Vers une ingénierie de la production de linguiciels : spécification et réalisation d'un prototype de poste de travail linguistique pour la spécification de correspondances structurales

Yongfeng Yan

THESE

présentée par

Yongfeng YAN

pour obtenir le titre de DOCTEUR
de l'UNIVERSITE SCIENTIFIQUE, TECHNOLOGIQUE ET MEDICALE DE GRENOBLE
(arrêté ministériel du 5 juillet 1984)
Spécialité : **informatique**

VERS UNE INGENIERIE DE LA PRODUCTION DE LINGUICIELS

SPECIFICATION ET REALISATION D'UN PROTOTYPE DE POSTE DE TRAVAIL LINGUISTIQUE POUR LA SPECIFICATION DE CORRESPONDANCES STRUCTURALES

Date de soutenance : lundi 29 juin 1987

Composition du jury :

M. Christian BOITET (président)
M. François PECCOUD (directeur)
M. Jacques CHAUCHE (rapporteur)
M. Jacques COURTIN (rapporteur)
M. Patrice POGNAN (rapporteur)
M. Vincent QUINT (examinateur)

THESE

présentée par

Yongfeng YAN

pour obtenir le titre de DOCTEUR
de l'UNIVERSITE SCIENTIFIQUE, TECHNOLOGIQUE ET MEDICALE DE GRENOBLE
(arrêté ministériel du 5 juillet 1984)
Spécialité : **informatique**

VERS UNE INGENIERIE DE LA PRODUCTION DE LINGUICIELS

SPECIFICATION ET REALISATION D'UN PROTOTYPE DE POSTE DE TRAVAIL LINGUISTIQUE POUR LA SPECIFICATION DE CORRESPONDANCES STRUCTURALES

Date de soutenance : lundi 29 juin 1987

Composition du jury :

M. Christian BOITET (président)
M. François PECCOUD (directeur)
M. Jacques CHAUCHE (rapporteur)
M. Jacques COURTIN (rapporteur)
M. Patrice POGNAN (rapporteur)
M. Vincent QUINT (examinateur)

UNIVERSITE SCIENTIFIQUE TECHNOLOGIQUE ET MEDICALE DE GRENOBLE

Président de l'Université :
M. TANCHE

Année Universitaire 1986 - 1987

MEMBRES DU CORPS ENSEIGNANT DE SCIENCES ET DE GEOGRAPHIE

PROFESSEURS DE 1ère Classe

ARNAUD Paul	Chimie Organique
ARVIEU ROBERT	Physique Nucléaire I.S.N.
AUBERT Guy	Physique C.N.R.S
AURIAULT Jean-Louis	Mécanique
AYANT Yves	Physique Approfondie
BARBIER Marie-Jeanne	Electrochimie
BARBIER Jean-Claude	Physique Expérimentale CNRS
BARJON Robert	Physique Nucléaire ISN
BARNOUD Fernand	Boichimie Macromoléculaire Végétale
BARRA Jean-René	Statistiques-Mathématiques Appliquées
BELORISKY Elie	Physique C.E.N.G- D.R.F.
BENZAKEN Claude	Mathématiques Pures
BERNARD Alain	Mathématiques Pures
BERTRANDIAS Françoise	Mathématiques Pures
BERTRANDIAS Jean-Paul	Mathématiques Pures
BILLET Jean	Géographie
BOELHER Jean-Paul	Mécanique
BONNIER Jane Marie	Chimie Générale
BOUCHEZ Robert	Physique Nucléaire ISN
BRAVARD Yves	Géographie
CARLIER Georges	Biologie Végétale
CAUQUIS Georges	Chimie Organique`
CHIBON Pierre	Biologie Animale
COHEN ADDAD Jean-Pierre	Physique
COLIN DE VERDIERE Yves	Mathématiques Pures
CYROT Michel	Physique du Solide
DEBELMAS Jacques	Géologie Générale
DEGRANGE Charles	Zoologie
DELOBEL Claude	Mathématiques Appliquées
DEPORTES Charles	Chimie Minérale
DESRE Pierre	Electrochimie
DOLIQUE Jean-Michel	Physique des Plasmas
DOUCE Rolland	Physiologie Végétale
DUCROS Pierre	Cristallographie
FONTAINE Jean-Marc	Mathématiques Pures
GAGNAIRE Didier	Chimie Physique
GERMAIN Jean-Pierre	Mécanique,
GIRAUD Pierre	Géologie
HICTER Pierre	Chimie
IDELMAN Simon	Physiologie Animale
JANIN Bernard	Géographie
JOLY Jean-René	Mathématiques Pures
KAHANE André, détaché	Physique
KAHANE Josette	Physique
KRAKOWIAK Sacha	Mathématiques Appliquées
KUPKA Yvon	Mathématiques Pures
LAJZEROWICZ Jeanine	Physique
LAJZEROWICZ Joseph	Physique
LAURENT Pierre-Jean	Mathématiques Appliquées
DE LEIRIS Joel	Biologie

LLIBOUTRY Louis	Géophysique
LOISEAUX Jean-Marie	Sciences Nucléaires I.S.N.
MACHE Régis	Physiologie Végétale
MAYNARD Roger	Physique du Solide
MICHEL Robert	Minéralogie et Pétrographie (Géologie)
OMONT Alain	Astrophysique
OZENDA Paul	Botanique (Biologie Végétale)
PAYAN Jean-Jacques	Mathématiques Pures
PEBAY-PEYROULA Jean-Claude	Physique
PERRIAUX Jacques	Géologie
PERRIER Guy	Géophysique
PIERRARD Jean-Marie	Mécanique
PIERRE Jean-Louis	Chimie Organique
RASSAT André	Chimie Systématique
RENARD Michel	Thermodynamique
RINAUDO Marguerite	Chimie CERMAV
ROSSI André	Biologie
SAKAROVITCH Michel	Mathématiques Appliquées
SAXOD Raimard	Biologie Animale
SENGEL Philippe	Biologie Animale
SERGERAERT Francis	Mathématiques Pures
SOUCHIER Bernard	Biologie
SOUTIF Michel	Physique
STUTZ Pierre	Mécanique
VALENTIN Jacques	Physique Nucléaire I.S.N.
VAN CUTSEM Bernard	Mathématiques Appliquées
VIALON Pierre	Géologie

PROFESSEURS de 2ème Classe

ADIBA Michel	Mathématiques Pures
ANTOINE Pierre	Géologie
ARMAND Gilbert	Géographie
BARET Paul	Chimie
BECKER Pierre	Physique
BEGUIN Claude	Chimie Organique
BLANCHI J.Pierre	STAPS
BOITET Christian	Mathématiques Appliquées
BORNAREL Jean	Physique
BRUANDET J.François	Physique
BRUN Gilbert	Biologie
CASTAING Bernard	Physique
CERFF Rudiger	Biologie
CHARDON Michel	Géographie
CHIARAMELLA Yves	Mathématiques Appliquées
COURT Jean	Chimie
DEMAILLY Jean-Pierre	Mathématiques Pures
DENEUVILLE Alain	Physique
DEPASSEL Roger	Mécanique des Fluides
DERRIEN Jacques	Physique
DUFREYNOY Alain	Mathématiques Pures
GASPARD François	Physique
GAUTRON René	Chimie
GENIES Eugène	Chimie
GIDON Maurice	Géologie
GIGNOUX Claude	Sciences Nucléaires
GILLARD Roland	Mathématiques Pures
GIORNI Alain	Sciences Nucléaires
GUIGO Maryse	Géographie
GUMUCHAIN Hervé	Géographie
GUITTON Jacques	Chimie
HACQUES Gérard	Mathématiques Appliquées

HERBIN Jacky	Géographie
HERAULT Jeanny	Physique
JARDON Pierre	Chimie
JOSELEAU Jean-Paul	Biochimie
KERCKHOVE Claude	Géologie
LEBRETON Alain	Mathématiques Appliquées
LONGEQUEUE Nicole	Sciences Nucléaires I.S.N.
LUCAS Robert	Physique
LUNA Domingo	Mathématiques Pures
MANDARON Paul	Biologie
MARTINEZ Francis	Mathématiques Appliquées
MASCLE Georges	Géologie
NEMOZ Alain	Thermodynamique CNRS - CRTBT
OUDET Bruno	Mathématiques Appliquées
PELMONT Jean	Biochimie
PERRIN Claude	Sciences Nucléaires I.S.N.
PFISTER Jean-Claude	Physique du Solide
PIBOULE Michel	Géologie
RAYNAUD Hervé	Mathématiques Appliquées
RIEDIMANN Christine	Mathématiques Pures
ROBERT Gilles	Mathématiques Pures
ROBERT Jean-Bernard	Chimie Physique
SARROT-REYNAULD Jean	Géologie
SAYETAT Françoise	Physique
SERVE Denis	Chimie
STOECKEL Frédéric	Physique
SOUTIF Jeanne	Physique
SCHOLL Pierre-Claude	Mathématiques Appliquées
SUBRA Robert	Chimie
VALLADE Marcel	Physique
VIDAL Michel	Chimie Organique
VIVIAN Robert	Géographie
VOTTERO Philippe	Chimie

MEMBRES DU CORPS ENSEIGNANT DE L' IUT 1

PROFESSEURS de 1ere Classe

BUISSON Roger	Physique IUT 1
DODU Jacques	Mécanique Appliquée IUT 1
NEGRE Robert	Génie Civil IUT 1

PROFESSEURS de 2 ème classe

BOUTHINON Michel	EEA. IUT 1
CHAMBON René	Génie Mécanique IUT 1
CHEHIKIAN Alain	EEA. IUT 1
CHENAVAS Jean	Physique IUT 1
CHOUTEAU Gérard	Physique IUT 1
CONTE René	Physique IUT 1
GOSSE Jean-Pierre	EEA.IUT 1
GROS Yves	Physique IUT 1
KUHN Gérard, (Détaché)	Physique IUT 1
MAZUER Jean	Physique IUT 1
MICHOULIER Jean	Physique IUT 1
MONLLOR Christian	EEA.IUT 1
NOUGARET Marcel	Automatique IUT 1
PEFFEN René	Métallurgie IUT 1
PERARD Jacques	EEA. IUT 1
PERRAUD Robert	Chimie IUT 1
TERRIEZ Jean-Michel	Génie Mécanique IUT 1
TOUZAIN Philippe	Chimie IUT 1
VINCENDON Marc	Chimie IUT 1

MEMBRES DU CORPS ENSEIGNANT DE MEDECINE

PROFESSEURS CLASSE EXEPTIONNELLE ET 1ère CLASSE

AMBLARD Pierre	Dermatologie	C.H.R.G.
AMBROISE-THOMAS Pierre	Parasitologie	C.H.R.G.
BEAUDOING André	Pédiatrie-Puericulture	C.H.R.G.
BEZEZ Henri	Orthopédie-Traumatologie	Hopital SUD
BONNET Jean-Louis	Ophtalmologie	C.H.R.G.
BOUCHET Yves	Anatomie	Faculté La Merci
	Chirurgie Générale et Digestive	C.H.R.G.
BUTEL Jean	Orthopédie-Traumatologie	C.H.R.G.
CHAMPETIER Jean	Anatomie-Topographique et Appliquée	C.H.R.G.
CHARACHON Robert	O.R.L.	C.H.R.G.
COUDERC Pierre	Anatomie-Pathologique	C.H.R.G.
DELORMAS Pierre	Pneumophtisiologique	C.H.R.G.
DENIS Bernard	Cardiologie	C.H.R.G.
GAVEND Michel	Pharmacologie	Faculté La Merci
HOLLARD Daniel	Hématologie	C.H.R.G.
LATREILLE René	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
LE NOC Pierre	Bactériologie-Virologie	C.H.R.G.
MALINAS Yves	Gynécologie et Qbstétrique	C.H.R.G.
MALLION Jean-Michel	Médecine du Travail	C.H.R.G.
MICOUD Max	Clinique Médicale et Maladies Infectieuses	C.H.R.G.
MOURIQUAND Claude	Histologie	Faculté La Merci
PARAMELLE Bernard	Pneumologie	C.H.R.G.
PERRET Jean	Neurologie	C.H.R.G.
RACHAIL Michel	Hépato-Gastro-Entérologie	C.H.R.G.
DE ROUGEMONT Jacques	Neurochirurgie	C.H.R.G.
SARRAZIN Roger	Clinique Chirurgicale	C.H.R.G.
STIEGLITZ Paul	Anestésiologie	C.H.R.G.
TANCHE Maurice	Physiologie	Faculté La Merci
VERAIN André	Biophysique	Faculté La Merci
VIGNAIS Pierre	Biochimie	Faculté La Merci

PROFESSEURS 2ème CLASSE

BACHELOT Yvan	Endocrinologie	C.H.R.G.
BARGE Michel	Neurochirurgie	C.H.R.G.
BENABID Alim Louis	Biophysique	Faculté La Merci
BENSA Jean-Claude	Immunologie	Hopital Sud
BERNARD Pierre	Gynécologie-Obstétrique	C.H.R.G.
BESSARD Germain	Pharmacologie	ABIDJAN
BOLLA Michel	Radiothérapie	C.H.R.G.
BOST Michel	Pédiatrie	C.H.R.G.
BOUCHARLAT Jacques	Psychiatrie Adultes	Hopital Sud
BRAMBILLA Christian	Pneumologie	C.H.R.G.
CHAMBAZ Edmond	Biochimie	C.H.R.G.
CHIROSSEL Jean-Paul	Anatomie-Neurochirurgie	C.H.R.G.
COLOMB Maurice	Immunologie	Hopital Sud
COMET Michel	Biophysique	Faculté La Merci
CONTAMIN Charles	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
CORDONNIER Daniel	Néphrologie	C.H.R.G.
COULOMB Max	Radiologie	C.H.R.G.
CROUZET Guy	Radiologie	C.H.R.G.
DEBRU Jean-Luc	Médecine Interne et Toxicologie	C.H.R.G.
DEMONGEOT Jacques	Biostatistiques et Informatique Médicale	Faculté La Merci

DUPRE Alain	Chirurgie Générale	C.H.R.G.
DYON Jean-François	Chirurgie Infantile	C.H.R.G.
ETERRADOSSI Jacqueline	Physiologie	Faculté La Merci
FAURE Claude	Anatomie et Organogénèse	C.H.R.G.
FAURE Gilbert	Urologie	C.H.R.G.
FOURNET Jacques	Hépato-Gastro-Entérologie	C.H.R.G.
FRANCO Alain	Médecine Interne	C.H.R.G.
GIRARDET Pierre	Anesthésiologie	C.H.R.G.
GUIDICELLI Henri	Chirurgie Générale et Vasculaire	C.H.R.G.
GUIGNIER Michel	Thérapeutique et Réanimation Médicale	C.H.R.G.
HADJIAN Arthur	Biochimie	Faculté La Merci
HALIMI Serge	Endocrinologie et Maladies Métaboliques	C.H.R.G.
HOSTEIN Jean	Hépato-Gastro-Entérologie	C.H.R.G.
HUGONOT Robert	Médecine Interne	C.H.R.G.
JALBERT Pierre	Histologie-Cytogénétique	C.H.R.G.
JUNIEN-LAVILLAUROY Claude	O.R.L.	C.H.R.G.
KOLODIE Lucien	Hématologie Biologique	C.H.R.G.
LETOUBLON Christian	Chirurgie Générale	C.H.R.G.
MACHECOURT Jacques	Cardiologie et Maladies Vasculaires	C.H.R.G.
MAGNIN Robert	Hygiène	C.H.R.G.
MASSOT Christian	Médecine Interne	C.H.R.G.
MOUILLON Michel	Ophtalmologie	C.H.R.G.
PELLAT Jacques	Neurologie	C.H.R.G.
PHELIP Xavier	Rhumatologie	C.H.R.G.
RACINET Claude	Gynécologie	C.H.R.G.
RAMBAUD Pierre	Pédiatrie	C.H.R.G.
RAPHAEL Bernard	Stomatologie	C.H.R.G.
SCHAERER René	Cancérologie	C.H.R.G.
SEIGNEURIN Jean-Marie	Bactériologie-Virologie	Faculté La Merci
SELE Bernard	Cytogénétique	Faculté La Merci
SOTTO Jean-Jacques	Hématologie	C.H.R.G.
STOEBNER Pierre	Anatomie Pathologique	C.H.R.G.
VROUSOS Constantin	Radiothérapie	C.H.R.G.

MEMBRES DU CORPS ENSEIGNANT PHARMACIE

AGNIUS-DELORD Claudine	Physique	Faculté La Tronche
ALARY Josette	Chimie Analytique	Faculté La Tronche
BERIEL Hélène	Physiologie et Pharmacologie	Faculté La Tronche
BOUCHERLE André	Chimie et Toxicologie	Faculté Meylan
CUSSAC Max	Chimie Therapeutique	Faculté La Tronche
DEMENGE Pierre	Pharmacodynamie	Faculté La Tronche
JEANNIN Charles	Pharmacie Galénique	Faculté Meylan
LATURAZE Jean	Biochimie	Faculté La Tronche
LUU DUC Cuong	Chimie Générale	Faculté La Tronche
MARIOTTE Anne-Marie	Pharmacognosie	Faculté La Tronche
MARZIN Daniel	Toxicologie	Faculté Meylan
RENAUDET Jacqueline	Bactériologie	Faculté La Tronche
ROCHAT Jacques	Hygiéne et Hydrologie	Faculté La Tronche
SEIGLE-MURANDI Françoise	Botanique et Cryptogamie	Faculté Meylan
VERAIN Alice	Pharmacie Galénique	Faculté Meylan

REMERCIEMENTS

Les travaux rapportés dans cette thèse ont été menés au sein du laboratoire GETA (Groupe d'Etudes pour la Traduction Automatique), où j'ai été accueilli par le professeur Bernard Vauquois qui m'a fait l'honneur de prendre en charge la direction de ma thèse. Je regrette de ne pas avoir eu le temps de lui exprimer ma profonde reconnaissance.

Je tiens à remercier

Monsieur François PECCOUD, professeur à l'Université II de Grenoble, qui a assuré la suite de la direction de mes travaux, et m'a prodigué les meilleurs conseils pour les faire aboutir.

Monsieur Christian BOITET, professeur à l'USTMG et directeur du GETA, qui m'a encouragé tout au long de cette thèse en m'apportant ses critiques et ses réflexions et m'a fait l'honneur de présider le jury.

Messieurs Jacques CHAUCHE, professeur à l'IUT du Havre, Jacques COURTIN, professeur à l'Université II de Grenoble, Patrice POGNAN, professeur à l'Université de Bordeaux et à l'INALCO, et Vincent QUINT, directeur de recherche à l'INRIA, qui ont accepté d'étudier ce travail et d'y apporter leurs suggestions.

Je voudrais enfin remercier tous les membres du GETA et de la société B'VITAL pour les conseils qu'ils m'ont prodigués et l'accueil que j'ai reçu, Rémi ZAJAC et ZAHARIN pour les discussions constructives que j'ai pu avoir avec eux, ainsi que Nadine CHRISTIN et Christine MAIDA pour les corrections et la mise en page de la présente thèse. Enfin, je remercie Dorien LEVENBACH et Elizabeth GUILBAUD, qui n'ont pas hésité à utiliser le prototype que j'ai réalisé, et dont les remarques ont été des plus judicieuses.

-o-o-o-o-o-o-o-o-o-o-

AVANT PROPOS

A mon arrivée au GETA (Groupe d'Etudes pour la Traduction Automatique), je voulais réaliser un prototype de système de Traduction Assistée par Ordinateur (TAO) sur le chinois [Yan 83]. Après avoir étudié le système ARIANE-78, je me suis intéressé aux nouvelles approches en TAO, et surtout à celles menant à des systèmes de TAO de "troisième génération" [Wilks 75, et Schank 75]. Mais je me suis rendu compte qu'une étude approfondie du chinois était nécessaire, et qu'un outil de spécifications pour la description des langues était fortement souhaitable.

N'étant pas linguiste de formation, je me suis tourné vers le deuxième point, et plus précisément vers la spécification et la réalisation d'un environnement pour le développement de Grammaires Statiques de Correspondances Structurales (GSCS), utilisées en particulier pour spécifier des "grammaires dynamiques" utilisées en TAO.

L'idée des GSCS a d'abord été étudiée par S. Chappuy, sous la direction de B.Vauquois, vers 1982-1983 [Chappuy 83, Vauquois & Chappuy 85]. Il s'agit de décrire des langages d'une manière naturelle et cependant formalisée, en définissant une relation entre les chaînes et les structures multiniveaux de la langue utilisées comme structures interfaces dans des systèmes de TAO à "transfert".

Afin qu'un système informatique puisse gérer (intelligemment) des GSCS, il faut que les GSCS soient lisibles par la machine. A l'époque des premières études, aucune étude formelle complète sur la syntaxe ni la sémantique des GSCS n'avait été faite. Nous avons donc commencé par étudier le formalisme des GSCS. Après plus d'une année d'études sur le formalisme et sur sa sémantique usuelle, plus intuitive que mathématique, nous avons abouti à un langage, baptisé LSCS (Langage de Spécification de Correspondances Structurales), et à une première maquette d'environnement pour les GSCS (ESCS) [Yan 86].

Cette thèse est consacrée à l'ingénierie de la production de linguiciels. L'introduction passe en revue les principales techniques utilisées en TAO, justifie l'importance des GSCS, et conclut sur la nécessité d'une approche génie logiciel. Le chapitre I présente les GSCS au moyen d'une petite GSCS pour les nombres en chinois. Le chapitre II examine, du point de vue du génie logiciel, les tâches importantes pour la réalisation d'un système de TAO. Le chapitre III présente la première version d'un poste de travail linguistique facilitant certaines tâches examinées dans le chapitre précédent. La conclusion présente le bilan de ce qui a été réalisé, les perspectives de développement du prototype, et quelques idées sur ce que pourrait être le poste de travail linguistique du futur. On trouvera en annexe une définition de la structure des Bases Linguicielles (BSCS) et la syntaxe du LSCS.

TABLE DES MATIERES

INTRODUCTION

Cette introduction commence par donner un bref aperçu sur l'histoire de la TAO, puis présente l'approche en TAO du GETA, et le système ARIANE. Enfin, l'importance de la spécification et des outils correspondants est mise en relief, justifiant ainsi le recours aux techniques issues du génie logiciel.

1. TAO en général

L'histoire de la TAO est presque aussi ancienne que celle de l'informatique. On classe les systèmes de TAO en trois générations, selon les techniques utilisées, dont nous allons rappeler les principes.

1.1. Etat de l'art de la TAO

1.1.1. Première génération

L'idée d'utiliser l'ordinateur pour la traduction automatique est apparue vers 1945 (memorandum de Warren Weaver). La première démonstration a eu lieu en 1954. Il s'agissait du système de Georgetown, caractéristique de la première génération, dont les systèmes SYSTRAN présentent aujourd'hui des représentants évolués.

Un système de ce type est bilingue et fonctionne grâce à plusieurs dictionnaires et à un ensemble de programmes. Il n'est fondé ni sur des théories linguistiques ni sur des théories de compilation des langages formels (qui n'existaient pas en 1950-1962). Le processus de traduction consiste essentiellement à remplacer des mots de la langue source par ceux de la langue cible selon les dictionnaires, et en quelques arrangements locaux effectués par des programmes *ad hoc*.

L'aspect purement bilingue de ces systèmes exige le recensement des phénomènes linguistiques les plus intéressants, sur un couple de langues, et leur codification sous forme de dictionnaires et de programmes. Peu à peu, l'accumulation des phénomènes linguistiques devient si importante que l'insertion de nouveaux phénomènes dans les programmes de plus en plus complexes devient pratiquement impossible. Par conséquent, le niveau de qualité arrive assez vite à sa limite.

