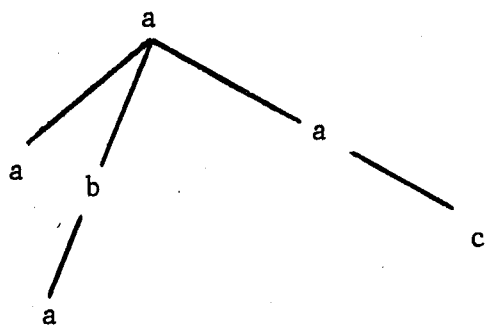


Figure 81

sera notée : ((a) ((a) b) a (a (c)), chaque forme parenthésée décrit ne sous arborescence complète.

A chaque niveau de parenthèse correspond un élément terminal et un seul. La justification rigoureuse de ce qui suit est donnée en annexe III.

Si nous faisons appel à une grammaire mise sous forme normale, il est possible de transformer cette grammaire pour produire de telles chaines parenthésées.

Soit : $G = (V_T , V_N , S, P)$ et une interprétation I (P), tel que I (p) = 'gauche' ou I (p) = 'droite' si p est une production non terminale.

Par exemple :

$$V_N = \{A, B, C, S\} \quad , V_T = \{a, b, c\}$$

P : r_1 S → AB I = 'droite'

r_2 B → SC I = 'droite'

r_3 C → AC I = 'gauche'

A → a

B → b

C → c

et la dérivation de la chaine a a b a a c

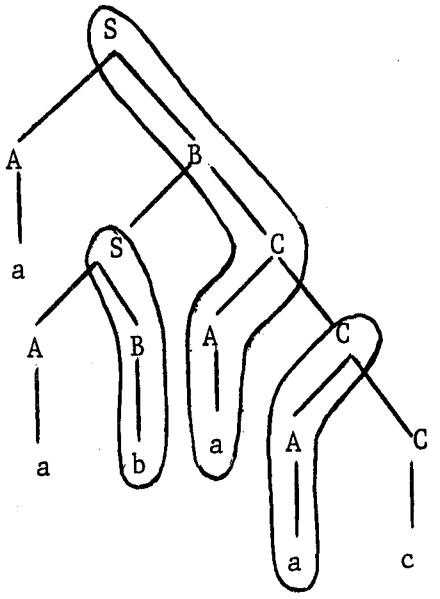


Figure 82

Cette structure est la structure de dépendance de la figure 81 .

Pour produire la chaine parenthésée, nous associerons à G la grammaire $G' = [V'_T , V'_N , S_0 , P']$ telle que :

$V'_T = V_T \cup \{ (,) \}$

$V'_N = V_N \cup \{ S_0 \}$

P' est telle que : si $[A \rightarrow BC] \in P$ et si
I $[A \rightarrow BC]$ = 'gauche', alors $[A \rightarrow B\ (C)] \in P'$

si I $[A \rightarrow BC]$ = 'droite' , $[A \rightarrow (B)\ C] \in P'$

si $[A \rightarrow a] \in P$, $[A \rightarrow a] \in P'$

Enfin P' contient en outre la production $S_0 \rightarrow (S)$. En poursuivant l'exemple, nous obtiendrons la grammaire G' :

$S_0 \rightarrow (S)$

$S \rightarrow (A)\ B \qquad A \rightarrow a$

$B \rightarrow (S)\ C \qquad B \rightarrow b$

$C \rightarrow A\ (C) \qquad C \rightarrow c$

Et la dérivation :

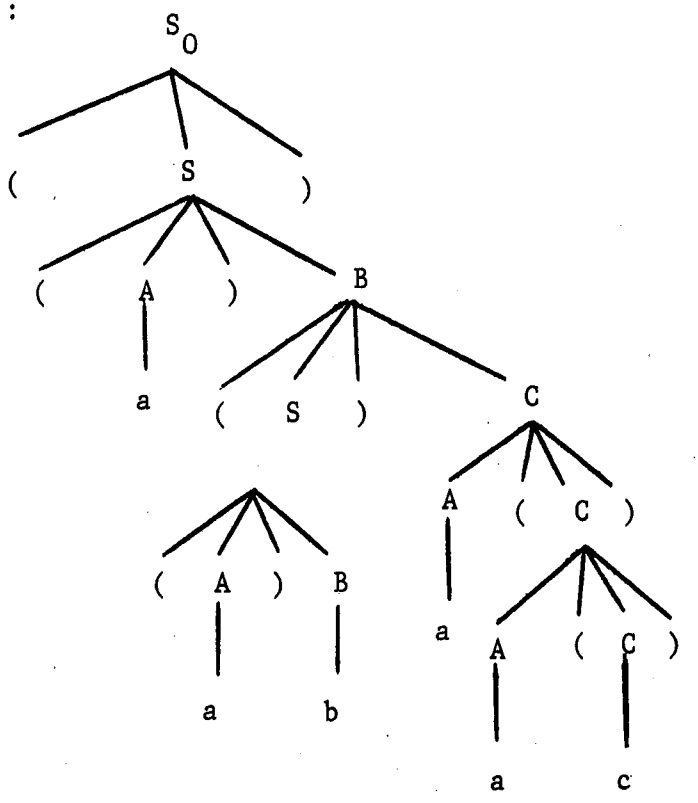


Figure 83

Chaque niveau de parenthésage correspond effectivement à un changement de niveau sur la structure de constituants.

L'on peut déduire de la grammaire G' un transducteur produisant la chaine parenthésée. Le transducteur devra analyser la chaine en parcourant l'arbre de dérivation, et produire les parenthèses à chaque changement de niveau.

2.5.2. TRANSDUCTEUR [Annexe III]

A toute grammaire G sous forme normale, il est possible [CH47] d'associer un automate à pile reconnaissant le langage L_G.

Soit $M = (Q, V, V_A, \delta, q_0, Z_0, F)$ tel que :

Si $G = (V, V_N, S, P)$

$Q = \{A^1, A^2 \mid A \in V_N\}$

$V_A = V_N$

δ est telle que

si $(C \rightarrow AB) \in P$,

$\delta\ (C^1, \xi, \gamma) = (A^1, C\gamma)$ $\quad \gamma = Z_0$ si $C^1 = S^1$, $\gamma \in V_A^*$ sinon

$\delta\ (A^2, \varepsilon, \gamma) = (B^1, \gamma)$

$\delta\ (B^2, \varepsilon, C\gamma) = (C^2, \gamma)$

si $(A \rightarrow a) \in P$

$\delta\ (A^1, a, \gamma) = (A^2, \gamma)$

$q_0 = S^1$, $F = \emptyset$

Le parcours de l'arborescence est représenté sur la figure 84

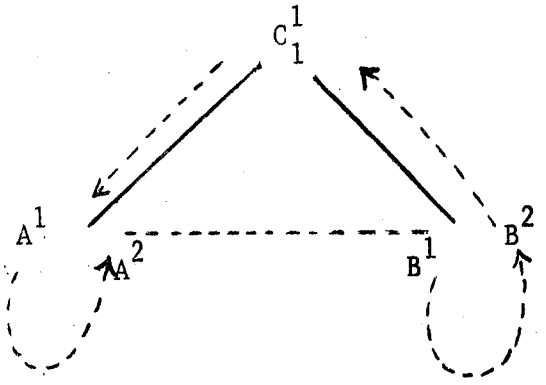


Figure 84

Nous déduirons de l'automate M et de l'interprétation I de la grammaire un transducteur $T = (Q', V, V'_N, V_S, \delta, q_0, Z_0)$, dans lequel $V'_N = V_N \cup \{S_0\}$, $Q' = Q \cup \{S_0^1, S_0^2\}$

$$V_S = V_T \cup \{(,)\}$$

A toute règle $C \rightarrow AB$, il correspond :

$\delta\ (C^1, \varepsilon, \gamma) = (A^1, C\gamma, '(')$ si I (C → AB) = 'droite'

$\delta\ (C^1, \varepsilon, \gamma) = (A^1, C\gamma, \varepsilon)$ si I (C → AB) = 'gauche'

$\delta\ (A^2, \varepsilon, \gamma) = (B^1, \gamma, ')')$ si I (C → AB) = 'droite'

$\delta\ (A^2, \varepsilon, \gamma) = (B^1, \gamma, '(')$ si I (C → AB) = 'gauche'

$\delta\ (B^2, \varepsilon, C\gamma) = (C^2, \gamma, \varepsilon)$ si I (C → AB) = 'droite'

$\delta\ (B^2, \varepsilon, C\gamma) = (C^2, \gamma, ')')$ si I (C → AB) = 'gauche'

$q_0 = S_0^1$,

$\delta\ (S_0^1, \varepsilon, Z_0) = (S^1, S_0 Z_0, '(')$

$\delta\ (S^2, \varepsilon, S_0 Z_0) = (S_0^2, Z_0, ')')$

si $(A \rightarrow a) \in P$

$\delta\ (A^1, a, \gamma) \rightarrow (A^2, \gamma, a)$

Exemple : En reprenant la grammaire G donnée en 5.1, nous aurons :

$\Delta (S_0^1 , \varepsilon , Z_0) = (S^1 , S_0 Z_0, '(')$

$\Delta (S^2 , \varepsilon , S_0\gamma) = (S_0^2 , \gamma , ')')$

$\Delta (S^1 , \varepsilon , \gamma) = (A^1 , S\gamma , '(')$

$\Delta (A^2 , \varepsilon , \gamma) = (B^1 , \gamma , ')')$

$\Delta (B^2 , \varepsilon , S\gamma) = (S^2 , \gamma , \varepsilon)$

$\Delta (B^1 , \varepsilon , \gamma) = (S^1 , B\gamma , '(')$

$\Delta (S^2 , \varepsilon , \gamma) = (C^1 , \gamma , ')')$

$\Delta (C^2 , \varepsilon , B\gamma) = (B^2 , \gamma , \varepsilon)$

$\Delta (C^1 , \varepsilon , \gamma) = (A^1 , C\gamma , \varepsilon)$

$\Delta (A^2 , \varepsilon , \gamma) = (C^1 , \gamma , '(')$

$\Delta (C^2 , \varepsilon , C\gamma) = (C^2 , \gamma , ')')$

$\Delta(A^1 , a , \gamma) = (A^2 , \gamma , a)$

$\Delta(C^1 , c , \gamma) = (C^2 , \gamma , c)$

$\Delta(B^1 , b , \gamma) = (B^2 , \gamma , b)$

Transduction de la chaine ≠ a a b a a c ≠

Etat	Entrée	Pile	Sortie
S_0^1	a a b a a c ≠	Z_0	
S^1	a a b a a c ≠	$S_0 Z_0$	(
A^1	a a b a a c ≠	S $S_0 Z_0$	((
A^2	a b a a c ≠	S $S_0 Z_0$	((a
B^1	a b a a c ≠	S $S_0 Z_0$	((a)
S^1	a b a a c ≠	B S $S_0 Z_0$	((a) (
A^1	a b a a c ≠	S B S $S_0 Z_0$	((a) ((
A^2	b a a c ≠	S B S $S_0 Z_0$	((a) ((a
B^1	b a a c ≠	S B S $S_0 Z_0$	((a) ((a)
B^2	a a c ≠	S B S $S_0 Z_0$	((a) ((a) b
S^2	a a c ≠	B S $S_0 Z_0$	((a) ((a) b
C^1	a a c ≠	B S $S_0 Z_0$	((a) ((a) b)
A^1	a a c ≠	C B S $S_0 Z_0$	((a) ((a) b)
A^2	a c ≠	C B S $S_0 Z_0$	((a) ((a) b) a
C^1	a c ≠	C B S $S_0 Z_0$	((a) ((a) b) a (
A^1	a c ≠	C C B S $S_0 Z_0$	((a) ((a) b) a (
A^2	c ≠	C C B S $S_0 Z_0$	((a) ((a) b) a (a
C^1	c ≠	C C B S $S_0 Z_0$	((a) ((a) b) a (a (
C^2	≠	C C B S $S_0 Z_0$	((a) ((a) b) a (a (c
C^2	≠	C B S $S_0 Z_0$	((a) ((a) b) a (a (c)
C^2	≠	B S $S_0 Z_0$	((a) ((a) b) a (a (c))
B^2	≠	S $S_0 Z_0$	((a) ((a) b) a (a (c))
S^2	≠	$S_0 Z_0$	((a) ((a) b) a (a (c))
S_0^2	≠	Z_0	((a) ((a) b) a (a (c)))

2.5.3. TRANSDUCTION DE L'ARBRE D'ANALYSE

Etant donné le procédé adopté pour l'analyse syntaxique, nous déterminerons la structure de dépendances après cette analyse, c'est-à-dire à partir de la structure de constituants proprement dite. Nous devrons donc effectuer la transduction à partir de l'arborescence binaire. L'interprétation est fournie sous la forme de l'information 'gauche' ou 'droite' sur chaque sommet non terminal. Cette arborescence est donnée sous la forme préfixée classique.

En reprenant la structure fournie en 5.1. , l'arborescence peut être notée : S A a B S A a B b C A a C A a C c, ou en notant directement l'interprétation et en négligeant les productions terminales : δ a δδ a b γ a γ a c

δ= 'DROITE' , γ = 'GAUCHE' ce qui correspond à l'arborescence

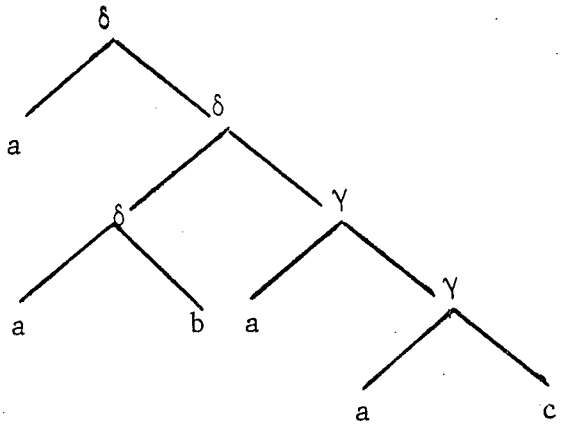


Figure 85

Nous pouvons déduire du transducteur 5.2. un transducteur T' associant à cette chaine préfixée une chaine parenthésée. Ce transducteur ne dépend plus de la grammaire, puisque la structure est donnée.

Il suffit de considérer un parcours équivalent de l'arborescence. Ce transducteur est le modèle du programme d'interprétation (Annexe II). Les 2 états correspondent aux 2 possibilités que l'on y rencontre. L'ouverture d'une parenthèse se traduit par la création d'un nouveau sommet de l'arborescence.

$$T' = \{Q''\ V'',\ V''_N,\ V''_S\ \pi,\ q_0,\ Z_0\}$$

$$V'' = \{\ \gamma,\ \delta\ \} \cup V$$

$$V''_N = \{\ \gamma,\ \delta,\ \gamma'\ ,\ \delta',\ S\}$$

$$V''_S = V_T \cup \{(\ ,\ )\}$$

$$Q\ \ = \{q_0,\ q_1,\ q_2\}$$

π est telle que :

$\pi\ (\ q_0,\ \varepsilon\ ,\ Z_0) = (q_1\ ,\ SZ_0,\ '('\ )$

$\pi\ (\ q_1,\ \delta\ ,\ \varphi\ ) = (q_1\ ,\delta\varphi\ ,\ '('\ )\quad \varphi \in V''^{*}_N$

$\pi\ (\ q_1,\ \gamma\ ,\ \varphi\ ) = (q_1\ ,\gamma\varphi\ ,\ \varepsilon\ )$

$\pi\ (\ q_1,\ t\ ,\ \varphi\ ) = (q_2\ ,\ \varphi\ \ ,\ t\ )\quad t \in V$

$\pi\ (\ q_2,\ \varepsilon\ ,\delta\varphi\ ) = (q_1\ ,\ \delta'\varphi\ ,\ ')'\ )$

$\pi\ (\ q_2,\ \varepsilon\ ,\gamma\varphi\ ) = (q_1\ ,\gamma'\varphi,\ '('\ )$

$\pi\ (\ q_2,\ \varepsilon\ ,\delta'\varphi) = (q_2\ ,\ \varphi\ \ ,\ \varepsilon\ )$

$\pi\ (\ q_2,\ \varepsilon\ ,\gamma'\varphi) = (q_2\ ,\ \varphi\ \ ,\ ')'\ )$

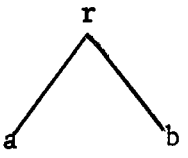
$\pi\ (\ q_2,\ \varepsilon\ ,S_0\varphi) = (q_0\ ,\ \varphi\ \ ,\ \varepsilon\ )$

Transduction de 'δ a δ δ a b γ a γ a c'

Etat	Entrée	Pile	Sortie
0	δ a δ δ a b γ a γ a c	Z_0	
1	δ a δ δ a b γ a γ a c	S Z_0	(
1	a δ δ a b γ a γ a c	δ S Z_0	((
2	δ δ a b γ a γ a c	δ S Z_0	((a
1	δ δ a b γ a γ a c	δ' S Z_0	((a)
1	δ a b γ a γ a c	δ δ' S Z_0	((a) (
1	a b γ a γ a c	δδ δ' S Z_0	((a) ((
2	b γ a γ a c	δδ δ' S Z_0	((a) ((a
1	b γ a γ a c	δ 'δδ' S Z_0	((a) ((a)
2	γ a γ a c	δ' δδ' S Z_0	((a) ((a) b
2	γ a γ a c	δ δ' S Z_0	((a) ((a) b
1	γ a γ a c	δ 'δ' S Z_0	((a) ((a) b)
1	a γ a c	γ δ 'δ' S Z_0	((a) ((a) b)
2	γ a c	γ δ 'δ' S Z_0	((a) ((a) b) a
1	γ a c	γ'δ 'δ' S Z_0	((a) ((a) b) a (
1	a c	γ γ'δ 'δ' S Z_0	((a) ((a) b) a (
2	c	γ γ'δ 'δ' S Z_0	((a) ((a) b) a (a
1	c	γ'γ'δ 'δ' S Z_0	((a) ((a) b) a (a (
2		γ'γ'δ 'δ' S Z_0	((a) ((a) b) a (a (c
2		γ'δ 'δ' S Z_0	((a) ((a) b) a (a (c)
2		δ 'δ' S Z_0	((a) ((a) b) a (a (c))
2		δ' S Z_0	((a) ((a) b) a (a (c))
2		S Z_0	((a) ((a) b) a (a (c))
0		Z_0	((a) ((a) b) a (a (c)))

2.5.4. AFFECTATION DES NUMEROS DE REGLE

Il existe un algorithme simple permettant, à partir de la chaine préfixée, de définir le nom de règle affecté à chaque terminal. Ceci est lié à la remarque suivante :

Soit la configuration 

dans laquelle r est un nom de règle, a et b les éléments terminaux. La notation préfixée sera 'r a b'. Si r est 'gauche', a gouverne b et le nom r sera affecté à b. Si r est droite, b est gouverneur et r est affecté à a. Si l'on possède par réductions successives, en remplaçant toute chaine de type ' r a b' par son gouverneur après avoir choisi l'affectation de règle, nous réduirons toute l'arborescence.

Exemple : soit l'exemple de la figure 8 2 , noté en préfixé :

$r_1\ a\ r_2\ r_1\ a\ b\ r_3\ a\ r_3\ a\ c$, I (r_1) = droite, I (r_2) = droite ,
I (r_3) = gauche

$\underline{r_3\ a\ c} \rightarrow a$, r_3 affecté à c ; d'où $r_1\ a\ r_2\ r_1\ a\ b\ r_3\ a\ \underline{r_3\ a\ c}$

$\longrightarrow r_1\ a\ r_2\ r_1\ a\ b\ \underline{r_3\ a\ a} \rightarrow r_1\ a\ r_2\ \underline{r_1\ a\ b\ a} \rightarrow r_1\ a\ \underline{r_2\ b\ a} \rightarrow \underline{r_1\ a\ a} \rightarrow a$

Nous transposerons cette remarque au niveau de l'algorithme, en déterminant l'affectation en même temps que la structure.

2.5.5. AFFECTATION DES SYNTAGMES

La structure de dépendance doit contenir, pour chaque sommet, les informations syntaxiques contenues dans les syntagmes.

A un sommet donné correspond un chemin de la structure de constituants, et plusieurs syntagmes. Nous retiendrons les informations liées au syntagme le plus haut de la sous arborescence. En effet, les règles de grammaires interviennent comme des filtres sur les variables et permettent de lever certaines ambiguïtés : 'Bois' substantif peut être singulier ou pluriel , 'Les bois' est spécifiquement pluriel.

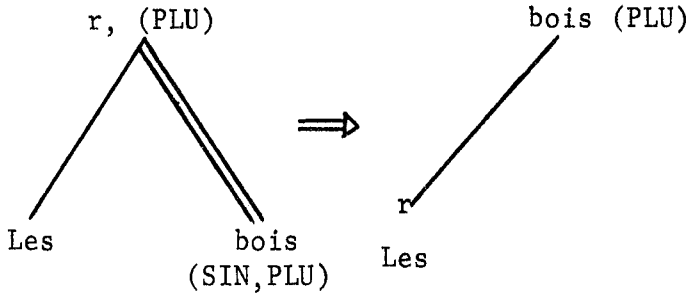


Figure 86

Cependant, les informations lexicales, telles que le numéro d'unité lexicale (référence au dictionnaire), sont liées à l'élément terminal, symbolisé par le mot 'bois' dans la figure 86 .

Finalement, nous retiendrons les deux types d'information, que nous noterons "syntagme haut" et "syntagme bas".

2.5.6. COMPARAISON DES STRUCTURES

Dans de nombreux cas, l'analyse syntaxique fournit plusieurs structures homographes pour une phrase donnée. Dans la suite du traitement, ces structures seront examinées indépendamment. Certaines peuvent alors être rejetées en fonction d'incohérences qui échappaient au modèle d'analyse. Souvent, ces homographies peuvent être éliminées a priori à la suite d'une comparaison des structures entre elles. C'est pourquoi nous avons introduit un filtre supplémentaire complétant l'analyse syntaxique. Ce filtre compare les structures homographes entre elles.

Actuellement, le système de comparaison est extrêmement simplifié. Il procède de la façon suivante :

1° Les structures de constituants d'une phrase donnée sont comparées

deux à deux, sans tenir compte des symboles affectés aux sommets.

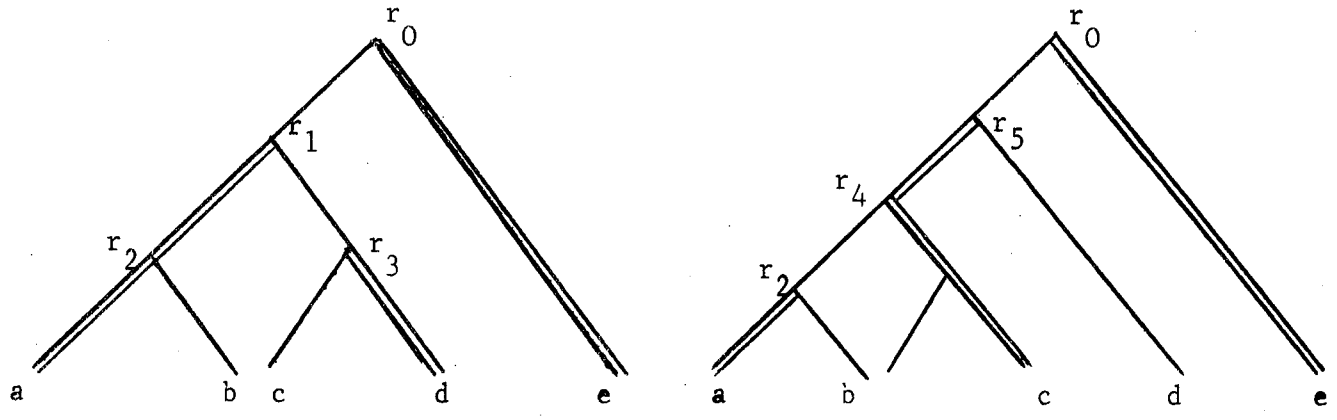
2° Les noms de règles affectés aux éléments terminaux sont comparés pour les mêmes structures.