Dans ces systèmes, les programmes linguistiques sont directement écrits dans les langages de programmation générale, souvent de bas niveau, et il n'y a aucun logiciel spécial pour la programmation linguistique.

Malgré la qualité déficiente de la traduction, ces systèmes satisfont certains utilisateurs, qui préfèrent de mauvaises traductions à rien du tout, en particulier pour des besoins de renseignement ou de veille scientifique et technique.

1.1.2. Deuxième génération

On a cherché une autre approche pour améliorer la qualité de la TAO et on a abouti aux méthodes de la deuxième génération, qui se sont développées pendant la période de 1960-1970. La caractéristique initiale commune à ces systèmes est la division du processus de traduction en trois phases logiques : analyse, transfert et génération, illustrées par la figure ci-après.

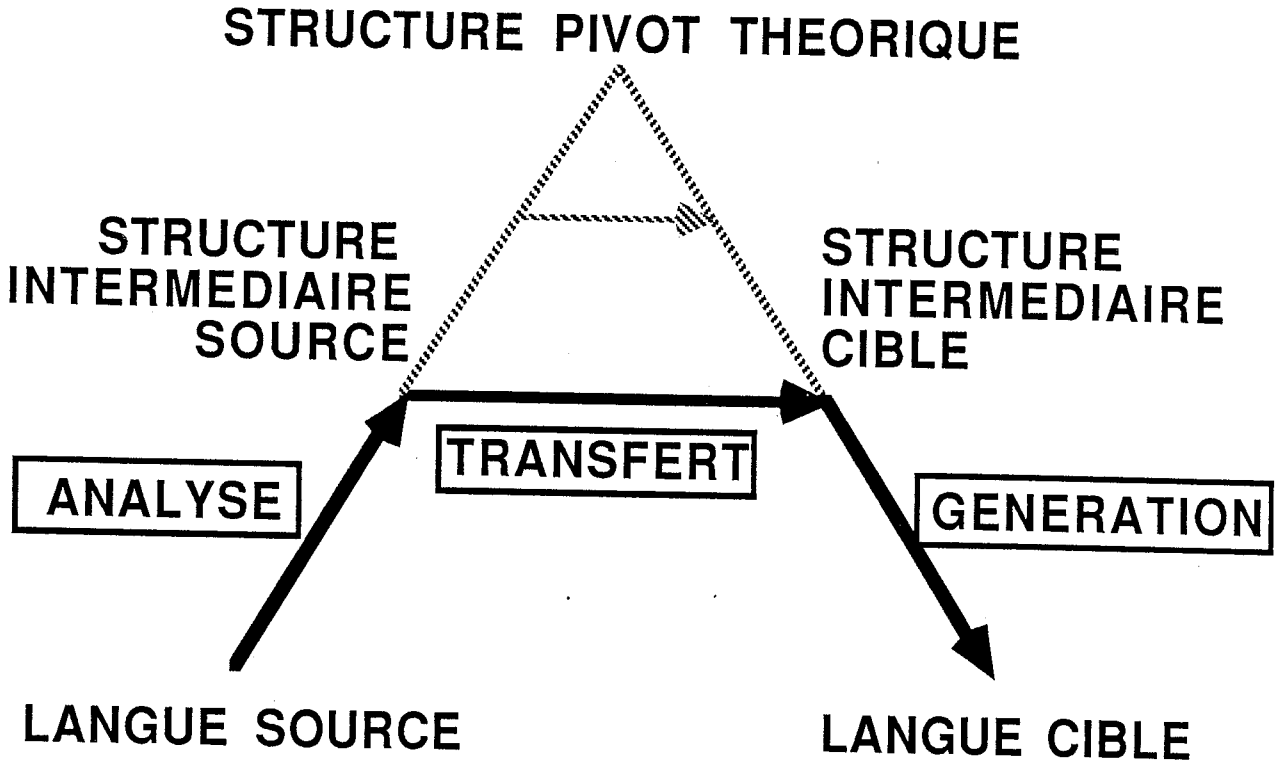


Figure 1 : Les trois phases de la TAO de deuxième génération.

La sortie de l'analyse est un descripteur structural du texte d'entrée, ce dernier étant transformé en un descripteur structural équivalent dans la langue cible par la phase de transfert. Ce descripteur cible est alors transformé en texte de sortie par la phase de génération.

Dans cette division, la première et la troisième phase ne concernent qu'une langue chacune, seule la deuxième fait intervenir le couple de langues. Plus l'analyse est profonde, plus la distance entre les deux descripteurs structuraux est courte. Nous pourrions imaginer un niveau pivot où ils seraient identiques.

Pendant cette période se sont développées les théories de la compilation et des langages formels, qui offrent des méthodes systématiques pour construire ces systèmes de TAO. Les algorithmes, à cette époque, fonctionnaient de façon combinatoire, c'est-à-dire que les analyseurs essayaient toutes les possibilités, quel que soit le texte. C'est le premier stade de la deuxième génération.

A partir des années 70, on a recherché des algorithmes plus puissants, fondés sur un modèle de transduction, ce qui donne la possibilité de contrôler les processus par la définition de stratégies et d'heuristiques. Le concept de grammaires modulaires a été mis en oeuvre, pour faciliter la construction et la maintenance de grosses grammaires.

Des outils logiciels ont été développés pour réaliser de tels transducteurs, tels que les SYSTEMES-Q de A.Colmerauer [Colmerauer 70], les ATN de W.Woods [Woods 70], et ATEF [Chauché 72], CETA [Chauché 75], puis ROBRA [Boitet, Guillaume et Quézel-Ambrunaz 78 et GETA-DSE1 82] du GETA. Ces systèmes permettent la séparation des données linguistiques (principalement des dictionnaires et des grammaires) et des programmes (compilateurs et moteurs des transducteurs). C'est une autre caractéristique distinguant bien ces systèmes de ceux de la première génération.

Cette séparation permet d'utiliser le même outil logiciel pour toutes les langues, et la division du processus de traduction en trois phases est bien adaptée à la traduction multilingue.

Toutefois, dans un système de deuxième génération, la sémantique n'est exprimée que par des traits sémantiques qui sont analogues à des traits grammaticaux. Le problème des ambiguïtés est souvent difficile à résoudre, à cause du manque de critères linguistiques ou sémantiques pertinents. Ce type de systèmes ne peut comprendre "explicitement" le texte à traduire.

1.1.3. Troisième génération

Le principe des systèmes de troisième génération est de viser une compréhension explicite du texte source :

- le texte source est analysé et transformé en une représentation conceptuelle indépendante de la langue, il s'agit d'une *interprétation* (au sens logique) dans un modèle formalisé ;
- à l'aide de divers mécanismes d'inférence, on utilise des connaissances extra-linguistiques ainsi que le contexte pour enrichir la représentation à l'aide de l'information qui était implicite dans le texte source ;
- à partir de la structure obtenue, on produit le texte dans la langue cible.

Il n'existe aujourd'hui aucun système complet de troisième génération. Les études et les réalisations actuelles ne sont que des approches partielles de certains aspects du problème, et cela pour des textes concernant des domaines très restreints. La difficulté principale est de déterminer le niveau de compréhension nécessaire (et suffisant), et de trouver comment l'obtenir à partir du texte.

1.1.4. Perspectives

Les systèmes de première génération tournent très vite, et ont des gros dictionnaires. Mais quel que soit l'effort de développement consenti, ils ne peuvent dépasser les limites inhérentes à leur conception. Par conséquent, sauf s'il s'agit de sous-langages artificiels ou quasi artificiels, on ne peut les utiliser que pour de l'acquisition d'information, et pas pour de la diffusion : avec de l'entraînement, les traductions sont lisibles mais pas révisables (il vaut très souvent mieux retraduire à la main que de tenter de corriger).

Les systèmes de troisième génération sont à l'étude. Il pourra d'ailleurs s'agir, soit de systèmes totalement liés à un domaine d'expertise, soit de systèmes interactifs, soit du couplage de systèmes de deuxième génération avec des systèmes experts (ou de simples bases de connaissances).

Malgré leurs limites, les techniques de deuxième génération permettent d'ores et déjà de construire des systèmes industriels de TAO économiquement rentables. D'où les deux axes de recherche principaux en TAO :

1. **Qualité :** découvrir de nouvelles méthodes pouvant améliorer la qualité de la traduction (études linguistiques, intégration de techniques d'intelligence artificielle) ;

2. **Rentabilité :** développer les techniques actuellement connues afin de construire des systèmes pouvant remplacer partiellement les traducteurs humains. Les exemples de ces derniers sont TAUM-METEO [Chandioux 78, Chevalier &al 78, Isabelle 80, Isabelle 84], METAL [Slocum 84], le projet MU [Nagao et al. 85] et CALLIOPE-AERO (PN-TAO français [Boitet 86]).

1.2. Approche du GETA

L'approche du GETA est principalement fondée sur les techniques de deuxième génération, avec quelques points originaux :

1) Structure intermédiaire multiniveau,
2) Transducteurs au lieu d'analyseurs,
3) Utilisation de LSPLs (Langages Spécialisés pour la Programmation Linguistique).

En plus, l'unité de traduction n'est pas réduite à la phrase. Les dictionnaires sont organisés autour de la notion d'unité lexicale (famille dérivationnelle de lemmes).

1.2.1. Structure multiniveau

Les niveaux suivants sont distingués : morphologique, syntagmatique, syntaxique, logique, sémantique et pragmatique. Il est utile de garder les informations des différents niveaux calculables, parce que d'une part, elles sont complémentaires les unes des autres et que, d'autre part, si l'on ne peut pas atteindre un niveau plus haut, les analyses des niveaux plus bas peuvent donner tout de même une traduction servant alors de "filet de sécurité".

L'idée de structure intermédiaire multiniveau [Vauquois 78] consiste à représenter dans un seul descripteur structural d'un texte les informations de plusieurs niveaux. Actuellement, les informations sont largement exploitées aux niveaux morphologique, syntagmatique, syntaxique, logique et sémantique (traits et relations). Le niveau pragmatique n'est pas abordé.

1.2.2. Transducteurs plutôt qu'analyseurs

Le choix de transducteurs, plutôt que d'analyseurs, s'appuie sur le fait que la structure produite par un analyseur est un sous-produit du processus d'acceptation, puisque le rôle de l'analyseur est d'accepter ou de refuser son entrée : en cas d'échec, rien n'est produit.

Or, malheureusement, les grammaires ne sont jamais parfaites et les textes d'entrée sont souvent incorrects par rapport aux grammaires ou même à l'usage courant. Un transducteur peut traiter des entrées imparfaites et donner un résultat dans tous les cas.

Dans le cas d'un analyseur, la grammaire impose la structure obtenue, alors que, par transduction, on définit les structures d'entrée et de sortie. Deux autres arguments en faveur des transducteurs sont les possibilités de modularité et de programmation heuristique.

1.2.3. Utilisation des LSPLs

Du point de vue mathématique ou informatique, le processus de la traduction est une séquence de transformations d'arbres dont la forme et la décoration codent les informations multiniveau. Le système ARIANE du GETA offre quatre LSPLs spécialement conçus pour ce processus.

1.3. Développement d'un système TAO sous ARIANE

Après une étude de faisabilité, le développement d'un système de TAO contient les étapes suivantes :

1. **Préparation des données linguistiques :** choix et étude comparative des langues source et cible, étude des corpus en identifiant les phénomènes présents, etc.

2. **Formalisation linguistique :** choix de la théorie linguistique et de la représentation interne des données, et spécification de la correspondance entre le texte et la représentation interne.

3. **Programmation linguistique :** transformation de la formalisation en un programme, écrit dans un ou plusieurs LSPLs.

4. **Intégration et validation du système :** enchaînement des modules et test du système sur des corpus typiques (jeu d'essai).

Durant l'exploitation et la maintenance du système, il peut y avoir des modifications du système à cause d'erreurs survenues au cours de l'exécution, de nouveaux phénomènes linguistiques rencontrés, de demandes d'amélioration de traitement , etc. Ces modifications peuvent provoquer des retours en arrière dans toutes les étapes précédentes.

1.3.1. Fonctions d'ARIANE

Le système ARIANE est un système destiné à développer des programmes de traduction de documents écrits dans diverses langues naturelles. Mis au point au GETA en 1978, il appartient à la deuxième génération, bien que beaucoup d'améliorations lui aient été apportées. La version 78.4 d'ARIANE est actuellement en service, la version 85.2 devant lui succéder bientôt.

Le système ARIANE est un environnement complet de programmation pour la TAO. Il comporte quatre langages spécialisés pour la programmation linguistique, un moniteur interactif et un ensemble de programmes utilitaires facilitant la mise au point et l'exploitation de programmes linguistiques.

Le moniteur est l'interface entre les utilisateurs et tout le système. Il dialogue avec les utilisateurs en demandant des paramètres et enchaîne les programmes du système en réalisant des tâches complexes en fonction des paramètres obtenus. Il possède diverses fonctions permettant aux utilisateurs de :

- préparer des textes à traduire et des données linguistiques,
- compiler des données linguistiques et les exécuter sur des textes indiqués,
- réviser des traductions en affichant sur l'écran le texte source, le texte traduit et des dictionnaires,
- imprimer ces textes en les formatant.

1.3.2. Processus de traduction en ARIANE-78.4

Le processus de traduction d'une langue source vers une langue cible est divisé en trois étapes logiques: analyse, transfert et génération, chacune étant décomposée en au moins deux phases successives, comme l'illustre le diagramme ci-dessous. Les phases optionnelles sont marquées du signe "-".

Le texte d'entrée est soumis à l'analyseur morphologique (AM), écrit en ATEF, qui produit comme résultat un arbre décoré.

Ce résultat est ensuite transformé en un autre arbre (descripteur multiniveau) par l'analyseur structural (AS), écrit en ROBRA.

Le transfert lexical (TL) est écrit en EXPANS. Il transforme l'arbre, au moyen d'une consultation de dictionnaires bilingues multichoix. Le traitement des polysémies et des tournures complexes est possible.

Le transfert structural (TS) traite les différences structurales entre les deux langues, en rendant la structure linguistique intermédiaire conforme à la nature de la langue cible. Cette phase finit également une partie du TL (choix d'équivalents en fonction d'un contexte étendu, traitement des tournures "prédites" par le TL), et complète éventuellement les informations non trouvées par l'AS et nécessaires pour la génération.

La génération syntaxique (GS) calcule, à partir du descripteur produit par le transfert, une structure de surface qui est une paraphrase possible (contrainte par les impératifs de la traduction) du texte. Concrètement, elle recalcule les fonctions, et les classes syntaxiques à partir des relations logiques et sémantiques, en respectant les ordres éventuels codés dans la structure par le transfert, et en cherchant à suivre les "conseils" du transfert. Elle s'occupe aussi, bien sûr, de l'ordre des mots, de la fabrication des mots outils absents, et de toutes les questions liées à l'actualisation.

La génération morphologique (GM) travaille sur les feuilles de l'arbre produit par la GS, et produit des formes morphologiques correctes constituant un texte.

Chaque étape est écrite dans un LSPL, réalisé à l'aide d'un compilateur et d'un "moteur" interprétant les tables produites par le compilateur.

PHASE LOGIQUE	PHASE PHYSIQUE	+/-	LANGAGE (LSPL)	FICHIERS LINGUICIELS TYPE (NOMBRE)
ANALYSE	Analyse Morphologique AM	+	ATEF	Variables (2) Formats (3) Grammaire (1 à 7) Dictionnaire (1 à 7)
	Complément Lexical AX	-	EXPANS	Variables (1) Formats (2) Procédures (1) Dictionnaire (1 à 7)
	Complément Lexical AY	-	EXPANS	Comme AX
	Analyse Structurale AS	+	ROBRA	Variables (1) Formats (1) Procédures (1) Grammaire (1 à 7)
TRANSFERT	Transfert Lexical TL	+	EXPANS	Comme AX
	Complément Lexical TX	-	EXPANS	Comme AX
	Transfert Structural TS	+	ROBRA	Comme AS
	Complément Lexical TY	-	EXPANS	Comme AX
GENERATION	Complément Lexical GX	-	EXPANS	Comme AX
	Génération Structurale GS	+	ROBRA	Comme AS
	Complément Lexical GY	-	EXPANS	Comme AX
	Génération Morphologique GM	+	SYGMOR	Variables (1) Formats (1) Conditions (1) Grammaire (1 à 10) Dictionnaire (1 à 7)

Figure 2 : Phases de la traduction en ARIANE-85.2.

(repris de [Boitet 86], avec l'aimable autorisation de l'auteur)

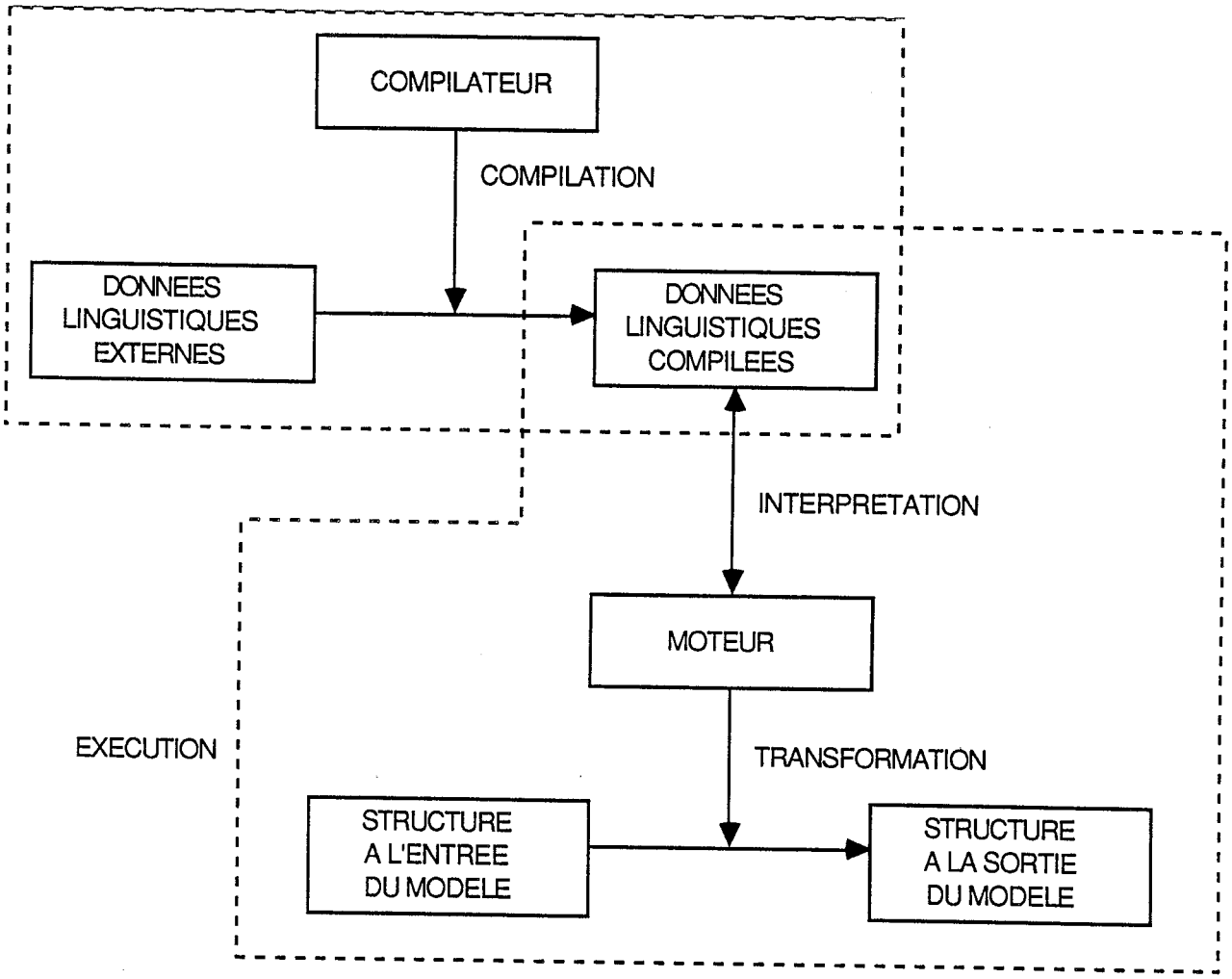


Figure 3 : une phase de la TA.

1.4. Caractéristiques de la TAO

1.4.1. Masse d'informations

Un système de TAO nécessite une grande quantité d'informations. Ces informations sont composées :

- *de données linguistiques :* des grammaires avec des centaines de règles, des dictionnaires de dizaines de milliers d'unités lexicales, correspondant à des centaines de milliers de mots.
- *de programmes :* le système ARIANE contient environ 200.000 lignes source (en assembleur, PL 360, EXEC 2/XEDIT, PL/I et PASCAL).
- *de textes :* un système de TAO opérationnel doit traduire un grand nombre de pages par jour, pour être économiquement viable.
- *de documents :* les spécifications, les manuels d'utilisation et de maintenance sont très volumineux.

1.4.2. Interdisciplinarité

La réalisation d'un système de TAO nécessite des compétences en linguistique, en lexicographie, en informatique et en mathématiques. Il est difficile sinon impossible d'exiger de quiconque une compétence dans tous ces domaines. Des méthodes et des outils permettant une organisation modulaire où chacun s'occupe de sa spécialité sont donc essentiels.

1.4.3. Méthode empirique

La TAO est encore beaucoup plus technique que scientifique. Aujourd'hui, personne ne sait faire traduire l'ordinateur parfaitement, même si le temps et la mémoire ne sont pas limités. Bien que la qualité de la traduction soit difficile à juger, il paraît évident qu'aucun ordinateur ne traduit mieux qu'un traducteur humain. Par contre, dans d'autre domaines de l'intelligence artificielle, par exemple le jeu des échecs, des programmes peuvent déjà battre des joueurs professionnels, et on connaît des algorithmes qui peuvent toujours gagner si la mémoire et le temps sont suffisants.

La cause des problèmes est que la langue est vivante. L'entrée d'un système TAO est imprévisible et intrinsèquement ambiguë.

2. Génie logiciel et TAO

Le génie logiciel a pour but de répondre à la crise du logiciel provoquée par les problèmes suivants: la faisabilité, les échéanciers, la qualité et le coût du logiciel. Il s'agit de l'**application pratique de connaissances scientifiques pour la production de logiciels de qualité industrielle** [Boehn 76, Pressman 82, Krakowiak 86, et GEC 86]. Ici, le logiciel comprend non seulement le programme, mais aussi les données et la documentation permettant le développement, l'exploitation et la maintenance d'un système informatique.

2.1. Principes du génie logiciel

Nous pouvons résumer les principes du génie logiciel en deux mots : méthodologie et outils informatisés.

La méthodologie se traduit par :

- **des bornes tout au long du cycle de vie** :
  en divisant un gros travail en tâches plus simples, les phases du cycle de vie donnent des points de repère permettant de vérifier et de contrôler l'avancement du projet. La production d'un système est un processus de découverte et de résolution des problèmes. Le système part d'idées floues et devient de plus en plus concret au fur et à mesure du développement. La vérification et la révision à la fin de chaque étape et la limitation du retour en arrière, niveau par niveau dans le cycle, évitent la modification d'une partie sans mise à jour du reste.

- **la documentation normalisée à chaque borne :**
  à la fin de chaque phase du cycle de vie, un document normalisé doit être produit. Ce document est aussi bien le résultat de l'étape courante que le point de départ pour l'étape suivante. Il permet de garder la trace du système à construire.