Quatre résultats sont possibles à la suite de ces deux tests. Si les structures et les affectations de règles sont identiques, une des deux structures est éliminée. Si les règles sont identiques avec des structures différentes, les deux structures sont adoptées (exemple de la figure 77).

Dans les deux autres cas, l'on consulte une liste de priorités. Cette liste contient des couples de noms de règle (r_i , r_j) associés à un entier p_{ij} appelé poids de la priorité de r_i sur r_j .

Pour chaque sommet de la structure interprétée, le choix entre deux noms de règle différents peut être décidé par une priorité. La structure retenue sera celle qui contiendra la priorité de plus grand poids.

Exemple : Soient les structures :



Les structures sont différentes. Les règles affectées sont :

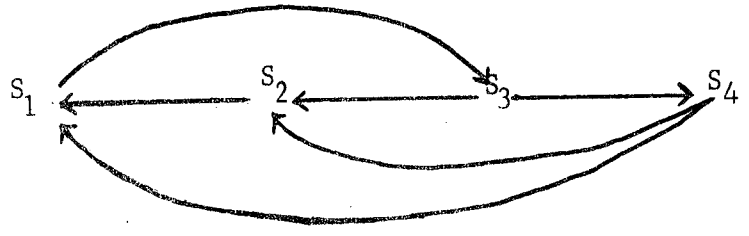
première structure		deuxième structure	
a	r_0	a	r_4
b	r_2	b	r_2
c	r_3	c	r_0
d	r_1	d	r_5
e	$\emptyset$	e	$\emptyset$

Les comparaisons nous montrent les choix :

a	r_0	r_4
c	r_3	r_0
d	r_1	r_5

Si l'on a les priorités : $(r_0 , r_4 , 12)$, $(r_3 , r_0 , 5)$, $(r_5 , r_1 , 100)$, la deuxième structure est conservée (priorité 100)

Ce système permet de lever un grand nombre de fausses ambiguïtés et s'est révélé très efficace. Cependant, il comporte beaucoup de défauts. D'une part, il procède par élimination. De ce fait, le résultat peut être fonction de l'ordre dans lequel sont réalisées les comparaisons : si l'on a quatre structures S_1 , S_2 , S_3 , S_4 , il est possible d'avoir le graphe suivant :



où les arcs montrent les éliminations possibles. Si l'on opère dans l'ordre S_1, S_2, S_3, S_4, la structure retenue sera S_3, mais l'ordre S_1 S_3 S_4 S_2 nous conduirait à retenir S_4.

En examinant tous les couples possibles, nous pourrions effectuer une élimination exacte, en tenant compte du 'poids' de chaque arc. Une telle solution n'a pu être retenue car elle conduit à un temps de traitement trop important, et oblige à stocker toutes les solutions.

Une meilleure méthode consisterait à tenir compte des particularités des structures, en ne considérant que les sous arborescences différentes.

C H A P I T R E III

MODELES TRANSFORMATIONNELS (MODELES ARBRE-ARBRE)

3.1.1. NOTION DE REGLE DE TRANSFORMATION

Pour obtenir la structure décrivant la phrase en langage pivot, et pour reconstruire ensuite la structure syntaxique française, il est nécessaire de disposer d'un modèle établissant des correspondances entre arborescences. Nous avons déjà rencontré un modèle de ce type dans le cas de la transformation des structures de constituants en structures de dépendance, et dans le cas des transformations liées aux variables véhiculaires. Les transformations utilisées alors présentaient un aspect systématique. Les configurations demandant de telles transformations sont très simples. Dans un système plus général de transformations de structures arborescentes, nous devons faire appel à des règles plus puissantes.

Une grammaire de transformation se présente sous la forme d'une liste de règles. Une règle de transformation est elle-même constituée de deux parties : le schéma de figure, qui définit la configuration de la structure pour laquelle on doit appliquer la transformation et la transformation elle-même, qui est constituée par une suite de transformations élémentaires.

En d'autres termes, le schéma de figure permet d'isoler certains éléments de la structure, et les transformations définissent les nouvelles relations qui doivent intervenir entre ces éléments, et, éventuellement, d'autres éléments. Ainsi, nous pouvons considérer qu'une règle syntaxique de type 'hors contexte' est une transformation et écrire :

A B ⟵ C sous la forme :

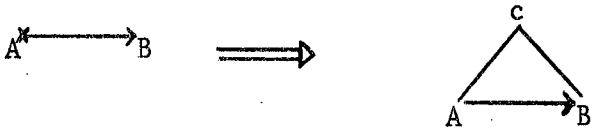
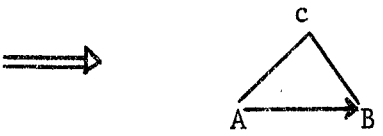


Figure 87

en reconnaissance

ou : C ⟶ AB sous la forme :



Pour construire la structure associée à la dérivation dans le cas de l'analyse syntaxique, nous avons pu définir un algorithme d'analyse global, indépendant du contenu de la grammaire.

En ce qui concerne les transformations, nous ne pouvons utiliser de tels algorithmes, la structure analysée étant plus complexe (arborescence) que la structure de chaîne ; de plus, les transformations ne sont pas systèmatiques mais sont fonction des schémas de figures.

En pratique, ces structures comportent plusieurs relations, autres que celle de l'arborescence (relation d'ordre sur la chaîne, alphabet, substitution). C'est pourquoi le système que nous avons défini permet de traiter des structures plus complexes.

Dans ce paragraphe, nous définissons la notion de schéma de figure, et le processus de détermination d'une figure dans une structure donnée.

3.1.2. FIGURE ET SCHEMA

3.1.2.1. Une structure est un ensemble muni de relations. Une relation i-aire sur un ensemble X est une partie de X^i

Parmi les relations, nous distinguerons une base R, et une fermeture $\tilde{R}$, ensemble des relations que l'on peut dériver de R.

Un schéma de figure Σ est une relation i-aire, $r_i \in \tilde{R}$. Une figure de schéma Σ est un i-uple $\xi^i \in X^i$ tel que $\xi^i \in r_i$.

Nous nous proposons ici de définir une notation des schémas permettant de donner un algorithme de recherche d'une figure, si elle existe, à partir d'un schéma donné.

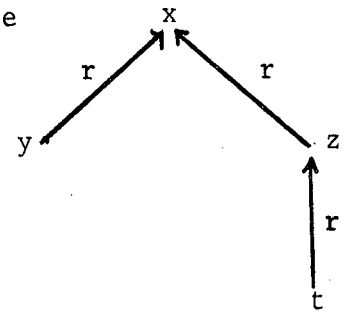
Dans ce but, nous définirons le schéma comme une structure, dont les relations ont une interprétation dans la structure principale.

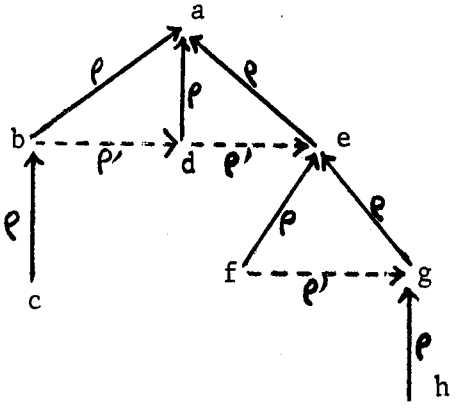
3.1.2.2. Nous nous limiterons au cas où les relations sont binaires, ce qui n'est pas une restriction; nous noterons $\tilde{R}_2$ les relations binaires de $\tilde{R}$. $\tilde{R}_2$ est donc un ensemble de parties de X^2.

Définition : Soit une structure (X, R), les relations binaires associées $\tilde{R}_2$, et un schéma $\Sigma = (S, r)$, ensemble muni d'une seule relation binaire, tels que $X \cap S = \emptyset$, nous appellerons interprétation de r dans $\tilde{R}_2$ une application p de r dans $\tilde{R}_2$, et figure de schéma Σ conformément à l'interprétation p, une application ϕ de Σ sur une partie F de X telle que :

si $(s_i, s_j) \in r$, $(\phi(s_i), \phi(s_j)) \in p(r)$

Exemple : Soient $\Sigma = (\{x, y, z, t\}, r)$ et $E = (\{a, b, c, d, e, f, g, h\}\ \{\rho, \rho'\})$ conformément à la figure





Soit $p(y, x) = \hat{\rho}$, $p(z, x) = \hat{\rho}$, $p(t, z) = \rho'$ dans lequel $\hat{\rho}$ représente la fermeture transitive de ρ.

Alors $\phi(x, y, z, t) = \{a, b, e, d\}$ est une figure mais aussi $\phi(x, y, z, t) = \{a, c, g, f\}$ ou $\{e, f, g, f\}$ ... etc.

Soit encore $p(y, x) = p(z, x) = \hat{\rho}^{-1}$

$p(t, z) = \hat{\rho}'$

Alors nous aurons $\phi(x, y, z, t) = \{g, a, e, d\}$

mais aussi $\phi(x, y, z, t) = \{f, e, e, b\}$

$\phi(x, y, z, t) = \{f, a, e, d\}$

... etc

En pratique, nous nous limiterons à écrire les schémas sous la forme de structures arborescentes, en utilisant des relations binaires et des relations unaires. Cependant, une relation d'identité, entre deux sommets, traitée ici comme une relation unaire, permet de tenir compte de structures plus complexes (cycles).

3.1.2.3. Notation des schémas

Nous noterons les schémas en utilisant une structuration en liste des relations elles-mêmes. Les relations intervenant dans de tels schémas seront unaires ou binaires, ce qui ne constitue pas une restriction de puissance. Pour décrire cette notation, nous utiliserons les méthodes syntaxiques classiques (forme de Backus).

< schéma de figure > ::= < parenthèse ouverte > < liste de propriétés >
< parenthèse fermée >

< liste de propriétés > ::= < propriété > | < propriété > < séparateur >
< liste de propriétés >

< propriété > ::= < relation unaire > | < relation binaire > | < relation binaire >
< schéma de figure >

< parenthèse ouverte > ::= (

< parenthèse fermée > ::=)

< séparateur > ::= ,

En notant Σ, Σ' et Σ'' le schéma, la liste de schémas et le schéma élémentaire, les 3 règles de production seront notées :

règle 1 $\quad \Sigma ::= (\Sigma')$
règle 2 $\quad \Sigma' ::= \Sigma'' \mid \Sigma'', \Sigma'$
règle 3 $\quad \Sigma'' ::= r_1 \mid r_2 \mid r_2\ \Sigma$

Exemple :

Soient ρ_1, ρ'_1 les relations unaires (r_1) et ρ_2, ρ'_2 les relations binaires (r_2) :

$\Sigma_1 = (\rho_1)$ est un schéma

$\Sigma_2 = (\rho_2(\rho_1), \rho'_2\ (\rho_2, \rho'_2\ (\rho'_1)))$ est un schéma

3.1.2.4. Interprétation des schémas

Dans ce qui suit, nous noterons ξ^n un n-uple : $\xi^n = (x_1, x_2, \ldots, x_n)$. A un schéma de figure $\Sigma = (S, R')$ nous associons un prédicat $P_\Sigma(\xi^n)$ si Card (S) = n , $\xi^n \in X^n$.

A chaque règle de production (3.1.2.3.) nous associons une combinaison de prédicats de la façon suivante :

règle 1 : $P_\Sigma(\xi^n) = P_{\Sigma'}(\xi^n)$

règle 2 : $P_{\Sigma'}(\xi^n) = P_{\Sigma''}(\xi^n)$ si $\Sigma' ::= \Sigma''$

ou $P_{\Sigma'}(\xi^n) = P_{\Sigma'}(x, \xi^{n-1}) = P_{\Sigma''}(x, \xi^k) \,\&\, P_{\Sigma'}(x, \xi^{n-k-1})$

si $\Sigma' ::= \Sigma'', \Sigma'$

règle 3 : $P_{\Sigma''}(x, \xi^k) = P_{r_1}(x)$ si $\Sigma'' ::= r_1$ $(k = 0)$

$P_{\Sigma''}(x, \xi^k) = P_{r_2}(x,y)$ si $\Sigma'' ::= r_2$ $(k = 1)$

$P_{\Sigma''}(x, \xi^k) = P_{r_2}(x,y) \,\&\, P_\Sigma(y, \xi^{k-1})$ si $\Sigma'' ::= r_2$ $(k \geqslant 1)$

Les prédicats $P_{r_1}(x)$ et $P_{r_2}(x,y)$ sont les prédicats associés aux relations r_1 et r_2.

Exemple : Reprenons les deux schémas donnés en 3.1.2.3. :

à $\Sigma_1 = (\rho_1)$ nous associerons $P_\Sigma(x) = P_{\rho_1}(x)$

à $\Sigma_2 = (\rho_2\ (\rho_1)\ ,\rho'_2\ (\rho_2\ ,\rho'_2\ (\rho'_1)))$. Nous associerons

$$P_\Sigma(\xi^5) = (P_{\rho_2}(x_1,x_2) \,\&\, P_{\rho_1}(x_2)) \,\&\, (P_{\rho'_2}(x_1,x_3) \,\&\, (P_{\rho_2}(x_3,x_4) \,\&\, (P_{\rho'_2}(x_3,x_5) \,\&\, P_{\rho'_1}(x_5))))$$

Les parenthèses indiquant la correspondance entre le schéma et l'articulation des prédicats

Nous pouvons représenter le schéma de figure sous la forme d'un graphe arborescent faisant apparaître les relations binaires et les sommets.

Exemple : Σ_2 sera représenté par l'arborescence de la figure

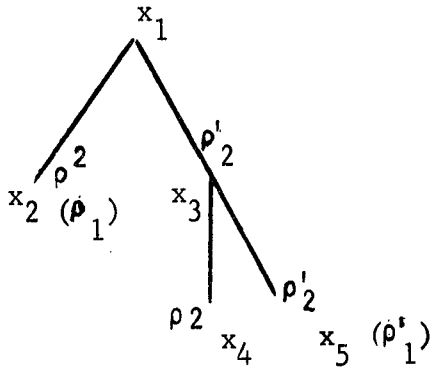


Figure 88

3.1.3. RECHERCHE D'UNE FIGURE DE SCHEMA DONNE

Rechercher une figure revient à déterminer une valeur de ξ^n, soit un n-uple $(a_1, a_2 \ldots, a_n) \in X^n$ tel que $P_\Sigma (a_1, a_2 \ldots a_n)$ soit vrai.

3.1.3.1. Enumération

Le calcul d'un prédicat nécessite de pouvoir énumérer effectivement les éléments de son extension. Dans le cas de la recherche d'une figure, nous voulons obtenir un élément et un seul de l'extension, caractérisant l'application φ de S sur $F \subset X$.

Nous associerons à tout prédicat une fonction d'énumération, qui permet de déterminer l'application, en prenant le premier élément de l'énumération.

3.1.3.2. Fonctions d'énumération

A une structure donnée Δ , on associe une relation d'ordre totale sur X, définissant une fonction d'énumération ("fonction de balayage").

- Relations unaires

La fonction d'énumération de l'extension d'un prédicat unaire $P_{r_1}(x)$ est la fonction de balayage.

- Relations binaires

A toute relation binaire r_2 nous associerons une fonction d'énumération $h_{r_2}(x,i)$ telle que :

$$\{ y \mid (x,y) \in r_2 \} = \{ h_{r_2}(x,i) \mid 0 \leqslant i < \text{Card } X \}$$

Cette fonction est une fonction récursive définie à l'aide de deux fonctions f_{r_2} et g_{r_2} telles que :

$$h_{r_2}(x,0) = f_{r_2}(x)$$

$$h_{r_2}(x,i+1) = g_{r_2}(h_{r_2}(x,i))$$

3.1.3.3. Détermination d'une figure

Etant donné un sommet $a \in X$, ou pivot, et un schéma Σ, nous associons au prédicat $P_\Sigma(a, \xi^{n-1})$ une fonction $\phi_\Sigma(a)$ à valeurs dans X^{n-1}, donnant un (n-1)uple u tel que $P_\Sigma(a,u)$

$\phi_\Sigma(a)$ est définie suivant les règles syntaxiques données en 3.1.2.3. et utilisées pour définir les prédicats :
Nous noterons Σ, Σ', Σ'' les schémas, listes de propriétés et propriétés, en leur associant : ϕ_Σ, $\phi_{\Sigma'}$, $\phi_{\Sigma''}$
Nous ferons correspondre la définition récursive suivante de $\phi_\Sigma(a)$:

Le (n-1)uple u est construit par l'opérateur non commutatif " , ", dont nous noterons l'élément neutre ϵ .

Ainsi : (x_1, x_2) , $(x_3, x_4) = (x_1, x_2, x_3, x_4)$

(x_1, x_2) , $\epsilon = \epsilon$, $(x_1, x_2) = (x_1, x_2)$ ...

Règle 1 : $\phi_\Sigma(a) = \phi_{\Sigma'}(a)$

Règle 2 : $\phi_{\Sigma'}(a) = \phi_{\Sigma''}(a)$ $\qquad \Sigma' ::= \Sigma''$

$= \phi_{\Sigma''}(a)$, $\phi_{\Sigma'}(a)$ $\qquad \Sigma' ::= \Sigma'', \Sigma'$

Règle 3 : $\phi_{\Sigma''}(a) = \epsilon$ si $\Sigma'' ::= r_1$

$\phi_{\Sigma''}(a) = h_{r_2}(a,0)$ si $\Sigma'' ::= r_2$

$\phi_{\Sigma''}(a) = h_{r_2}(a,k)$, $\phi_\Sigma(h_{r_2}(a,k))$

$k = \min_i (h_{r_2}(a,i) \mid (\exists u \quad P_\Sigma(h_{r_2}(a,i) , u))$ si $\Sigma'' :: r_2 \Sigma$

Exemple :

Reprenons l'exemple de la figure 89, de schéma : $\Sigma = (r' (r'))$ correspondant au graphe :

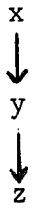


Figure 89

Si $p(r') = r^{-1}$ (relation inverse de celle de l'arborescence), nous devons avoir une fonction d'énumération définissant un ordre total sur l'ensemble : $\{ y \mid (x,y) \in r^{-1} \}$. Si nous figurons cette relation selon la figure 90 , $\phi_\Sigma (a) = (b,c)$.

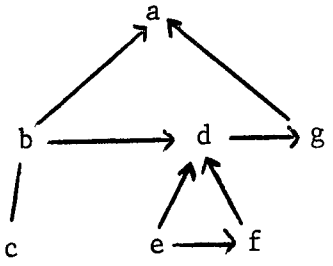


Figure 90

3.1.3.4. Choix du pivot

En utilisant la relation d'ordre totale définie sur x, il est possible d'associer au schéma de figure Σ un élément unique a, tel qu'il existe une figure et de déterminer cette figure en prenant l'application de S sur $(a, \phi_\Sigma (a))$.

Cependant, nous verrons qu'il est peu intéressant de choisir systématiquement cet élément. En effet, la grammaire contiendra plusieurs règles de transformations, donc plusieurs schémas de figure. L'on préfère alors rechercher sur chaque élément de la structure toutes les figures possibles. Cet élément servira de "pivot", et l'énumération joue alors sur ces pivots, et non sur la détermination d'une figure de schéma donné.

3.1.3.5. Algorithme de recherche

Les règles de transformations et les algorithmes correspondants ont été écrits en macroassembleur MAP.

Nous donnons ici une description de l'algorithme de recherche du schéma en ALGOL 68.

Le langage nous permet en effet de définir le schéma avec une écriture très proche de celle que nous avons décrite par les règles syntaxiques (3.1.2.3.). Un schéma est défini par un tableau à une dimension, dont les éléments sont des noms de propriété . Chaque propriété est elle-même une structure, composée d'un nom de relation et d'un schéma.(règle 3 de 3.1.2.3.). Pour alléger la description, nous avons supposé qu'il n'y avait que des relations binaires. L'adjonction de relations unaires introduit de nouvelles déclarations de structures et des conditions supplémentaires dans les procédures.

La procédure 'phiprim' a pour paramètres un schéma et un entier caractérisant le pivot, sommet à partir duquel nous recherchons une figure.

Le tableau 'index' est destiné à décrire la figure, sous la forme de noms de sommets de la structure. A l'appel de la procédure 'phiprim', index [pivot] doit contenir le nom du sommet pivot.

Après exécution de 'phiprim' (procédure booléene appelée par valeur), si 'phiprim' est vraie, le tableau index contient les noms de sommets de la figure. La borne supérieure du tableau est donnée par la valeur de l'entier 'pile'

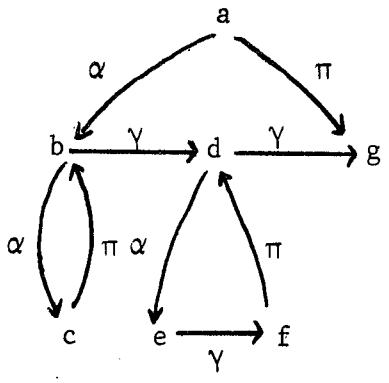
Chaque relation binaire r est caractérisée par deux fonctions f et g définissant la fonction récursive h_r.

Cette présentation est indépendante de la nature de la structure elle-même. Nous n'avons pas cherché à réaliser un programme dont la compilation et l'exécution soient efficaces, mais plutôt à décrire l'algorithme en suivant le plus possible les définitions de la fonction ϕ_Σ donnée en (3.1.3.3.).

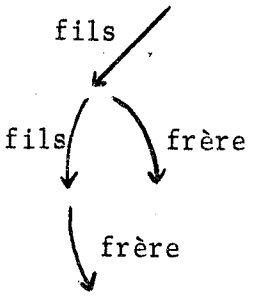
```
begin
    ¢ recherche d'une figure ¢

        mode schéma = [:] ref propriété ;
        struct propriété = (ref relation r, schéma s) ;
               relation = (ref proc f, ref proc g) ;

        [0 : 10 ] ref som index ; int pile ; ¢ 10 est une borne a priori du
                                              nombre de sommets dans un
                                              schéma ¢

        proc phiprim = (schéma sch, int pivot) bool :
             begin pile := pivot ;
                  if upb sch ≠ 0 then
                      if phisecond (sch [1]) then phiprim (sch [2 : at 1], pivot)
                                             else pile  := pivot ; false
                      fi
                                    else true
                  fi
             end
        proc phisecond = (propriété p) bool :
             begin if (index [pile + 1] := (f of r of p) (index [pivot])):≠:nil
                       then while ¬ phiprim (s of p, pile + 1)
                            do if
                     (index [pile + 1] := (g of r of p)(index [pile + 1])):≠: nil
                                then true else false
                                fi
                       else : false
                   fi
             end

    ¢ exemple de recherche   (conforme aux notations des arborescences en 3.1.4.)¢

        struct som = (ref som alpha, ref som gammapi, bool aine, bool benj) ;
        som a = (b, nil, true, true), b = (c, d, true, false) ,
            c = (nil, b, true, true), d = (e, g, false, false) ,
            e = (nil, f, true, false), f = (nil, d, false, true) ,
            g = (nil, a, false, true) ;
```

```
rel fils = (faine, cadet), frère = (cadet, cadet) ;
proc faine = (ref som x) ref som : (alpha of x) ;
proc cadet = (ref som x) ref som :
      (if  benj of x then  gammapi of x else nil fi) ;
index ⌈0⌉= a ;
  phiprim ((fils, ((fils, ((frère,))), (frère, )))), 0) ;
end
```

La structure S a la forme de la figure suivante



Le schéma paramètre de phiprim a la structure :



Le pivot est le sommet a ; après application de phiprim, phiprim = true et index = (a, d, e, f, g), pile = 4.