- **l'utilisation de méthodes bien définies**
  comme la programmation structurée, HIPO [Stay 76] et PDL [Caine et Gordon 75] pour la conception, le plan de test, etc.

La méthodologie est mise en oeuvre par des outils automatisés. Il est difficile de suivre une méthode systématique sans un outil pertinent à cause de travaux apparemment triviaux et redondants, cependant importants. Toutes les phases du cycle de vie ont besoin d'outils informatisés.

2.2. Importance de la spécification et outils correspondants en TAO

Un système de TAO est un logiciel de taille industrielle, destiné à l'application linguistique, donc un *linguiciel*. Le passage des techniques de l'état de l'art en TAO vers une ingénierie de la production de linguiciels fait naturellement appel aux techniques du génie logiciel.

Le terme logiciel a été pendant longtemps synonyme de programme. La méthodologie ainsi que les outils du génie logiciel étaient donc concentrés sur la programmation. On s'est rendu compte de l'importance des autres phases et surtout de celle de la spécification [Caplain 78]. Des outils ont été produits pour la spécification de programmes généraux SADT [Ross, Schoman 77], PSL/PSA [Teichroew et Hershey 77], etc.. Le problème de la spécification est aussi important, sinon plus, pour les linguiciels que pour les logiciels.

C'est pourquoi, dans cette thèse, nous nous intéressons à l'ingénierie en TAO en général et à la spécification en particulier.

Dans le chapitre I, nous rappelons le formalisme de GSCS qui a été étudié dans deux thèses [Zaharin 86] et [Zajac 86], et présentons notre étude sur ce formalisme de spécification d'une manière informelle et pédagogique à travers une GSCS du chinois.

Dans le chapitre II, en analysant le projet national français de TAO, nous justifions la nécessité d'une approche inspirée du génie logiciel, nous identifions les besoins de chaque phase d'un linguiciel et nous donnons les principes d'une méthodologie.

Dans le chapitre III, nous présentons notre poste de travail linguistique en illustrant comment réaliser des tâches décrites selon la méthodologie du chapitre II.

CHAPITRE I RAPPEL SUR LES GSCS

Dans ce chapitre, nous discutons d'abord des notions de grammaire. Nous présentons ensuite les idées fondamentales des GSCS et donnons un aperçu général d'une GSCS à travers un exemple traitant les groupes cardinaux du chinois.

1. Notions de grammaire

A partir du 19ème siècle, le terme grammaire a été utilisé pour dénoter la connaissance scientifique d'une langue ou d'un ensemble de langues (excluant en général l'étude du vocabulaire) [Larousse 80]. Une grammaire est composée d'un ensemble de règles générales définissant la correction de la langue écrite ou parlée, ainsi que des exceptions à ces règles.

En TALN (traitement automatique des langues naturelles), on peut distinguer trois sens du terme grammaire, selon qu'il s'agit de grammaires traditionnelles, de grammaires "statiques" ou de grammaires "dynamiques".

1.1. Grammaire traditionnelle

C'est le sens originel du terme. Les **grammaires traditionnelles** sont toujours écrites en langue naturelle sans aucun formalisme particulier. Elles visent à décrire les divers phénomènes linguistiques et à en dégager les lois. Elles sont très utiles pour la mise en oeuvre de systèmes de TAO, par exemple, la "grammaire de l'Académie d'URSS" est utilisée de façon intensive au GETA, pour le travail sur l'analyseur du russe du système russe-français.

1.2. Grammaire statique

En informatique, le terme grammaire peut désigner un texte écrit dans un certain formalisme, et permettant de décrire un langage : ensemble d'entités respectant certaines contraintes. Ces entités peuvent être les phrases ou les mots corrects d'une langue, les programmes d'un langage de programmation ou une structure de données.

Comme exemples de tels formalismes, on peut citer les grammaires syntagmatiques de Chomsky [Chomsky 57], les GPSGs (Generalized Phrase Structure Grammars) [Gazdar, Pullum 82], les grammaires d'unification [Kay 82] et les grammaires fonctionnelles d'unification [Kay 84], les GSCS (Grammaires Statiques de Correspondances Structurales [Vauquois, Chappuy 85] etc..

Ces grammaires ne définissent pas seulement un langage, elles le décrivent, au sens où elles associent un ou plusieurs descripteurs à chaque élément du langage. Nous les appelons **grammaires statiques,** parce qu'elles donnent une description déclarative du langage considéré, sans préciser le processus dynamique de calcul.

1.3. Grammaire dynamique

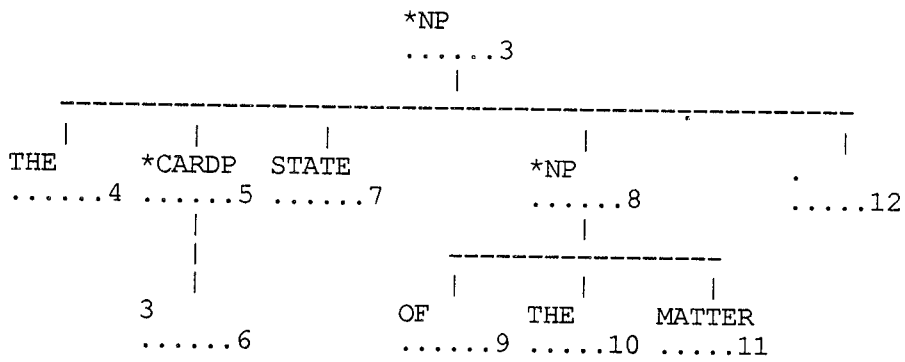
Nous appelons **Grammaires Dynamiques (GD)** les programmes permettant d'obtenir les résultats spécifiés (ou décrits) par des grammaires statiques. Elles doivent posséder une structure de données et un algorithme de calcul efficace (stratégie de contrôle) et sont écrites dans un langage spécialisé pour la programmation linguistique (LSPL), c'est-à-dire fondé sur des règles de production, analogues aux instructions des langages algorithmiques classiques.

Par analogie avec la séparation entre un programme et sa spécification, une distinction nette entre les grammaires statiques et les grammaires dynamiques est très importante.

2. Grammaire Statique de Correspondances Structurales (GSCS)

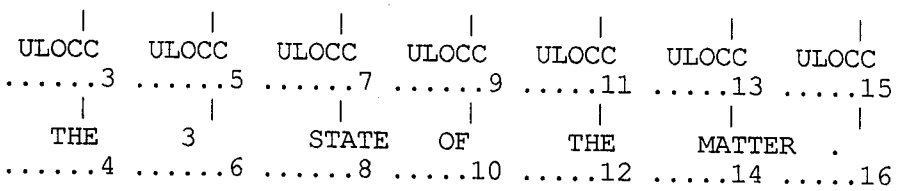
Les GSCS ont pour but d'établir les correspondances entre les chaînes valides des langues et leurs représentations structurales. Elles peuvent être utilisées aussi bien en analyse qu'en génération. Elles ont été étudiées d'abord par S.Chappuy (1983) sous la direction de B.Vauquois [Vauquois & Chappuy 1985], puis par Y.Zaharin (1986) et R.Zajac (1986). Des GSCS ont été écrites : pour le français et l'anglais dans le projet ESOPE [GETA 83] et le PN-TAO [Boitet 86] en France et pour le malais [Tong 86] ; des GSCS sont en cours de développement pour le russe, l'allemand, le néerlandais, le chinois et le thaï.

Les représentations structurales sont des descripteurs arborescents [Vauquois 1978, Zaharin 1987]. Par exemple, la représentation de la chaîne valide "The three states of the matter." est l'arbre décoré donné dans la figure suivante. Cet arbre est produit par une grammaire dynamique spécifiée par la GSCS BXFXeng [GETA 83] à partir de la chaîne ULOCC (Unité Lexicale de l'OCCurence) sortie par l'analyse morphologique (voir la figure suivante). Cet arbre contient des informations multiniveaux, allant du niveau morphologique jusqu'au niveau logico-sémantique.



SOMMET 3 ' ': UL('*NP'),K(NP),CAT(N),SUBN(CN),NUM(PLU),SEM(ABST),VL1(N).
SOMMET 4 '*THE': UL('THE'),SF(DES),TYPOG(CAP),CAT(D),SUBD(ART),NUM(PLU).
SOMMET 5 ' ':UL('*CARDP'),RS(QUAL),K(CARDP),SF(ATG),CAT(A),SUBA(CARD),NUM(PLU).
SOMMET 6 'THREE': UL('3'),SF(GOV),CAT(A),SUBA(CARD),NUM(PLU).
SOMMET 7 'STATES': UL('STATE'),SF(GOV),CAT(N),SUBN(CN),NUM(PLU),SEM(ABST).
SOMMET 8 ' ': UL('*NP'), RS(QUAL),UNSAFE(RS),K(NP),
SF(COMP),CAT(N),SUBN(CN),NUM(SIN),SEM(CONC),SEMCO(SUBST), VL1(OF).
SOMMET 9 'OF': UL('OF'),RS(QUAL),UNSAFE(RS),SF(REG),CAT(S),SUBS(PREP),VL1(OF).
SOMMET 10 'THE': UL('THE'),SF(DES),CAT(D),SUBD(ART),NUM(SIN).
SOMMET 11 'MATTER':UL('MATTER'), SF(GOV),CAT(N),SUBN(CN),NUM(SIN),SEM(CONC),SEMCO(SUBST).
SOMMET 12 '.': UL('.'),CAT(P).

Sortie de l'analyse structurale pour "the three states of the matter."

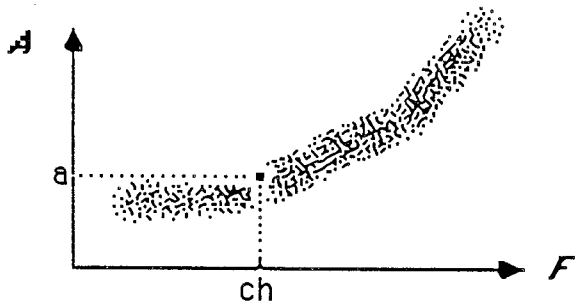


SOMMET 3 ' ': UL('ULOCC').
SOMMET 4 '*THE': UL('THE'),STATE(1),TYPOG(CAP),CAT(D),SUBD(ART),NUM(SIN,PLU).
SOMMET 5 ' ': UL('ULOCC').
SOMMET 6 'THREE': UL('3'),STATE(1),CAT(A),SUBA(CARD),NUM(PLU).
SOMMET 7 ' ': UL('ULOCC').
SOMMET 8 'STATES':UL('STATE'), STATE(3),CAT(N),SUBN(CN),NUM(PLU),NEND(1),SEM(ABST).
SOMMET 9 ' ': UL('ULOCC').
SOMMET 10 'OF': UL('OF'),STATE(1),CAT(S),SUBS(PREP),VL1(OF).
SOMMET 11 ' ': UL('ULOCC').
SOMMET 12 'THE': UL('THE'),STATE(1),CAT(D),SUBD(ART),NUM(SIN,PLU).
SOMMET 13 ' ': UL('ULOCC').
SOMMET 14 'MATTER': UL('MATTER'), STATE(3),CAT(N),SUBN(CN),NUM(SIN),NEND(1),SEM(CONC),SEMCO(SUBST).
SOMMET 15 ' ': UL('ULOCC').
SOMMET 16 '.': UL('.'),STATE(1),CAT(P).

Sortie de l'analyse morphologique pour "the three states of the matter."

Figure 1 : Correspondance chaîne-arbre

Plus généralement, on peut dire qu'une GSCS définit une relation entre un ensemble de forêts et un ensemble d'arbres, où les forêts représentent les chaînes valides et où les arbres sont les représentations linguistiques structurales correspondantes. La figure suivante schématise cette idée.



L'axe ***A*** représente le domaine des arbres décorés (ex. représentation structurale).
L'axe ***F*** représente le domaine des forêts (ex. représentation morphologique).
Un point dans le plan représente donc un élément d'une relation entre chaînes et arbres.

Figure 2 : Relation définie par une GSCS.

Afin de définir une relation entre deux domaines, on doit les préciser et disposer de moyens pour spécifier leur relation. C'est pourquoi une GSCS est divisée *grosso modo* en deux parties: la **déclaration** définissant les arbres décorés qui constituent les deux domaines, et l'ensemble de règles, appelées **planches**, définissant la relation entre ces domaines.

2.1. Déclarations

La déclaration d'une GSCS a en général deux parties : décoration des arbres (type d'attribut, étiquette et attributs) et entités auxiliaires utilisées dans les planches (propriétés, constantes, fonctions booléennes, hiérarchie).

2.1.1. Objet de base : arbre décoré

Chaque noeud d'un arbre décoré porte une "décoration", qui est une liste d'**attributs.** Un attribut est un couple <nom : valeur>. La notion d'attribut a été introduite par Knuth dans les "grammaires avec attributs" [Knuth 68]. Son usage est général en linguistique informatique. La valeur d'un attribut peut être de type simple (entier, booléen, identificateur, chaîne, ensemble, scalaire), ou de type composé. Un attribut de type composé a pour valeur une liste d'attributs, et possède donc une

hiérarchie interne, comme un terme en PROLOG. Par exemple, "personne(3)" est un attribut de type simple, dont le nom est "personne" et la valeur "3" (l'interprétation linguistique peut être la 3ième personne) tandis que "Verbe(personne(3))" est un attribut de type composé, pouvant s'interpréter comme un verbe à la troisième personne.

En pratique, les attributs composés ne sont pas utilisés dans les GSCS déjà écrites, parce que les LSPLs actuels ne les supportent pas directement (le langage TETHYS [GETA-DSE2 82], en cours de construction, les supportera). Une décoration est alors une liste d'attributs simples contenant un attribut spécial, de type étiquette, qui est l'unité lexicale (UL).

2.1.2. Entités auxiliaires

Afin de faciliter l'écriture des planches, on peut déclarer des **constantes**, des **fonctions booléennes** (appelées aussi **prédicats**), et des **propriétés** (combinaisons de valeurs d'attributs, appelées aussi **formats**).

2.2. Planches

Chaque planche contient:

- une en-tête ;
- des références ;
- un schéma d'arbre ;
- un schéma de forêt ;
- des contraintes fines.

Une planche définit un ensemble d'arbres et un ensemble de forêts et établit une correspondance, éventuellement sous contexte, entre ces deux ensembles.

Dans l'en-tête d'une planche, seul le nom de la planche a une utilité formelle. Le reste de l'en-tête contient des informations destinées à faciliter la lecture ou la gestion. On y trouvera les cas traités par la planche, des exemples, la date et l'auteur de la création et de la modification et les applications de la planche (dans une ou plusieurs grammaires dynamiques).

Le nom de la planche sert de référence. On peut imposer des contraintes sur des sous-arbres dans les schémas au moyen des références : un arbre se référant à des planches doit appartenir à l'union des ensembles d'arbres définis par ces planches.

Les schémas expriment surtout la forme des arbres et des forêts. L'aspect géométrique d'un arbre peut être exprimé sous plusieurs formes : graphique, parenthésée ou indentée. Dans les schémas,

on utilise des désignateurs locaux à la planche pour des noeuds, des arbres et des forêts. Un désignateur dénote un ensemble d'éléments, ayant des caractéristiques communes, auquel on peut ajouter des contraintes.

On peut distinguer trois genres de contraintes sur les schémas : contraintes propres à un désignateur, contraintes entre désignateurs du schéma et contraintes fines portant à la fois sur le schéma d'arbre et sur le schéma de forêt. Les deux premières font partie de l'écriture des deux schémas.

2.3. Quelques remarques

2.3.1 A propos de la syntaxe de LSCS (*)

La syntaxe d'écriture des GSCS a quelque peu évolué, au fur et à mesure de leur utilisation et des études menées pour leur implémentation. La dernière en date présente les différences suivantes par rapport à celle du LSCS [Zajac 86]. Ces différences sont dues à l'utilisation de More (voir page 66).

i) Dans la description géométrique de l'arbre, les parenthèses et les virgules sont remplacées par l'indentation et la fin de ligne. Cela permet de visualiser l'arbre sous la forme graphique par une commande More.

ii) Les contraintes propres sont placées après le désignateur. Cela permet de mieux voir la liaison entre les contraintes et le désignateur.

iii) Le schéma de forêt peut être optionnel, grâce à la numérotation des éléments dans le schéma d'arbre selon leurs positions dans le schéma de forêt et à la précision sur des sous-arbres référants dans les références.

iv) Certains symboles spéciaux composés sont remplacés par des symboles simples, par exemple ^= par ≠.

v) On peut déclarer les axiomes et les terminaux. Cela permet, d'une part, de mieux préciser les axiomes et les terminaux, et, d'autre part, de rendre les deux domaines de la relation spécifiée par la GSCS plus clairs, aussi bien pour l'auteur que pour les lecteurs.

(*) Voir aussi l'annexe II.

2.3.2. GSCS et grammaires de Chomsky

On peut établir un parallèle entre les grammaires formelles de Chomsky et les GSCS . La notion de GSCS est une généralisation de celle de grammaire de Chomsky. Les objets de base sont des arbres, et non des symboles. Il faut remarquer que, dans les grammaires de Chomsky, les arbres syntaxiques, qui structurent les chaînes, ne sont que des produits secondaires des dérivations, alors que, dans les GSCS, les arbres associés aux chaînes sont des objets de la grammaire, spécifiés directement par l'auteur de la GSCS. De plus, des désignateurs (qui ne sont pas des constantes comme les symboles) peuvent être utilisés, et les planches peuvent renvoyer les unes aux autres, de manière plus fine que les grammaires usuelles. En effet, la notion de "référence", appliquée aux grammaires syntagmatiques, permettrait de spécifier, dans une partie droite, qu'un symbole non-terminal ne peut être développé que par certaines des règles ayant ce symbole en partie gauche.

Dans les GSCS déjà écrites, V_t et V_n sont implicites. V_t est l'ensemble des arbres pouvant s'instancier avec un des schémas de forêts de toutes les planches d'une GSCS, et V_n l'ensemble des arbres pouvant s'intancier avec un des schémas d'arbres. La déclaration des axiomes et des terminaux est un moyen complémentaire pour préciser les arbres pouvant être dans les deux domaines de la relation spécifiée par une GSCS. Le tableau suivant donne une comparaison :

GSCS	GRAMMAIRES DE CHOMSKY	
$G = (V, V_t, A, P)$	$G = (V_n, V_t, S, P)$	
ARBRES DECORES	SYMBOLES	$(V = V_t \cup V_n)$
A	S	(AXIOMES)
$A_p x F_p : V \supseteq A_p, F_p$	$V_1 x V_2 : V_1, V_2 \in V^*$	(P)
UNIFICATION	PRODUCTION	
REFERENCE	$\emptyset$	
$V_t \cap V_n = ?$	$V_t \cap V_n = \emptyset$	
RELATION	LANGAGE	
$\{<a,f> \mid a \xrightarrow{*} f, f \in V_t^*, a \in A\}$	$\{ x \mid S \xrightarrow{*} x, x \in V_t\}$	

3. UN EXEMPLE DE GSCS DU CHINOIS

Sur cet exemple, nous allons montrer les principes et le mécanisme des GSCS, ainsi que le processus de développement des GSCS. Nous expliquons d'abord les règles de numération en chinois. Nous montrons également comment utiliser cette GSCS aussi bien en analyse qu'en génération.

3.1 Phénomènes linguistiques traités

Nous traitons ici les groupes cardinaux du chinois pouvant exprimer tous les nombres naturels dont la valeur est inférieure à 10^{20}. Les groupes cardinaux sont des suites de noms cardinaux. Un nom cardinal est un caractère représentant une unité de base.

3.1.1. Unité de base

Il y a 15 noms cardinaux présentés dans le tableau ci-dessous:

Caractère		Prononciation	Valeur	Rang
Simple	Complexe			
〇	零	Ling	0	0
一	壹	Yi1	1	1
二	贰	Er	2	1
三	叁	San	3	1
四	肆	Si	4	1
五	伍	Wu	5	1
六	陆	Liu	6	1
七	柒	Qi	7	1
八	捌	Ba	8	1
九	玖	Jiu	9	1
十	拾	Shi	10	2
百	佰	Bai	100	3
千	仟	Qian	1000	4
万	萬	Wan	10000	5
亿	億	Yi	10**8	6

Note : 1. La prononciation est donnée en pinyin.
2. '**' veut dire 'puissance'.

Figure 3 : Unités de base de la numération chinoise

Les caractères pour 1 et 10^8 ont la même transcription en pinyin, mais leurs tons diffèrent. Afin de les distinguer, nous avons ajouté le chiffre 1 à la transcription du caractère 1, ce qui donne : Yi1.

3.1.2. Décomposition d'un nombre

Voici les règles de décomposition des nombres en chinois [LIU 83].

1. Les nombres en chinois sont en base dix.

2. Un nombre est représenté comme une expression polynomiale dont chaque terme se décompose en deux facteurs: une unité de base et un coefficient. Par exemple, ShiWuWanLingEr (10, 5, 10000, 0, 2 vaut 150002) a deux termes (10 + 5)*10000 et 2. Le coefficient du premier terme est encore une expression: 10 + 5. Dans l'expression du coefficient, on utilise des unités inférieures ou égales à l'unité suivante.

3. L'expression est écrite par ordre décroissant selon les rangs des unités de base. Quand les rangs de deux termes voisins ne sont pas consécutifs, on peut ajouter un zéro (Ling) entre les deux termes (voir l'exemple ci-dessus).

4. Deux caractères consécutifs doivent appartenir à des rangs différents sauf WanWan (=Yi =cent millions). Par exemple "BaiBai", "ErSan" sont incorrects.

5. Pour deux cardinaux consécutifs c_1c_2, si $c_1>c_2$ alors, $c_1c_2=c_1+c_2$ sinon $c_1c_2=c_1*c_2$. Nous avons vu dans l'exemple ci-dessus: 10+5 (10>5) et 5*10000 (5<10000).
6. Devant: Shi (10), Bai (100) et Qian (1000), on n'utilise que Yi1 à Jiu (1 à 9) comme coefficient. Par exemple, "SanShi" (3,10=30) est correct mais "SanShiQian" (3,10,1000) est incorrect. On peut utiliser un nombre comme coefficient devant Wan et Yi. Par exemple, WuWan (5,10000= 50000) et SanWanYi (3,10000,10**8=$3x10^{12}$) sont corrects.

Donnons d'abord une grammaire hors-contexte décrivant les chaînes correctes, mais pas leur "sémantique" (arbres abstraits). Nous verrons ensuite qu'une seule GSCS peut décrire à la fois la syntaxe et la sémantique.