3.1.4. NOTATION DES ARBORESCENCES

3.1.4.1.

Il est nécessaire de définir une fonction d'énumération pour chaque relation. C'est pourquoi nous associons à la relation de base de l'arborescence une relation d'ordre partielle. Si (X,U) est une arborescence (I.3.2), nous lui associerons une seconde relation de base O telle que pour tout x, l'ensemble $\{ y \ / \ (y,x) \in U \}$ soit ordonné totalement par O . Cette relation est celle que nous avons définie sur la figure 91.

3.1.4.2. REPRESENTATION ET FONCTIONS DE BASES

Afin de définir des fonctions d'énumération, nous représenterons les arborescences biordonnées à l'aide de trois fonctions (correspondant à trois relations binaires particulières) et deux propriétés (relations unaires). Nous utiliserons les noms de relations "généalogiques" pour représenter les relations de l'arborescence.

Propriétés (relations unaires) :

Soit $x \in X$, nous aurons la propriété a(x) si $O^{-1}(x) = \emptyset$ ("ainé")

b(x) si $O(x) = \emptyset$

(benjamin)

Fonctions (relations binaires) :

$\alpha(x) = y$ si et seulement si $(y, x) \in U$ et si a(y) ("fils ainé")

$\gamma(x) = y$ si et seulement si $(x, y) \in O$ ("frère cadet")

$\pi(x) = y$ si et seulement si b(x) et $(x, y) \in U$ ("père du benjamin")

Exemple : Soit l'arborescence biordonnée

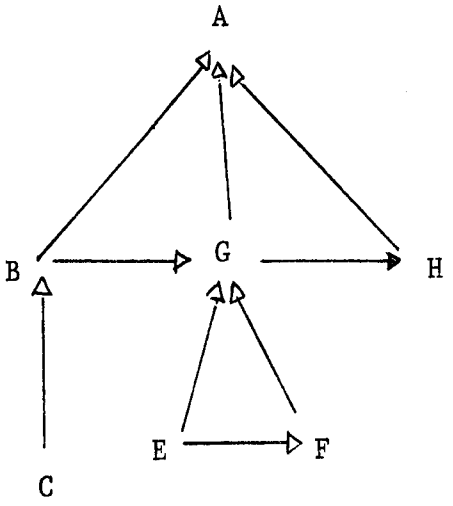


figure 91

Nous aurons la représentation suivante :

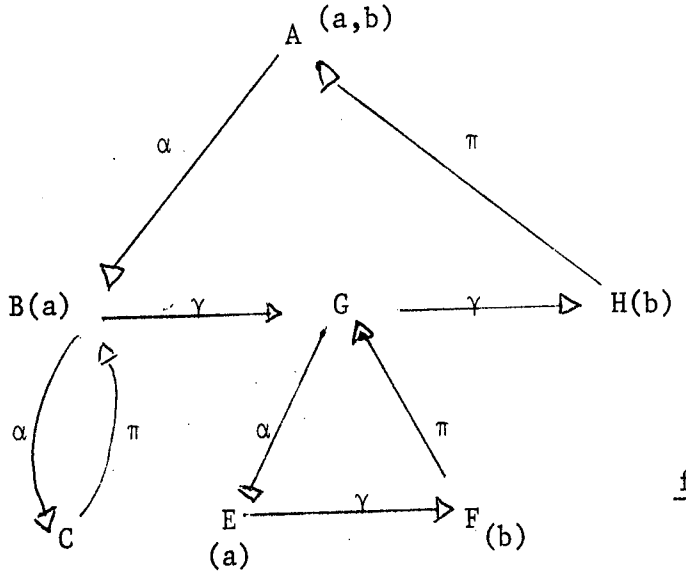


figure 92

3.1.4.3. RELATIONS ET FONCTIONS D'ENUMERATION

Une telle représentation permet d'associer un chemin élémentaire unique à tout couple de points x, y de la structure et de caractériser ce chemin par une composition de α , γ et π , ou par un prédicat. Ceci nous permet de définir des schémas de figure pour toutes les configurations.

Pour chaque relation binaire, nous définirons une fonction récursive h à l'aide des deux fonctions f et g.

Notation. Soit δ une fonction, nous noterons $\delta^{\cdot}(x)$ la fonction suivante :

$$\delta^{\cdot}(x) = \delta^{n}(x) \neq \emptyset \text{ , n étant tel que } \delta^{n+1}(x) = \emptyset$$

Nous pouvons alors définir les relations suivantes :

Relation "PERE" (identique à U) : $f = \pi_{\gamma}^{\cdot}$, $g = \emptyset$

Relation "CADET" (identique à 0) : $f = \gamma$, $g = \emptyset$

Relation "ANCETRE" (identique à $\hat{U}$) : f = g = PERE

Relation "FRERE" (identique à $\hat{0}$) : f = g = CADET

Relation "FILS" $f = \alpha$, $g = \gamma$

Nous pouvons définir une fonction d'énumération des sommets (ordre total) correspondant à l'ordre préfixé des arborescences : soit "ENUM" cette fonction.

ENUM(x) = si $\alpha(x) \neq \emptyset$, $\alpha(x)$

sinon si $\gamma(x) \neq \emptyset$, $\gamma(x)$

sinon si il existe r tel que $\gamma.\pi^{r}(x) \neq \emptyset$, $\gamma.\pi^{r}(x)$

Exemples : soit la structure de la figure T10

PERE(E) = G (car $\gamma^{\cdot}(E) = F$)

ANCETRE(E) = G , A

FRERE(B) = G , H

FILS(A) = B , G , H ...

3.1.5. APPLICATION AUX ARBORESCENCES ETIQUETEES (I.4.2.3)

En pratique, nous traiterons des structures plus complexes, utilisant d'autres relations. En particulier, les arborescences sont étiquetées, et les schémas de figure comportent des propriétés (relations unaires) relatives au "nom" des sommets (ou étiquettes).

Par convention, ces propriétés sont notées en même temps que la relation binaire. Cela n'est pas contradictoire avec la syntaxe du schéma si l'on associe au symbole "relation binaire" la production terminale suivante :

<relation binaire > ::= <nom de relation > , < étiquette >

Chaque règle de transformation comporte une nouvelle affectation de symbole, ce qui permet d'assurer que la même figure ne sera pas sélectionnée plusieurs fois.

Exemple pratique de schéma de figure :

Soit V = {a, b, c}

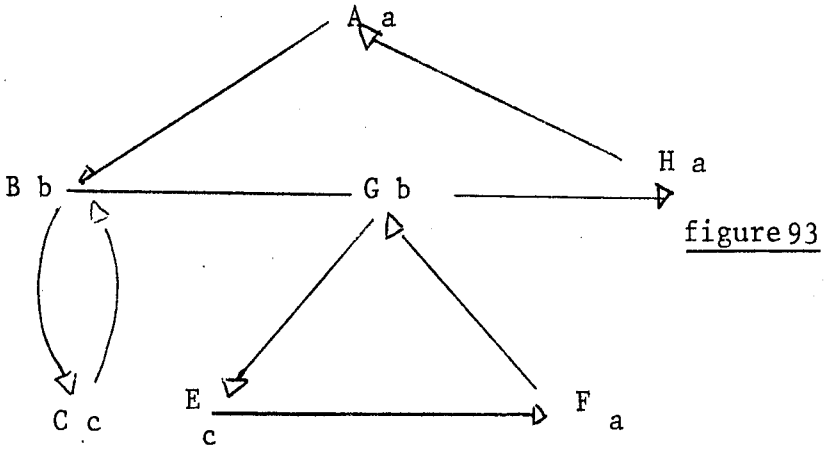


figure 93

Σ_1 = (FILS, b (FILS, c, CADET, a (ANCETRE, a)), FILS, a)

Le schéma Σ_1 appliqué au pivot A donne la figure :

(A, G, E, F, A, H)

Σ_2 = (FILS, b (FILS, b))

Le schéma Σ_2 appliqué au pivot A donne la figure :

(A, B, C)

3.1.6. CYCLES

Le schéma Σ_1 de l'exemple précédent fait apparaître un "cycle" dans la figure associée ; le sommet A apparaît plusieurs fois dans la figure. Avec ce même schéma, une autre structure aurait pu ne pas comporter un tel cycle.

Afin de contrôler l'existence des cycles nous avons défini une propriété (relation unaire) vérifiant l'identité (ou la non identité) du sommet sur lequel porte la propriété , et d'un sommet de rang inférieur dans le schéma. Ainsi, dans le schéma Σ_1 , nous aurions pu noter :

Σ_1 = (FILS, b (FILS, c, CADET, a (ANCETRE, a (DIF (0)))), FILS,a)

La propriété DIF(0) signifiant que le sommet (ici de rang 4) doit être différent du sommet de rang 0. Ici, aucune figure de schéma Σ_1 n'aurait alors été obtenue.

Cette propriété est en fait une relation binaire, mais elle n'est pas traitée comme telle dans l'évaluation de ϕ_Σ , car elle ne comporte pas la recherche d'un nouvel élément.

3.2.1. INDEX STRUCTURAL

Une figure est caractérisée par un n-uple ou index structural. Les transformations porteront sur les sommets nommés dans la figure. Chaque sommet est caractérisé par son rang dans la figure, numéroté à partir de zéro (le sommet de rang zéro est le pivot).

Certaines transformations comportent l'insertion de nouveaux sommets dans l'arborescence. Dans ce cas, nous considérons que la figure comporte, outre les n sommets de l'index structural, un nombre non borné de sommets virtuels, caractérisés par des rangs supérieurs à n.

3.2.2. DETACHEMENT

Etant donné un sommet $x \in X$ de l'arborescence $\mathcal{A}$, il lui correspond une sous-arborescence complète $\mathcal{A}_x$ de racine x. L'opération de détachement transforme l'arborescence $\mathcal{A}$ en deux arborescences, $\{ \mathcal{A} - \mathcal{A}_x , \mathcal{A}_x \}$, obtenues en supprimant la relation $(x,y) \in U$, la structure obtenue étant alors $(X, U - (x,y))$.

Pendant l'application des transformations, nous considérerons que les opérations portent sur un ensemble d'arborescences que l'on peut obtenir par des opérations de détachement. Tout sommet virtuel (3.2.1) constitue une arborescence réduite à sa racine.

Nous appellerons arborescence principale l'arborescence qui a même racine que l'arborescence initiale. Nous noterons ω sa racine.

3.2.3. INSERTION

Etant données deux arborescences $\mathcal{A} = (X, U, O)$ et $\mathcal{A}' = (X', U', O')$ de racines x et x', l'opération d'insertion de l'arborescence $\mathcal{A}'$ dans l'arborescence $\mathcal{A}$ associe aux deux arborescences une arborescence unique $\mathcal{A}''(X \cup X', U'')$ de racine x, préservant les relations des deux arborescences (en tenant compte de la transitivité de la relation d'ordre O).

Cette contrainte implique que la racine x' soit insérée en position terminale (feuille de l'arborescence). Dans l'arborescence $\mathcal{A}''$, $\mathcal{A}'$ constitue une

sous arborescence propre. L'insertion se présente donc comme une opération inverse du détachement.

L'opération d'insertion doit spécifier la position de la sous arborescence $\mathcal{A}''$ par la donnée d'un nom de fonction, et d'un nom de sommet de l'arborescence $\mathcal{A}'$. L'on utilisera les fonctions simples (α: "fils ainé", γ : "frère cadet"), ou des fonctions plus élaborées. La fonction π (père du benjamin) ne peut être utilisée car elle ne correspond pas à une position de sommet terminal.

3.2.4. TRANSFORMATION PAR INSERTION

Etant donné un index structural décrivant une figure, une transformation par insertion comporte un détachement suivi d'une insertion.

Par convention, une telle transformation est notée seulement par le nom de fonction, indiquant une nouvelle affectation.

Exemple

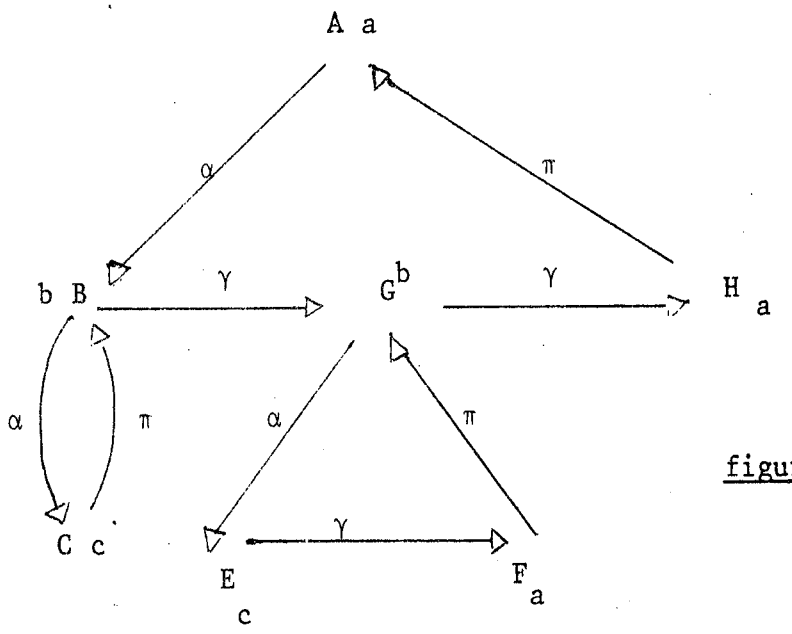


figure 94

Et l'index correspondant à la figure du schéma Σ

0	A
1	G
2	E
3	F
4	A
5	H

La transformation par insertion 1 = FAINE (5)

comporte : détachement de 1 : γ (B) = H

(G) γ (G) = $\emptyset$

insertion de 1 'fils ainé' de 5 :

γ (G) = α(H)

α (H) = G

a (G) ; si γ (G) = $\emptyset$, π(G) = H, b (G)

d'où la structure :

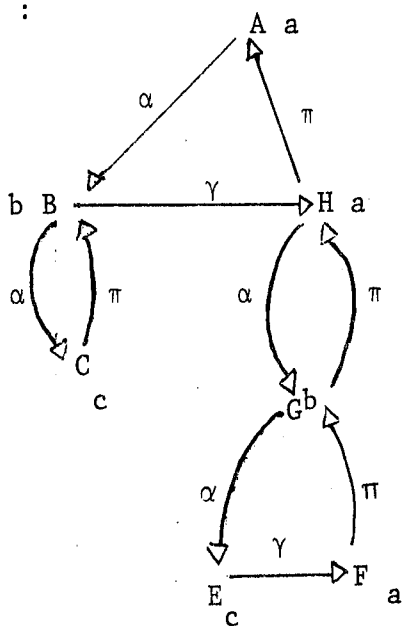


figure 95

Soit maintenant la transformation : 6 = CADET (1)

Le rang 6 étant supérieur à l'index structural, le sommet 6 est virtuel. Il y a en fait création d'un nouveau sommet I, qui est donc déjà détaché.

- création du sommet I (mise à jour de l'index)
- insertion de I : $\gamma(I) = \gamma(G)$

$$\gamma(G) = I$$

si $\gamma(I) = \emptyset$, b (I), $\pi(I) = \pi(G)$

$$\pi(G) = \emptyset$$

D'où la structure et le nouvel index.

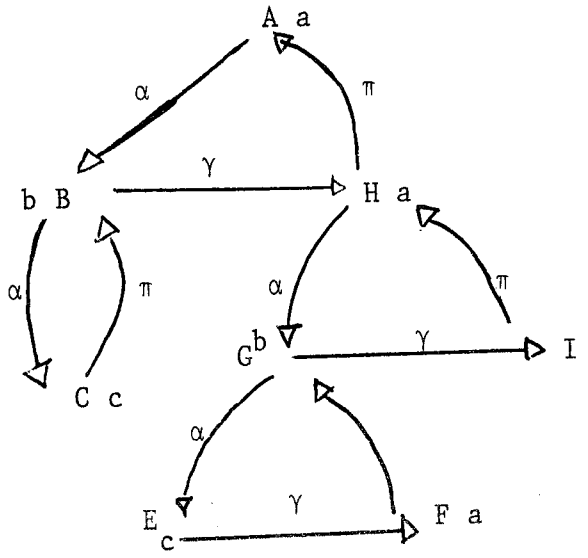


figure 96

0 A
1 G
2 E
3 F
4 A
5 H
6 I

3.2.5. TRANSFORMATIONS SANS DETACHEMENTS

Deux transformations ne font pas intervenir de détachement, mais modifient l'arborescence elle-même.

3.2.5.1. Echange : L'échange consiste à échanger deux sous arborescences complètes. Les deux sous arborescences sont spécifiées par leurs racines x_1 et x_2. L'effet de la transformation échange sera le suivant :

Soient α, γ, π les fonctions sur l'arborescence $\mathcal{A}$ initiale, α', γ', π' les fonctions sur l'arborescence $\mathcal{A}'$ transformée :

$$\text{si } \alpha(x) \neq x_1 , x_2, \quad \alpha'(x) = \alpha(x)$$

$$\text{si } \alpha(x) = x_1 , \alpha'(x) = x_2$$

$$\text{si } \alpha(x) = x_2 , \alpha'(x) = x_1$$

$$\text{si } \gamma(x) \neq x_1, x_2 \quad \gamma'(x) = \gamma(x)$$

$$\text{si } \gamma(x) = x_1 , \gamma'(x) = x_2$$

$$\text{si } \gamma(x) = x_2 , \gamma'(x) = x_1$$

$$\text{si } x \neq x_1 , x_2 , \pi'(x) = \pi(x)$$

$$\gamma'(x) = \gamma(x)$$

$$\gamma'(x_1) = \gamma(x_2) \qquad \pi'(x_1) = \pi(x_2)$$

$$\gamma'(x_2) = \gamma(x_1) \qquad \pi'(x_2) = \pi(x_1)$$

L'échange n'est possible que si le résultat de l'opération est encore une arborescence.

De ce fait, les deux opérandes x_1 et x_2 ne doivent pas être comparables par U.

3.2.5.2. Absorption : L'absorption est une transformation qui ne fait intervenir qu'un seul paramètre. Si x_1 est ce paramètre, α, γ, π les trois fonctions de $\mathcal{A}$ α', γ', π' les fonctions de la transformée $\mathcal{A}'$, il vient :

$$\alpha'(x_1) = \gamma'(x_1) = \pi(x_1) = \emptyset$$

si $\alpha(x) = x_1$, $\alpha(x_1) \neq \emptyset$, $\alpha'(x) = \alpha(x_1)$

$\alpha(x_1) = \emptyset$, $\alpha'(x) = \gamma(x_1)$

si $\alpha(x) \neq x_1$, $\alpha'(x) = \alpha(x)$

si $\gamma(x) = x_1$, $\alpha(x_1) \neq \emptyset$, $\gamma'(x) = \alpha(x_1)$

$\alpha(x_1) = \emptyset$, $\gamma'(x) = \gamma(x_1)$

si $\gamma(x) = \emptyset$, $\pi'(x) = \pi(x_1)$

si $\pi(x) = x_1$, $\gamma(x) = \gamma(x_1)$

$\pi'(x) = \pi(x_1)$

Exemple : En reprenant le résultat de la figure 96, soit la transformation absorption (1)

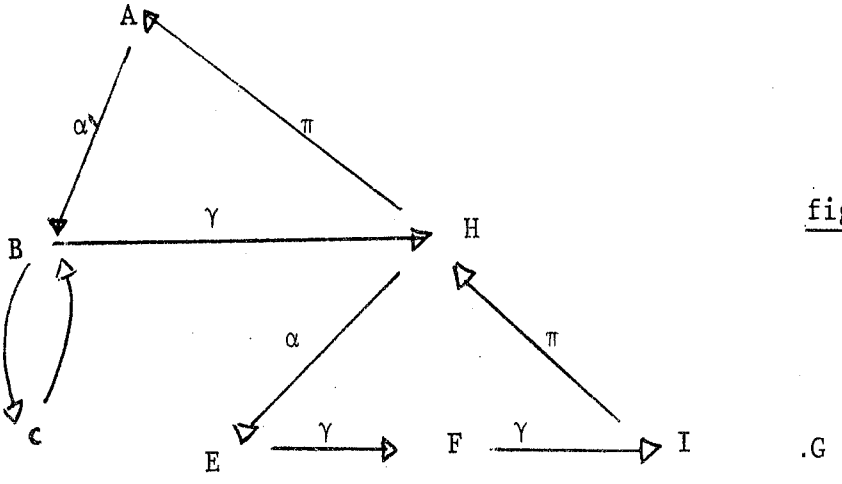


figure 97

G (référence 1) est détaché, ses "fils" prennent sa place.

3.2.6. REGLE DE TRANSFORMATION

Une règle de transformation comporte : un schéma de figure, une liste de transformations élémentaires, une liste d'affectations de symboles.

Le schéma de figure constitue la partie "reconnaissance" de la règle ; si il existe une figure de schéma donné, l'index structural est constitué et les transformations et affectations sont effectuées séquentiellement.

A la fin de l'application d'une règle, plusieurs arborescences peuvent exister. Par convention, seule l'arborescence principale (3.2.2.) est conservée.

L'exemple précédent peut se résumer par la règle de transformation suivante :

Schéma : (FILS, b (FILS, c, CADET, a (ANCETRE, a)), FILS, a)

Transformations :

(1, FAINE, 5)

(6, CADET, 1)

(, ABS, 1)

Chaque transformation est notée avec deux opérandes et un nom de transformation. Nous ne traiterons l'affectation des symboles qu'en fonction de l'application dans le système de traduction. Appliquée au pivot A sur la structure de la figure 94 , cette règle de transformation fournit la structure de la figure 97 .

3.3.1. PRINCIPE GENERAL

Dans le programme de traduction automatique, les grammaires de transformation interviennent dans deux phrases différentes : la construction de la représentation en langage pivot et la production de la structure syntaxique cible. Ces deux phrases sont symétriques, mais, tenant compte des particularités du problème, elles ne sont pas réversibles.

A ce niveau, nous n'avons pas pu définir une grammaire indépendante des algorithmes de traitement, car nous ne connaissons pas de modèle de description des arborescences permettant de définir des algorithmes d'analyse globaux analogues à ceux que l'on utilise en analyse syntaxique. De ce fait, la grammaire se présente sous la forme d'une suite séquentielle de règles, dont l'ordre d'application est défini.

De plus, certaines difficultés ont été rencontrées dans le choix des pivots. En effet, certaines règles, appliquées à un pivot x donné, peuvent conduire à détacher (3.2.4.) celui-ci. Il est impossible dans ce cas de poursuivre les transformations à moins de repartir au début de l'arborescence principale (3.2.2.).

C'est pourquoi, le choix du pivot suivant doit pouvoir être indiqué dans une règle de transformation.

Le balayage de la structure est contrôlé par la grammaire à l'aide de ce choix de pivot. Ainsi, cette grammaire est un véritable programme, dont l'exécution est initialisée par la donnée d'une structure au moyen de sa racine. Après exécution de ce programme, le contrôle est rendu au système qui transmet la structure transformée et initialise un nouveau traitement.

3.3.2. ECRITURE DE LA GRAMMAIRE

Le langage d'écriture de la grammaire a été défini pour le système 7044 IBM (Annexe II.2) en utilisant les possibilités du macro-assembleur. Une nouvelle

version, destinée à être exploitée en 360, est en cours d'élaboration [St], les spécifications en seront différentes. Nous nous limiterons ici à donner une définition générale du langage, afin de montrer le fonctionnement de la grammaire, indépendamment d'une réalisation donnée. On trouvera en annexe II la description du langage que nous avons réalisé sur 7044 IBM.