<Naturel> ::= "Ling"
| <de 1 à 10**8>
| <de 10**8>

<de 1 à 10**8> ::= <de 1 à 10000>
| <de 10000 à 10**8>

<de 1 à 10000> ::= <de 1 à 1000>
| <de 1000 à 10000>

<de 1 à 1000> ::= <de 1 à 100>
| <de 100 à 1000>

<de 1 à 100> ::= <de 1 à 10>
| <de 10 à 100>

<de 1 à 10> ::= "Yi1" | "Er" | "San" | "Si" | "Wu" | "Liu"
| "Qi" | "Ba" | "Jiu"

<de 10 à 100> ::= "Shi" | "Shi"<de 1 à 10> | <de 1 à 10> "Shi"
| <de 1 à 10> "Shi" <de 1 à 10>

<de 100 à 1000> ::= <de 1 à 10> "Bai" | <de 1 à 10> "Bai" <Ling><de 1 à 10>
| <de 1 à 10> "Bai" <de 10 à 100>

<Ling> ::= ∅ | "Ling"

<de 1000 à 10000> ::= <de 1 à 10> "Qian" | <de 1 à 10> "Qian" <Ling><de 1 à 100>
| <de 1 à 10> "Qian" <de 100 à 1000>

<de 10000 à 10**8> ::= <de 1 à 10000> "Wan" | <de 1 à 10000> "Wan" <Ling><de 1 à 1000>
| <de 1 à 10000> "Wan" <de 1000 à 10000>

<de 10**8> ::= <de 1 à 10**8> "Yi" | <de 1 à 10**8> "Yi" <Ling><de 1 à 10000>
| <de 1 à 10**8> "Yi" <de 10000 à 10**8>

Le sens d'un groupe cardinal est sa valeur numérique. Les règles sémantiques des groupes cardinaux en chinois sont similaires à celles du français et de l'anglais, seules les unités de base diffèrent.

Soit N un nombre naturel syntaxiquement correct en chinois, $N = c_1c_2...c_n$ où c_i est un cardinal parmi les quinze donnés plus haut et n le nombre de cardinaux. La valeur de N peut se calculer récursivement selon la fonction V ainsi définie:

R1 n =1 : $V(N) = V(c_1) = c_1$

R2 n >1 : $c_i = \max(c_1, c_2, ..., c_n)$

$c_1 = 0$:	$V(N) = V(c_2...c_n)$	-- Zéro est facultatif
$i = 1$:	$V(N) = c_i + V(c_{i+1}...c_n)$	-- Sans coefficient
$i = n$:	$V(N) = V(c_1c_2...c_{i-1})*c_i$	
$1 < i < n$:	$V(N) = V(c_1c_2...c_{i-1})*c_i + V(c_{i+1}...c_n)$	
	-- coefficient multiplié par unité de base plus le reste	

Par exemple, pour la chaîne : "ShiWuWanLingEr" (10,5,10000,0,2), la valeur sera calculée comme suit :

V(ShiWuWanLingEr)	
= V(10,5,10000,0,2)	-- R2 : n = 5, i = 3
= V(10,5)*10000 + V(0,2)	-- R2 : n = 2 & i=1
= (10 + V(5))*10000 + V(0,2)	-- R1
= (10 + 5)*10000 + V(0,2)	-- R2 : $c_1 = 0$
= 150000 + V(2)	-- R1
= 150000 + 2 = 150002	

3.2. Objets de base

Bien que cette GSCS ne soit qu'un exemple, nous l'avons écrite de telle manière qu'elle puisse s'intégrer dans une GSCS complète du chinois.

ETIQUETTE DES : étiquette
COM : Dans cette GSCS, l'étiquette porte la Valeur NUMérique entière du cardinal.

vnum = ENTIER

DECORATION :
DES : Décoration
COM : Nous déclarons seulement les attributs utilisés dans les cardinaux

```
var : ENSEMBLE
    (
    CAT  = SCALAIRE     -- CATégories
           (N           -- Nom
            A           -- Adjoint
            ...         -- autres catégories
           CAT0         -- sans catégorie
           )
    SUBA = SCALAIRE     -- Sous-catégorie d'adjoint
           (CARD,       -- nom CARDinal
           ...          -- autres sous-catégories
           SUBA0        -- sans sous-catégorie d'adjoint
            )
    K    = SCALAIRE     -- Classe syntagmatique
           (GCARD,      -- Groupe CARDinal
            ...         -- autres classes
           K0           -- sans classe
           )
    FS   = SCALAIRE     -- Fonction Syntaxique
           (GOV,        -- GOuVerneur du groupe syntagmatique
           ...          -- autres fonctions
           FS0          -- fonction vide
           )
    )
```

3.3 Arbres terminaux

Dans cet exemple, un arbre représentant un des quinze caractères est un arbre terminal. Ces terminaux sont des arbres triviaux (ayant un seul noeud). Chaque noeud contient des informations pouvant être obtenues par une consultation de dictionnaire ou une analyse morphologique. Voici la déclaration des arbres terminaux :

TERMINAUX :

.mot : ¬CAT(CAT0), K(K0)

-- ',' remplace 'ET' (forme conjonctive implicite)

3.4 Arbres axiomes

L'ensemble des arbres axiomes est l'union des ensembles d'arbres définis par les trois planches. Il n'est donc pas nécessaire de les déclarer, bien que nous l'ayons fait pour la clarté de l'exposé :

AXIOMES :

.GC : K(GCARD)

àcoef? : K(GCARD)

.gov : FS(GOV),SUBA(CARD)

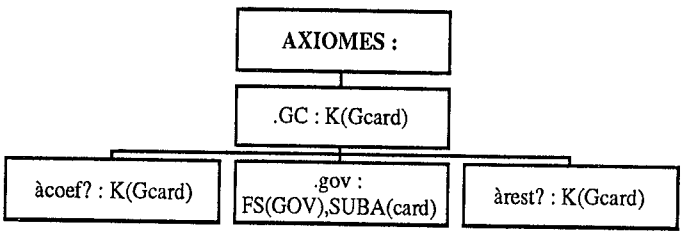
àrest? : K(GCARD) (*)

CSCHEMA

vnum(.gov) > vnum(àcoef?)

vnum(.gov) > vnum(àrest?)

Ce schéma se visualise graphiquement comme suit :



(*) à introduit un désignateur d'arbre,

. introduit un désignateur de noeud,

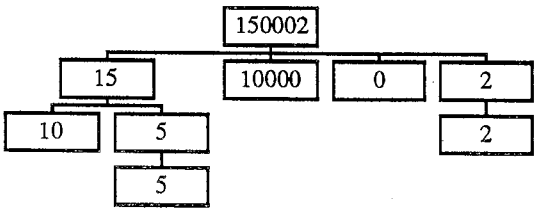
? marque l'optionnalité

Notre GSCS établit une correspondance entre chaque forêt d'arbres terminaux représentant un nombre chinois et un arbre axiome. La valeur de l'attribut vnum de la racine de l'arbre axiome sera la valeur numérique du groupe cardinal représenté par la forêt des arbres terminaux.

Par exemple, l'arbre (axiome) correspondant au groupe cardinal "Shi Wu Wan Ling Er" sera, sous forme indentée :

```
+ 150002
   + 15
      - 10
      + 5
         - 5
   - 10000
   - 0
   + 2
      - 2
```

ou sous forme graphique



Remarquons que, dans un arbre correspondant à un groupe cardinal, chaque noeud interne domine un groupe cardinal, chaque feuille est un nom cardinal et la valeur de l'attribut vnum est la valeur numérique du groupe ou du nom cardinal.

Par la suite, nous simplifions l'écriture des noeuds en donnant seulement les valeurs des attributs qui nous intéressent. Par exemple, pour la racine d'un groupe cardinal : <vnum(15),K(GCARD),CAT(A), ...> nous notons <15> parce que 15 ne peut être que valeur de l'attribut vnum et que les autres attributs sont évidents.

3.5 Planches

Voici les trois planches décrivant la correspondance :

PLANCHE PGCe

TYPE : GCe -- GROUPE CARDINAL élémentaire

CAS TRAITES : nombre entre zéro et neuf

EXEMPLES :
1) "ling"
REF. CORPUS GSchi, TEXTE N°1, PAGE 3
2) "jiu"
REF. CORPUS GSchi, TEXTE N°1, PAGE 3

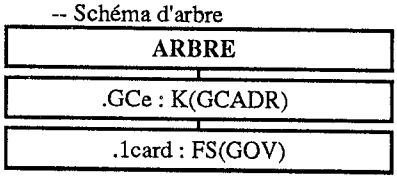
PLANCHES APPELANTES : PGCc1, PGCc2

Création
Par : Y.Yan Date : Sept. 1986
Modification
Par : Y.Yan Date : Mardi 17 mars 1987 Cause : mise en page en MORE
Vérification
Par: Date :
Version : 1

Application :
Adresse :
Code langue : Phase ARIANE :
Grammaires : Règles :

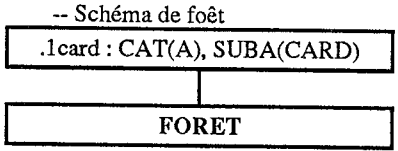
REFERENCES : aucune -- Cette planche ne se réfère à aucune planche

-- Schéma d'arbre



CSCHEMA -- Contraintes sur le SCHEMA d'arbre, ici aucune.

-- Schéma de foêt



-- L'élément est un nom cardinal comme Lin ou Jiu, etc.

CSCHEMA -- Contraintes sur le SCHEMA de forêt
$0 \leq$ vnum(.1card) ≤ 9
-- La valeur du cardinal est comprise entre 0 et 9.

CFINES -- Contraintes FINES portant sur les deux schémas à la fois.

UNIF(.GCe, .1card)
-- Le prédicat prédéfini UNIF garantit que les instances de deux désignateurs s'unifient de façon maximale
-- en respectant les contraintes dans les schémas.

PLANCHE PGCc1

TYPE : GCe -- GROUPE CARDINAL complexe

CAS TRAITES : nombre entre 10 et 9999

EXEMPLES :

1) "shi (dix=10)"
   REF. CORPUS GSchi, TEXTE N°1, PAGE 3
2) "shi wu (dix cinq= 15)"
   REF. CORPUS GSchi, TEXTE N°1, PAGE 3
3) "si qian ling er shi er (quatre mille zéro deux dix deux = 4022)"
   REF. CORPUS GSchi, TEXTE N°1, PAGE 3
4) "wu bai (cinq cent =500)"
   REF. CORPUS GSchi, TEXTE N°1, PAGE 3

PLANCHES APPELANTES : PGCc1, PGCc2

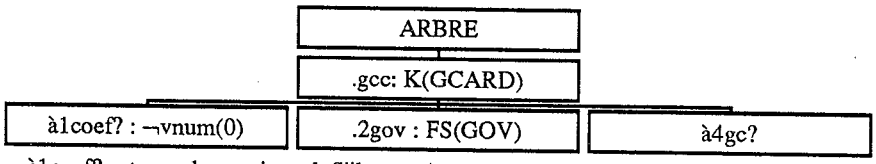
Création
Par : Y.Yan Date : Sept. 1986
Modification
Par : Y.Yan Date : Mardi 17 mars 1987 Cause : mise en page en MORE
Vérification
Par: Date :
Version : 1

Application :
Adresse :
Code langue : Phase ARIANE :
Grammaires : Règles :

REFERENCES

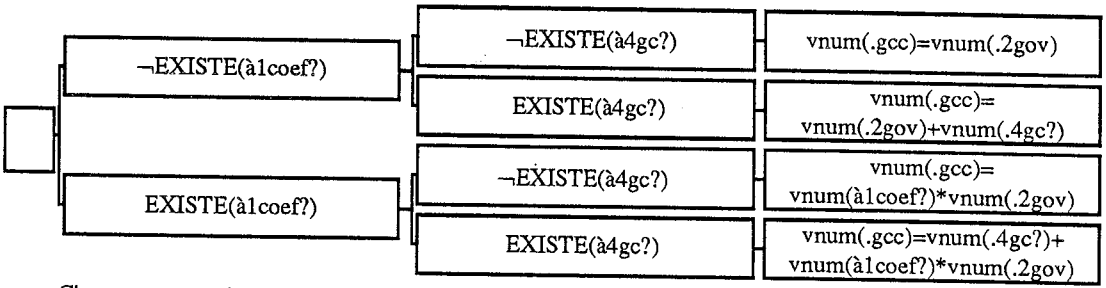
à1coef? : PGCe
-- exemples : wu, si. à1coef (référant) se réfère à la planche PGNe (référée)

à4gc? : PGCe, PGCc1
-- exemples : wu, er shi er

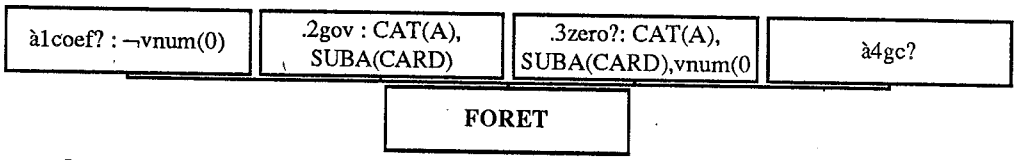


-- à1coef? est un arbre optionnel. S'il est présent, la valeur de vnum ne doit pas être zéro.

CSCHEMA



-- C'est une expression logique sous forme d'arbre ET-OU. L'opérateur entre père et fils est ET
-- et celui entre les frères est OU. Vide est toujours vrai.



-- Le 1er élément optionnel de la forêt (à1coef?) est un groupe cardinal élémentaire différent de zéro.

-- Le 2ième élément (.2gov) est un nom cardinal gouverneur. Par exemple : 10, 100 et 1000.

-- Le 3ième élément optionnel (.3zéro?) est le nom cardinal Lin.

-- Le 4ième élément optionnel (à4gc?) est un groupe cardinal défini par d'autres planches.

CSCHEMA

vnum(.2gov) ∈ (10, 100, 1000)	-- La valeur du cardinal gouverneur est 10, 100 ou 1000.
vnum(.2gov)≠10 => EXISTE(à1coef?)	-- Si la valeur du cardinal gouverneur est différent de 10, -- le coefficient est obligatoire.
EXISTE(.3zero?) => EXISTE(.4gc?)	-- La présence de zéro implique celle du groupe cardinal suivant.
vnum(.2gov) > vnum(.4gc)	-- La valeur du gouverneur est toujours supérieure à celle du -- groupe cardinal.
EXISTE(.3zero) => vnum(.2gov) > 10*vnum(.4gc)	-- La présence de zéro implique que le cardinal gouverneur est 10 -- fois plus grand que le groupe cardinal.

CFINES

UNIF(.2gov, .gcc)

PLANCHE PGCc2

TYPE : GCe -- GROUPE CARDINAL complexe

CAS TRAITES : nombre ≥ 10000

EXEMPLES :

1) "shi wu wan ling er (dix cinq dix-mille zéro deux =150002)"
   REF. CORPUS GSchi, TEXTE N°1, PAGE 3
2) "shi wu wan (dix cinq dix-mille =150000)"
   REF. CORPUS GSchi, TEXTE N°1, PAGE 3
3) "shi yi (dix cinq-million = 10**9)"
   REF. CORPUS GSchi, TEXTE N°1, PAGE 3

PLANCHES APPELANTES : PGCc2

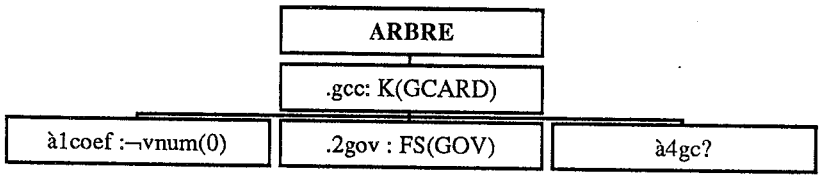
Création
Par : Y.YAN Date : Sept. 1986
Modification
Par : Y.Yan Date : Mardi 17 mars 1987 Cause : mise en page en MORE
Vérification
Par: Date :
Version : 1

Application :
Adresse :
Code langue : Phase ARIANE :
Grammaires : Règles :

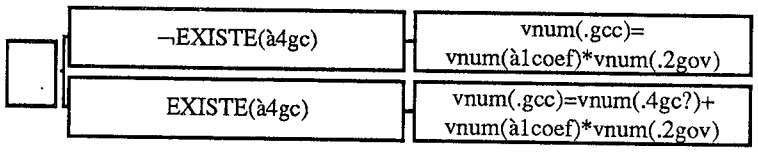
REFERENCES

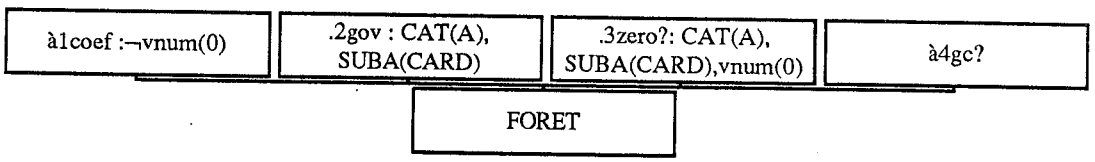
à1coef : PGCe, PGCc1, PGCc2
-- exemples : er, shi, wu wan,

à4gc? : PGCe, PGCc1, PGCc2
-- exemples : wu, er shi er, er wan



CSCHEMA





-- L'élément à1coef est un groupe cardinal élémentaire différent de zéro.

-- L'élément .2gov est un nom cardinal gouverneur, par exemple 10000, 10**8.

-- L'élément .3zéro? est le nom cardinal zéro.

-- L'élément à4ge? est un groupe cardinal défini par d'autres planches.

CSCHEMA

vnum(.2gov) ∈ (10000, 10**8	-- La valeur du cardinal gouverneur est 10000 ou 10**8.
vnum(à1coef) < 10**8	-- Contrainte propre au coefficient (inferieur à 10**8).
EXISTE(.3zero?) => EXISTE(.4gc?)	-- La présence de zero implique celle du groupe cardinal suivant.
vnum(.2gov) > vnum(.4gc)	-- La valeur du gouverneur est toujours supérieure à celle du groupe cardinal.
EXISTE(.3zero?) => vnum(.2gov) > 10*vnum(.4gc?)	-- La présence de zéro implique que le cardinal gouverneur est 10 fois plus grand que le groupe cardinal.

CFINES

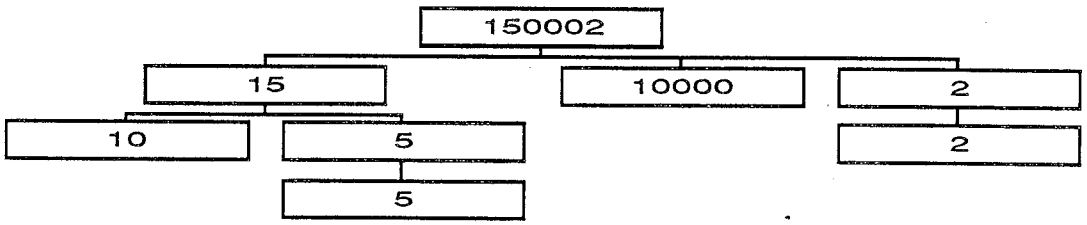
UNIF(.2gov,.gcc)

3.6 Interprétation

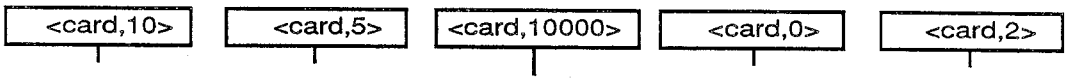
Nous allons voir dans cette sous-section comment interpréter la GSCS dans les deux sens (analyse et génération) en utilisant le même exemple que ci-dessus. L'application d'une planche consiste à remplacer une forêt (ou un arbre) par un arbre (ou une forêt) qui vérifie la correspondance établie par la planche. Une étude plus formelle de la sémantique des GSCS, légèrement différente de la présente, se trouve dans [Zaharin 86, 87a].

3.6.1. GSCS en analyse

Nous allons voir comment obtenir l'arbre (nommé A_4)



à partir de la forêt



au moyen d'une dérivation, c'est-à-dire suite d'applications de planches.

Cette analyse peut être divisée en quatre étapes. A chaque étape, on applique l'une des trois planches de la GSCS. Chaque élément de la forêt de départ doit appartenir à l'ensemble des terminaux. L'application d'une planche sur la forêt donne une autre forêt équivalente, dans laquelle la sous-forêt appartenant à l'ensemble des forêts définies par la planche est remplacée par l'arbre correspondant. Le nombre d'éléments de la nouvelle forêt n'est donc jamais plus grand que celui de l'ancienne. Quand la nouvelle chaîne ne contient qu'un élément, et si celui-ci appartient à l'ensemble des axiomes, l'analyse est réussie. La forêt de départ et l'axiome obtenu constituent un élément de la relation définie par la GSCS.

A chaque étape, l'analyse procède de la manière suivante:

- recherche d'une occurrence du schéma de forêt ;
- vérification des contraintes sur le schéma de forêt et les références ;
- choix d'un arbre satisfaisant les contraintes sur le schéma d'arbre et sur les références.

Nous ne donnons pas les applications qui aboutissent à des impossibilités, par exemple celles de la planche PGCe aux cardinaux 10, 10 000 et 0.

Etape 1 : application de la planche PGCe.

L'élément <card, 5> est une occurrence du schéma de forêt de la planche PGCe. Ce schéma n'a qu'un élément: .1card. La condition propre au désignateur sur le schéma: "la valeur de l'attribut CAT est égale à A et celle de SUBA est égale à CARD" est satisfaite par l'élément <card, 5>.

Les contraintes sur le schéma : "0<=vnum(.1card) <10" sont également vérifiées. Il n'y pas de référence dans cette planche.

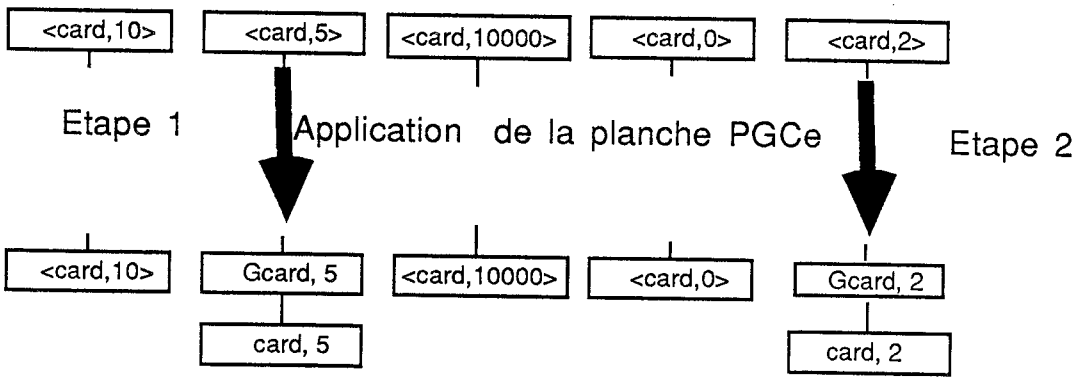
L'arbre correspondant doit avoir deux noeuds. Le premier est presque le même que l'élément de la forêt (en lui ajoutant la Fonction Syntaxique GOuVerneur). L'autre est la racine, décorée par "K(GCARD)" selon le schéma d'arbre. Dans les contraintes fines, le prédicat prédéfini UNIF(.GCe,.1card) garantit que les décorations de .GCe et de .1card s'unifient de façon maximale en respectant les contraintes explicites dans les deux schémas. L'arbre ne peut être que <Gcard,5>(<gov, 5>).

En remplaçant <card, 5> par <Gcard, 5>(<card, 5>), nous obtenons donc une nouvelle forêt.

Etape 2 : application de la planche PGCe.

De façon analogue à la première étape, en remplaçant <card,2> par <Gcard,2>(<card,2>) dans la nouvelle forêt obtenue, nous obtenons une seconde forêt (nouvelle forêt dans la figure ci-dessous).

Forêt de départ



Nouvelle forêt (après deux étapes)

Etape 3 : application de la planche PGCc1.

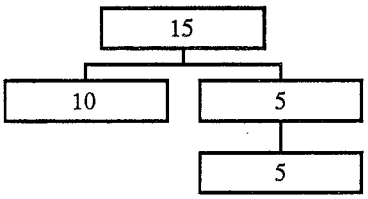
La sous-forêt "<card,10>,<Gcard,5>(<card,5>)" est une occurrence du schéma de forêt de la planche PGCc1. La coïncidence (pattern matching) est montrée par le tableau suivant :

occurrence	schéma de forêt
ϕ, <card, 10>, ϕ, Gcard, 5 card, 5	à1coef? .2gov <.3zéro?, à4gc?

L'élément, remplaçant à4gc?, <Gcard, 5>(<card,5>) réfère aux planches PGCe ou à la planche elle-même. Nous avons vu à l'étape 1 qu'il est dans l'ensemble d'arbres définis par la planche PGCc1. Les contraintes sur le schéma de forêt sont vérifiées :

- la valeur du cardinal unifié avec le gouverneur est 10 ∈ (10,100,1000).
- quand le coefficient est absent, toutes les conditions posées sur lui sont considérées comme satisfaites.
- la 3-ième contrainte est vérifiée grâce à l'absence du coefficient.
- la valeur du gouverneur est supérieure à celle du groupe cardinal suivant.
- l'élément .3zéro est absent (remplacé par Ø); la 5ième contrainte est vérifiée : la fonction booléenne prédéfinie : EXISTE (<désignateur>) vaut FAUX si <désignateur> est Ø, et VRAI sinon.

Maintenant, nous allons choisir un arbre correspondant à la forêt en respectant les contraintes sur le schéma d'arbre. La seule contrainte donne quatre cas exclusifs pour la valeur de vnum selon la présence de àcoef? et de .4gc?. Dans cet exemple, c'est le deuxième cas: vnum(.gcc)=vnum(.2gov)+vnum(.gc) =10+5=15. En tenant compte des contraintes propres dans le schéma, nous obtenons l'arbre correspondant:



La nouvelle forêt est :

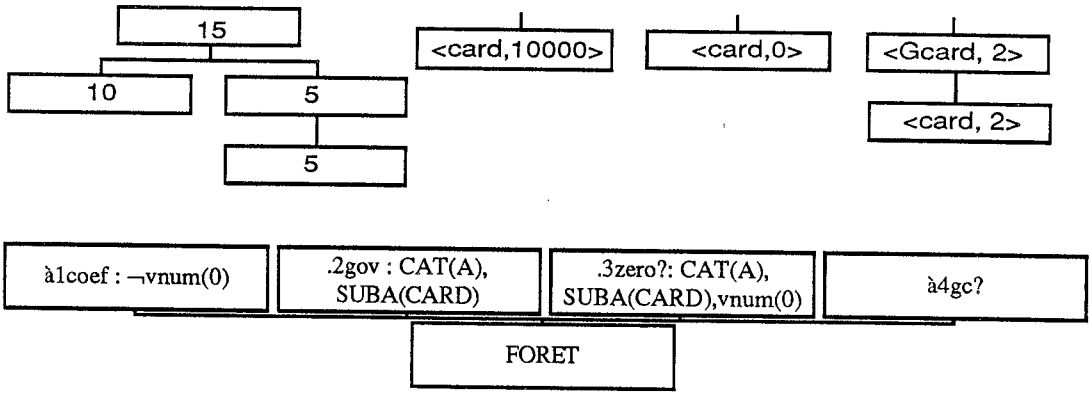


Schéma de forêt de la planche PGCc2

Etape 4 : application de la planche PGCc2.

Nous arrivons à la dernière étape de la dérivation. La nouvelle forêt est une occurrence du schéma de forêt de la planche PGCc2. Dans la figure ci-dessus, nous voyons que la forêt courante peut s'instancier avec le schéma de forêt de la planche.

Le désignateur .3zero? désigne en fait une constante (le mot cardinal zéro). Il est optionnel et sa présence peut influencer la présence des autres.

Les éléments à1coef et à4gc réfèrent à une même liste de planches, y compris la planche PGCc2 elle-même. Nous savons que l'instance de à1coef est un arbre défini par PGCc1 (étape 3) et que celle de à4gc? est l'arbre défini PGCe (étape 2). Les contraintes sur le schéma de forêt dans cette planche sont similaires à celles de la planche PGCc1. Donc, nous ne répétons pas la vérification de ces contraintes. Ce qui diffère dans cette étape, c'est la présence d'un coefficient et de zéro. La contrainte

"EXISTE(.3zero) => vnum(.2gov) > 10*vnum(.4gc?)"

exprime la règle de décomposition des nombres en chinois : si zéro (Ling=0) est présent, il se trouve entre une unité de base (>10) et un nombre ; le nombre doit être dix fois inférieur à l'unité précédente. C'est aussi le cas parce que le nombre 2 est dix fois inférieur à 1000 dans cette instance. La présence du zéro implique celle du groupe suivant. Il y a en effet un groupe dans cette instance. Nous avons vérifié que la forêt faisait partie de l'ensemble de forêts définies par la planche.

Le choix de l'arbre correspondant est analogue à celui de l'étape 3.

Nous avons obtenu une forêt ne contenant que l'arbre A_4 (voir page 36).

Il est trivial de montrer que cet arbre est un axiome et que chaque élément de la forêt de départ est un terminal. Donc, le couple forêt-arbre est un élément de la relation spécifiée par la GSCS.

3.6.2. GSCS en génération

La génération est à peu près l'inverse de l'analyse. On part d'un arbre, et, par application de planches, on obtient l'une des forêts correspondantes.

Pour le même couple forêt-arbre, les quatre étapes sont donc inversées. Le processus de chaque étape est également inversé. On cherche d'abord une occurrence du schéma d'arbre. On vérifie ensuite les contraintes sur le schéma d'arbre. On choisit à la fin l'une des forêts respectant les contraintes sur le schéma et la référence.

Etape 1 : application de la planche PGCc2.

L'arbre A_4 est une occurrence du schéma d'arbre de la planche PGCc2. Cela est évident, grâce à l'étape 4 en analyse.

Pour satisfaire la contrainte sur le schéma d'arbre, il y a apparemment un choix: l'arbre à4gc? peut être présent ou absent. En fait, si à4gc? est absent, il faut vnum(àgcc)=150002 =vnum(à1coef)*vnum(.2gov). Or, dans les contraintes sur le schéma de forêt, une contrainte exige que vnum(.2gov) soit égal à 10000 ou à 10**8. Il est donc impossible de trouver un entier vnum(à1coef), solution de cette équation. En conséquence, l'arbre àgcard doit être présent. La solution pour l'équation: "vnum(à4gc)=vnum(à1coef)*vnum(.gov)+vnum(àgc)" est unique: vnum(à1coef)=15, vnum(.2gov)=10000 et vnum(à4gc)=2.

La présence de zéro dans la forêt est optionnelle. C'est-à-dire qu'il y a au moins deux forêts correspondant à cet arbre. Cela reflète le phénomène linguistique de synonymie. Le caractère Ling dans un groupe cardinal est facultatif. Sa présence rend le nombre plus lisible. En analyse, nous devons donc accepter les nombres sans Ling (zéro), tandis qu'en génération il vaudra mieux les rajouter.

La forêt correspondante est donc la même que la forêt de départ dans l'étape 4 de l'analyse.

Les autres étapes sont tout à fait analogues. Nous ne les détaillons donc pas.

Remarquons enfin que, pour obtenir une chaîne de noms cardinaux correspondant à un arbre défini par cette GSCS, il suffit de prendre ses feuilles de gauche à droite. On peut ajouter un zéro entre le mot cardinal gouverneur et le groupe cardinal suivant, si la valeur de vnum du gouverneur est dix fois plus grande que celle du groupe.

Si l'arbre de départ n'a pas cette structure (transfert à partir d'une autre langue), on peut la construire facilement, si l'on connaît sa valeur numérique Vn, de la manière suivante :

si Vn est l'une des unités de base : l'arbre est une racine dominant un seul noeud, sinon :

1) diviser Vn par les unités de base dans l'ordre décroissant jusqu'à ce que le quotient Q diffère de zéro,

2) prendre l'unité de base (diviseur) comme gouverneur ; de la même manière (récursive), fabriquer le sous-arbre coefficient avant le gouverneur avec le quotient et le sous-arbre après le gouverneur avec le reste, s'il est différent de zéro.

CHAPITRE II. INGENIERIE EN TAO

La mise au point du formalisme des GSCS a comme enjeu la possibilité de donner les propriétés statiques des grammaires dynamiques dans un objectif général du génie logiciel. Il faut poursuivre cet objectif pour voir comment insérer la manipulation de ce formalisme dans un poste de travail adapté à la réalisation d'un linguiciel.

Dans ce chapitre, nous passons d'abord en revue les enjeux d'une approche du type génie logiciel pour un projet de linguiciel. Puis nous indiquons les différents types de logiciels qui devraient composer un atelier de génie linguiciel. Nous précisons enfin les objectifs retenus pour les logiciels qui constituent le poste de travail que nous avons mis au point.

1. Enjeux d'un système industriel de TAO

1.1. Etude de cas : le linguiciel CALLIOPE-AERO

En définissant les grammaires statiques, nous avons rappelé leur intérêt essentiel comme moyen de spécification des caractéristiques linguistiques du couple de langues pour lesquelles est construit un linguiciel de TAO dans un domaine et une typologie déterminés.

En ce sens, nous témoignons d'un souci d'organiser la conduite d'un projet de fabrication d'un tel linguiciel selon les méthodes plus générales mises en évidence dans le cadre du génie logiciel.

Ces méthodes ont été élaborées progressivement pour permettre de faire face aux défis présentés par la fabrication de gros logiciels. Amélioration de la productivité des équipes, normalisation des méthodes, augmentation de la fiabilité des produits fabriqués, telles sont les principales motivations qui ont inspiré la mise au point de méthodes et outils regroupés sous le terme générique de "génie logiciel" et qui doivent permettre de rendre industrielle, donc rentable, la production de logiciels.

Dans le cadre du GETA, les aspects d'industrialisation d'un linguiciel ont dû être pris en compte à l'occasion du Projet National de TAO (dit PN-TAO), dont le principal objectif a été de construire un système commercialisable de TAO. Le linguiciel concerné, baptisé CALLIOPE-AERO, permet de traduire du français vers l'anglais des textes techniques en aéronautique.

A partir de l'expérience accumulée durant ce projet, nous pouvons présenter les enjeux de chaque phase de construction d'un linguiciel de TAO. Pour chaque phase, nous donnerons :

a) un rappel du contenu opératoire de la phase,

b) l'ordre de grandeur du travail qu'elle implique,

c) les difficultés techniques rencontrées,

d) les méthodes et outils relevant d'une approche en génie logiciel qui permettraient de faire face à ces difficultés.

1.1.1. Rappel de quelques ordres de grandeur économiques

A partir d'un projet comme CALLIOPE-AERO, nous pouvons évaluer les coûts de construction d'un linguiciel de TAO, SR1-CB1, qui serait nouveau tant pour la langue source SR1 que pour la langue cible CB1.

Dans ces coûts, nous ne comptons pas toutes les opérations qui, dans le projet, ont été réalisées pour améliorer ARIANE-78, l'environnement logiciel de base de préparation et d'exécution du linguiciel.

Phase du projet	**U O Unité d'oeuvre**	**coût complet de l'U O (en KF)**	**Nombre d' UO**	**coût total de la phase (en KF)**	**% du total**
Etude de faisabilité	mois d'ingénieur	60	3	**180**	**3**
.Spécification linguistique	mois d'ingénieur	60	20	1.200	20
.Implémentation	mois d'ingénieur	60	20	1.200	20
Sous-total pour la conception et la programmation du linguiciel				**2.400**	**40**
. Formation et suivi des équipes d'indexation	mois d'ingénieur	60	6	360	6
. Construction du dictionnaire général	terme	0,080 (*)	7.000	560	9,3
. Construction du dictionnaire terminologique	terme	0,040 (**)	20.000	800	13,4
Sous-total pour la fabrication des dictionnaires				**1.720**	**28,7**
Tests et mise au point	mois d'ingénieur	60	15	**900**	**15**
Energie informatique	mois (forfait)	35	23	**800**	**13,3**
		TOTAL		**6.000**	**100**

Légende (*) soit la codification de 12 termes/jour au prix de revient du jour d'indexeur de 1000 francs.

(**) soit la codification de 50 termes/jour au prix de revient du jour d'indexeur de 2000 francs.

TABLEAU I

A partir de cette évaluation de l'investissement, il est possible de procéder à celle du prix de revient de la traduction brute, avant révision, d'une page standard de 250 mots, en fonction du volume de documentation à traduire.

(en franc / page)	Volume de la documentation (en nombre de pages à traduire) 10.000	20.000	40.000	50.000	100.000
Consommation de temps de calcul	40	40	40	40	40
Amortissemement de l'investissement de réalisation du linguiciel	600	300	150	120	60
Prix de revient d'une page traduite sans révision	**640**	**340**	**190**	**160**	**100**

TABLEAU II

Si, par contre, après la réalisation de ce premier linguiciel SR1-CB1, nous devions réaliser un nouveau linguiciel SR1-CB2 avec la même langue source SR1 et une autre langue cible CB2 pour le même domaine et la même typologie, le tableau serait le suivant :

Phase du projet	**U O Unité d'oeuvre**	**coût complet de l'U O (en KF)**	**Nombre d' UO**	**coût total de la phase**	**% du total**
.Etude de faisabilité	mois d'ingénieur	60	2	**120**	**3,3**
.Spécification linguistique	mois d'ingénieur	60	10	600	16,2
.Implémentation	mois d'ingénieur	60	10	600	16,2
Sous-total pour la conception et la programmation du linguiciel				**1200**	**32,4**
.Formation et suivi des équipes d'indexation	mois d'ingénieur	60	4	360	9,7
.Construction du dictionnaire général	terme	0,050 (*)	7000	350	9,5
.Construction du dictionnaire terminologique	terme	0,030 (**)	20000	600	16,2
Sous-total pour la fabrication des dictionnaires				**1310**	**35,4**
.Tests et mise au point	mois d'ingénieur	60	10	**600**	**16,2**
Energie informatique	mois (forfait)	35	13,5	**470**	**12,7**
		TOTAL		**3.700**	**100**

Légende (*) soit la codification de 20 termes/jour, uniquement dans les dictionnaires de transfert et de génération, au prix de revient du jour d'indexeur de 1000 francs
(**) soit la codification de 70 termes/jour uniquement dans les dictionnaires de transfert et de génération, au prix de revient du jour d'indexeur de 2000 francs

Tableau III

Le calcul du volume minimum de traduction à assurer avec ce nouveau linguiciel modifie sensiblement le niveau du point mort obtenu en fonction du prix de la traduction humaine (entre 200 et 400 francs par page).

	Volume de la documentation (en nombre de pages à traduire)				
(prix en franc / page)	**10.000**	**20.000**	**40.000**	**50.000**	**100.000**
Consommation de temps de calcul	40	40	40	40	40
Amortissement de l'investissement de réalisation du linguiciel	370	185	92	74	37
Prix de revient d'une page traduite sans révision	**410**	**225**	**132**	**114**	**77**

Tableau IV

1.1.2. Conclusion

Ces deux exemples d'analyse des coûts de développement d'un linguiciel mettent en évidence l'ordre de grandeur des investissements dans ce développement et l'importance du volume de traduction à assurer pour rentabiliser ces investissements. Ils permettent de mesurer l'obligation fondamentale d'améliorer par tous les moyens la productivité de la réalisation, et donc de faire appel à toutes les approches adaptables provenant du génie logiciel. Notons enfin que l'amortissement de l'investissement de fabrication du linguiciel représente quasiment toujours plus de la moitié du prix de revient de la page traduite, hors révision.

1.2. Analyse des besoins selon chaque phase d'un projet de linguiciel

1.2.1. Etude d'opportunité du linguiciel

a - Contenu de la phase

La perspective de construction d'un linguiciel s'analyse toujours en précisant :

- le couple langue source - langue cible (par exemple anglais - français) ;
- le domaine technologique couvert qui déterminera la nature du vocabulaire terminologique à prendre en compte (par exemple les techniques de la construction aéronautique) ;
- la typologie des documents (par exemple des manuels d'entretien d'avions), à savoir les formes particulières de constructions grammaticales à prendre en compte ;
- le volume prévisible de documentation susceptible d'être traduite ultérieurement par le système de TAO.

Comme nous l'avons vu précédemment, ce volume permet de faire un calcul prévisionnel du prix de revient d'une page de traduction. L'amortissement de l'investissement de fabrication du linguiciel représentant presque toujours plus de la moitié de ce prix de revient.

b - Importance du travail

Selon que l'expérience dans le couple de langues existe déjà ou n'existe pas, les conditions de cette étude sont très différentes :

- CAS 1 : S'il existe déjà un linguiciel de TAO dans le même couple de langues, pour un domaine et une typologie différents, le travail sera d'examiner les possibilités de récupération des acquis linguistiques de ce premier linguiciel pour en fabriquer le second.

- CAS 2 : Si la langue source (cible) est la même, et si la langue cible (source) est nouvelle, on limitera l'étude aux possibilités de récupérer les connaissances et algorithmes disponibles pour l'analyse (génération) de la langue source (cible) en vérifiant les variables et attributs nécessaires pour permettre le transfert vers (depuis) la nouvelle langue cible (source).

- CAS 3 : Si les deux langues, source et cible, sont nouvelles, il faudra sans doute fabriquer un premier prototype qui permette d'explorer les difficultés de traitement des nouvelles constructions grammaticales rencontrées.

c - Difficultés spécifiques rencontrées

Dans les deux premiers cas précédents, ces difficultés proviennent essentiellement de l'obligation de pouvoir récupérer toutes les connaissances existantes sur les langues déjà étudiées. La possibilité de faire cette récupération est associée à l'existence d'une documentation exhaustive et cohérente disponible sur les linguiciels déjà réalisés.

d - Objectifs : méthodes et outils

Pour un linguiciel déjà existant, il faut disposer d'un système de documentation automatique gérant les corpus, les spécifications et les jeux d'essai de ce linguiciel. Ce système permettrait de prendre connaissance, de façon descendante, de toutes les caractéristiques du linguiciel. Pour la construction d'un autre linguiciel devant reprendre une partie du premier, un tel système faciliterait grandement la récupération rapide et efficace de toutes les spécifications à conserver.

1.2.2. Planification et cahier des charges

a - Contenu de la phase

Une fois connues les possibilités de récupération d'acquis linguistiques et mesurés les volumes de terminologie à codifier dans le domaine, il est possible de faire une évaluation prévisionnelle du projet tant en termes de coût que de délai.

Mais le principal reste à faire : constituer une équipe compétente qui prenne en compte à la fois les problèmes de construction du linguiciel et de codification des dictionnaires.

b - Importance du travail

La constitution de cette équipe suppose en général le recrutement de différentes catégories de personnel. Citons notamment :

- au moins un ingénieur-linguiste (nous parlerons de "linguicien"), maîtrisant les problèmes d'analyse de la langue source avec ses difficultés de traitement des ambiguïtés ;
- au moins un linguicien chargé de prendre en compte la génération dans la langue cible et une bonne partie des questions relatives au transfert ;
- un responsable de la codification du vocabulaire général, animant l'équipe de codage des dictionnaires généraux ;
- une équipe de codage des dictionnaires généraux, constituée de 4 à 6 personnes ;

- une équipe de codage de la terminologie spécialisée du domaine, dont l'effectif dépendra à la fois du volume de terminologie et des délais imposés pour la phase de codage.

c - Difficultés spécifiques rencontrées

La constitution d'une telle équipe ne se réalise pas rapidement, car elle suppose effectuées les opérations de recrutement et de formation des personnels.

Le recrutement de "linguiciens" est très difficile, car ce profil de poste demande une double compétence à la fois en informatique et en linguistique. Une telle formation peut être trouvée soit chez des linguistes qui ont une deuxième formation en informatique, soit chez des informaticiens connaissant plutôt le domaine de l'intelligence artificielle et qui se passionnent pour la linguistique, même si leur formation initiale reste faible. Tout ingénieur recruté selon l'un ou l'autre des deux profils doit apprendre le fonctionnement d'un système important comme ARIANE, et doit expérimenter effectivement les possibilités de représentation des phénomènes linguistiques qu'il permet. Cela ne peut se faire en moins de trois mois en supposant une formation intensive et une excellente productivité.

d - Objectifs : méthodes et outils

Pour réaliser cette formation, la meilleure approche semble être de mettre à la disposition des nouveaux membres de l'équipe une maquette de TAO qui soit un modèle simplifié de système de traduction illustrant les principales possibilités techniques d'ARIANE et les principales difficultés de modélisation linguistique rencontrées pendant la construction d'un système de TAO. Dans le cadre du projet ESOPE de l'ADI (préparant le PN-TAO), une telle maquette pédagogique, baptisée BEX-FEX, a été réalisée. Elle permet la traduction de l'anglais vers le français d'un corpus expérimental de 500 mots.

Une maquette doit être documentée de manière particulièrement claire, complète et cohérente, car sa documentation doit permettre :

- de prendre rapidement connaissance du modèle de manière interactive,
- de donner aux nouveaux membres de l'équipe des habitudes correctes sur les normes de documentation.

Il s'avère alors très souhaitable de disposer d'un système de documentation automatique qui contiendrait le corpus d'essai, la documentation proprement dite (spécification et programmmes commentés) et des aides interactives pour interroger en cas de besoin les modes opératoires préconisés.

Cet outil documentaire devrait de plus assurer l'enregistrement des extensions de la maquette réalisées par chaque stagiaire dans le cadre de travaux pratiques.

1.2.3. Conception

a - Contenu de la phase

Elle doit aboutir à une spécification complète des éléments permettant l'analyse, le transfert et la génération. Il faut donc :

- décrire les planches statiques de la grammaire de la langue source et la grammaire de spécification des désambiguïsation ;
- décrire les planches de la langue cible en ne prenant pas en compte les problèmes de désambiguïsation ;
- dessiner les bordereaux d'indexation pour les dictionnaires d'analyse, de génération et de transfert ;
- tester ces bordereaux sur un échantillon de mots complexes du vocabulaire de base pour vérifier la possibilité de les faire remplir par des indexeurs en obtenant de surcroît des exemples significatifs permettant de procéder à leur formation.

b - Importance du travail

Dans un projet de taille industrielle, il faut prévoir

1) pour les GSCS (en comptant 5 pages par planche)
   - langue source :
     = 150 planches de construction syntaxique, soit 750 pages
     = 500 planches de désambiguïsation, soit 2 500 pages
   - langue cible :
     = 100 planches de construction, soit 500 pages
2) pour les dictionnaires, 100 types de bordereaux(*) répartis ainsi
   - termes généraux : 15 x 2 monolingues et 30 bilingues
   - termes techniques : 10 x 2 monolingues et 10 bilingues

Il faut remplir 20 bordereaux pour chaque type, simplement pour la validation des dessins de bordereaux. Rappelons que dans un système de TAO, il faut remplir environ 30 000 bordereaux.

(*) masque de saisie permettant d'acquérir les informations linguistiques associées à un terme.

c - Difficultés spécifiques rencontrées

Cette modélisation linguistique s'opère en rattachant les phénomènes linguistiques identifiés aux textes du corpus où ils apparaissent. De plus, la spécification définitive des planches ne se réalise que progressivement ; on constate en effet que les équipes linguistiques notent en permanence des difficultés qu'elles ne savent pas résoudre immédiatement et dont elles reportent la solution à une phase ultérieure d'affinage de leur modèle.

d - Objectifs: méthodes et outils

Il est donc indispensable de donner aux équipes de conception un logiciel de gestion de leurs spécifications, qui autorise :

- le rapprochement entre les textes du corpus et leurs spécifications,
- la gestion de plusieurs niveaux de spécifications dans un ordre croissant de formalisation,
- l'organisation d'un bloc-notes électronique dans lequel on puisse ranger les difficultés linguistiques non encore traitées,
- l'organisation facile du cycle "lecture-écriture". Par ce terme, on désigne la procédure qui garantit que toute spécification réalisée par un premier analyste doit être relue et commentée par un second analyste, puis corrigée par le premier, avant d'être définitivement validée. Il est évident qu'elle revêt une importance particulière dans la spécification d'un linguiciel, du fait que la cohérence d'un modèle linguistique est le fruit d'une maturation lente de l'équipe qui l'a en charge.