Ce langage pourrait être un sous ensemble d'un langage de programmation, dans la mesure ou les schémas de figure pourraient y être représentés (ALGOL 68).

3.3.2.1. Syntaxe

Nous ne donnons ici qu'une structure simplifiée du langage, et non une définition formelle complète.

Nous utilisons pour cela la forme normale de Backus.

< grammaire de transformation > ::= < règle de transformation > | <règle de transformation > < grammaire de transformation >

< règle de transformation > ::= < début de règle > < règle > < fin de règle >

< début de règle > ::= < symbole règle > < nom de règle > < alternant >

< alternant > ::= < nom de règle > | < symbole suite >

< règle > ::= < question > | < application > | < question > < application >

< question > ::= < symbole gauche > < schéma de figure >

< application > ::= < transfert > | < modifications > | < modifications > < transfert >

< modifications > ::= < transformations de structure > | < affectations de symboles > | < transformations de structure > < affectations de symboles >

< transformations de structures > ::= < symbole STRUCT > < liste de transformations >

< liste de transformations > ::= < transformation élémentaire > | < transformation élémentaire > < liste de transformation >

< affectations de symboles > ::= < affectation élémentaire > | < affectation élémentaire > < affectations de symbole >

< affectation élémentaire > ::= < symbole symbo > < entier > < nouveau symbole >

< transfert > ::= < changer de pivot> | < aller à > < changer de pivot > | < aller à >

< changer de pivot > ::= < symbole pivot > < entier >

< aller à > ::= < symbole aller à > < nom de règle >

< fin de règle > ::= < symbole fin de règle > < nom de règle >

Dans cette grammaire, nous n'avons pas donné les règles terminales, qui dépendent d'une réalisation donnée, ni l'écriture du schéma [3.1.3.5.] et des transformations élémentaires [3.2.3.]. Les composants d'une règle sont facultatifs. Schématiquement, une règle de transformation comporte les parties suivantes :

1) Début de règle : REGLE n_1, $\begin{cases} n_2 \\ \text{SUITE} \end{cases}$ obligatoire

2) Question GAUCHE 'sch' facultatif

3) Transformation de structure STRUCT $(..(e_1, r, e_2)..)$

4) Affectations de symbole SYMBØ e_3, S

5) Changer de pivot PIVOT e_4

6) Aller à (A, n_3)

7) Fin de règle FREGLE n_1 obligatoire

n_1, n_2, n_3 sont des noms de règles. Par définition, n_1 est le nom de la règle encadrée par REGLE n_1, x et FREGLE n_1 ; e_1, e_2, e_3, e_4 sont des entiers ; r est un nom de relation et S un symbole, sch est un schéma.

3.3.2.2. Sémantique

1) Lorsque la grammaire est appelée, un vecteur index contient en index [0] l'adresse de la racine de la structure (3.3.3.5.) . Le contrôle est donné à la première règle de la grammaire (début de règle).

2) Début de règle : si il existe une figure de schéma sch avec le pivot index [0], le contrôle est donné à 3 sinon, si l'alternant est un nom de règle n_2 , le contrôle est donné à n_2, sinon, l'alternant est suite et le contrôle est donné à la première règle suivant FREGLE n_1. Si il n'y a pas de question 2, le contrôle est donné à 3.

3) Si il n'y a pas de transformation, le contrôle est donné en 4, sinon les transformations (e_1, r, e_2) sont effectuées : (3.1.2.3.) les sommets index [e_2] sont mis en relation r avec les sommets index [e_1] . Si e_2 est supérieur au rang du dernier sommet de la figure, un nouveau sommet est créé. Le contrôle est donné à 4.

4) Si il n'y a pas d'affectation, le contrôle est donné à 5, sinon, les symboles S sont affectés aux sommets Index [e_3], puis le contrôle est donné à 5.

5) Si il y a changer de pivot, l'adresse donnée en index [e_4] est placée en index [0], et le contrôle est donné à 6, sinon le contrôle est donné à 6.

6) Si il y a aller à, le contrôle est donné à la règle de nom n_3, sinon, il est donné à la règle suivant FREGLE n_1.

3.3.3. EXEMPLES

La suite de règles donnée en exemple est utilisée pour définir le balayage de la structure. La convention d'écriture est celle que nous avons prise dans la réalisation en macroassembleur MAP.

```
REGLE    R1, SUITE
GAUCHE   ((FS, 0))        si le pivot à un fils, celui-ci est pris
CONDIT                    comme nouveau pivot, la grammaire est
PIVOT    1(A, GRAM)       appelée
FREGLE   R1
REGLE    R2, SUITE
GAUCHE   ((FR, 0))        si le pivot n'a pas de fils, son frère
CONDIT                    est pris comme pivot
PIVOT    1(A, GRAM)
FREGLE   R2
REGLE    R3, STOP         sinon, si le pivot a un père, on prendra
GAUCHE   ((PERE, 0))      le frère de celui-ci comme pivot ; sinon,
CONDIT                    le balayage est terminé.
PIVOT    1(A, R2)
FREGLE   R3
```

L'instruction CONDIT est un simple séparateur, nécessité par l'écriture des macro-instructions. Le symbole aller à n'est remplacé par (A, n).

La règle R_1 est appelée après chaque exécution de la grammaire sur un pivot donné. Le balayage obtenu est le balayage habituel (de haut en bas et de gauche à droite). La grammaire, dont la première règle est GRAM, est appelée pour chaque choix de pivot. Le traitement s'arrête au cas où la règle R_3 ne peut pas s'appliquer (STOP).

3.3.4. EXTENSION DU LANGAGE

En pratique, nous rencontrons des symboles complexes sur chaque sommet (étiquettes, variables, etc...). Pour alléger l'écriture, plusieurs règles élémentaires sont regroupées. Pour un schéma de figure simple commun, plusieurs modifications peuvent alors être appelées. Le choix est réalisé à l'aide de conditions booléennes sur les propriétés de chaque sommet (annexe II.2).

3.4. APPLICATIONS

3.4.1. LE MODELE D'ETIQUETAGE

3.4.1.1. Nature des structures

Conformément à la définition du langage pivot (1.3.3., figure 33), la structure comporte :

1) Une arborescence étiquetée (X, U)
2) Deux relations d'équivalence sur X (élision E et substitution).
3) Un ensemble totalement ordonné (occurrences), muni d'informations grammaticales et de références au lexique.
4) Une application de chaque classe de sommets de X sur l'ensemble des occurrences.

La structure interprétée, obtenue à l'issue de l'analyse syntaxique, est de ce type. Cependant, aucune relation d'équivalence (élision, substitution) n'y existe. Les étiquettes affectées aux sommets de X sont des noms de règles syntaxiques comme dans l'exemple traité en annexe I. L'ensemble des occurrences est l'image de la chaine de mots donnée en langue source.

Le rôle de la grammaire d'étiquetage est de transformer cette structure en structure du langage pivot.

3.4.1.2. Nature des relations

A chaque sommet de X correspond une occurrence unique, donc un rang dans l'ordre séquentiel. Cette occurrence est celle correspondant à sa classe d'équivalence.

Nous avons noté les relations d'équivalence E et S sous la forme de relations d'ordre. En effet, ceci permet d'affecter une occurrence à un seul des sommets, et de la retrouver facilement à partir d'un sommet quelconque -comme dans la figure 33- .

Cette restriction est sans importance, car la notion d'équivalence est ici purement interprétative. Réciproquement, chaque occurrence ne correspond qu'à un seul sommet de X, celui où elle apparaissait effectivement dans la réalisation, donc dans la structure d'entrée.

Nous pouvons ainsi utiliser les relations de l'arborescence, notées conformément aux conventions données en 3.1.4., la relation d'ordre total sur les occurrences, et toutes les relations complexes que l'on peut en déduire.

Les règles d'étiquetages peuvent ainsi utiliser des schémas de figure concernant la structure syntaxique (arborescence) et l'ordre séquentiel des mots.

3.4.1.3. Transformations

Les transformations ne jouent que sur la structure arborescente, l'ordre séquentiel étant une donnée de la langue que l'on ne peut modifier.

3.4.1.4. Symboles

Les symboles comportent, pour chaque sommet de X, son étiquette, et les informations de l'occurrence correspondante. Ces informations contiennent, en particulier, les données du lexique définies par une référence. Chaque unité du lexique peut correspondre à plusieurs homographes (polysémie).

Seuls les homographes dont le code est conforme aux questions rencontrées au cours de la grammaire sont retenus (1.4.1.3.).

Les affectations de symboles concernent essentiellement les symboles du langage pivot : étiquettes et variables persistantes (1.4.1.3.).

3.4.2. LE MODELE DE PRODUCTION SYNTAXIQUE

Le modèle de production syntaxique doit associer à toute structure du langage pivot une structure syntaxique en langue cible.

Pratiquement, nous devons dans cette phase déterminer les relations syntaxiques, choisir les catégories syntaxiques pour chaque unité du lexique à l'intérieur d'un paradigme donné (figure 31) et définir l'ordre total caractérisant la phrase.

Partant d'une structure pivot, il est relativement aisé d'en déduire une structure de dépendances par transformations successives, et de déterminer les relations syntaxiques et les catégories. La définition de l'ordre total peut être plus complexe.

En effet, nous pouvons construire un ensemble d'occurrences totalement ordonné à partir de l'ensemble fourni par le langage pivot. Ceci nous oblige à opérer sur une chaine, c'est-à-dire à nous heurter aux phénomènes combinatoires de l'analyse. Il est préférable de rester au niveau de la structure syntaxique de dépendance, en définissant chaque niveau de l'arbre à la manière d'une règle de dépendance (2.2.3).

La chaine finale peut alors être déduite par application de la règle de projectivité (2.1.1.4.).

Cependant, un tel procédé ne nous évite pas de manipuler là encore une chaine. Ainsi la production de phrases telles que :

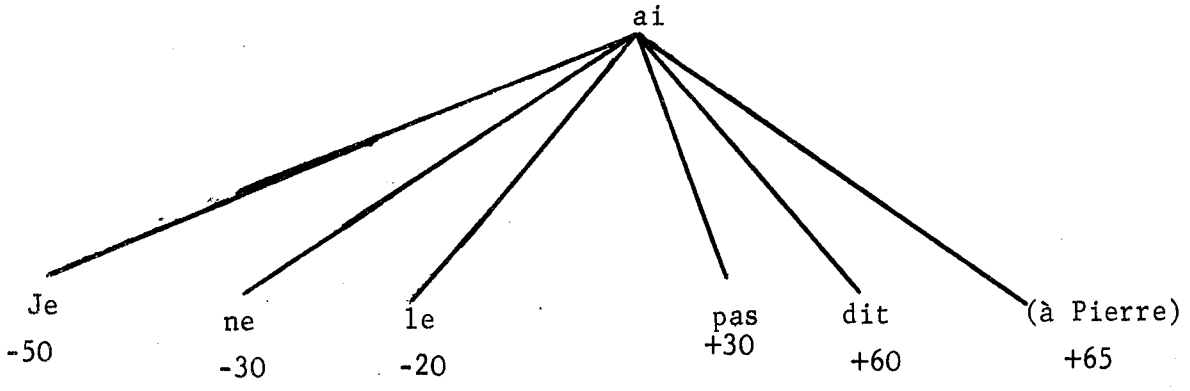
"Je l'ai dit à Pierre"
"Je ne l'ai pas dit à Pierre"
"Je ne le lui ai pas dit" ...

qui correspondent à des structures semblables en langage pivot, devront faire appel

à des règles de transformations différentes afin de tenir compte des différents ordres possibles, bien qu'elles ne diffèrent que par la nature d'un actant ("Pierre" et "lui") ou par la présence de la négation.

Afin d'éviter cette prolifération de règles, ou d'éventuelles transformations sur la chaine, nous construisons une structure de dépendance en affectant à chaque sommet un "poids" caractéristique de sa position. Chaque poids est en fait caractéristique d'une fonction syntaxique, et sert de nom de fonction.

Exemple : Reprenant la phrase :"Je ne l'ai pas dit à Pierre" la structure de dépendance sera :



Chaque sommet est affecté d'un poids ; les poids négatifs correspondent aux mots placés à gauche du gouverneur, et les poids positifs à droite. Le gouverneur a un poids zéro par rapport à ses dépendants, et la détermination de la chaine finale utilise l'ordre ainsi défini à chaque niveau.

Cette solution permet de traiter indépendamment les différentes relations syntaxiques à un même niveau.

Certaines transformations se ramènent à changement de poids : le remplacement du troisième actant nominal ("à Pierre") par un pronom (lui) consiste à remplacer le poids +65(caractéristique de l'objet indirect) par le poids -10.

CONCLUSION

Nous n'avons pu donner ici qu'un aspect de l'ensemble des travaux qui ont permis la réalisation du programme de traduction automatique. Cependant, nous pouvons en dégager quelques principes.

Du point de vue mathématique, tout en nous efforçant de conserver une certaine orthodoxie dans les modèles que nous avons définis, il a fallu, dans bien des cas, utiliser des méthodes adaptées aux circonstances. En ce qui concerne l'analyse syntaxique, nous avons ressenti la nécessité de mettre l'accent sur la notion de structure plus que sur la notion de grammaires génératives, ces dernières n'étant considérées que comme un outil. De plus, l'emploi nécessaire de ces grammaires ne nous a pas semblé être des plus efficaces au niveau linguistique ; car la mise au point sur un corpus est longue et coûteuse. Il pourrait être plus intéressant de développer des méthodes de description fondées sur les structures, en leur associant, si cela est nécessaire, un programme de construction des grammaires permettant de réaliser un algorithme utilisant un modèle génératif. Ce point de vue est corroboré par l'inefficacité des modèles génératifs pour décrire certains cas, finis, de structures non projectives ou non continues. Ceci nous a amené à définir les variables véhiculaires, extension des grammaires hors contexte. Plusieurs auteurs ont déjà étudié de telles extensions, et un travail de recherche permettrait peut être de définir des classes de langages sous contexte analysables d'une façon efficace, en déterminant les limites exactes de l'utilisation de contraintes sur les dérivations.

Les modèles de transformation que nous avons définis sont en fait réalisés comme des programmes. Cette solution n'est pas très satisfaisante; nous y avons été contraints par la complexité des schémas de figure proposés par le linguiste, et par l'aspect non-systématique de l'ordre dans lequel doivent s'effectuer les transformations.

Plusieurs travaux sont actuellement consacrés à ce type de problème. A notre connaissance, les seuls cas pour lesquels on pourrait développer une méthode conservant l'aspect formel, et descriptif, indépendamment de l'algorithme, sont les cas où l'on manipule des arborescences, et où les schémas de figure sont relativement

simplifiés. Les transformations que nous avons mises en oeuvre pour fournir une structure de dépendance en sont un exemple.

Les méthodes de comparaison des structures entre elles, en faisant éventuellement appel à la notion de schéma de figure, pourraient, elles aussi, faire l'objet d'une recherche plus approfondie. L'emploi de méthodes statistiques permettrait de définir les caractéristiques de certaines figures privilégiées dans les structures syntaxiques, caractéristiques d'une langue ou d'un style.

Du point de vue linguistique, nous pourrions faire des constatations analogues. La réalisation d'un système complet, analysant des textes non choisis, conduit à prendre une position qui n'est pas toujours orthodoxe. Ainsi la grammaire hors contexte a pour rôle d'analyser toutes les phrases, et non de décrire la langue ; de même le langage pivot est une notation intermédiaire entre les langues, dans lequel le contenu n'est analysé que si la traduction l'impose.

La réalisation sur ordinateur a été faite en tenant compte des deux impératifs suivants : d'une part obtenir un temps de traitement raisonnable, d'autre part faciliter l'accès de la machine pour les linguistes. Ce choix était dicté par la nécessité d'opérer sur des textes importants : les modèles réduits ne sont pas significatifs dans l'étude d'une langue. De ce fait, les programmes ont été écrits dans le langage permettant la plus grande vitesse d'exécution et le plus faible volume (l'assembleur) ; la stratégie a été dictée par la taille des fichiers à traiter et par le mode d'utilisation de la machine (Batch processing). Cependant, cette stratégie reste la seule valable dans le cadre du traitement de textes importants, utilisant le dictionnaire complet.

Nous avons pu définir un métalangage efficace et facilement modifiable en utilisant le macroassembleur pour les règles de transformation. Malheureusement, cet outil est toujours très coûteux, les temps d'assemblage étant toujours plus élevés que les temps de compilation, dans les systèmes que nous utilisons. Bien que le programme ait une efficacité satisfaisante, les choix que nous avions faits se sont révélés très coûteux dans la partie la plus importante de son utilisation (jusqu'à maintenant) qui est la mise au point linguistique : mise à jour des dictionnaires et

modification des grammaires.

Nous avons utilisé ALGOL 68 pour décrire les algorithmes que nous avons définis. Ce langage nous permet en effet de manipuler tous les types d'information que nous utilisons, même au niveau linguistique. Il est douteux que l'on puisse utiliser effectivement ce langage pour réaliser complètement un programme exploitable en traduction automatique. Cependant, la définition d'un sous-ensemble adapté à ce problème, utilisant des sous programmes spéciaux, permettrait peut être d'obtenir une vitesse, et un volume raisonnables.

Les limites de ce système sont liées aux classes de modèles que nous avons choisis. Les améliorations devront faire intervenir des modèles sémantiques plus élaborés, permettant par exemple d'analyser la détermination (choix des articles). Les divers algorithmes que nous avons développés peuvent être utilisés dans d'autres domaines, car ils permettent de réaliser des analyses syntaxiques sur des corpus importants, ce qui pourrait fournir une donnée intéressant la mécanolinguistique. Le codage en langage pivot peut être utilisé dans les travaux de documentation automatique ou dans la définition des langages d'accès aux ordinateurs.

BIBLIOGRAPHIE

Ab - ABRAHAM
'Some questions of phrase structure grammars'
Computational Linguistics
Computing Center of the Hungarian Academy of Sciences
Budapest - 65

AH - AHO
'Indexed grammars' -'an extension of Context Free grammars'
J. Acm 15 (Octobre 68) pages 647-671

An - ANDREEV
'Intermediary language as the focal point of machine translation'
Machine translation - Booth editor - North Holland

Be - BERGE
'Théorie des graphes et ses applications'
Dunod 63

CH_1 - CHOMSKY
'Syntactic Structures' Mouton

CH_2 - CHOMSKY
'On Certain formal properties of grammars'
Information and Control Vol. 2 1959

CH_3 - CHOMSKY
'Aspects of theory of syntax' - The M.I.T. Press - 65

CH_4 - CHOMSKY
'Formal properties of grammars'
Handbook of mathematical Psychology
Vol. 2 Wiley 1963

Cn - COHEN
'Langages pour l'écriture des compilateurs
thèse U - GRENOBLE Juin 1967

Co - COURTIN - RIEU - SGALL
'Métalangage pour l'analyse morphologique'
Document C.E.T.A. G 2500 A 1969

Da - DAVIS — 'Computability and unsolvability'
Mc Graw - Hill- 58

Do - DOSTERT — 'Georgetown University, Machine translation project'
General report - Juin 1963

Ga_1 - GARVIN — 'Computer aided research in machine translation'
Final report
Bunker ramo corporation

Ga_2 - GARVIN — 'The fulcrum syntactic analyser for Russian'
Conférence internationale sur le traitement automatique des langues - GRENOBLE 1967

Gf - GAIFMAN — 'Dependency system and phrase structure system'
Information and control Vol. 8 n° 3 - Juin 1965

Gi_1 - GINSBURG — 'The mathematical theory of contex free language'
Mc Graw Hill

Gi_2 - GINSBURG - SPANIER — 'Control set grammars'
Mathematical system theory - Vol. 2 n° 2

Gd - GRANDJEAN — 'Compilateur syntaxique' - C.E.T.A. GTC 17
'Programme d'analyse syntaxique'
C.E.T.A. GTC 18

Gr - GREIBACH & HOPCROFT — 'Scattered context grammars'
IFIP Congress 68 - Edinburgh - August 1968

Gp - GRIFFITHS - PETRICK — 'On the relative efficiencies of context-free grammar recognizers - Comm. - ACM - Mai 1965

Ha - HARRIS — 'Structural linguistics'
The University chicago press

Hy_1 - HAYS — 'Dependency theory' : a formalism and some observations
Language 40 (4) pages 511-525

Hy_2 - HAYS
'Research procedure in machine translation'
Natural langage and the computer
Garvin édit - Mc Graw Hill

Ku - KUNO
'The predictive analyser in readings in automatic'
'Language processing'
ELSEVIER (Hays Editor)

La - LAMB
'Stratificational linguistics as a basis for méchanical translation'
Mechanolinguistics project. University of California Berkeley - mars 1964

Le - LECERF et IHM
'Eléments pour une grammaire générale des langues projectives' Rap CETIS n° I - Evratom

Ma - MARCUS
'Algebraïc linguistics" Analytical models
Academic press

Md - Mc DANIELL
'An evaluation of usefulness of machine translation produced at Teddington, and an account of translation method'Conf. Int. sur le traitement automatique des langues - Grenoble 1967

Me - MELTCHOUK - JOLKOVSKY
'Essai d'une théorie sémantique applicable au traitement des langues'
Conférence Internationale sur le Traitement Automatique des Langues - GRENOBLE 1967

Pa_1 - PAIR
'Etude de la notion de pile, application à l'analyse syntaxique'
Thèse - Faculté des Sciences de l'Université de Nancy 1965

Pa_2 - PAIR - QUERE
'Définition et étude des bilangages réguliers
Information and Control 13-68 - pp. 565-593

Ra - ROUAULT
Quelques applications de la logique à la sémantique des langues naturelles
Int. Conf. Computational Linguistics
Stockholm 1969

Ro - ROBINSON
'Endocentric constructions and the Cocke parsing logic'
The Rand corporation - Mars 1965 - page 3 101

Sa - SAKAI
'Syntax in Universal Translation
Proc. 1961 - Int. Conf. on Machine Translation
Her Majesty's Stationary office
London 1962 - Pages 593 - 608

Te - TESNIERES
'Eléments de syntaxe structurale'
Klincksieck 1959

Va_1 - VAUQUOIS
'Langages artificiels, systèmes formels et traduction automatique'
Ecole d'été - Venise - Juillet 1962

Va_2 - VAUQUOIS - VEILLON VEYRUNES
'Application des grammaires formelles aux modèles linguistiques en traduction automatique'
Colloque de Prague - Septembre 1964

Va_3 - VAUQUOIS
'Le système de traduction automatique du C.E.T.A.'
Com. Conf. sur la traduction automatique Erevan
U.R.S.S. Avril 1967

Va_4 - VAUQUOIS - VEILLON VEYRUNES
'Syntax and Interpretation'
Int. Conf. Comp. linguistics - New York 1965

Va_5 - BOURGUIGNON - NEDOBEJKINE VAUQUOIS - VEILLON
'Une notation des textes hors des contraintes morphologiques et syntaxiques de l'expression'
Int. Conf. Computational linguistics
Stockholm 1969

V_1 - VEILLON 'Le langage pivot du C.E.T.A.
TA informations - n° 1

V_2 - VEILLON 'Consultation du dictionnaire et analyse morphologique en traduction automatique'
Thèse de 3e Cycle - Université de Grenoble 1962

V_3 - VEILLON 'Présentation de l'analyse morphologique et du programme de dictionnaire allemand'
document C.E.T.A. G 500 A

V_4 - VEILLON - VEYRUNES 'Etude de la réalisation pratique d'une grammaire Context Free et de l'algorithme associé'
La traduction automatique - cinquième année n° 3