1.2.4. Codage

a - Contenu de la phase

Elle regroupe à la fois :

- le codage des grammaires dynamiques du linguiciel dans les langages spécialisés de programmation linguistique ;
- la construction d'un jeu d'essai de dictionnaires suffisamment complet pour le vocabulaire de base et le vocabulaire terminologique, afin de permettre ultérieurement une phase de tests relativement significative.

b - Importance du travail

1) La programmation d'un modèle français-anglais de taille industrielle contient approximativement, selon l'examen des chiffres tirés de l'expérience CALLIOPE-AERO :

20.000 lignes pour l'analyse du français ,

3.000 lignes pour le transfert,

6.000 lignes pour la génération de l'anglais .

Il s'agit ici uniquement des grammaires structurales, écrites en ROBRA (langage de'écriture de systèmes transformationnels).

On retiendra donc un ordre de grandeur de 30.000 lignes de code, dont 65% sont consacrés à l'analyse, 10% au transfert et 25% à la génération. C'est dire explicitement et quantitativement l'importance de l'analyse et des stratégies de désambiguïsation qu'elle implique.

Au niveau de la productivité, la codification de 30.000 lignes en 20 homme x mois (cf. tableau I, page 45), soit en 400 jours, donne une productivité de codage de 75 lignes de code par jour. Cet ordre de grandeur, comparable à celui du codage d'autres applications importantes dans des langages de haut niveau comme PASCAL, implique alors que les méthodes pour améliorer la productivité des équipes soient aussi comparables.

2) Pour la codification des bordereaux, il est judicieux de retenir :

- 3.000 unités lexicales du vocabulaire général, soit, au rythme de 12 UL /jour, 250 jours de travail ou 12 mois d'indexage ;
- 5.000 unités lexicales de terminologie, soit, à la cadence de 50 UL / jour, 100 journées d'indexage ou 4 mois de travail terminologique.

c - Difficultés spécifiques rencontrées

Les objectifs de fabrication du code sont ici ceux que l'on retrouve dans l'écriture d'autres gros logiciels : nous voulons obtenir un code structuré, respectant des normes de commentaires précises.

Par contre, l'importance de la fabrication des dictionnaires est tout à fait originale. Il faut gérer cette production d'unités lexicales au même rythme que l'écriture du code, sous peine de ne pas disposer de tests d'un volume suffisant d'unités, au démarrage, et de ne pas pouvoir analyser un nombre suffisant de pages de corpus.

d - Objectifs: méthodes et outils

Pour la production du code, les outils disponibles sont ceux directement intégrés sous ARIANE ; en particulier, la modularité du code est facilitée par la possibilité d'exécution séparée de chaque phase du système de TAO.

Pour la production des dictionnaires, il est difficile et coûteux de demander à des indexeurs de codifier directement les unités lexicales sous leur format ARIANE. Aussi s'avère-t-il indispensable de construire une base de données intermédiaire, appelée base lexicale, où les unités sont codées selon leurs propriétés statiques. Ensuite, un traducteur transforme cette base en dictionnaires ARIANE. Dans le PN-TAO, une première version d'un tel traducteur a été réalisée par la société SG2, pour le système CALLIOPE-AERO. Ce traducteur concerne uniquement les dictionnaires d'analyse et de génération.

1.2.5. Tests et mise en oeuvre

a - Contenu de la phase

Il s'agit de mettre au point un modèle en lui soumettant pour essai des lots de textes issus du corpus d'essai. Ces tests doivent permettre de trouver les origines multiples des causes d'échec. Elles sont de deux sortes :

- inadéquation du linguiciel dans une des phases d'analyse, de transfert ou de génération ;
- mauvaise codification des dictionnaires.

En fait, la mise au point d'un linguiciel consiste à :

- tester d'abord l'adéquation du code et des dictionnaires au modèle linguistique spécifié,
- enrichir ensuite ce modèle au fur et à mesure que sont rencontrées dans les jeux d'essai des constructions linguistiques qui n'étaient pas initialement prévues.

Cet enrichissement implique donc un très important processus d'apprentissage de l'équipe de développement et d'évolution du modèle linguistique. Ce processus est analogue à celui rencontré dans la mise au point de systèmes d'intelligence artificielle.

b - Importance du travail

Pour réaliser une première mise au point d'un linguiciel qui autorise le démarrage de son utilisation opérationnelle, nous considérons, à partir de l'expérience de mise au point de CALLIOPE-AERO, qu'il est nécessaire de soumettre le linguiciel à un volume de tests d'au moins 1.000 pages de 250 mots, soit 250.000 mots à traduire.

Le processus d'organisation des lots de tests peut être le suivant :

- construction d'un lot de 50 pages environ, exécution de la traduction et édition des traces d'exécution ;
- analyse des erreurs de traduction par le futur utilisateur du linguiciel (2 homme x jours de travail) ;
- analyse par l'équipe de développement du modèle linguistique des causes d'erreurs préalablement détectées (10 homme x jours) ;
- correction des erreurs, soit au niveau du modèle linguistique, soit au niveau de l'indexation des dictionnaires, réalisée conjointement par l'équipe de développement et par l'utilisateur (4 homme x jours).

S'il faut ainsi consacrer 16 jours de tests pour un lot de 50 pages, il faut prévoir environ 320 jours, soit 16 hommes x mois, pour la mise au point de 1000 pages.

c - Difficultés spécifiques rencontrées

La principale cause de ralentissement dans la mise au point n'est pas à chercher dans l'identification des causes d'erreurs, qui sont assez facilement détectables à la lecture comparative du texte source et du texte traduit. Elle réside dans la multiplicité des formes d'erreurs qui peuvent provenir soit de la spécification, soit du processus d'analyse, de transfert ou de génération, soit d'une mauvaise indexation des dictionnaires, soit enfin d'une combinaison des erreurs précédentes. Or, toute correction sur l'un de ces points ne doit pas remettre en cause la cohérence du nouveau modèle obtenu après correction.

d - Objectifs: méthodes et outils

Il faut donc impérativement pouvoir revenir aux spécifications du modèle et pouvoir les comparer aux traces d'exécution que fournit ARIANE. Toute modification du source ARIANE doit être précédée d'une modification préalable des spécifications. C'est dire que le système de

être précédée d'une modification préalable des spécifications. C'est dire que le système de documentation permettant d'accéder aux spécifications doit être utilisable simultanément avec l'éditeur d'ARIANE.

1.2.6. Maintenance évolutive

a - Contenu de la phase

Lorsqu'un projet se situe dans les CAS 1 ou 2 évoqués au § *1.2.1.* (page 48), il s'agit de récupérer tous les acquis linguistiques d'un précédent projet pour minimiser l'effort de spécification du nouveau projet.

Toute la phase de conception et une partie de la phase de codage vont consister à prendre connaissance de ces acquis, à sélectionner, à la lecture du nouveau corpus, ceux qui restent valables et à les extraire de l'ancienne documentation pour en faire les fondements de la nouvelle. Il s'agit en quelque sorte d'assurer, non pas la portabilité d'un progiciel classique d'une machine vers une autre, mais la portabilité d'un linguiciel concernant le transfert :

- d'un même couple de langues fonctionnant dans un domaine et une typologie vers un autre domaine et une autre typologie (CAS 1) ;
- de l'analyse de la langue source employée en association avec une première génération de langue cible vers celle d'une autre langue cible (CAS 2).

Cette analogie approximative avec les problèmes de portabilité classique en génie logiciel se justifie par le fait que, dans les deux cas, les possibilités sont associées aux mêmes caractéristiques du produit : modularité des spécifications, exhaustivité et qualités de structuration de la documentation, codage dans un langage de haut niveau.

b - Importance du travail

Il n'y a pas de critère pour déterminer le coût de ce portage.

CAS 1 :

- selon que les domaines sont voisins (aéronautique et mécanique) ou éloignés (aéronautique et chimie), le coût d'actualisation du vocabulaire de base à modifier sera très variable.
- selon que les typologies sont voisines (deux types de documentation d'entretien) ou

assez différentes (cas d'une documentation technique par rapport à celui d'une documentation d'utilisation), le nombre de tournures grammaticales présentes dans une typologie et absentes dans l'autre peut beaucoup changer.

CAS 2 :

- Si les deux langues cibles sont "voisines" (passage d'une génération de l'anglais à une génération de l'allemand ou du danois), il est évident que le nombre d'attributs syntaxiques et sémantiques issus de l'analyse et réutilisables pour la nouvelle génération sera important . Dans le cas où nous avons des langues éloignées (comme dans le cas du passage d'une génération de l'anglais à une génération de l'arabe ou du chinois), il n'est pas possible de dire, *a priori*, l'importance des acquis réutilisables.

c - Difficultés spécifiques rencontrées

Un des problèmes principaux dans l'obligation de faire cet inventaire des acquis linguistiques réutilisables est qu'il est conduit à partir d'une lecture du nouveau corpus. Il faut identifier les phénomènes linguistiques qu'il présente, et, pour chacun d'entre eux, examiner dans la documentation du premier linguiciel s'il a déjà été traité ou non. On ne peut pas procéder *a contrario* en examinant chaque élément de cette documentation pour parcourir ensuite le corpus et voir si l'élément reste pertinent.

d - Objectifs: méthodes et outils

Il faut donc disposer d'un logiciel de consultation documentaire, qui optimise les parcours de recherche dans la documentation volumineuse du premier linguiciel. Il doit également permettre de gérer en parallèle plusieurs documentations de plusieurs versions, pour qu'on assure facilement la copie de tout élément de documentation d'un premier linguiciel qui s'avère utile pour le second en cours de conception.

2. Inventaire des outils de génie logiciel nécessaires à la productivité d'une équipe de fabrication d'un linguiciel

2.1. Bilan

A l'issue de l'analyse des spécificités rencontrées dans l'approche de la conduite d'un projet de linguiciel selon les méthodes du génie logiciel, il est possible de dresser un tableau récapitulatif qui regroupe tous les outils de génie logiciel disponibles ou souhaitables.

Phase	Moyens manuels	Outils logiciels disponibles sous ARIANE	Outils logiciels généraux utilisables	Outils logiciels à créer spécifiquement
Etude de faisabilité	Annotation manuelle des corpus	Liste de mots concordances	Manipulation du corpus par traitement de texte	*Système de gestion de la documention Corpus /Phénomènes linguistiques*
Planification	Suivi de projet manuel	----	Tableur, PERT	Suivi automatique de la production lexicographique
Formation des équipes	Manuels de présentation	Maquettes de formation	----	Système de gestion de toute la documentation technique
Conception	Rédaction de bordereaux de planches statiques	----	----	*Documentation automatique Phénomènes linguistiques Planches statiques*
Codage	----	ATLAS VISULEX	----	Système d'assistance à la codification de lexiques Visualisation synthétique de tout ou partie du lexiques d'un système de TAO écrit en ARIANE
Tests	Etude des listings de trace	Contrôle pas à pas	----	*Edition en parallèle de grammaires dynamiques, traces et planches statiques*
Maintenance évolutive				Système de gestion de toute la documentation technique

Tableau V

Nous avons fait figurer en italique dans ce tableau les deux outils de génie logiciel que nous avons spécialement étudiés pour constituer une première version d'un poste de travail destiné à un "linguicien". Ils font tous les deux appel à une utilisation de la connaissance linguistique sous forme de planches statiques. Examinons plus en détail leurs enjeux fonctionnels avant de décrire au chapitre suivant les mécanismes de leur réalisation.

2.2. Gestion des spécifications initiales

Elle doit assurer la prise en charge de tout le mécanisme de conception, depuis l'étude de faisabilité par analyse du corpus jusqu'à la construction complète des planches statiques.

Il faut donc permettre à un "linguicien" les opérations suivantes :

- Aide à la prise en charge de l'outil par consultation à la demande d'un manuel d'utilisation disponible au poste de travail,
- Etude interactive du corpus stockable et stocké sur support magnétique,
- Classification des principaux mécanismes identifiés au cours de cette étude,
- Planification des planches statiques à écrire en spécifiant initialement leur hiérarchie,
- Ecriture des planches de grammaires statiques sous une double forme de représentation textuelle ou graphique,
- Assurer la coordination de l'équipe de spécification et permettre le cycle "lecture- écriture" en donnant à chaque "linguicien" la possibilité de signer chaque planche qu'il spécifie ou vérifie.

2.3. Gestion des tests et des versions de linguiciels

Nous avons constaté qu'au moment des tests, il faut pouvoir rapprocher les arbres obtenus comme traces de l'exécution d'ARIANE du contenu des planches statiques ayant servi à programmer les grammaires dynamiques correspondantes.

Le poste de travail devra donc être connectable au système ARIANE pour pouvoir récupérer l'exécution des jeux d'essai.

Cette connexion étant un préalable, elle nous permettra alors d'assurer au niveau du poste :

- toutes les modifications des spécifications statiques,
- l'édition de modifications dans les grammaires dynamiques,
- la soumission de nouveaux jeux d'essai.

CHAPITRE III. POSTE DE TRAVAIL LINGUISTIQUE

Dans ce chapitre, nous présentons une méthodologie de travail en montrant comment atteindre les buts définis dans le chapitre précédent. La méthodologie est influencée par les outils utilisés. Nous l'illustrons avec une première version d'un poste de travail linguistique implémenté sur MacIntosh. Cette version n'utilise que des outils informatiques disponibles sur le marché. Les utilisateurs sont des linguistes censés connaître les outils employés. L'accent est mis sur l'utilisation des outils informatiques dans la réalisation de tâches linguistiques. L'aspect linguistique des tâches n'est donc pas développé ici.

Dans l'introduction, nous définissons le scénario suivi et les tâches faisant l'objet de l'illustration. Dans la section 1, nous présentons brièvement les outils informatiques employés et les structures de données utilisées. Les autres sections suivent l'ordre chronologique des tâches, tel qu'il a été défini dans le chapitre précédent, en détaillant plus particulièrement l'écriture d'une planche.

Dans les sections suivantes, nous allons parcourir le chemin grisé de la figure ci-dessous qui illustre le scénario (simplifié) de la production d'un linguiciel pour une langue donnée.

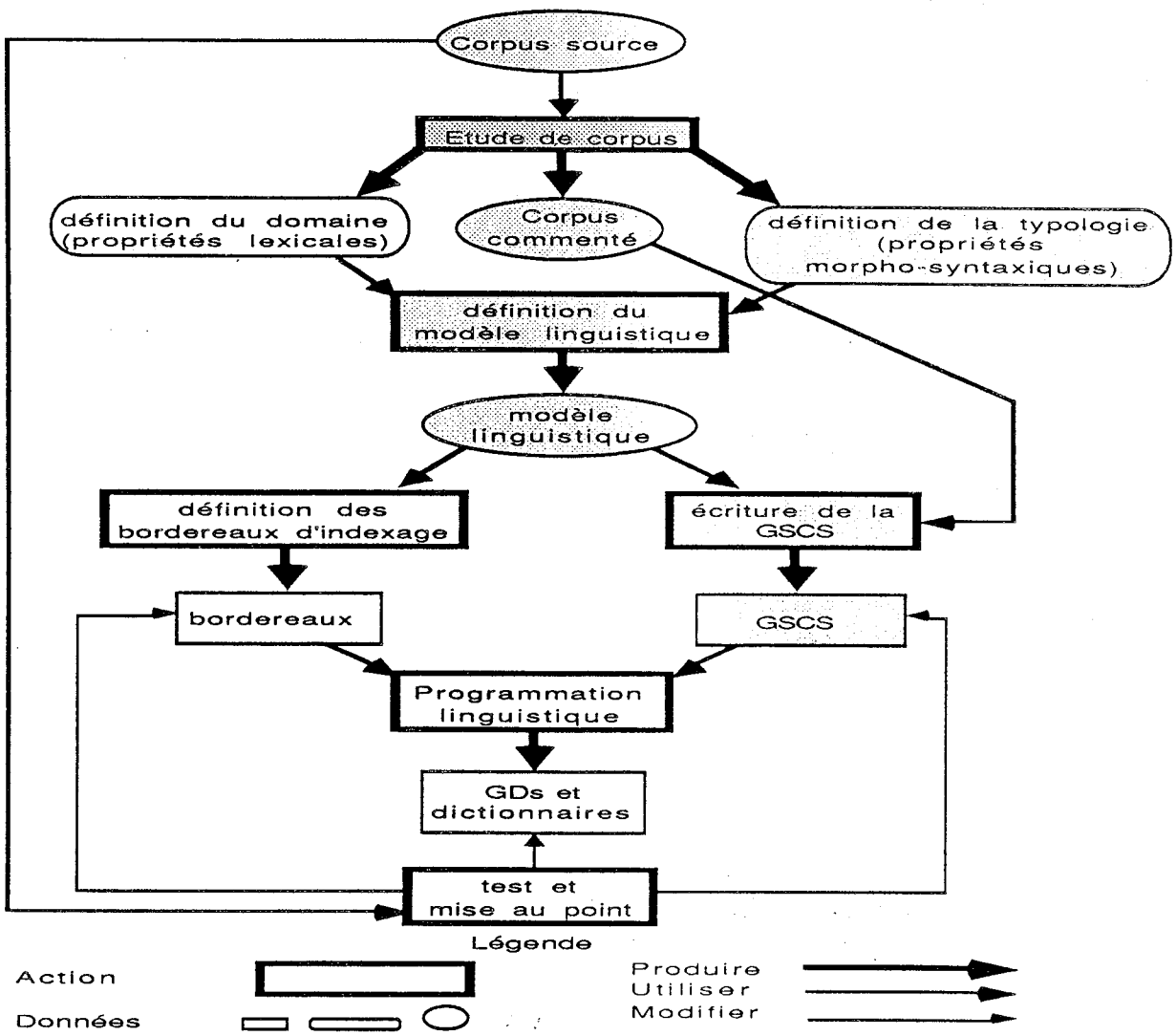


Figure 1 : Scénario simplifié d'utilisation du poste de travail pour l'élaboration d'un modèle linguistique monolingue. Les points en grisé font l'objet de l'illustration.

L'étude du corpus source fournit trois documents contenant :

1) la définition du domaine (qui concerne essentiellement les propriétés lexicales des textes du corpus),
2) la définition de la typologie (qui concerne essentiellement les propriétés syntaxiques),
3) le corpus commenté de manière à en faciliter la consultation (les cas représentatifs sont soulignés).

En tenant compte des propriétés lexicales et structurales recueillies et en utilisant des connaissances en linguistique computationnelle, la définition du modèle linguistique fournit un document qui décrit les structures linguistiques qui seront utilisées dans le linguiciel pour la représentation des textes. La GSCS établit les correspondances formelles entre les textes et leurs représentations linguistiques. La programmation linguistique utilise la base lexicale (créée à partir de bordereaux) et la spécification linguistique (GSCS et stratégies de désambiguïsation) pour produire des programmes exécutables, dont les composants principaux sont des dictionnaires et des grammaires dites dynamiques. La partie test et mise au point consiste à exécuter les programmes sur une partie des corpus sources (jeux de test) et à corriger les programmes (et éventuellement la GSCS et les bordereaux) selon les résultats.

Le scénario est principalement une boucle activité-produit : l'action produit ou modifie des objets. Parallèlement, il y a deux aspects, dynamique et statique, dans la méthodologie, et donc dans le poste de travail. L'aspect dynamique est l'évaluation des objets : comment les manipuler (produire, modifier, utiliser, etc.). L'aspect statique est l'organisation ou le stockage des objets et la structure interne de chaque objet. Le poste de travail se compose d'un ensemble d'outils matériels et logiciels facilitant les actions, et d'une base de données contenant les objets produits par l'utilisateur ou servant à la production d'autres objets. Parmi ces objets, il y a le fichier 'Plans de travail', qui est un guide méthodologique, linguistique et informatique, et le fichier 'Définition de BSCS' (Base de Spécification de Correspondances Structurales), où l'on définit la position et la nature de chaque objet et des modèles d'objets utilisés fréquemment (GSCS, planche, etc.) pour la création de nouveaux objets du même genre.

Ce chapitre est dérivé du fichier 'Plans de travail'. Présentant les outils employés, la sous-section 1.1. correspond à la partie guide informatique de 'Plans de travail'. La sous-section 1.2. donne un aperçu et les principes de la BSCS dont la définition détaillée se trouve en annexe 1. Les autres sections illustrent une partie des guides méthodologique et linguistique dans les plans de travail.

1. Outils et structures de données employés

1.1. Outils

1.1.1. Matériel

L'environnement matériel nécessite :

- un MacIntosh Plus,
- un disque dur 20 de mégaoctets,
- une imprimante graphique,
- l'accès à partir du MacIntosh à une machine sur laquelle tourne le système ARIANE.

Le système ARIANE auquel nous avons accès tourne sur une machine IBM 4331 sous le système VM/SP CMS. Il peut fonctionner également sur un PC/AT 370. Le poste de travail fonctionne principalement sur le MacIntosh.

1.1.2. Logiciel

Le poste de travail utilise les logiciels suivants :

1) le système d'exploitation du MacIntosh,
2) un traitement de textes : MacWrite,
3) un éditeur d'arbres : More,
4) un gestionnaire de dictionnaires : WinTool ,
5) un programme de communication entre les machines : Kermit.

Ces programmes ne sont pas conçus pour un poste de travail linguistique. Nous ne prétendons pas qu'ils sont les meilleurs, car d'une part ils évoluent eux-mêmes, et d'autre part, il y a de nouveaux produits qui arrivent. Néanmoins, nous montrons qu'une utilisation judicieuse de ces programmes nous permet déjà d'améliorer considérablement l'efficacité du travail.

Il existe un manuel d'utilisation fourni par l'auteur de chaque programme ci-dessus [Apple 85], [Wigginton et al. 85], [Minkley 86], [Gaddas 87], [Catchings et al. 85]. Nous en donnons ici un résumé en mettant en relief les fonctions qui nous seront utiles.

1.1.2.1. Système MacIntosh

Le système d'exploitation du MacIntosh est un produit Apple réputé pour sa convivialité : menus déroulants, interactivité, graphisme, son, souris, multifenêtrage, presse-papiers (tampon accessible partout et facile à modifier via les commandes Couper/Copier/Coller), etc.. Ce qui nous sert surtout c'est d'une part, le menu dont les commandes, entre autres WinTool, sont accessibles à partir de presque tous les programmes d'application, y compris bien entendu MacWrite et More, et d'autre part, l'organisation hiérarchique en fichiers et dossiers présentée dans le bureau électronique sous forme graphique.

Se présentant comme des fichiers analogues aux données de l'utilisateur, les programmes d'application peuvent être appelés par la commande **Ouvrir** dans le menu Fichier ou par un double-clic sur eux ou encore sur un fichier créé par eux. Les commandes du système comme Copier, Effacer, Déplacer, etc. sont très faciles à utiliser pour qui ne connaît pas l'informatique. Les commandes, qu'elles soient du système ou des applications, peuvent être appelées soit par la souris soit par une seule frappe (raccourci) qui peut être définie facilement au gré de l'utilisateur via le programme ResEdit (Ressources Edition). Les raccourcis sont toujours affichés dans les menus. Le programme Switcher [Hertzfeld 86] chargeant plusieurs programmes dans la mémoire centrale du MacIntosh permet d'en utiliser un sans quitter les autres.