V_5 - VEILLON - VAUQUOIS VEYRUNES 'Un métalangage de grammaires transformationnelles'
Conf. Int. traitement automatique des langues
Grenoble 1967

VW - VAN WIJGAARDEN - MAILLOUX - PECK - KOSTER
'ALGOL 68'
MR 101 - Mathematisch centrum - AMSTERDAM - octobre 1969

Documents linguistiques C.E.T.A. (utilisation des modèles syntaxique et étiquetage)

GALICHET 'Etudes sur l'analyse de la phrase allemande dans le système context-free
Thèse - Faculté des Lettres et Sciences Humaines de l'Université de Grenoble - novembre 1968

MAKINOUCHI 'Algorithmes de traduction automatique du japonais en français' - Thèse - Faculté des Sciences de l'Université de Grenoble - mars 1970

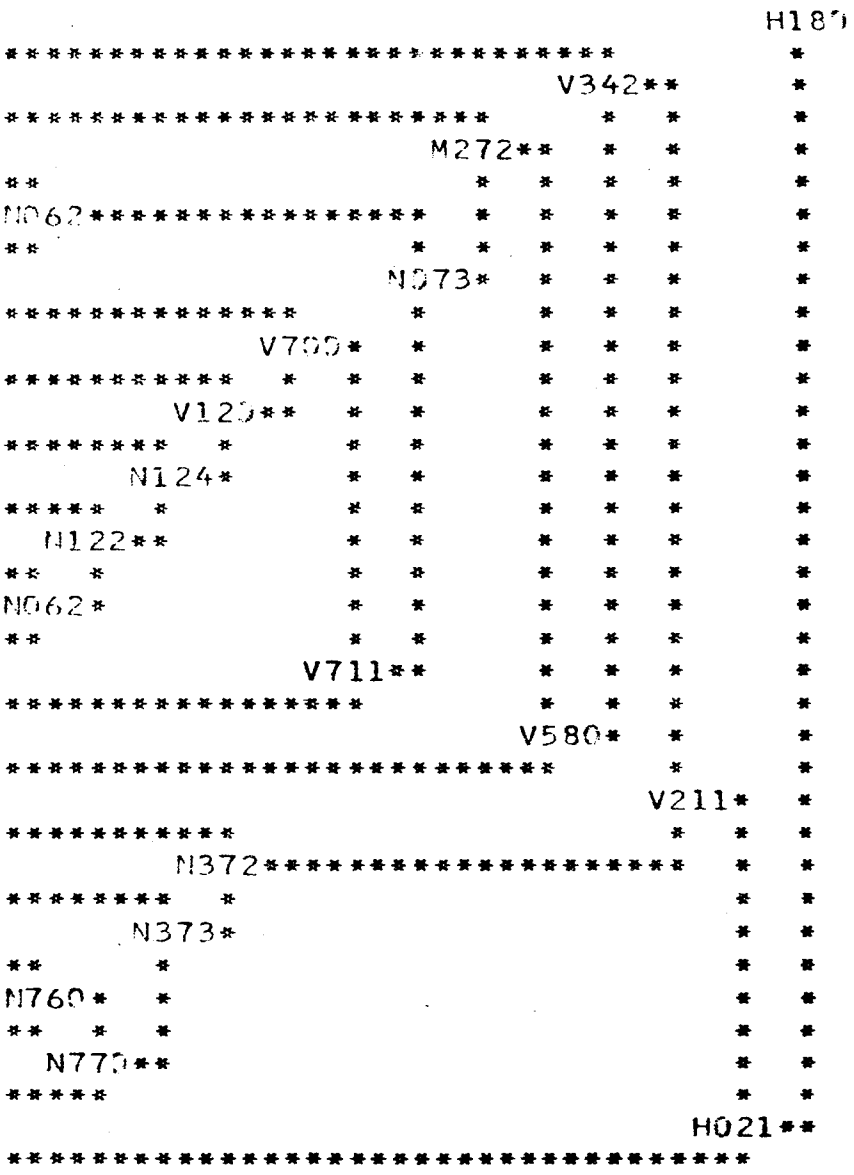
NEDOBEJKINE 'Modèle de la syntaxe russe - Structures abstraites dans une grammaire "Context-Free" - décembre 1964

STIERS 'Reconnaissance syntaxique des constituants de la phrase allemande par un modèle context-free' - mars 1969

A N N E X E I

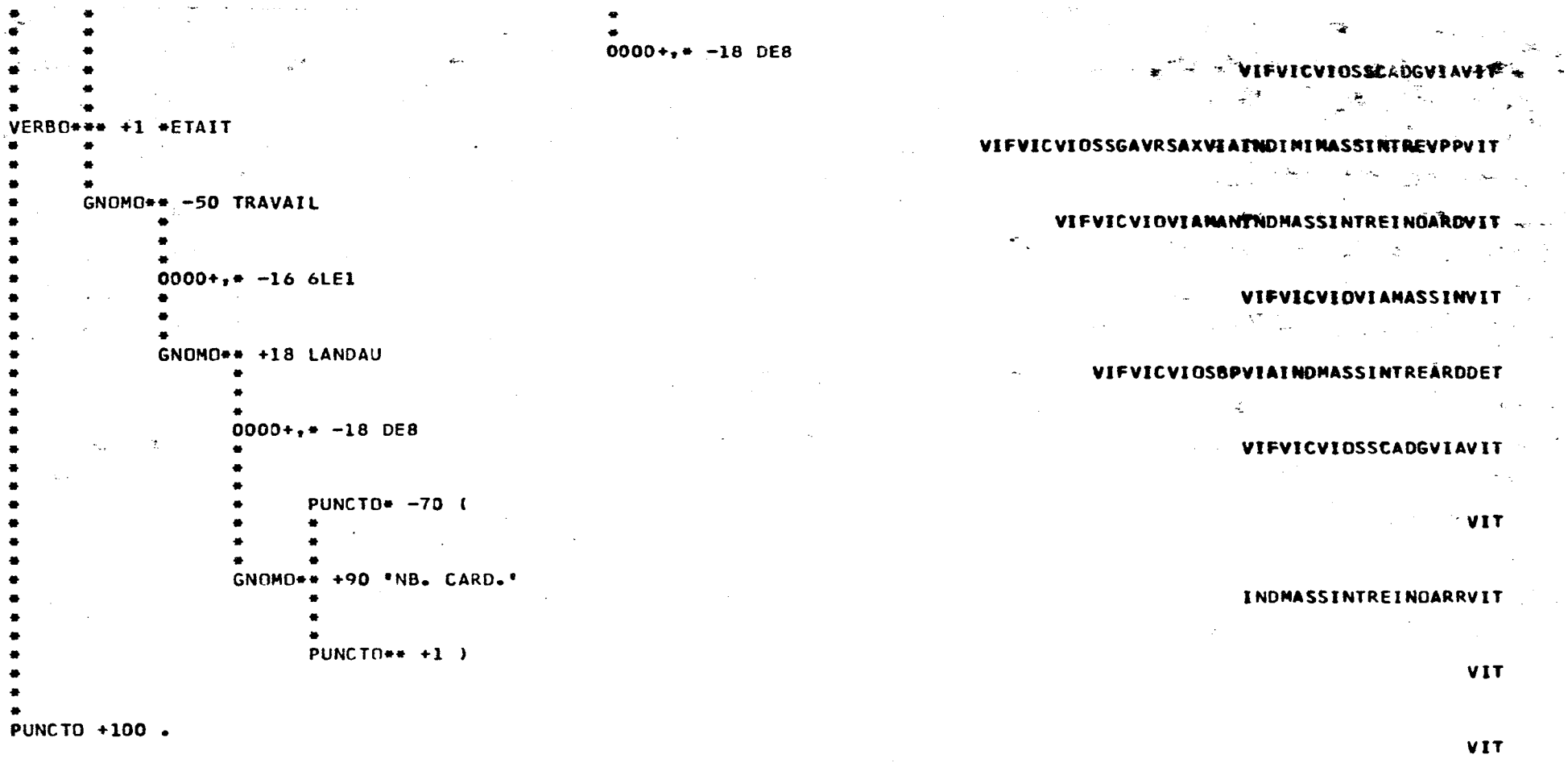
RESULTATS

```
000017
DEBUT 'DEBUT DE PHRASE'
PHRAS
GNOMO ODNOJ
VERBO
ADVGO IZ
ADVGO
VERBO PJERVYKH
GNOMO
GNOMO RABOT
GNOMO
COCOO ,
VERBO
VERBO POSVJASHCHJENNYKH
VERBO
GNOMO ISSLJEDOVANIJU
GNOMO
GNOMO ZATUKHANTJA
GNOMO
VERBO UDARNYKH
GNOMO
GNOMO VOLN
VERBO
COCOO ,
VERBO
VERBO BYLA
VERBO
GNOMO RABOTA
GNOMO
GNOMO LANDAU
GNOMO
TERMN (
GNOMO
GNOMO 'NB. CARDINAL FINI 1'
GNOMO
TERMN )
PHRAS
TERMN .
000041        ' SEC 0 DIX
000001 STRUCTURES IDENTIQUES
```



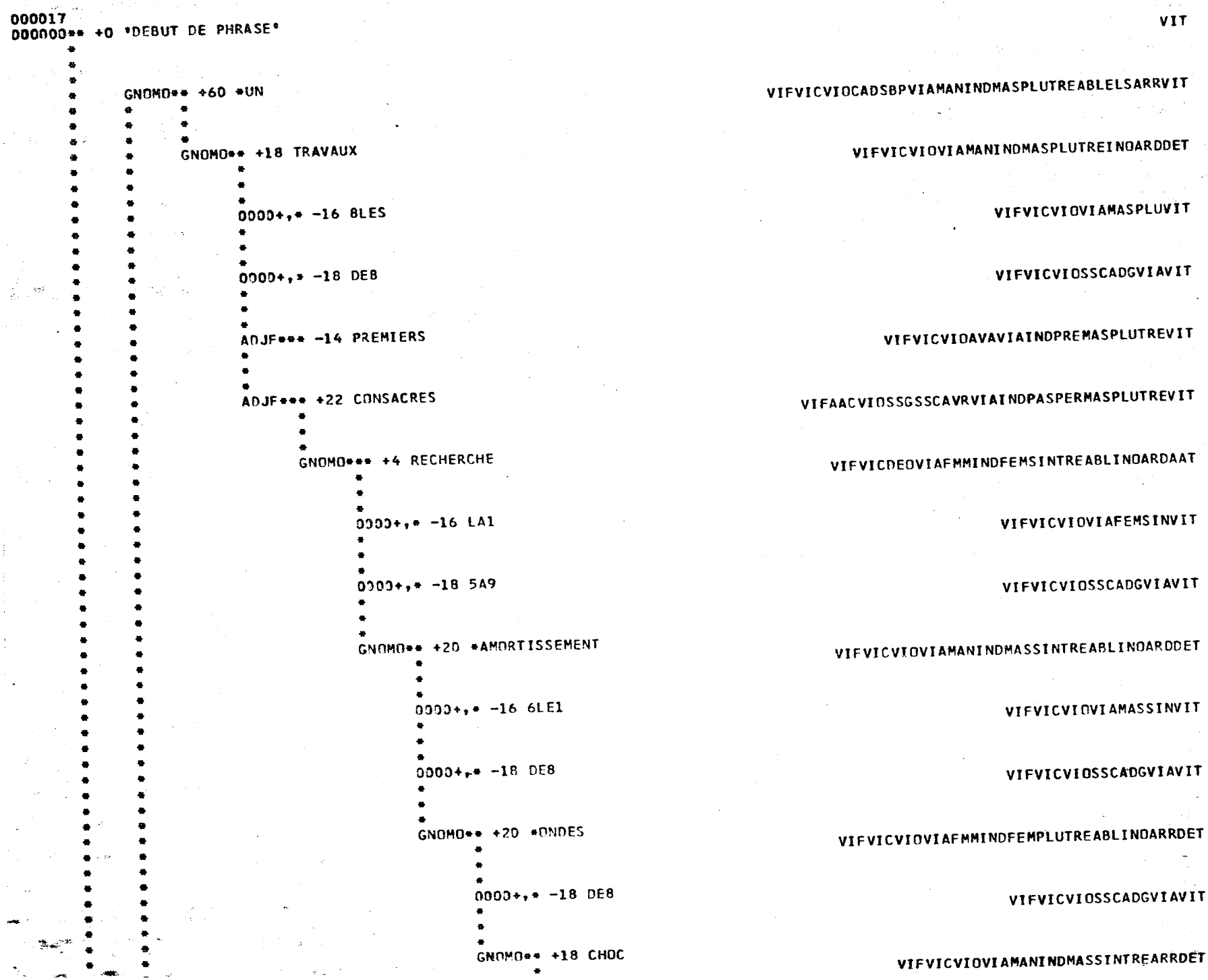
ANNEXE I.1. : STRUCTURE DE CONSTITUANTS DE LA PHRASE 17

```
000017
0000***'DEBUT DE PHRASE'
      *
      *
      *            V342***ODNOJ
      *            *
      *            *
      *            V580***IZ
      *            *     *
      *            *     *
      *            *     *     N062***PJERVYKH
      *            *     *     *
      *            *     *     *
      *            *     M272***RABOT
      *            *           *
      *            *           *
      *            *           N073***,
      *            *                 *
      *            *                 *
      *            *                 V700***POSVJASHCHJENNYKH
      *            *                 *     *
      *            *                 *     *
      *            *                 *     V120***ISSLJEDOVANIJU
      *            *                 *           *
      *            *                 *           *
      *            *                 *           N124***ZATUKHANIJA
      *            *                 *                 *
      *            *                 *                 *
      *            *                 *                 *     N062***UDARNYK
      *            *                 *                 *     *
      *            *                 *                 *     *
      *            *                 *                 N122***VOLN
      *            *                 *
      *            *                 *
      *            *                 V711***,
      *            *
      *            *
      *      H021***BYLA
      *      *     *
      *      *     *
      *      *     V211***RABOTA
      *      *           *
      *      *           *
      *      *           N372***LANDAU
      *      *                 *
      *      *                 *
      *      *                 *     N760***(
      *      *                 *     *
      *      *                 *     *
      *      *                 N373***'NB. CARDINAL FINI 1'
      *      *                       *
      *      *                       *
      *      *                       N770***)
      *      *
      *      *
      H180***.
```


VERBO*** +1 *ETAIT
GNOMO** -50 TRAVAIL
0000+,* -16 6LE1
GNOMO** +18 LANDAU
0000+,* -18 DE8
PUNCTO* -70 (
GNOMO** +90 'NB. CARD.'
PUNCTO** +1)
PUNCTO +100 .
0000+,* -18 DE8
VIFVICVIOSSCADGVIAVIT
VIFVICVIOSSGAVRSAXVIAINDIMIMASSINTREVPPVIT
VIFVICVIOVIAMANINDMASSINTREINOARDVIT
VIFVICVIOVIAMASSINVIT
VIFVICVIOSBPVIAINDMASSINTREARDDET
VIFVICVIOSSCADGVIAVIT
VIT
INDMASSINTREINOARRVIT
VIT
VIT

```
000017
000000*'DEBUT DE PHRASE'
      *
      *
      *            NOMBRE*ODNOJ  *UN
      *            *
      *            *
      *      ACT2  'ELIDE'
      *      *     *
      *      *     *
      *      *     *           ACT2  'SUBSTITUE'
      *      *     *           *
      *      *     *           *
      *      *     *     EPITHE*PJERVYKH  *UN
      *      *     *     *
      *      *     *     *
      *      *     DETERM*RABOT  TRAVAILLER
      *      *           *
      *      *           *
      *      *           *     ACT2  'SUBSTITUE'
      *      *           *     *
      *      *           *     *
      *      *           EPITHE*POSVJASHCHJENNYKH  CONSACRER
      *      *                 *
      *      *                 *
      *      *                 ACT3  *ISSLJEDOVANIJU  CHERCHER
      *      *                       *
      *      *                       *
      *      *                       ACT2  *ZATUKHANIJA  *AMORTIR
      *      *                             *
      *      *                             *
      *      *                             *     DETERM*UDARNYKH  PRODUIRE
      *      *                             *     *     *
      *      *                             *     *     *
      *      *                             *     *     SPECIF'VIRTUEL'
      *      *                             *     *
      *      *                             *     *
      *      *                             ACT1E2*VOLN  *ONDE
      *      *
      *      *
      PHRASE*BYLA  *ETRE
      *     *
      *     *
      *     ACT1  *RABOTA  TRAVAILLER
      *           *
      *           *
      *           DETERM*LANDAU  LANDAU
      *                 *
      *                 *
      *                 *     PUNCTU*(
      *                 *     *
      *                 *     *
      *                 FORMUL*'NB. CARDINAL FINI 1'
      *                       *
      *                       *
      *                       PUNCTU*)
      *
      *
      PUNCTU*.
```

ANNEXE I-4. : STRUCTURE SYNTAXIQUE FRANCAISE DE LA PHRASE 17



ANNEXE I.5 : RESULTAT

Les phrases erronées détectées par les modèles sont imprimées à gauche.

ARTICLE 17

PAGE 3

THEORIE DE L'AMORTISSEMENT D'ONDES STATIONNAIRES ET NON

STATIONNAIRES DE CHOC DANS LES MILIEUX HETEROGENES JU. L. ZHILIN

INTRODUCTION

LE TRAVAIL DE LANDAU (1) ETAIT UN DES PREMIERS TRAVAUX CONSACRES A LA RECHERCHE DE L'AMORTISSEMENT D'ONDES DE CHOC.

(00042) EN UTILISANT LA RESOLUTION EXACTE CONNUE DE RIEMANN DES EQUATIONS DE LA DYNAMIQUE DE GAZ POUR L'ONDE SIMPLE, LANDAU A ETUDIE LE TABLEAU ASYMPTOTIQUE DU MOUVEMENT QUI APPARAIT PENDANT LA PROPAGATION DANS LE MILIEU HOMOGENE DES ONDES PLANES*77 , ET SPHERIQUES SUR LES GRANDES DISTANCES DE SOURCE DE LEUR APPARITION.

/E1

/E1

POUR LA PREMIERE FOIS, LANDAU A MONTRE QUE PENDANT LA PROPAGATION DANS LE MILIEU HOMOGENE DES ONDES CYLINDRIQUES ET SPHERIQUES SUR LES DISTANCES ASYMPTOTIQUEMENT GRANDES DE SOURCE DE LEUR APPARITION LE TABLEAU LIMITE DU MOUVEMENT EN FORME D'ONDE APPELE PAR LA SUITE SE REALISE. LE MOUVEMENT DU GAZ DANS L'ONDE EN FORME DE SE CARACTERISE SELON LE MILIEU NON PERTURBE PAR CE QUE L'ONDE DE CHOC DE TETE APRES LAQUELLE L'ECOULEMENT DE RAREFACTION AVEC UN PROFIL LINEAIRE DE LA CHUTE DE LA PRESSION EXCEDANTE SE REALISE, SE PROPAGE. L'ONDE DE CHOC DE QUEUE QUI RETABLIT LA PRESSION AVANT LA PRESSION DANS LE MILIEU NON PERTURBE, TERMINE CE MOUVEMENT. L'AMORTISSEMENT D'ONDES DE CHOC DANS LES MILIEUX HOMOGENES ETAIT EXAMINE DANS LES TRAVAUX DE SJEDOV (6), DE KHRISTIANOVICH (7) ET D'AUTRES AUTEURS. AUSSI LES RECHERCHES DES LOIS DE L'AMORTISSEMENT D'ONDES DE CHOC DANS LES MILIEUX HETEROGENES ONT OBTENU LE DEVELOPPEMENT ESSENTIEL.

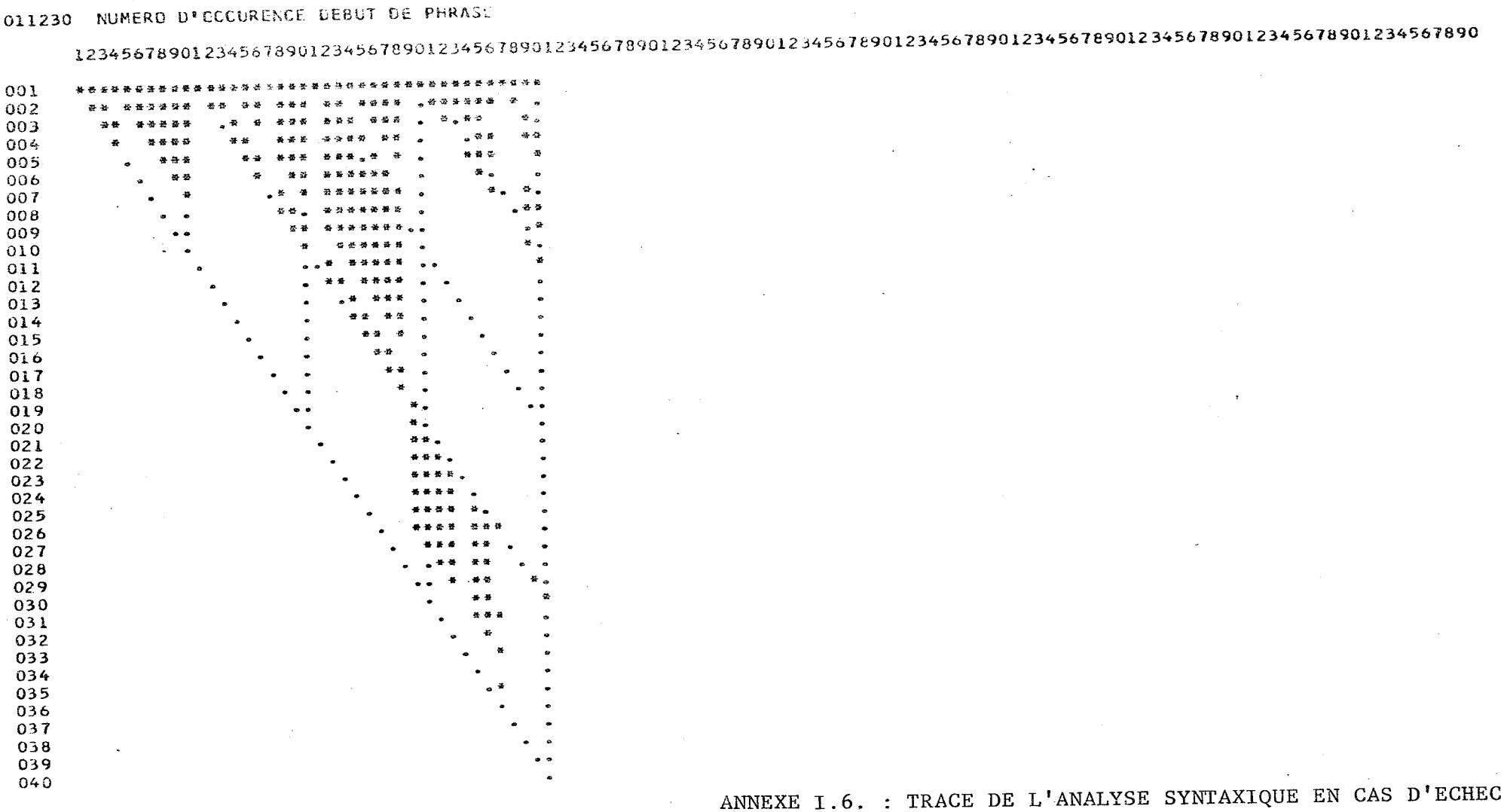
PAGE :4

PENDANT CETTE SUPPOSITION LES PERTURBATIONS FAIBLES DANS LE MILIEU HETEROGENE SE PROPAGENT DE MANIERE QUASI-UNIDIMENSIONNELLE LE LONG DES RAYONS CARACTERISTIQUES. RYJOV A REMARQUE QUE PENDANT LA RECHERCHE DE L'AMORTISSEMENT D'ONDES DE CHOC IL EST NECESSAIRE QU'IL CONSIDERE LA DIVERGENCE LIEE A SON HETEROGENEITE QUI SE PRODUIT DE MANIERE PLUS RESTREINTE DANS L'APPROXIMATION DE L'ACOUSTIQUE GEOMETRIQUE, DES SURFACES CARACTERISTIQUES DU MILIEU NON PERTURBE. DANS CES TRAVAUX CEPENDANT ON N'A OBTENU QUE LA LOI ASYMPTOTIQUE DE L'AMORTISSEMENT D'ONDE DE CHOC DANS LE MILIEU HETEROGENE SUR LES GRANDES DISTANCES DE SOURCE DE SON APPARITION ET ON A ETABLI LA DEPENDANCE DE SON AMPLITUDE PAR RAPPORT A LA

```
 1
 2
 3
 4
 5
 6 *CAS
 7
 8
 9
10 /E1 *GNR
11
12
13
14
15
16
17
18
19
20
21
22 /E1
23
24
25
```

FORME DU CORPS OU PAR RAPPORT A LA DISTRIBUTION INITIALE DE LA PRESSION EXCEDANTE. CETTE LIAISON A ETE ETABLIE DANS LE TRAVAIL COLLECTIF DE FRIEDMAN, DE KANE ET SIGALLA (12) ET DANS LE TRAVAIL DE GUIROUD (13). DANS LE TRAVAIL (12) ON EXAMINE LA PROPAGATION QUASI-UNIDIMENSIONNELLE DES PERTURBATIONS DANS LE TUBE RADIAL. LA RESOLUTION OBTENUE DANS DEPEND D'UNE CONSTANTE INDETERMINEE QUI SE TROUVE A PARTIR DE LA CONDITION DE LA JONCTION DE LA RESOLUTION OBTENUE DANS LE CAS DU MILIEU HOMOGENE AVEC UNE FORMULE ASYMPTOTIQUE DE WHITHAM POUR L'ONDE EN FORME DE . GRACE A DANS LA FORMULE ASYMPTOTIQUE POUR L'AMPLITUDE DE L'ONDE DE CHOC LA DEPENDANCE PAR RAPPORT A LA FORME DU CORPS APPARAIT. DANS LE TRAVAIL DE GUIROUD POUR LA RECHERCHE DE L'AMORTISSEMENT DE PERTURBATIONS PROVOQUEES PAR LE MOUVEMENT DU CORPS, DANS LE MILIEU HETEROGENE ON APPLIQUE LA METHODE DES DEVELOPPEMENTS INTERNES ET EXTERNES. DANS LA PREMIERE APPROXIMATION EN UTILISANT LES VARIABLES CARACTERISTIQUES, GUIROUD CONSIDERE LES EFFETS NON LINEAIRES QUI APPARAISSENT PENDANT LA PROPAGATION DES ONDES COURTES DANS LE MILIEU HETEROGENE. DANS CE TRAVAIL ON A EXAMINE LE CAS ASYMPTOTIQUE DE L'AMORTISSEMENT D'ONDES DE CHOC OU SUR LES GRANDES DISTANCES DE SOURCE DES PERTURBATIONS L'ONDE EN FORME DE SE FORME. DANS LE TRAVAIL DE GUIROUD ON A CONSIDERE L'INFLUENCE MENTIONNEE PLUS HAUT DE LA DIVERGENCE DES SURFACES CARACTERISTIQUES DU MILIEU NON PERTURBE SUR L'AMORTISSEMENT D'ONDES DE CHOC.

FIN DE TRAVAIL



ANNEXE I.6. : TRACE DE L'ANALYSE SYNTAXIQUE EN CAS D'ECHEC

A N N E X E II

PROGRAMMATION

II.1. ANALYSE SYNTAXIQUE ET INTERPRETATION

Nous donnons ici une description des programmes en utilisant ALGOL 68. L'écriture effective en langage assembleur correspond, sauf dans certains cas ('garbage') à une stratégie identique.

II.1.1. Structure

La structuration de la mémoire est celle que nous avons défini en 2.4.4. Cependant, nous avons fait appel au mode union pointe afin de permettre la récupération de la structure utile à la fin de l'analyse. Le programme que nous avons décrit est constitué en effet de deux blocs imbriqués. Dans le bloc extérieur sont déclarées les informations communes à la syntaxe et à l'interprétation. Le bloc 'syntaxe' comporte la grammaire, le programme de balayage et le programme de 'ramassage' (garbage).

Ce dernier ne délivre au bloc extérieur que les structures utiles, connectées à la référence "racine".

II.1.2. Grammaire

Le syntagme comporte un nom de catégorie, un vecteur booléen représentant les variables qui constituent les informations purement grammaticales.

En pratique, le vecteur booléen devrait être structuré lui-même pour tenir compte de tous les noms de variables et des saturations [V_4]

L'entier occ contient le numéro d'occurrence, rang du mot dans la phrase. Il n'a de signification qu'au niveau terminal et n'est pas recopié lors de l'application des règles non terminales, de même que l'entier ul, représentant la référence au dictionnaire.

Chaque règle de grammaire est représentée par une procédure, décrite par le linguiste. Nous avons simplifié cette partie de la description. En fait, l'adressage d'une famille de règles, qui ne fait intervenir ici que les noms de catégorie, doit aussi tenir compte des 'saturations'.

Pour un couple de noms de catégories, plusieurs règles peuvent évidemment intervenir. Chaque règle est caractérisée par un numéro d'identification, qui est spécifié par le linguiste dans un paramètre de la procédure 'range'.

Cette organisation assure une certaine indépendance entre la partie linguistique (la liste des règles), et l'algorithme. La même grammaire pourrait être appliquée avec un algorithme ascendant différent, la seule modification intervenant dans le contenu de la procédure range, qui ne fait pas partie de la grammaire.

Les règles terminales constituent un vecteur (un seul élément en partie gauche).

Le test d'accélération tient compte de la possibilité, pour un symbole B donné d'intervenir comme élément de droite dans une règle non terminale AB $\rightarrow$ C.

II.1.3. Algorithme de balayage

L'algorithme de balayage utilise un tableau triangulaire T de référence aux différents sommets.

A chaque sommet correspond une liste de symboles possibles, tant au niveau terminal qu'au niveau non terminal.

L'organisation est conforme aux descriptions données en 2.4.

La mise en liste des solutions est assurée dans les procédures range (non terminaux) et ranget (terminaux).

Le balayage de ces listes intervient dans gram.

L'entrée supposée est formée d'une chaine de syntagmes. Deux syntagmes sont homographes si ils portent le même numéro d'occurrence. Nous nous sommes fixés une borne supérieure ℓ du nombre de mots d'une phrase, conformément à la méthode utilisée en pratique. Ceci fixe la taille d'un vecteur de pointeurs (référence).

II.1.4. 'Ramassage des miettes' (garbage collector)

A la fin de l'analyse syntaxique, seules les structures conduisant à 'phrase' sont retenues. Les arborescences correspondantes ont toutes pour racine le sommet d'adresse T [i][i], et dont le syntagme porte la valeur 'phrase'. Les structures à conserver sont celles qui sont connectées à ce sommet par l'intermédiaire de 'const', 'suivante', 'gauche' et 'droite'.

Dans la version MAP 7044, le programme procède par une recopie complète des structures retenues. Ce programme utilise une pile de contrôle contenant les adresses des sommets.

La structure globale, contenant toutes les solutions, n'est plus arborescente, puisqu'un sommet peut intervenir dans plusieurs règles. L'information 'cote' indique si un sommet a déjà été rencontré. Dans ce cas, la référence 'autre' contient l'adresse où ce sommet est recopié. Dans la version MAP 7044, ces structures recopiées et condensées sont ensuite extraites sur bande magnétique. Le système procède en effet par chargements successifs, en traitant tout le texte à chaque chargement.

Dans la description ALGOL 68, nous utilisons une procédure récursive ramassage, équivalente à la gestion de la pile. Nous ne recopions pas les informations du type 'liste', qui seront conservées par le 'garbage collector' du système.

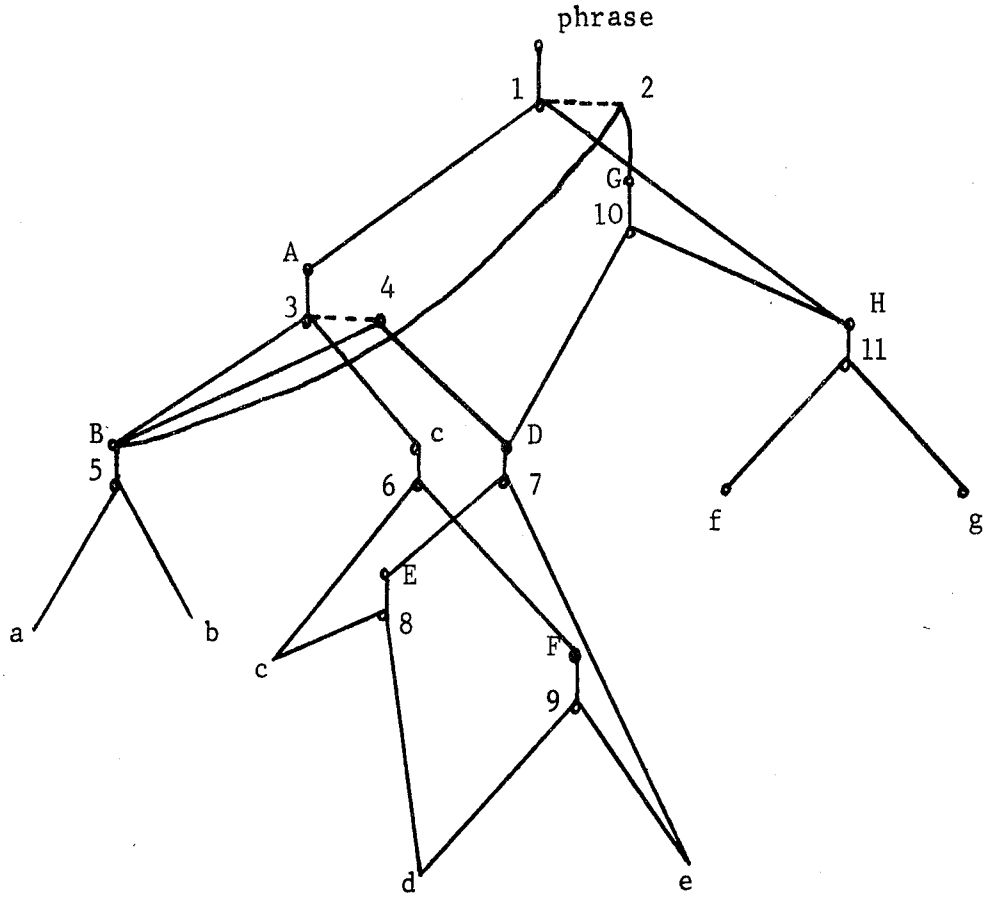
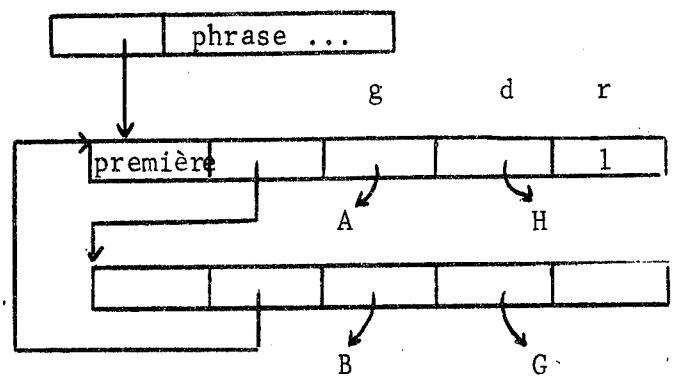


Figure II.1

Exemple de structure globale liée à phrase. Les traits verticaux ⎮ représentent la liaison par 'const', les traits pointillés o..o représentent les liaisons par 'suivante'.

Dans la structure recopiée, la liste des règles (liées par 'suivante') est cyclique, la dernière ayant pour suivante la première, qui est marquée par un booléen.



II.1.5. Le programme suivant a pour rôle d'extraire de la structure globale les diverses solutions une par une. Ce programme est en fait utilisé selon plusieurs utilisations, comme la sortie en clair des résultats d'analyse ou l'interprétation. Nous avons limité ici l'utilisation à l'interprétation proprement dite, sans donner l'algorithme de comparaison de structures (2.5.6.). Ce programme utilise la procédure récursive 'parcours'. Il construit une pile de référence aux sommets de la structure. Chaque solution est fournie par l'état de la pile, mais aussi par l'état de pointeurs "const" des différents sommets. Dans l'exemple de la figure 2.1., la pile contiendra successivement :

1		2		3	
Etat de Pile	Valeurs de const	Pile	Const	Pile	Const
Phrase	1	Phrase	1	Phrase	2
A	3	A	4	B	5
B	5	B	5	a	
a		a		b	
b		b		G	10
C	6	D	7	D	7
c		E	8	E	8
F	9	c		c	
d		d		d	
e		e		e	
H	11	H	11	H	11
f		f		f	
g		g		g	

Le programme modifie à chaque pas la valeur d'un pointeur const, chacun de ceux-ci parcourant ainsi tous les éléments de liste de chaque sommet avant de reprendre la valeur initiale ("première").

```
begin  ¢ traitement syntaxique ¢
    struct sommet = (pointe autre, ref liste const, syntagme synt, bool cote),
           liste = (bool première, ref liste suivante, pointe gauche, pointe droite,
                    int règle),
           syntagme = (int catégorie, bits variables, int occ, int ul),
           som = (ref liste const, syntagme synt),
    mode   pointe = union (ref sommet, ref som) ;
    begin ¢ analyse ¢
          ¢ 1-Définition de la grammaire ¢
          ref sommet g, d ; int i, j, k ;
          [1 : n, 1 : n] ref proc grammaire ¢ table des règles non terminales,
                                              n est le nombre de catégories ¢
          int gnom = 1, gverb = 2,...    phrase=n ¢ valeur des noms de catégories ¢
          for i to n do for j to n do grammaire [i, j]:= nil ;
          ¢ règles syntaxiques ¢
          grammaire [gnom, gnom] := loc proc := (:(if condition1 (g,d) then range
                                                    résultat1 ,1) fi ;
                                                  if condition2 (g,d) then range
                                                  résultat2 ,2) fi ;
                                                  ....      fi))
          ¢ conditioni représente une certaine condition sur variables of synt of g
            et variables of synt of d, résultati est un syntagme calculé à partir
            de g et d ; le numéro associé indique le numéro de règle ¢
          grammaire [gnom, gverb] := loc proc := (:(if conditionp ...
                                                                    ))
           ...
          [1 : m] ref proc term ; ¢ tableau des règles terminales a ⟶ A , m est
                                    le nombre de catégories terminales ¢
          int substantif = 1, adjectif = 2,...        ¢ valeurs des catégories
                                                        terminales ¢
          term [substantif]:= ref proc := (:(if condition1 (t) then
                                               ranget (résultat 1,1) fi
                                             if ...                  fi))
```

```
¢ mêmes conventions que pour les règles non terminales ¢

¢ préparation du test d'accélération ¢
    [1 : n] bool pdt ¢ tableau indiquant si une catégorie donnée peut intervenir
                      comme élément de droite ¢
        for i to n do (pdt [i] := true for j to n while
                        if grammaire [i,j] :=: nil then true else pdt [i]:=false
                            do  skip fi

¢ procédure de test de pdt : balayage d'une liste de sommets liés par autre ¢
    proc pasdroit  = (ref sommet x) bool :
        begin ref sommet y := x ;
        e : if pdt [catégorie of synt of y] then
                if (y := autre of y) :≠: nil then e else true fi
            else false
            fi
        end

¢ 2. Algorithme d'analyse ¢

    [1 : b] ref [  ] ref sommet t, ¢ table d'analyse, b est une borne fixée sur
                                    la longueur de phrase ¢
                int nocc := 1 ¢ numéro d'occurrence courant ¢ ; syntagme s ;
                                                          read (s) ;
    op = = (syntagme a, b) bool : ((catégorie of a = catégorie of b) ∧
                                    (variables of a = variables of b))
    for i to b while catégorie of synt of t [i][i] ≠ phrase do
    begin t [i]:= loc [1:i] ref sommet ; for j to i do t [i][j ]:= nil ;
    appel (t [i][1]) ;
                if pas droit (t [i][j]) then skip else
                   for j to i-1 do
                          if pasdroit  (t [i][j]) then skip else
                          for k to i-j do
                          gram (t [i-j][k], t [i][j], t [i][j+k])
                          fi ;
                fi ;
    end
```

```
¢ procédure gram : appel de la grammaire pour tous les couples possibles ¢
    proc gram = (ref sommet ga, dr, c) :
        begin  if ga :=: nil ∪ dr :=: nil then exit else
                 ref sommet g := ga, d := dr, ref ref sommet res := c ;
               while g :≠: nil do
                  begin while d :=: nil do
                        (grammaire (catégorie of synt of g, catégorie of synt
                                                              of d) ;
                         g := autre of g
                  end
        end

¢ range : est appelé par la règle de grammaire : construit le nouveau syntagme
  et la règle associée ¢
    proc range = (syntagme s, int r) :
    begin  liste x := (false, nil, g, d, r)
           if res :=: nil then res := loc sommet = (nil, x,s, false) else
              ref sommet c := res ;
           boucle : if s = synt of c then suivante of x := const of c ;
                           const of c := x
                                   else if (pointe := autre of c) :≠: nil then
                                                              c := autre of c ;
                                   boucle fi
                    else autre of c := loc sommet = (nil, x, s, false)
                    fi
           fi
     end
    proc ranget = (syntagme s) :
        begin ref sommet c := loc sommet = (nil, nil, s, false) ;
                    if  res :=: nil then skip else autre of c := res fi
              res := c
        end
```

```
proc appel = (ref sommet c) :
     begin   ref ref sommet res := c
          while occ of s = nocc do
          (term (catégorie of s) ; read (s))
          nocc := occ of s
      end
```

```
¢ Ramassage (garbage) ¢

Ramassage : ref sommet courant := T [i][i] ; ref liste u ;
        racine := (nil, synt of courant) ; ref som som ; autre of courant := racine ;
        ramassage (courant) ; ref som scour ;

proc ramassage = (ref sommet cour) :
     begin ref sommet x := cour ;
          continuer : if const of x :=: nil then fin ¢ sortie si sommet terminal ¢
                      else ¢ placer la nouvelle règle en liste ¢
                      u := const of autre of x ; const of autre of x := const of x;
                      const of x := suivante of const of x ;
                      suivante of const of autre of x := u ;
                          courant := gauche of const of autre of x ;
                      if cote of courant ¢ déjà rangé ? ¢ then suite
                      else autre of courant := scour := (nil, synt of courant),
                           cote of courant := true
                      fi ;
                 suite : gauche of const of autre of x := autre of courant ;
                      if const of courant :=: nil ¢ terminal ou épuisé ¢
                        then droite else ramassage (courant) fi
                 droite : courant := droite of const of autre of x ;
                      if cote of courant then suite2 else
                        autre of courant := scour := (nil, synt of courant) ;
                        cote of courant := true ;
                      fi ;
          suite2 : droite of const of autre of x := autre of courant ;
                      if const of courant :=: nil then boucler else ramassage
                                                    (courant) fi
          boucler : ¢ constituer une liste cyclique sur suivante ¢
                      if const of x :≠: nil then continuer else
                        while suivante of u :≠: nil do
                              u := suivante of u ; suivante of u := const of
                                   autre of x ;
                              première of suivante of u := true
                      fi
```

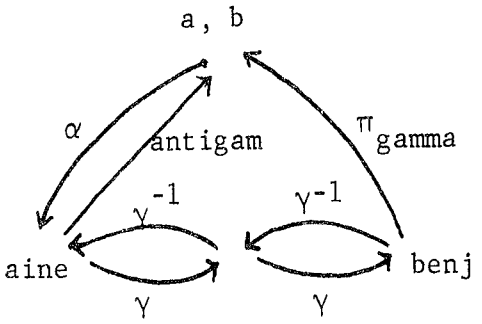
```
    fin : skip
    end
end ¢ fin du traitement syntaxique ¢

¢ programme de parcours de la structure syntaxique, délivre une à une les structures
  par pile ¢
 [1 : 2b -1] ref som pile ; int i := 1 ; pile [i] := racine ;

      départ : parcours (i) ; interp (pile) ;
      boucle : i := i - 1 ; ifconst of pile [i] :=: nil then boucle
               else const of pile [i] := suivante of const of pile [i];
                    if première of const of pile [i] then
                         if i = 1 then exit else boucle fi
                         else depart fi
                    fi

proc parcours = (int j) :
    begin ¢ parcours de la pile ¢ ref som x := pile [j] ;
        if const of x :=: nil then fin else
            j := j+1 ; pile [j]:= gauche of const of x ; parcours (j) ;
            j := j+1 ; pile [j]:= droite of const of x ; parcours (j) ;
        fin : skip
    end
```

II.1.6. INTERPRETATION : STRUCTURE DECRIVANT LES ARBORESCENCES

Chaque sommet ("non terminal") est décrit à l'aide de trois pointeurs (gamma, antigam, alpha) et de deux booléens (ainé, benj), dont la signification est la suivante : (3.1.4.).

si $\neg$ benj, gamma of x = $\gamma(x)$
si benj, gamma of x = $\pi(x)$
si $\neg$ aine, antigam of x = $\gamma^{-1}(x)$
si aine, antigam of x = $\gamma \circ \pi(x)$ = pere (x)
alpha of x = $\alpha(x)$

Exemple :



De plus, chaque sommet est muni d'une 'étiquette' (nom de relation), et est en correspondance avec un élément terminal (occurrence) par 'term'. Cette correspondance peut être indirecte, si sbt (substition) ou elp (élision) est vrai.

Un élément terminal est caractérisé par un rang (entier caractérisant sa position dans la chaine, et deux syntagmes (haut et bas) contenant les informations grammaticales (2.5.5.).

En pratique, l'ensemble des sommets et de leurs terminaux est adressé par un vecteur 'Ordre' qui indique un ordre total sur l'ensemble {terminaux, non terminaux}. Cet ordre est celui de la chaine préfixée décrivant la structure syntaxique.

Le rang de chaque terminal est donc sa position dans ce tableau ordre. Nous avons choisi cette disposition en ALGOL 68 afin de rester proche de la description utilisée en assembleur, dans laquelle la progression séquentielle des pointeurs nous indiquait aussi le rang.

II.1.7. PRINCIPE DE LA CORRESPONDANCE

Etant donnée une structure arborescente syntaxique binaire, dont l'interprétation est marquée sur la figure par des doubles traits, nous lui ferons correspondre une structure interprétée telle que : à chaque sommet non terminal de la structure syntaxique utilisant la règle r, correspond un non terminal de la structure interprétée, dont l'étiquette est r. A chaque terminal syntaxique correspond un terminal dont le syntagme bas est celui du terminal en syntaxe.

La relation γ est compatible avec le rang des terminaux correspondants.

Il existe un sommet racine (étiquette phrase) qui ne correspond à aucun non terminal de la structure syntaxique.

Cette configuration est illustrée par la figure suivante, dans laquelle nous avons noté par des lettres majuscules L les noms de sommets syntaxiques, des lettres minuscules les syntagmes correspondants, les lettres L' représentent les sommets de la structure de dépendances.

Chaque terminal est caractérisé par les syntagmes haut et bas .