1.1.2.2. MacWrite

MacWrite, produit de Encore Systems, est un traitement de texte destiné au MacIntosh. Il hérite la convivialité du MacIntosh et les fonctions classiques pour l'édition et l'impression en général, et possède en particulier les fonctions suivantes :

- mise en page directe sur l'écran et identique à celle de l'impression ;
- recherche et copie d'une chaîne via les commandes **Couper/Copier/Coller** ;
- disponibilité de différentes polices (mathématique, cyrillique, grecque, graphique etc.) ;
- disponibilité de différents styles de caractères (taille, gras, italique, indice, exposant, souligné, etc.).

Comme MacWrite est connu de tous les utilisateurs du MacIntosh, nous ne présentons pas ses fonctions ici, mais seulement une copie d'écran.

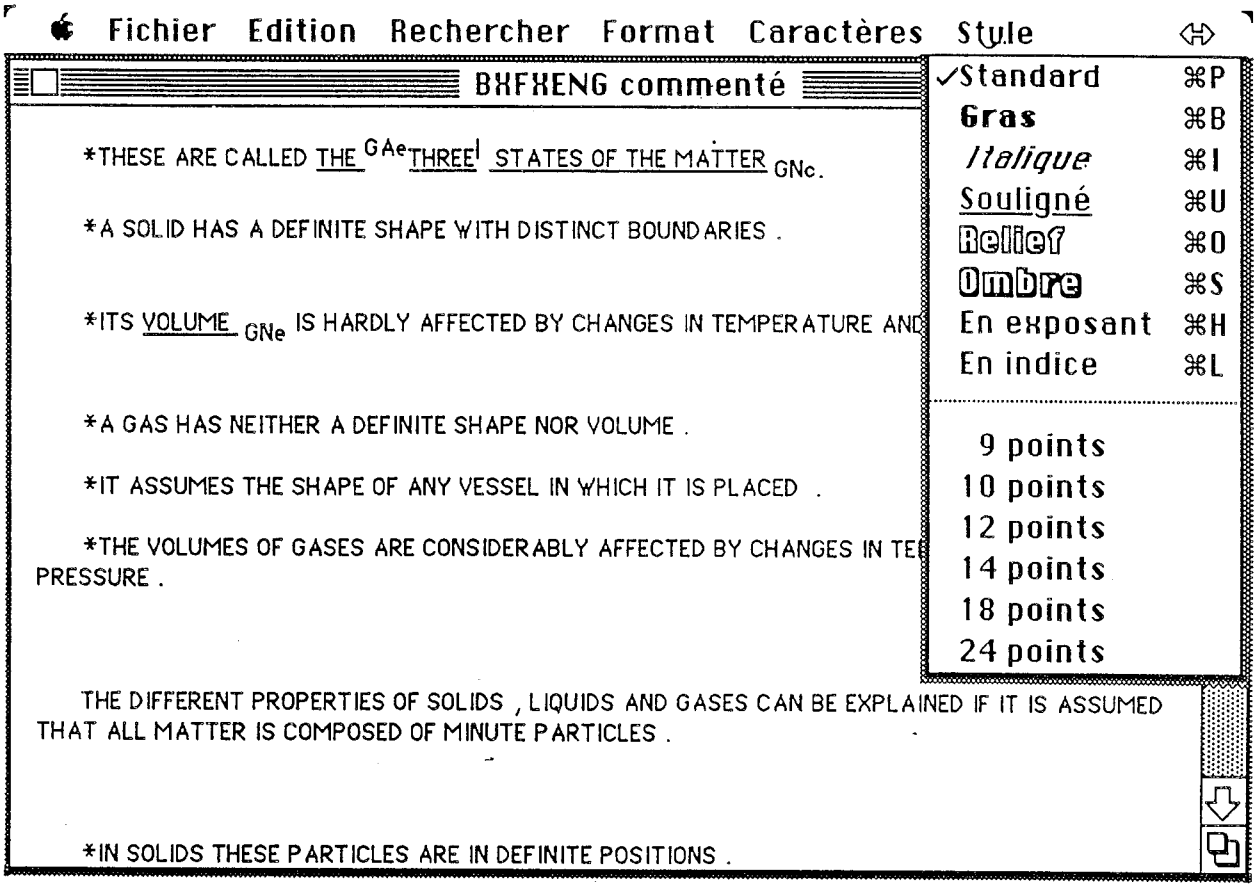


Figure 2 : Une fenêtre MacWrite : exemple de corpus commenté

1.1.2.3. More

More, successeur de Thinktank (Boite à idées) [Hershey 84] est conçu pour l'organisation des idées qui sont ensuite exprimées dans un document. Il permet essentiellement de manipuler des squelettes de documents, où la hiérarchie des composantes du document est clairement indiquée sur l'écran par leurs titres sous l'une des trois formes de visualisation disponibles. Une composante consiste en un titre, une figure ou un texte et éventuellement des sous-composantes ayant la même structure. La structure des squelettes est en fait un arbre. Chaque composante est un sous-arbre. Chaque noeud consiste en un titre et une figure ou un texte.

Les opérandes des commandes de More sont à deux niveaux : arbre et noeud. Au niveau de l'arbre, l'opérande est le sous-arbre courant, où se trouve le curseur. Au niveau du noeud, l'opérande est l'élément sélectionné. Le noeud courant (racine du sous-arbre courant) peut être changé par les quatres touches de flèches : ↑, ←, → et ↓ ou par un clic de la souris sur le noeud.

Nous utilisons More comme un éditeur d'arbre en profitant de ses trois possibilités principales :

- l'affichage graphique d'un arbre à partir de sa forme indentée ;
- l'édition structurée (par sous-arbre) : déplacement, copie, suppression, ajout ... ;
- le multifenêtrage pour la comparaison des arbres.

More est bien adapté à tout ce qui a une structure d'arbre, comme la hiérarchie entre les planches, les squelettes d'arbres, les schémas d'arbres et de forêts, les contraintes fines etc. d'une GSCS. Afin de simplifier la manipulation, nous écrivons une planche entière sous More. Nous pouvons appeler More pour visualiser d'autres éléments qui ne sont pas dans une planche et récupérer le résultat en utilisant les commandes **Couper/Coller[Copier]** via le presse-papiers.

FONCTIONS

Outre le menu , More offre sept menus dont les deux premiers sont communs à presque tous les programmes d'application sur le MacIntosh : File (Fichier) et Edit.

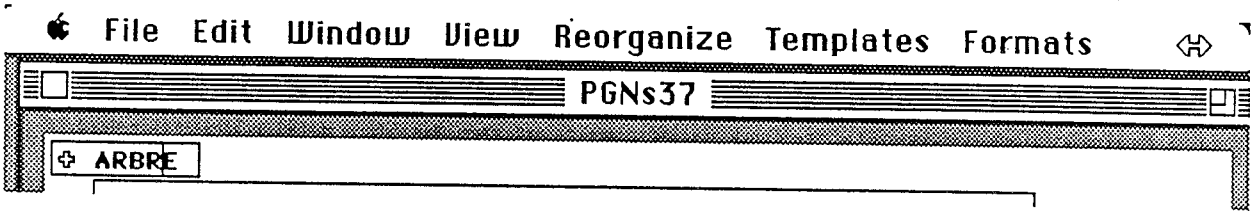


Figure 3 : Menus de More

1) Le menu File contient les commandes de gestion de fichiers. Outre les commandes classiques : ouvrir, fermer et sauvegarder, il y a trois commandes :

- **Send to** : permet de créer un fichier contenant le sous-arbre courant afin de l'envoyer aux autres versions More (Thinktank et IBM/PC) ou de l'intégrer plus tard dans un autre arbre.
- **Receive** : permet de créer un sous-arbre en tant que frère droit du noeud courant s'il est une feuille ou son premier fils, sinon, à partir d'un fichier créé par la commande **Send to** ou d'autres versions.
- **Export** : permet de créer un fichier contenant le sous-arbres courant formaté, afin qu'il soit lisible par d'autres types d'applications comme MacWrite, MacDraw, MicrosoftWord ou Jazz, ou sans formatage (ne contenant que le texte).

2) Dans le menu Edit, il y a d'abord les trois commandes figurant dans tous les programmes : **Cut (Couper), Paste (Coller)** et **Copy (Copier).** La commande **Clear** permet l'effacement.

Toutes ces commandes prennent comme opérandes aussi bien un sous-arbre entier qu'un élément sélectionné à l'intérieur d'un noeud, par exemple une chaîne de caractères. Dans ce menu, on a aussi les commandes (**Search, Find next** et **Change to)** permettant de rechercher et de changer une chaîne de caractères dans le document. Il y a encore des utilitaires comme la gestion d'agenda, le tampon de date (**Date stamp**) et d'heure (**Time Stamp**), etc..

3) Le menu Window contient les commandes pour la gestion de fenêtres.

Plusieurs fichiers peuvent être ouverts simultanément. Chaque fichier est affiché dans une fenêtre. La taille et la position d'une fenêtre peuvent être réglées via la souris. La commande **Zoom in/out** bascule la taille de la fenêtre courante entre sa taille réglée et la taille maximale (plein écran). Les commandes **Tile vertical** ou **horizontal** divisent verticalement ou horizontalement l'écran en fenêtres ayant la même taille. La commande **Overlay diagonal** met les fenêtres sur la diagonale de l'écran. Les noms des fichiers ouverts sont aussi affichés avec un numéro dans ce menu. On peut rendre courant un fichier en lâchant la souris sur son nom dans ce menu ou en tapant COMMAND+<Numéro-du -fichier>.

4) Les commandes du menu View permettent d'avoir différentes vues sur le fichier associé à la fenêtre courante.

- Les commandes **Expand/Collapse heading** permettent de voir ou de cacher les titres des fils du noeud courant.
- Les commandes **Expand/Collapse document** permettent de voir ou de cacher le texte ou la figure du noeud courant. La commande **Hoist** permet de ne voir que le sous-arbre courant (**Dehoist** fait l'inverse).
- Les commandes **Outline, Tree chart** et **Bullet chart** permettent de visualiser le sous-arbre courant sous l'une des trois formes (voir la figure suivante) : indentée, graphique et par niveau. L'arbre graphique affiché peut être copié dans le presse-papiers, puis collé comme figure d'un noeud sous forme indentée. On peut utiliser MacDraw pour modifier les arbres graphiques produits par More.

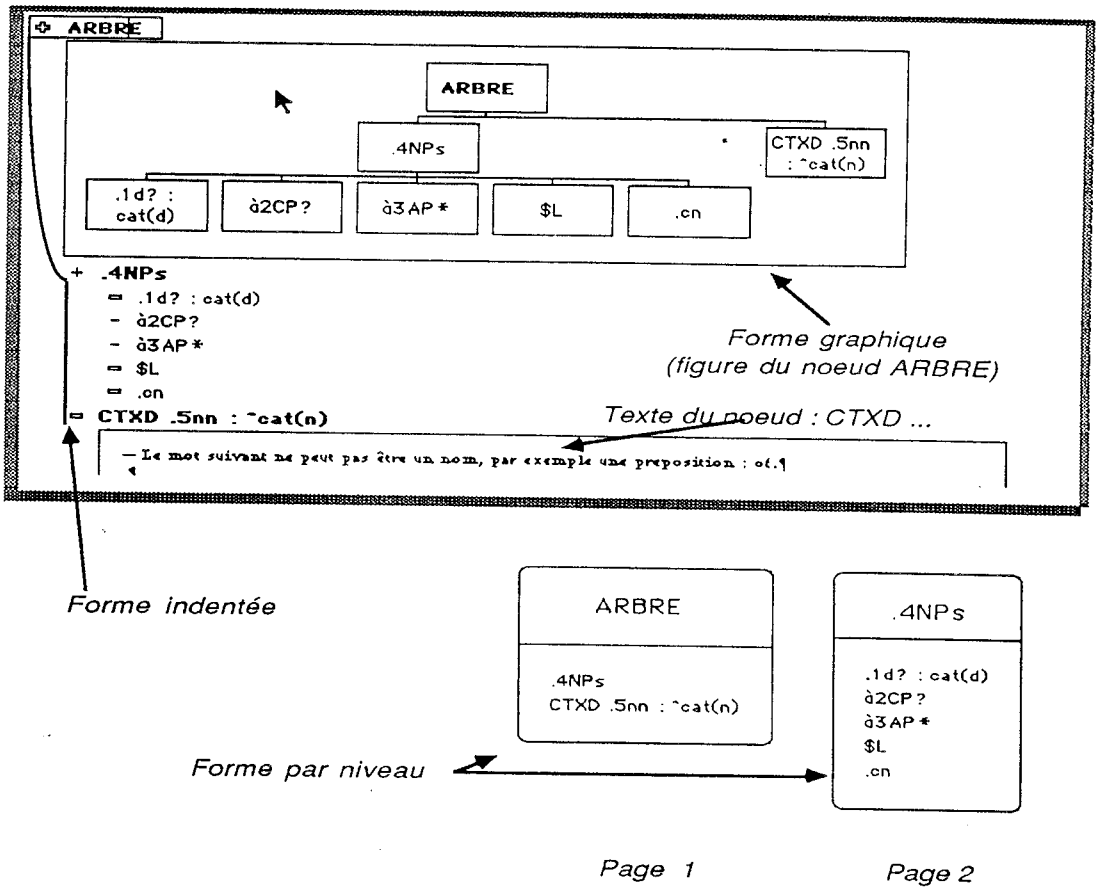


Figure 4 : Trois visualisations d'un arbre

5) Les commandes du menu Reorganize permettent de réorganiser l'arbre dans la fenêtre courante.

Les commandes **Up, Down, Left** et **Right** déplacent le sous-arbre courant en haut, en bas, à gauche et à droite (on peut obtenir le même effet en déplaçant le sous-arbre avec la souris). La commande **Promote** fait que les fils du noeud courant deviennent ses frères droits et la commande **Demote** fait que les frères droits du noeud courant deviennent ses fils. La commande **Clone** rajoute un sous-arbre comme frère droit du sous-arbre courant. Un sous-arbre cloné est toujours identique à son original, même s'il est déplacé et modifié plus tard. Les arbres marqués par la commande **Mark** peuvent être déplacés, clonés ou copiés via la commande **Gather** (ramasser). Les sous-arbres ramassés (gathered) seront placés comme frères droits du noeud courant et dominés par un noeud nommé 'gathered headlines'. Les deux dernières commandes **Sort ascendant/descendant** remettent les fils du noeud courant dans l'ordre alpha-numérique.

6) Dans le menu Templates, on trouve les commandes permettant d'installer ou de supprimer un template (un sous-arbre) et les noms des templates installés.

On retrouve les noms des templates installés pendant la session précédente en rentrant dans More. On peut ajouter un template comme frère de la feuille courante ou comme premier fils du noeud interne courant en lâchant la souris sur son nom dans le menu Templates.

7) Le menu Format contient des commandes définissant divers paramètres de présentation de l'arbre sur l'écran et l'imprimante.

On peut définir

- la taille, le style et la police de caractères pour le titre et le texte de chaque sous-arbre ;
- la marge, l'interligne, l'indentation, la numérotation des titres, l'en-tête, le pied de page et la table des matières, la couleur et le niveau de l'impression ;
- la taille du noeud, la distance verticale et horizontale entre les noeuds, le fond, le contour, etc. de l'arbre sous forme graphique ;
- l'option de l'affichage automatique niveau par niveau de l'arbre (Bullet chart) ;
- des préférences comme mode d'insertion ou de surfrappe, affichage des symboles de tabulation et de fin de ligne, forme de la fenêtre, etc.

1.1.2.4. WinTool

WinTool est un produit de Winsoft conçu pour la consultation et la modification d'un ensemble de fiches, appelé *base documentaire* dans le vocabulaire de son auteur. Chaque fiche est constituée d'une clé (nom de la fiche) et d'un contenu (texte). Dans une base, les fiches sont triées automatiquement par ordre alpha-numérique.

Une base, nommée 'WT modèle' par exemple, se présente comme deux fichiers : 'WT modèle' et 'WT modèleInd' sur le bureau du système MacIntosh. L'un contient toutes les fiches de la base et l'autre les index (clés).

Nous utilisons WinTool comme un gestionnaire de dictionnaires. Nous appelons donc une base un dictionnaire, une fiche un article. Les facilités offertes par WinTool sont :

- le tri et la recherche automatique des informations selon le mot-clé ;
- l'accessibilité à partir de n'importe quelle application.

FONCTIONS

Voici un exemple d'écran, produit par l'appel de WinTool depuis le 'Finder' sur deux dictionnaires 'bloc-note' et 'WT modèle' :

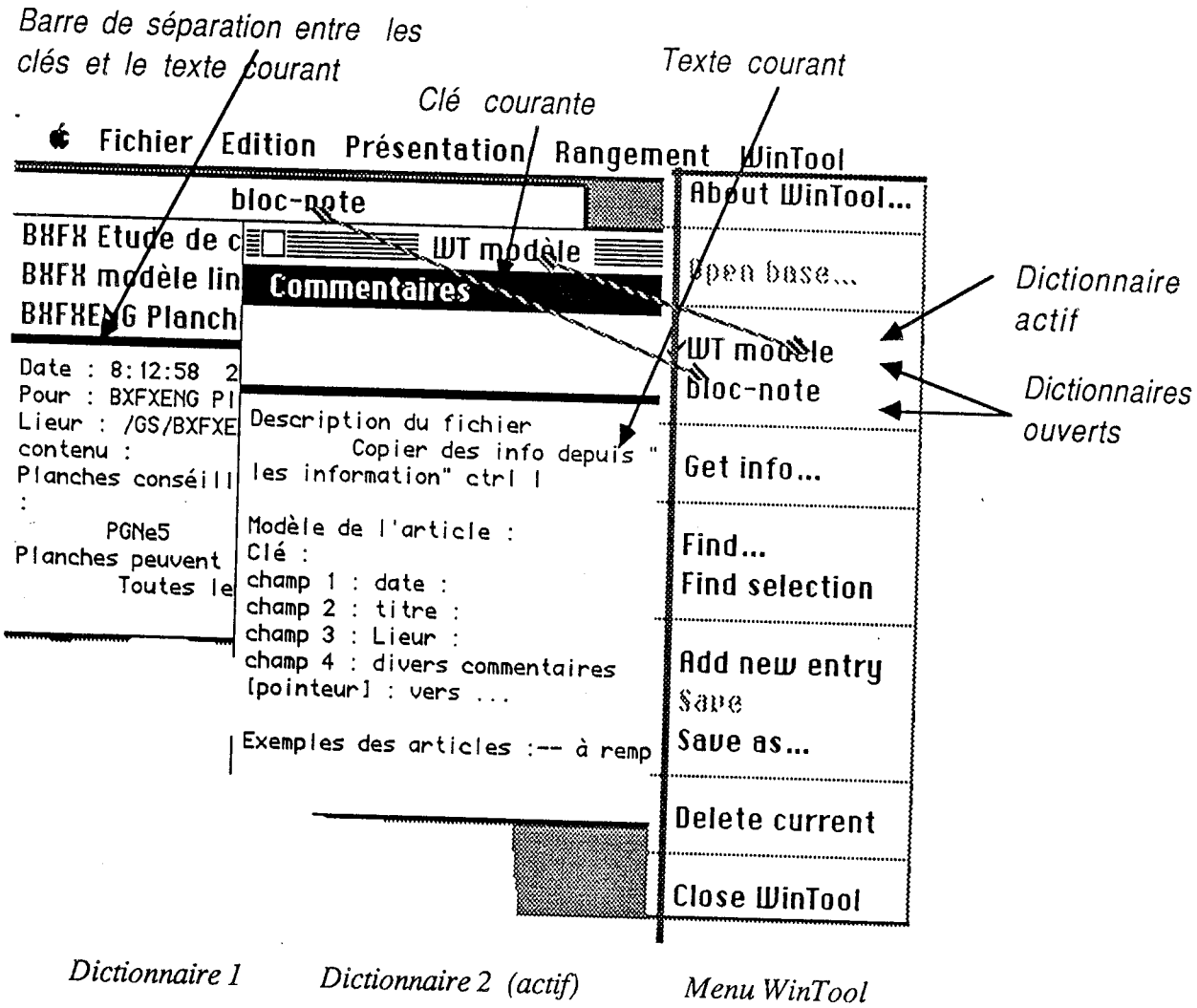


Figure 5 : Menu et fenêtres de WinTool

Dans le menu de WinTool,

- la commande **Open base...** permet d'ouvrir un dictionnaire.
- les noms des dictionnaires ouverts sont affichés (deux en même temps au maximun, la sélection d'un nom rend le dictionnaire correspondant actif).
- la commande **Get info ...** permet d'afficher à propos du dictionnaire activé :
  - . le nombre d'articles contenus ;
  - . le nombre d'articles détruits ;
  - . le nombre d'octets libérés en le compactant.

- la commande **Find ..** permet de rechercher un article selon sa clé donnée par l'utilisateur.
- la commande **Find selection** permet de rechercher un article selon sa clé donnée par une chaîne de caractères sélectionnée.
- la commande **Add new entry** permet d'ajouter un article.
- la commande **Save** permet d'enregistrer l'article courant.
- la commande **Save as ...** permet d'enregistrer l'article courant sous un nom à choisir.

- la commande **Delete current** permet de détruire l'article courant.
- la commande **Close WinTool** permet de fermer les dictionnaires ouverts et de faire disparaître le menu de WinTool.

Lorsque WinTool est appelé (le menu de WinTool est rajouté dans la barre des menus), les commandes dans les menus Fichier et Edition sont aussi accessibles. Les opérandes de ces commandes bien connues de l'utilisateur du MacIntosh sont les objets courants ou selectionnés pouvant être un dictionnaire, un article ou une chaîne de caractères. Tout en gardant la touche OPTION enfoncée, on obtient les effets spéciaux suivants :

- la commande **Couper** enlève l'article courant du dictionnaire actif et le met dans le presse-papiers (1ère ligne : la clé, lignes suivantes : le contenu).
- la commande **Copier** met l'article courant dans le presse-papiers.
- la commande **Coller** ajoute un article dans le dictionnaire actif à partir du presse-papiers dont la 1ère ligne sera prise comme clé et le reste comme contenu.
- la commande **Delete current** détruit l'article courant sans demander de confirmation.

Dans la fenêtre de WinTool, en déplaçant la barre entre la partie des clés et celle réservée au contenu de la fiche sélectionnée, on peut changer les tailles respectives de ces deux parties. On peut aussi utiliser l'ascenseur pour la recherche d'une fiche et cliquer sur la clé d'une fiche pour la selectionner.

Une application auxiliaire de WinTool, **DoctorBase,** permet de créer un dictionnaire, de compacter un dictionnaire (gagner de la place), d'exporter un dictionnaire au MacWrite et de fusionner deux dictionnaires.

1.1.2.5. Kermit

Nous utilisons Kermit pour connecter le poste de travail (MacIntosh) à la machine TAO supportant ARIANE (IBM 4331).

1.2. Structure de données

1.2.1. Principes

Toutes les données contenues dans le poste de travail constituent une Base de Spécifications de Correspondances Structurales – BSCS. La quantité de données dans une BSCS (contenant un seul projet) peut être assez importante (10 Mo) et la nature des objets diverse. Les liens entre objets sont complexes et de nature sémantique. Ces données sont hiérarchiquement structurées dans un arbre des objets qui est matérialisé par les dossiers et les fichiers du système Macintosh. Le schéma de l'arbre des objets est donné dans la figure suivante.