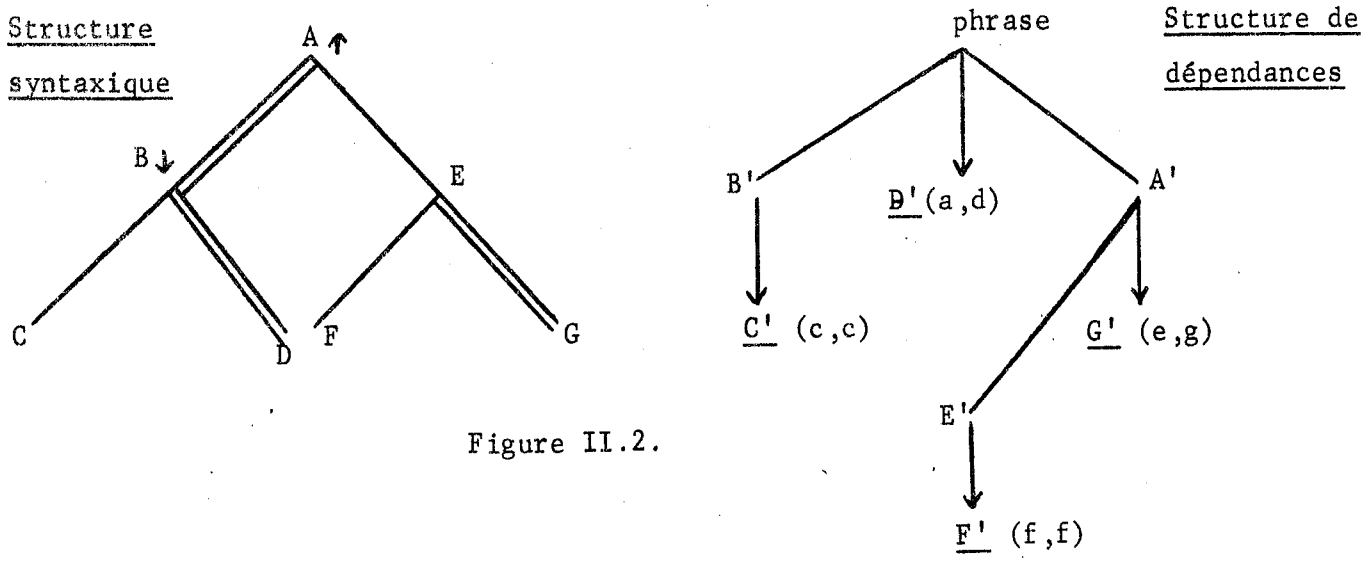


Figure II.2.

Les sommets soulignés sont les terminaux. Ainsi, phrase correspond à l'occurrence D et a pour syntagme haut celui de A, bas celui de D. A' a pour étiquette le nom de règle de A, pour terminal celui de G, et pour syntagme haut celui de E, bas celui de G. Le tableau ordre contient les pointeurs :

'Phrase'A' B' C' D' E' F' G' qui correspond à la chaine préfixée.'Phrase' a le rang zéro, et nous aurons pour terminaux :

C' (rang 3), D' (rang 4), F' (rang 6), G' (rang 7). Ces rangs nous permettront, par la suite, de faire référence à l'ordre entre les mots pour définir des relations de voisinage.

II.1.8. PRINCIPE DU PROGRAMME

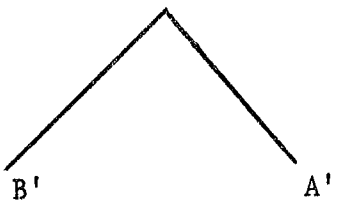
Le programme parcours 'pile' (fournit par 'parcours'), ce qui lui fournit les sommets de la structure syntaxique dans l'ordre séquentiel.

Une pile auxiliaire (pvar) permet de conserver les noms des non-terminaux partiellement traités. Quatre configurations sont possibles :

1) le sommet précédent dans la chaine est terminal, et la pile contient un non terminal "gauche" (état q_2 de l'automate 2.5.3.)
2) le sommet précédent est terminal, la pile contient un non-terminal "droit" (état q_2 de l'automate)
3) le sommet précédent est non terminal, et est "gauche". (état q_1)
4) le sommet précédent est non terminal, et est droit. (état q_1)

Dans chaque cas, nous distinguerons le cas où le sommet lui-même est terminal ou non. Les fonctions à appliquer sont de deux types : créer un dépendant (ouverture de parenthèse) insérer un cadet ou un 'anticadet'.

Exemple : reprenons la figure II.2 : la lecture de B (non terminal 'droit' suivant un non terminal) nous amène à créer B 'anticadet' de A :



la lecture de E, non terminal, suivant un terminal, nous amène à créer un dépendant de A (qui est dans la pile) et à éliminer A de cette pile.

II.1.9. TRAITEMENT DES VEHICULAIRES

A chaque règle correspond un nombre entier caractérisant le traitement des véhiculaires :
1 vidage
2 garnissage
3 transmission sur le gouverneur
4 transmission sur le dépendant

En utilisant une pile (vpil), on construit un ensemble de doublets "émetteurs-récepteurs" caractérisant les couples vidage-garnissage. Après le traitement, le programme 'rectif' transforme la structure. Ainsi, sur la figure II.2 nous avons donné un couple vidage-garnissage. Après le traitement, la pile contient le couple unique A'B' , et la transformation nous fournit la figure II.3. Cette transformation n'altère pas les rangs.

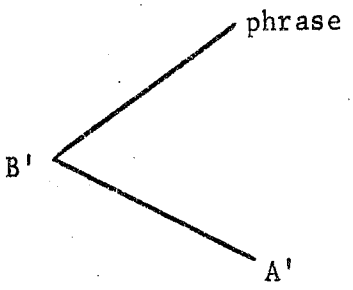


figure II.3

II.1.10. DONNEES LINGUISTIQUES

L'interprétation des règles est supposée fournie par deux tableaux : le tableau coté, contenant des booléens (vrai si gauche) et le tableau vh contenant des entiers (vidage, garnissage, transmission, gouverneur, transmission dépendant)

```
¢ interprétation : sous programme alerté par parcours ¢

proc interp =:
    begin
        ¢ définition des structures ¢
        struct noter = (int étiquette, bool sbt, bool elp, ref ter term, bool aine,
                        ref noter gamma, bool benj, ref noter antigam, ref noter
                        alpha),
             ter = (int rang, ref noter ante, syntagme haut,syntagme bas),
             syntagme vide = (0, 0, 0, 0) , [0:] elem ordre
             union (ter, noter) elem, bool terminal, [1:] ref noter pvar,
             int variable := 1, séquence := 0,
        ¢ déclarations pour le traitement des variables véhiculaires ¢
             struct couples = (ref noter émetteur, récepteur)
             [1:] couples couples ; [1:] ref noter vpile ;
             int c := 1 ; int vhpile := 1 ;
        ¢ données linguistiques ¢
             [1 : n] bool cote ; [1 : n] int vh
        ¢ interprétation : initialisation ¢
             ordre [0 ] := ref noter x := (phrase, false, false, (0, x, vide,
                                          vide), true, nil, true, nil, nil) ;
             dépendant (ordre [0]) ; pvar [1] := ordre [1];
        ¢ interprétation : programme ¢
             for séquence := 1 while variable ≠ 0 do
                  begin if terminal then
                        if cote [étiquette of pvar [variable]] then
                           dépendant (pvar [variable]) else droite fi
                              else
                        if cote [étiquette of ordre [séquence]] then
                           gauche else dépendant (ordre [séquence]) fi ;
                           variable :=+1
                        fi ;
                        if terminal then variable := -1
                                    else pvar [variable] := ordre [séquence +1]
                        fi
                  end
```

```
¢ créer un premier dépendant, si il est terminal, il n'y a pas de nouveau sommet
  mais une affectation de syntagmes ¢

proc dépendant = (ref noter x) :
    begin créer ; ref elem y ; if terminal then
          ordre [séquence + 1] := y := term of x ;
          haut of y := bas of y := syntagme of pile [séquence  + 1]
                              else
          aine of y:= ordre [séquence + 1] := true ; benj of y := true ;
          gamma of y := antigam of y := x
          haut of term of x := syntagme of pile [séquence + 1];
          véhic ; fi
    end

proc droite =:
    begin  créer ; ref noter x ;
          if terminal then
               x := pere (pvar [variable]) ; bas of term of x := syntagme of pile
                                                              [séquence + 1] ;
               ordre [séquence + 1]:= term of x ;
               rang of term of x := séquence + 1
                    else cadet (ordre [séquence + 1], pvar [variable]) ; vehic ;
          fi
    end

proc gauche =:
    begin créer ; ref noter x ;
         if terminal then x := père (ordre [séquence]) ;
            bas of term of x := syntagme of pile [séquence + 1];
            ordre [séquence + 1] := term of x ;
            rang of term of x := séquence + 1 ;
                  else
            anticadet (ordre [séquence + 1], ordre [séquence]) ; vehic
         fi
    end
```

```
proc créer =:
     begin if terminal := const of pile [séquence + 1] :=: nil
              then skip
              else ordre [séquence  + 1] := ref noter x :=
              (règle of const of pile [séquence + 1], false, false, (0, x, vide, vide)
              false, nil, nil, false, nil) ;
           fi
     end

proc père = (ref noter x) ref noter :
     begin ref noter y := x ;
          while benj of y do y := gamma of y ;
             gamma of y
     end

proc cadet = (ref noter x, y) :
     begin gamma of x := gamma of y ; antigam of x := y ; gamma of y := x ;
          if benj of x := benj of y then skip
               else antigam of gamma of x := x
          fi
     end

proc anticadet = (ref noter x, y) :
     begin antigam of x := antigam of y ; gamma of x := y ;
           antigam of y := x ;
           if aine of x := aine of y then alpha of antigam of x := x
                                     else gamma of antigam of x := x
           fi
      end
```

```
proc vehic =:
    begin                                   ref noter x := ordre [séquence]
        case vh [étiquette of x] in vidage, garnissage, transgouv, transdep out
                                    fin esac
        vidage : sbt of père (x) := true, vhpil := +1 ;
                 vpile [vhpile] := x ; fin ;
        garnissage : if sbt of père (x) then c := +1 ;
                     couples [c]:= (vpile [vhpile], x) ; vhpile := -1 ;
                     if vhpile = 0 then sbt of z := false fi ; fin ;
        transgouv : fin ;
        transdep : sbt of x := sbt of père (x) ; sbt of père (x) := false ;
        fin : skip ;
    end


Rectif : for vhpile := 0 to c do
    begin
        ref noter x := émetteur of couples [vhpile];
        if aine of x    benj of x
            then alpha of antigam of x := nil
            else if aine of x then alpha of antigam of x := gamma of x,
                antigam of gamma of x := antigam of x, aine of gamma of x = true
                else if benj of x then gamma of antigam of x :=
                    gamma of x, benj of antigam of x := true
                         else
                    gamma of antigam of x := gamma of x, antigam of gamma of x :=
                                                        antigam of x fi
                    ref noter y := récepteur of couples [vhpile];
                    if alpha of y :=: nil then alpha of y := x ;
                    gamma of x := antigam of x := y ; aine of x := benj of x :=
                                                                    true ;
                    else gamma of x := alpha of y ;
                    antigam of x := y, alpha of y := x ,
                    antigam of gamma of x := x ; aine of gamma of x := false ;
                    aine of x := true ;
    end
    end ¢ fin du traitement ¢
```

II.2. MODELES DE TRANSFORMATIONS, REALISATION PRATIQUE

La réalisation pratique des transformations utilise le macroassembleur MAP. Chaque relation est définie par la macro-instruction 'RELAT' dont les deux paramètres sont les fonctions f et g.

Le langage d'écriture des grammaires utilise lui aussi le macroassembleur. Chaque règle peut comporter, outre le schéma de figure et les transformations, une série de conditions booléennes appelées par 'SI'. En outre elles peuvent utiliser des procédures booléennes, définies par 'ligne'. Les principales macro-instructions et des exemples de règles sont données dans les pages suivantes.

```
77776       .L      BCCL    77776                                                   722
            TSX     OPD     007400000000,,2,3,6                                     723

            REGLE   MACRC   X,Y                                                     725
                    USE     .A                                                      726
            X       BSS     0               NCM DE REGLE                            727
                    NCCRS                   POUR PARTIE GAUCHE                      728
                    CLAL    C*X             QUALIFIER TCUTE LA REGLE                729
                    CHEF                    INITIALISATION POUR PARTIE GAUCHE       730
            .RECLR  SET     0                                                       731
                    TSL     INIT                                                    732
                    AXT     Y,              ADRESSE DE FIN DE REGLE                 733
                    SPACE   3                                                       734
                    ENDM    REGLE                                                   735


            FREGLE  MACRC   X                                                       737
                    USE     .A              REVENIR AU COMPTEUR PRINCIPAL           738
                    SPACE   3                                                       739
            SUITE   EQU     *                                                       740
                    ENDC    C*X             FIN DE ZONE REGLE                       741
                    ENDM    FREGLE                                                  742


            CHEF    MACRC                                                           744
                    LAC     IPILC,2                                                 745
                    ENDM    CHEF                                                    746
```

ANNEXE II : MACRO DEFINITION DU SCHEMA DE FIGURE

PARTIES GAUCHES DE REGLES

```
.YZ     MACRO   FONC,REGLE,SUITE                                        749
        TSX     FONC,4,,RECUR   APPEL RELATION, RECUR = PROF PARENTHESE 750
.T      SET     0                                                       751
        IRP     REGLE                                                   752
.T      SET     .T+1            Y A-T-IL PLUSIEURS ELEMENTS DANS REGLE  753
        IRP                                                             754
        IFT     .T=1                                                    755
        PZE     ,,REGLE         SINON NOTER REGLE                       756
        IFF     .T=1                                                    757
        .CHCT   REGLE                                                   758
.C      SET     0                                                       759
        IRP     SUITE                                                   760
.C      SET     .C+1            Y A-T-IL UNE SUITE                      761
        IRP                                                             762
.RECUR  SET     0                                                       763
        IFF     .C=0                                                    764
        GAUCHE  ('SUITE')       OUI, CONTINUER                          765
        ENDM    .YZ                                                     766


.CHCT   MACRO   REG,POIDS                                               768
        PZE     POIDS,,REG                                              769
        ENDM    .CHCT                                                   770


GAUCHE  MACRO   A               PRINCIPALE                              772
        IRP     A                                                       773
        .YZ     A               APPELLE .YZ                             774
.RECUR  SET     .RECUR+1        ET COMPTE LES PARENTHESES FERMEES       775
        IRP                                                             776
        SPACE   3                                                       777
        ENDM    GAUCHE                                                  778
```

MACRO DEFINITION DES RELATIONS

```
                CRGCRS                                                  RA101820  182


                                                                        RA101840  184
RIEN    MACRO   X                                                       RA101850  185
        TRA     X                                                       RA101860  186
        ENDM    RIEN


                                                                        RA101880  188
RELAT   MACRO   F,G,I,X                                                 RA101890  189
        MIT     ERREUR                                                  RA101900  190
        TRA     *+5             FAUX APPEL POUR REPRISE                 RA101910  191
        MSP     ERREUR                                                  RA101920  192
        CAL     ,FTR            REPARTIR SUR G                          RA101930  193
        FAX     ,SCR            AVEC LE SOMMET OBTENU                   RA101940  194
        TRA     X                                                       RA101950  195
I       TSL     FAR                                                     RA101960  196
        IRP     F                                                       RA101970  197
        F       NIE                                                     RA101980  198
        IRP                                                             RA101990  199
        TSL     COMPA           COMPARAISONS DE SOM                     RA102000  200
X       ECU     *                                                       RA102010  201
        IRP     G                                                       RA102020  202
        G       NIE                                                     RA102030  203
        IRP                                                             RA102040  204
        TRA     X-1                                                     RA102050  205
        ENDM    RELAT
```

ANNEXE II : MACRO DEFINITION DES RELATIONS

```
                                        NIB, CA N A PAS MARCHE                                  RA1CC55C  55
00C00 -1341 C7 C C1C31 1CCC1 NIB    PLT ERRCA2                                                  RA1CC56C  56
00C01  0020 CC C CCC11 1CCC1        TRA NIBCA2                                                  RA1C0570  57
00C02 -C5C0 CC 2 CCCCC 1CCCC        CAL ,FTR                                                    RA1C058C  58
00C03 -C734 CC 2 CCCCC 1CCCC        PDX ,FTR            ECHANGE                                 RA1CC59C  59
00C04 -C5C0 CC 2 CCCC1 1CCCC        CAL 1,FTR           APPEL REGLE C OU L ON VENAIT            RA1CC60C  60
00C05  C1CC CC C CCCCC 1CCCC INITA  TZE **              AC RANGEE PAR PROG    INIT              RA1C061C  61
00C06 -1623 C6 C CCC71 1CCC1        MSP ERREUR          PREVENIR                                RA1CC62C  62
00C07  C734 CC 4 CCCCC 1CCCC        PAX ,REG                                                    RA1CC63C  63
00C10  CC20 CC 4 CCCCC 1CCCC        TRA ,REG                                                    RA1CC64C  64
00C11  1 CCCC2 2 CCC12 1CCC1 NIBCA2 TXI **+1,PTR,+2                                             RA1CC65C  65
00C12 -C634 CC 2 CCC55 1CCC1        SXD ECRANTE,PTR                                             RA1CC66C  66
00C13 -C754 CC C CCCCC 1CCCC        ZAC                                                         RA1CC67C  67
00C14  C621 CC 2 CCCC2 1CCCC        STA 2,PTR           EFFACER L'AC. DU NOEUD                  RA1CC68C  68
00C15  C774 CC 1 CCCCC 1CCCC SCRCA2 AXT **,SCM                                                  RA1CC69C  69
00C16  C774 CC 2 CCCCC 1CCCC        AXT **,PTR                                                  RA1CC7CC  70
00C17  CC20 CC 4 CCCC2 1CCCC        TRA 2,REG                                                   RA1CC71C  71


                                        FONCTIONS F OU C                                        RA1CC73C  73
                                        PARTENT DE SOM, I F(SOM) DANS SOM ET DANS SOM (PTR)     RA1CC74C  74


                                ANT     MACRO   S,X                                             RA1CC76C  76
                                        CAL     1,SCM                                           RA1CC77C  77
                                        ANA     =C700000                                        RA1CC78C  78
                                        TZE     S                                               RA1CC79C  79
                                X       CAL     1,SCM                                           RA1CC80C  80
                                        ANA     =C77777                                         RA1CC81C  81
                                        PAX     ,SCM                                            RA1CC82C  82
                                        TZE     S                                               RA1CC83C  83
                                        CAL     1,SCM                                           RA1CC84C  84
                                        ANA     =C700000                                        RA1CC85C  85
                                        TNZ     X                                               RA1CC86C  86
                                        PXA     ,SCM                                            RA1CC87C  87
                                        STA     ,PTR                                            RA1CC88C  88
                                        ENDM    ANT                                             RA1CC89C  89


                                FAINE   MACRO   S                                               RA1CC91C  91
                                        CAL     ST,SCM                                          RA1CC92C  92
                                        PAX     ,SCM                                            RA1CC93C  93
                                        TXL     S,SCM,1     PAS E FILS                          RA1CC94C  94
                                        STA     ,PTR                                            RA1CC95C  95
                                        ENDM    FAINE                                           RA1CC96C  96
```

ANNEXE II : MACRODEFINITION DES RELATIONS : RELATIONS ELEMENTAIRES

```
FFAINE  MACRO   X                       RA1CC98C  98
X       CAL     ST,SCM                  RA1CC99C  99
        ANA     =O77777                 RA1CCCCC  100
        TZE     **3                     RA1CC01C  101
        PAX     ,SCM                    RA1CC02C  102
        TRA     X                       RA1CC03C  103
        ENDM    FFAINE                  RA1CC04C  104

FCADET  MACRO   S                       RA1CC06C  106
        PLT     ST,SCM                  RA1CC07C  107
        TRA     S                       RA1C108C  108
        CAL     ST,SCM                  RA1CC09C  109
        PDX     ,SCM                    RA1CC10C  110
        TXL     S,SCM,1                 RA1C111C  111
        PXA     ,SCM                    RA1CC12C  112
        STA     ,FTF                    RA1CC13C  113
        ENDM    FCADET                  RA1CC14C  114

FBENJ   MACRO   S,MACHIN,TRUC           RA1CC16C  116
MACHIN  FCADET  TRUC                    RA1C117C  117
        TRA     MACHIN                  RA1CC18C  118
TRUC    ECU     *                       RA1C119C  119
        ENDM    FBENJ                   RA1CC20C  120

FIBENJ  MACRO   S                       RA1CC22C  122
        FAINE   S                       RA1C123C  123
        FBENJ                           RA1CC24C  124
        ENDM    FIBENJ                  RA1CC25C  125

PERE    MACRO   S                       RA1CC27C  127
        FBENJ                           RA1CC28C  128
        TXL     S,SCM,1                 RA1CC29C  129
        CLA     ST,SCM                  RA1CC30C  130
        PDX     ,SCM                    RA1CC31C  131
        TXL     S,SCM,1                 RA1CC32C  132
        PXA     ,SCM                    RA1C133C  133
        STA     ,FTF                    RA1CC34C  134
        ENDM    PERE                    RA1CC35C  135
```

```
                          FS      RELAT   FAINE,FCAEET                                    RA1C250C  250

BINARY CARD CCCC4
                          FR      RELAT   FCAEET,FCAEET                                   RA1C251C  251

BINARY CARD CCCC5
                          AN      RELAT   FERE,FERE                                       RA1C252C  252

BINARY CARD CCCC6

BINARY CARD CCCC7
 00212  2 CCCCC C CCCCC CCCC1 PR      BSS     0                                          RA1C253C  253
                              PA      RELAT   FERE,RIEN                                  RA1C254C  254

BINARY CARD CCC1C

BINARY CARD CCC11
                          FN      RELAT   FAINE,RIEN                                      RA1C255C  255
                          VG      RELAT   VSG,RIEN                                        RA1C256C  256

BINARY CARD CCC12

BINARY CARD CCC13
                          VD      RELAT   VSE,RIEN                                        RA1C257C  257

BINARY CARD CCC14
                          ND      RELAT   (FCAEET,FAINE),RIEN                             RA1C258C  258

BINARY CARD CCC15
                          FB      RELAT   FIEENJ,RIEN                                     RA1C259C  259

BINARY CARD CCC16
                          GH      RELAT   VSG,VSG                                         RA1C260C  260

BINARY CARD CCC17

BINARY CARD CCC2C
                          DR      RELAT   VSE,VSE                                         RA1C261C  261

BINARY CARD CCC21

BINARY CARD CCC22
                          AC      RELAT   ANTICA,ANTICA                                   RA1C262C  262

BINARY CARD CCC23
                          AT      RELAT   ANT,RIEN                                        RA1C263C  263

BINARY CARD CCC24
```

ANNEXE II : REGLES DEFINISSANT LE BALAYAGE


```
                REGLE   FIN,SUITE                                    1495

                GAUCHE  ((FS,0))                                      1496

                CONDIT                                                1497

                PIVOT   1(A,CONTRO)                                   1498
                FREGLE  FIN                                           1499

                REGLE   R.F2,SUITE                                    1500
BINARY CARD C0326

                GAUCHE  ((FR,0))                                      1501

                CONDIT                                                1502

                PIVOT   1(A,CONTRO)                                   1503
                FREGLE  R.F2                                          1504

                REGLE   R.F3,FINET                                    1505

                GAUCHE  ((FA,0))                                      1506

                CONDIT                                                1507

                PIVOT   1(A,R.F2)                                     1508
                FREGLE  R.F3                                          1509
```

```
                        REGLE   R10,SUITE                                          1565

BINARY CARD CC332

                        GAUCHE  ((FS,CCCRCX)(FS,CCCRUN))                           1566

                        TABLO                                                      1567
                B       LIGNE   (R(FS,CCCRUN))                                     1568
                A       LIGNE   (R(FS,CCCRCX))                                     1569
                        CONDIT                                                     1570
                        SINON   (((P,-61)0)U((P,+85)0)U((E,CERONC)0))SUITE         1571
BINARY CARD CC334
                        SINON   (((NA,03)0)U((G,VIC)0))SUITE                       1572
                        SI      ((P,+22)0)BB                                       1573
                        SYMBO   2(EPI)                                             1574
                        SI      ((A,1)E(B,1))AA                                    1575
BINARY CARD 00335
                        STRUCT  ((0,SUBS,3)(3,AINE,0)(3,BENJ,2)(3,BENJ,1))         1576
                        STRUCT  ((0,BENJ,4))                                       1577
                        SYMBO   1((E,FUNCTC)(P,-100))                              1578
                        SYMBO   3((E,.,0)(P,.,0)(EVT,.,0))                         1579
BINARY CARD 00336
                        SYMBO   0((E,CCCRCX))                                      1580
                        SYMBO   4((E,CCRTRE)EPI)                                   1581
                        PIVOT   3(A,SUITE)                                         1582
                AA      STRUCT  ((0,SUBS,1)(1,AINE,0)(1,BENJ,2))                   1583
                        SYMBO   1((E,.,0)(P,.,0)(EVT,.,0))                         1584
BINARY CARD CC337
                        SYMBO   0((E,CCCRCX))                                      1585
                        PIVOT   1(A,SUITE)                                         1586
                BB      SI      ((A,1)E(B,1))AA                                    1587
                        STRUCT  ((0,SUBS,3)(3,BENJ,1)(3,AINE,0)(0,BENJ,4)(3,BENJ,2))  1588
BINARY CARD 00340
                        SYMBO   1((E,FUNCTC)(P,-100))                              1589
                        SYMBO   4((E,CCRTRE)(P,+2))                                1590
                        SYMBO   3((E,.,0)(P,.,0)(EVT,.,0))                         1591
                        SYMBO   0((E,CCCRCX))                                      1592
BINARY CARD CC341
                        PIVOT   3(A,SUITE)                                         1593
                        FREGLE  R10                                                1594
```

ANNEXE II : Exemple de règle de transformation (syntaxe française ; traitement des coordinations)

A N N E X E III

STRUCTURES DE CONSTITUANTS ET STRUCTURES DE DEPENDANCES

Cette formalisation m'a été suggérée par Monsieur Pair, qui a participé largement à son élaboration.

III.1.1 RAMIFICATION [Pa_1, Pa_2,]

Une ramification sur un vocabulaire V est une suite finie de pseudo-arborescences [1.4.4.] sur V. L'ensemble des ramifications sur V est noté $\hat{V}$, l'ensemble des pseudo arborescences sur V est noté a (V).

Algébriquement, les ramifications sont définies par

1) une loi de composition interne associative d'élément neutre Λ

2) une loi de composition externe, à opérateurs dans V, notée x, avec opérateur à gauche, telle que a x Λ = a ('enracinement')

Exemple : Soient r la ramification de la figure 1

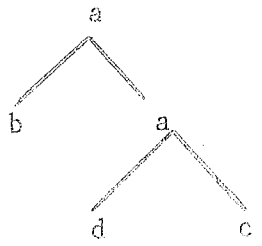


Figure 1

r' la ramification de la figure 2

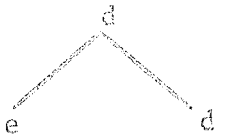


Figure 2

a x (r + r') est la ramification de la figure 3

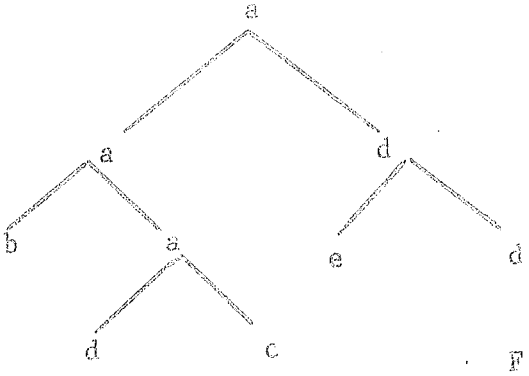


Figure 3

On montre que si un prédicat P est tel que

1) P (Λ) est vrai
2) ($\forall$ r $\in \hat{V}$; s $\in$ $\mathfrak{a}$ (V)) (P (r) x P (s) $\rightarrow$ P (r + s)) [1.4.2.6.]
3) ($\forall$ r $\in \hat{V}$, V a $\in$ V) (P (r) $\rightarrow$ P (a x r))

Alors P (r) est vrai pour tout r $\in \hat{V}$

III.1.2. NOMBRE DE SOMMETS

Soit n (r) une application de $\hat{V}$ dans $\mathbb{N}$ telle que :

1) n (Λ) = 0
2) n (r + s) = n (r) + n (s)
3) n (a x r) = n (r) + 1

n (r) est le nombre de sommets de r

III.1.3. MOT DES RACINES ET MOT DES FEUILLES

Le mot des racines de r$\in \hat{V}$ est le mot ρ (r) sur V défini par :

1) ρ (Λ) = Λ
2) ρ (r + s) = ρ (r) ρ(s)
3) ρ (a x r) = a

Le mot des feuilles de r $\in \hat{V}$ est le mot φ(r) sur V défini par

1) φ(Λ) = Λ
2) φ (r + s) = φ(r) φ(s)
3) φ (a x r) = φ(r) si r $\neq \Lambda$
= a si r = Λ

III.1.4. FAMILLES DE PREDECESSEUR a

Soit a $\in$ V et l'application F_a de $\hat{V}$ dans l'ensemble P (V^*) des parties de V^* telles que :

1) F_a (Λ) = Λ
2) F_a (r + s) = F_a (r) $\cup$ F_a (s)
3) F_a (b x r) = si b = a , alors F_a (r) $\cup$ {ρ(r)} sinon F_a (r)

Exemple : dans la ramification de la figure 3, F_a (a x (r + r')) = {ad, ba, dc}.
Les éléments de F_a (r) constituent les familles de prédécesseur a.

III.2.1. STRUCTURE DE DEPENDANCES SUR UN VOCABULAIRE V

Une structure de dépendances est une ramification sur V x $\mathbb{N}$ telle que si (a, i) est prédécesseur d'une famille φ , $1 \leqslant i \leqslant |\varphi| + 1$, la phrase associée à la ramification r est définie par l'application λ de r dans V^* telle que :

1) $\lambda(\Lambda) = \Lambda$
2) $\lambda((a, i) \times (r_1 + \ldots + r_p)) = \lambda(r_1)\,\lambda(r_2) \ldots \lambda(r_{i-1})\, a\, \lambda(r_i) \ldots \lambda(r$
3) $\lambda(a, 1) = a$

III.2.2. FORME PARENTHESEE

Soit $V_1 = V \cup \{[,]\}$, la forme parenthésée associée à une structure de dépendances r sur V est définie par l'application μ de r dans V_1^* telle que :

1) $\mu(\Lambda) = \Lambda$
2) $\mu((a, i) \times (r_1 + \ldots + r_p)) = [\mu(r_1)][\mu(r_2)] \ldots [\mu(r_{i-1})]a[\mu(r_i)] \ldots [\mu(r_p)]$

La forme ainsi obtenue est celle que nous avons décrite en 2.5.1. dans laquelle la dernière parenthèse a été supprimée. Nous utilisons cette forme, plus pratique, pour la suite de cette présentation.

Proposition 1 : Soit l'application e de V_1^* dans V^* définie par e ([) = e (]) = Λ, e (a) = a si a $\in$ V. Alors $\lambda = e \circ \mu$

Proposition 2 : Si r et s sont deux structures de dépendances distinctes, alors $\mu(r) \neq \mu(s)$

Ceci est vrai lorsque n (r) $\neq$ n (s), car $\lambda(r) \neq \lambda(s)$. Lorsque n (r) = n la proposition est vraie pour n = 0 et n = 1, car alors

r = (a, 1), s = (b, 1) , $r \neq s \rightarrow a \neq b$ et $\mu(r) \neq \mu(s)$

Soient alors $r' = r_1 + r_2 + \ldots r_p$, $s' = s_1 + s_2 + \ldots + s_q$, telles que $n(r_i) \leqslant n_0$, $n(s_i) \leqslant n_0$, et que $r_i \neq s_i \rightarrow \mu(r_i) \neq \mu(s_i)$.

Soient $r = (a, i) \times r'$ et $s = (b, j) \times s'$

$r \neq s$ implique soit $(a, i) \neq (b, j)$, soit $r' \neq s'$

Si $r' \neq s'$, il existe un entier $k = \min_{\ell} (r_\ell = s_\ell)$

Si $k < i$ et $k < j$, alors il existe une décomposition unique :

$$\mu(r) = \lceil \mu(r_1) \rceil \ldots \lceil \mu(r_k) \rceil \Psi = \alpha \lceil \mu(z_k) \rceil \Psi$$

$$\mu(s) = \lceil \mu(s_1) \rceil \ldots \lceil \mu(s_k) \rceil \Psi' = \alpha \lceil \mu(s_k) \rceil \Psi'$$

$\alpha \in V_1^*$, $\Psi, \Psi' \in V_1^*$, $\mu(r_k) \neq \mu(s_k)$ par hypothèse de récurrence et

$$\mu(r) \neq \mu(s)$$

Sinon, si $(a, i) = (b, j)$, le même raisonnement peut s'appliquer, en remarquant que si $k = p + 1$, $\mu(r) = \alpha$, $\mu(s') = \alpha \lceil \mu(s_k) \rceil \Psi$, sinon, si $(a, i) \neq (b, j)$, $i, j < k$, l'on peut noter :

$\mu(r) = \lceil \alpha \rceil a \Psi$, $\mu(s) = \lceil \beta \rceil b \Psi'$; si $i \neq j$, $\alpha \neq \beta$ et $\mu(r) \neq \mu(s)$; si $i = j$, $a \neq b$ et $\mu(r) = \lceil \alpha \rceil a \Psi$, $\mu(s) = \lceil \alpha \rceil b \Psi'$, $\mu(r) \neq \mu(s)$

III.2.3. RAMIFICATION ENGENDREE PAR UNE GRAMMAIRE HORS CONTEXTE $G = (V, V_N, P, S)$

C'est une ramification sur $V_N \cup V$ dans laquelle toute famille de prédécesseur $A \in V_N$. Soit telle que $(A \rightarrow \theta) \in P$, $\rho(r) = S$, $\varphi(r) \in V^*$.

Le langage associé est constitué par l'ensemble des mots des feuilles $\varphi(r)$

III.2.4. GRAMMAIRE INTERPRETEE

A une grammaire $G=(V, V_N, P, S)$ nous associons une interprétation I, application de P dans $\mathbb{N}$ telle que $I(A \rightarrow B_1 \ldots B_p) = j$, $1 \leqslant j \leqslant p$

III.2.5. TRANSFORMATION D'UNE RAMIFICATION ENGENDREE PAR UNE GRAMMAIRE INTERPRETEE EN STRUCTURE DE DEPENDANCES SUR V

Soit l'application g telle que :

1) $g(\Lambda) = \Lambda$

2) $g(a) = (a, 1) \quad a \in V$

3) $g(A \times (r_1 + \ldots + r_q)) = (a, j + k - 1) \times (g(r_1) \ldots + g(r_{j-1}) + \beta(g(r_j)) + g(r_{j+1}) \ldots + g(r_q))$

avec $j = I(A \rightarrow \rho(r_1) \ldots \rho(r_q))$

$\rho(r_j) = (a, k)$

et $\beta(r)$ telle que $\beta(c \times r) = r$ (suppression de la racine)

Exemple : Soit une grammaire binaire ; I (p) = 1 ('gauche') ou I (p) = 2 ('droite')

La ramification de la figure 4, compte tenu de l'interprétation, est transformée en celle de la figure 6

$I(S \rightarrow AB) = 2$

$I(A \rightarrow ab) = 1$

$I(B \rightarrow cd) = 2$

Proposition 3 : Le mot des feuilles $\varphi = \lambda \circ g(r)$

(la transformation préserve la notion de phrase associée).

En effet, ceci est vrai pour n (r) = 0, n (r) = 1, car $\varphi(a) = \lambda \circ g(a) = \lambda(a, 1)$

Soit alors $g(r_j) = (a, k) \times (s_1 + \ldots + s_\ell)$

et $g(r) = (a, j+k-1) \times (g(r_1) + \ldots + g(r_{j-1}) + s_1 + \ldots + s_\ell + g(r_{j+1}) + \ldots g(r_q)$

$\lambda \circ g(r) = \lambda \circ g(r_1) \ldots \lambda \circ g(r_{j-1}) \lambda(s_1) \ldots \lambda(s_{k-1}) a \lambda(s_k) \ldots \lambda(s_\ell) \lambda \circ g(r_{j+1}) \ldots \lambda \circ g(r_q)$

$= \lambda \circ g(r_1) \ldots \lambda \circ g(r_{j-1}) \lambda \circ g(r_j) \ldots \lambda \circ g(r_q)$

$= \varphi(r_1) \ldots \varphi(r_q)$ par hypothèse de récurrence

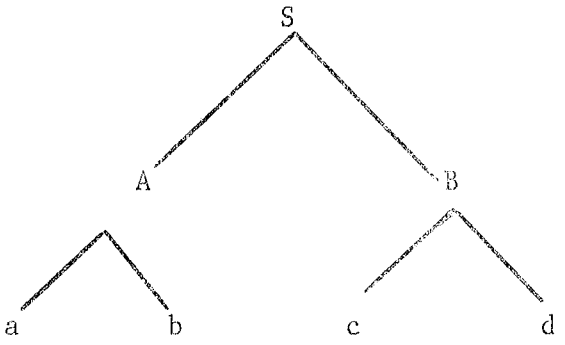


Figure 4

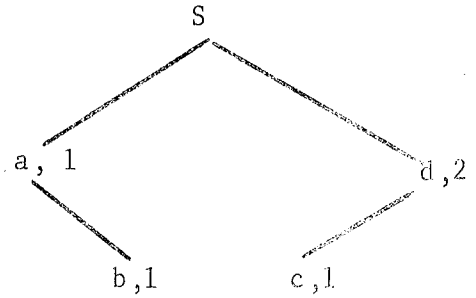


Figure 5

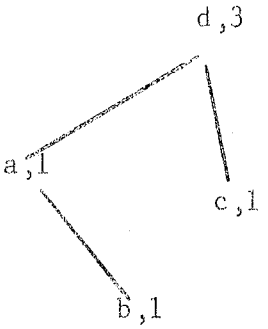


Figure 6

III.2.6. TRANSFORMATION DIRECTE EN FORME PARENTHESEE

Proposition 4 : La fonction $\nu = \mu \circ g$ est définie par

1) $\nu(\Lambda) = \Lambda$

2) $\nu(a) = a$

3) $\nu(A \times (r_1 + \ldots + r_q)) = [\nu(r_1)] \ldots [\nu(r_{j-1})]\ \nu(r_j)\ [\nu(r_{j+1})] \ldots [\nu(r_q)]$

si $j = I(A \rightarrow \rho(r_1) \ldots \rho(r_q))$

Ceci est vrai pour $n(s) = 0$ et $n(s) = 1$, $\mu \circ g(a) = \mu(a, 1) = a$

Soit alors $g(r_j) = (a, k) \times (s_1 + \ldots + s_\ell)$

et $g(r) = (a, j+k-1) \times (g(r_1) + \ldots + g(r_{j-1}) + s_1 + \ldots + s_\ell + g(r_{j+1}) + \ldots + g(r_q))$

$\mu \circ g(r) = [\mu \circ g(r_1)] \ldots [\mu \circ g(r_{j-1})]\ [\mu(s_1)] \ldots a \ldots [\mu(s_\ell)] \ldots [\mu \circ g(r_q)]$

$= [\mu \circ g(r_1)] \ldots [\mu \circ g(r_{j-1})]\ \mu \circ g(r_j)\ [\mu \circ g(r_{j+1})] \ldots [\mu \circ g(r_q)]$

$= [\nu(r_1)] \ldots [\nu(r_{j-1})]\ \nu(r_j)\ [\nu(r_{j+1})] \ldots [\nu(r_q)]$

Exemple : Reprenant les grammaires binaires utilisée en III.2.5. nous noterons

S → (A) B

A → a (b)

B → (c) d

Il vient (a (b)) (c) d

Conséquence : $\varphi = e \circ \nu$ car $\varphi = \lambda \circ g = e \circ \mu\, g = e \circ \nu$

III.3.1. DEFINITION DU TRANSDUCTEUR

Proposition 5 : L'ensemble M des mots $\nu(r)$ associés aux ramifications engendrées par une grammaire à contexte libre est un langage à contexte libre.

Soit L(A) l'ensemble des ramifications de racine A engendrées par la grammaire : $N(A) = \nu(L(A))$

Soit une règle $A \rightarrow B_1 \ldots B_q$, telle que $I(A \rightarrow B_1 \ldots B_q) = j$, et les ramifications de L (A) qui sont de la forme $A \times (r_1 + \ldots + r_q)$ avec $r_i \in L(B_i)$. ν transforme l'ensemble de ces ramifications en l'ensemble :

$$\lceil N(B_1)\rceil \ldots \lceil N(B_{j-1})\rceil\, N(B_j)\, \lceil N(B_{j+1})\rceil \ldots \lceil N(B_q)\rceil$$

et $N(A) = \bigcup \lceil N(B_1)\rceil \ldots [N(B_{j-1})]\, N(B_j)\, [N(B_{j+1})] \ldots [N(B_q)]$

$N(a) = \{a\}$, $a \in V$

Les ensembles N (A) constituent la solution d'un système qui est vérifié par les langages hors contexte ; cette solution est unique si il n'y a aucune règle du type $A \rightarrow \Lambda$ ou $A \rightarrow A'$, $A' \in V_N$; l'on peut toujours se ramener à ce cas. Les ensembles N (A) sont donc des langages hors contexte, et l'on peut les engendrer par une grammaire $G' = (V_1, V_N, P', S)$ telle qu'à toute règle $A \rightarrow B_1 \ldots B_q$ de G corresponde

$$A \rightarrow [B_1][B_2] \ldots [B_{j-1}]\, B_j\, [B_{j+1}] \ldots [B_q]$$

Application : L'ensemble N est reconnu par un automate à pile.

Nous cherchons un transducteur permettant de construire les mots $\beta \in N$ correspondant aux mots α engendrés par G, tels que $\alpha = e(\beta)$, ce que nous pouvons noter $\beta = N \cap e^{-1}(\alpha)$ pour un α donné.

Si M' est l'automate à pile reconnaissant N, il existe un transducteur T produisant N, on le construira en remplaçant toute chaine d'entrée a par e (a) et en produisant e (a).

Ainsi, si $M = (Q, V_1, V_N, \delta, q_0, Z_0, F)$

tel que $\delta(q, a, \gamma) = (q', \gamma')$

Alors $T = (Q, V, V_N, V_1, \pi, q_0, Z_0)$

tel que : $\pi(q, e(a), \gamma) = (q', \gamma', a)$

Nous pouvons facilement déduire l'automate M' de l'automate M reconnaissant le langage L_G : Soit, dans le cas binaire :

Si $(C \rightarrow AB) \in P$

$$M \quad \begin{cases} \delta\ (C^1, \xi, \gamma) = (A^1, C\gamma) \\ \delta\ (A^2, \xi, \gamma) = (B^1, \gamma) \\ \delta\ (B^2, \xi, C\gamma) = (C^2, \gamma) \end{cases}$$

$$M' \quad \begin{cases} \delta'(C^1, (, \gamma) = (A^1, C\gamma) \\ \delta'(A^2,), \gamma) = (B^1, \gamma) \\ \delta'(B^2, \xi, C\gamma) = (C^2, \gamma) \end{cases} \text{ si I = 2 ou } \begin{cases} \delta'(C^1, \xi, \gamma) = (A^1, C\gamma) \text{ si I=1} \\ \delta'(A^2, (, \gamma) = (B^1, \gamma) \\ \delta'(B^2,), C\gamma) = (C^2, \gamma) \end{cases}$$

D'où le transducteur

$$T \quad \begin{cases} \pi\ (C^1, \xi, \gamma) = (A^1, C\gamma, (\,) \text{ si I = 2 ou } \\ \pi\ (A^2, \xi, \gamma) = (B^1, \gamma,)\,) \\ \pi\ (B^2, \xi, C\gamma) = (C^2, \gamma, \xi) \end{cases} \begin{cases} \pi\ (C^1, \xi, \gamma) = (A^1, C\gamma, \xi) \\ \pi\ (A^2, \xi, \gamma) = (B^1, \gamma, (\,) \\ \pi\ (B^2, \xi, C\gamma) = (C^2, \gamma,)\,) \end{cases}$$

III.4. AFFECTATION DES NOMS DE RELATION

Une structure de dépendance munie de noms de relations est une structure sur V x R , R étant l'ensemble des noms de relation, donc une ramification $\widehat{V \times R \times \mathbb{N}}$

Chaque symbole comporte donc un élément du vocabulaire et un nom de relation r ∈ R.

Soit une grammaire G = (V, V_N, P, S) munie d'une interprétation I (p), nous lui associons une structure de dépendance munie de noms de relation R = P U $\{r_o\}$ P étant l'ensemble des productions P de G et r_o étant le nom de la relation 'vide' caractéristique de la racine de la structure de dépendances.

Soit g' la fonction associant une structure de dépendance munie de relations à une structure de constituants :

1) $g'(\Lambda) = \Lambda$

2) $g'(a) = (a, r_0, 1)$

3) $g'(A \times (r_1 + \ldots + r_q) = (a, p', j + k - 1) \times (\gamma(g'(r_1), p) + \ldots$
$+ \gamma(g'(r_{j-1}), p) + \beta(g'(r_j)) + \gamma(g'(r_{j+1}), p$
$+ \ldots + \gamma(g'(r_q), p)$

Dans lequel $\rho(g'(r_j)) = (a, p', k)$, β est le même qu'en III.2.5.

$p = (A \rightarrow \rho(r_1) \ldots \rho(r_q))$ et $j = I(p)$. γ est une application de $V \times P \times \mathbb{N} \times P$ dans $V \times P \times \mathbb{N}$ telle que

$\gamma(\Lambda, p) = \Lambda$

$\gamma((a, i, p') \times r, p) = (a, i, p) \times r$ (γ affecte un nouveau nom à la racine)

Exemple : Reprenons la grammaire binaire et la ramification de la figure 4.

Soient $p_1 = (S \rightarrow AB)$, $p_2 = (A \rightarrow ab)$, $p_3 = (B \rightarrow cd)$

Nous aurons la ramification finale de la figure 7

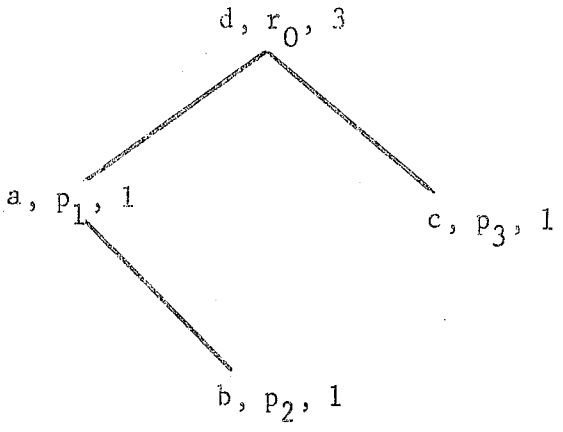


Figure 7

VU

Grenoble, le

Le Président de la Thèse

VU

Grenoble, le

Le Doyen de la Faculté des Sciences

Vu, et permis d'imprimer

Le Recteur de l'Académie de GRENOBLE