L'arbre des objets est nonordonné (l'ordre des fils d'un nœud n'est pas significatif). Un nœud contient quatre types d'informations :

- son nom et son type,
- une description en clair de la nature de l'objet (champ DESCRIPTION),
- un ensemble d'attributs caractérisant son contenu (champ ATTRIBUTS),
- son contenu.

Le nom est l'identificateur de l'objet. Son type indique le programme qui le gère. Il y a quatre types d' objets :

- DOSS (DOSSier géré par le système Macintosh),
- MW (manipulé au moyen du programme MacWrite),
- More (manipulé au moyen du programme More),
- WT (manipulé au moyen du programme WinTool).

Un noeud interne est toujours de type DOSS.

La description et les attributs sont contenus dans le 'Lire informations' de l'objet, accessible dans le bureau du système MacIntosh.

Le contenu d'une feuille est un fichier du type MW, More ou WT. Le contenu d'un noeud interne est l'ensemble de ses fils.

Chaque sous-arbre est un objet. Un objet est désigné par le chemin de la racine (BSCS par défaut) à la racine du sous-arbre. Par exemple, /GSCS/BXFXeng et BSCS/GSCS/BXFXeng désignent l'objet BXFXeng.

Le modèle hiérarchique permet une organisation modulaire des données. Les attributs peuvent être utilisés pour exprimer les liens entre les données. Bien qu'elle soit encore manuelle, l'utilisation des attributs contrôlables par l'utilisateur facilite la recherche des données. Les fichiers d'index permettent d'avoir une vue logique des données, différente de l'organisation physique (géométrie de l'arbre).

Une définition détaillée de la structure générique des arbres des objets est fournie en annexe II.

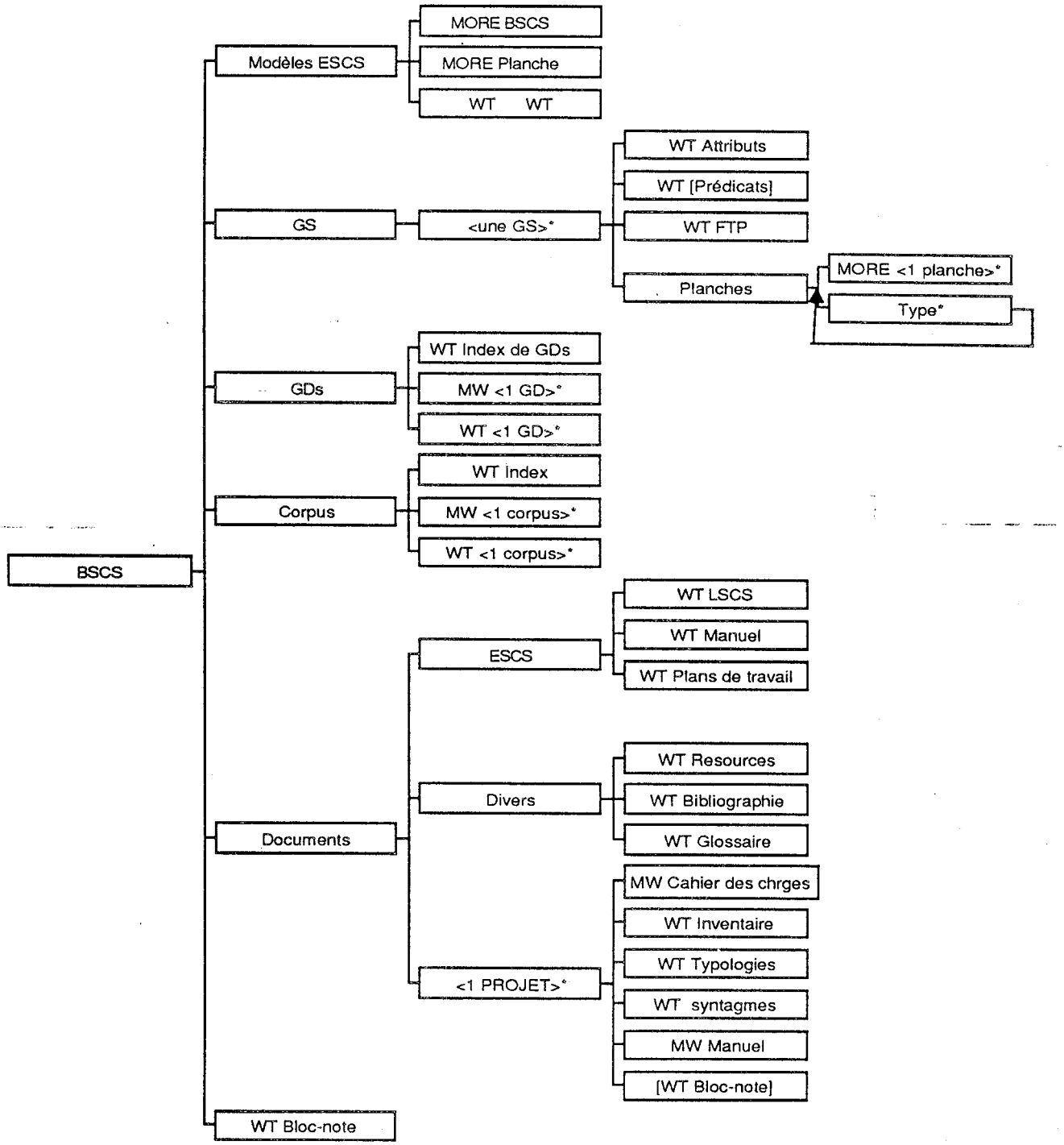


Figure 6 : Schéma de l'arbre des objets

1.2.2. Objets principaux

L'arbre des objets initial consiste en un modèle pédagogique (basé sur le projet BEX-FEX). Il y a un fichier/dossier exemple pour chaque type d'objet différent. La réalisation des tâches va le modifier au fur et à mesure de l'avancement des projets.

La racine de l'arbre des objets a toujours six fils :

1) 'Modèle ESCS',
2) 'GSCS',
3) 'GD',
4) 'Corpus',
5) 'Documents',
6) 'Bloc-note'.

1.2.2.1. 'Modèle ESCS'

Ce nœud est le père de 3 modèles de fichiers utilisés dans ESCS : 'Planche' pour créer une planche en la copiant; 'WT' pour créer un fichier du type WT en le copiant et 'BSCS' contenant la structure générique de BSCS, pour guider la création de divers fichiers dans l'arbre des objets.

1.2.2.2. 'GSCS'

Ce dossier contient toutes les GSCS. Il a autant de fils que de GSCS. Chaque GSCS est constituée :

- du fichier 'Attributs' de type WT pour les attributs d'une GSCS,
- éventuellement du fichier 'Prédicats' de type WT pour les prédicats,
- du fichier 'Formats, Types et Planches' de type WT pour l'organisation de la GSCS,
- du dossier 'Planches' qui peut contenir directement des fichiers du type More pour les planches ou encore des dossiers qui regroupent les planches d'un même type.

Du point de vue linguistique, un ensemble de planches décrit des phénomènes similaires, et du point de vue informatique, il est plus commode de manipuler les planches en les regroupant en modules. Dans un type de planches, il peut donc y avoir encore des sous-types et des planches.

1.2.2.3. 'GD'

On peut éditer des grammaires dynamiques sur le poste de travail et les envoyer à une autre machine pour la compilation et l'exécution. Les spécifications des grammaires dynamiques se trouvent dans la BSCS. Ce dossier contient donc un fichier du type WT 'index de GDs' enregistrant les liens entre les grammaires dynamiques et leurs spécifications. Les autres fichiers sont des fichiers des types MW et WT contenant les programmes sources.

1.2.2.4. 'Corpus'

Dans ce dossier 'Corpus', on trouve des textes représentatifs des documents à traduire. Les corpus sont classés selon les langues, les domaines et les typologies. Le fichier 'Index' du type WT permet de trouver rapidement les textes que l'on cherche. Un corpus peut être stocké dans un fichier de type MW ou de type WT. Il y a des corpus sources et des corpus commentés.

1.2.2.5. 'Documents'

Ce dossier contient toute la documentation de la base qui est divisée en sous-dossiers. Il y a deux dossiers spéciaux : 'ESCS' et 'Divers'. Le premier est la documentation du système ESCS et le deuxième la documentation des projets (il y a un sous-dossier par projet).

1) 'ESCS'

Le fichier 'LSCS' définit la syntaxe de LSCS. Le 'manuel' et le 'Plans de travail' sont deux fichiers ayant le même contenu, mais une présentation différente. Le premier a pour but la consultation sur papier, tandis que le deuxième est destiné à la consultation en ligne. Le 'Plans de travail' est un guide pour toutes les tâches d'un projet linguiciel. Il sert à appuyer la méthodologie de travail et il fournit une aide en ligne pour les débutants.

Ces plans ont au moins trois utilités :

- guide méthodologique : assurer la méthodologie de travail et définir les normes des produits (la syntaxe des objets n'est pas suffisante).
- guide linguistique : indiquer les données linguistiques, leurs références et leur organisation.
- guide informatique : expliquer certaines opérations de base comme copier un fichier, ouvrir un fichier, etc..

2) 'Divers'

Dans ce dossier, on trouve le fichier 'Ressource' (de type WT) contenant les informations sur toutes les ressources disponibles pour la création ou la modification d'un projet. Le fichier 'Bibliographie' du type WT contient toutes les références bibliographiques citées dans la base. Le fichier 'Glossaire' du type WT donne une explication succincte des abréviations et des notions utilisées dans la base.

3) Projet

Toute la documentation d'un projet est regroupée dans un dossier de même nom. L'utilisateur peut organiser son dossier à son gré (en créant des sous-dossiers par exemple). Les fichiers définis dans la structure sont des fichiers typiques dont l'utilisation sous cette forme est recommandée.

1.2.2.6. 'Bloc-note'

C'est un fichier du type WT qui sert d'aide-mémoire et permet de stocker des informations temporaires. On peut en créer un pour chaque dossier utilisateur.

2. Etude de corpus

Il s'agit ici de corpus extraits de grandes quantités de textes réels, et destinés à servir de référence et de source d'exemples aux constructeurs des GSCS.

2.1. Construction d'un corpus

Les corpus à étudier sont spécifiés dans le cahier des charges. Les critères pour choisir un corpus candidat à la TAO sont présentés dans le fichier 'Plans de travail'. Il y a deux moyens de construire un fichier corpus sur le Macintosh : support informatique préexistant (disquette, ligne télématique) ou entrée au clavier.

Un corpus contient plusieurs textes. La langue, le domaine et la typologie dans ces textes sont uniques. Ces informations se trouvent dans les attributs du corpus. Le découpage du corpus en textes est effectué selon des critères linguistiques et informatiques. Il est souhaitable qu'un texte corresponde à un petit nombre de phénomènes linguistiques du même genre. Il est plus commode de traduire ou de manipuler (lire, commenter, imprimer etc.) un texte de taille moyenne (de l'ordre d'une demi-page). Les corpus sont des fichiers du type MW. La taille idéale d'un corpus est de environ 10 pages. Les index de tous les textes sont dans le fichier 'Index de textes'.

Il est important de distinguer les corpus sources et les corpus commentés. Les premiers contiennent les textes originaux qui serviront plus tard à tester le système. Les deuxièmes contiennent des commentaires ajoutés pour faciliter la consultation des corpus, lors de la construction du système.

2.2. Méthode de l'étude

Nous ignorons ici l'étude lexicale de corpus qui est en général effectuée par des programmes classiques. Pour l'étude structurale de corpus, nous créons un fichier 'Typologie' où seront enregistrés les résultats de l'étude au fur et à mesure.

Nous lisons le corpus afin de collectionner les phénomènes linguistiques présents. Quand nous trouvons une chaîne de mots représentant un phénomène, nous la soulignons. Nous ajoutons un mot mnémonique en indice après la chaîne comme point de repère. Si à l'intérieur de la chaîne il y a des phénomènes à relever, nous ajoutons en exposant un mot avant et une barre verticale '|' après la sous-chaîne (voir l'exemple 'The three states of the matter' dans la figure suivante).

Nous créons ensuite un article dans le fichier 'Typologie', dont la clé est le mot notant le phénomène. Dans son contenu, nous indiquons :

- le type de phénomènes ;
- une brève description du phénomène ;
- l'exemple du phénomène ;
- la référence de l'exemple : les noms du corpus et du texte et le numéro de la page.

S'il existe déjà un article de même clé, nous pouvons créer plusieurs articles ayant la même clé ou rajouter l'exemple et la référence dans l'article existant. C'est ce dernier cas qui est illustré dans la figure suivante.

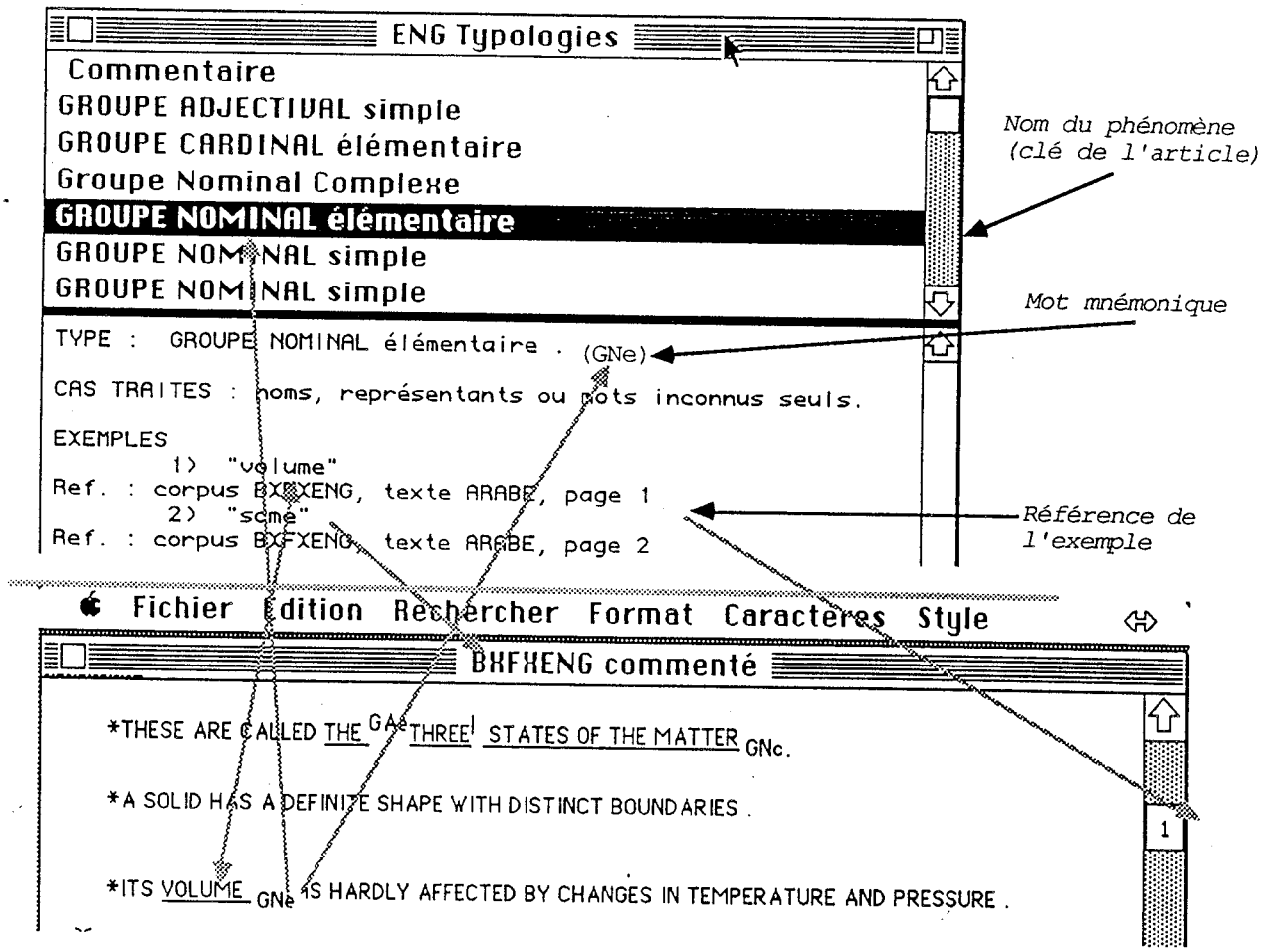


Figure 7 : Corpus commenté et typologie

En se fondant sur les phénomènes linguistiques répertoriés, on produit un document 'Typologie retenue' (MacWrite) qui est le bilan de l'aspect structural de l'étude du corpus. Dans ce document, on classe les phénomènes en constatant les phénomènes rares, fréquents et inexistants dans les corpus par rapport à la langue usuelle, et on consigne les observations ou les remarques pertinentes.

3. Définition d'un modèle linguistique

Dans le cahier des charges du système de TAO à développer, on a opté pour une théorie linguistique et un formalisme de grammaire qui fixent les grands principes du modèle linguistique défini dans cette section. La théorie et le formalisme sont souvent liés l'un à l'autre. Jusqu'à ce jour, les GSCS développées sont toutes fondées sur la théorie de l'école du GETA, baptisée "structure intermédiaire multiniveau", bien qu'il ne soit pas exclu d'utiliser le formalisme des GSCS pour d'autres théories. Dans la théorie du GETA, comme dans beaucoup d'autres d'ailleurs, l'unité de base de traitement est le groupe syntagmatique. Constitué de syntagmes de niveaux inférieurs (syntagmes lexicaux) et récursivement d'autres groupes syntagmatiques, il supporte des fonctions de niveaux supérieurs (syntaxique, logico-sémantique et pragmatique). La structure intermédiaire multiniveau est représentée par un arbre décoré par des attributs.

Quels que soient la théorie et le formalisme, la définition d'un modèle linguistique tient compte des particularités de la langue, du domaine et de la typologie. Fondée sur la typologie retenue, la définition d'un modèle linguistique produira un document 'Modèle linguistique' abordant les sujets suivants :

1) la qualité linguistique,
2) l'architecture et les principes du système,
3) la structure abstraite (formelle) de la représentation linguistique.

3.1. Qualité linguistique

Il s'agit de définir la qualité de traduction et le sous-langage à traiter : les types de phrases acceptées en analyse, les types de phrases produites par la génération (par exemple les phrases actives seulement) et la taille du dictionnaire, etc.

On définit aussi les jeux et le plan de test. Un jeu de test est un corpus source. La taille d'un texte dans ces corpus ne devrait pas dépasser trois phrases afin de faciliter le test.

3.2. Architecture et stratégie

Il s'agit de définir les composantes du système et leurs interfaces (en ARIANE, les six phases). On doit définir les tâches et les principes de chaque composant ainsi que l'interface entre eux.

3.3. Structure abstraite de la représentation

Cette structure doit assurer la compatibilité entre les composantes du système. On doit indiquer :

1) les informations codées par la forme (aspect géométrique) de l'arbre,

2) les informations codées par la décoration (attributs) de l'arbre.

Puisque les groupes syntagmatiques sont les unités de base pour le traitement, on donne aussi les squelettes d'arbre pour les groupes principaux et les informations que leurs racines doivent contenir. Par exemple, la hiérarchie des syntagmes est codée par la relation géométrique entre les noeuds. La fonction syntaxique est indiquée par un attribut dont l'interprétation dépend de la forme de l'arbre .

4. Ecriture d'une GSCS

L'écriture d'une GSCS est décomposée en quatre étapes, illustrées dans la figure suivante :

1) définition des attributs
2) définition des syntagmes
3) organisation des planches
4) écriture des planches

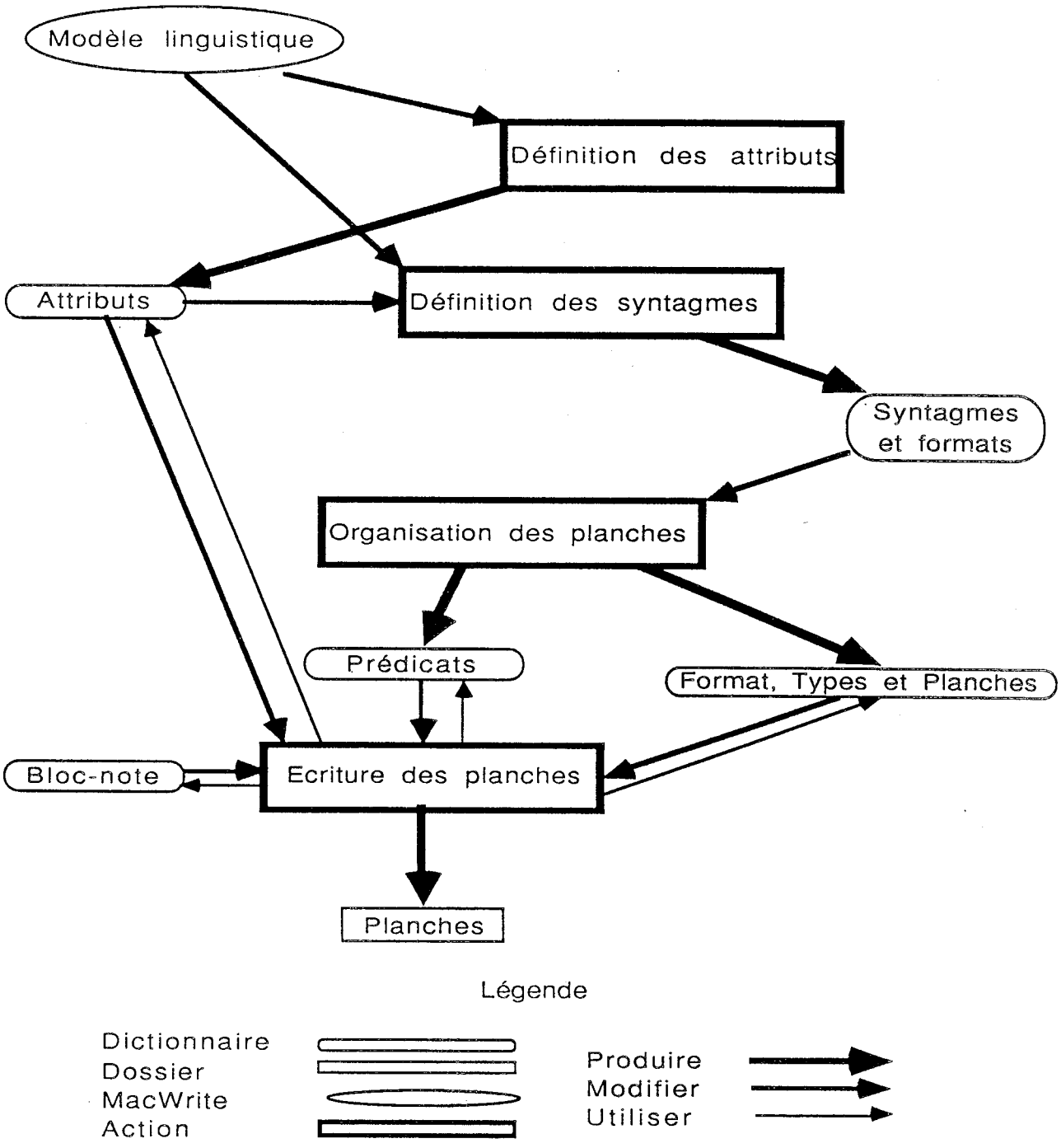


Figure 8 : Ecriture d'une GSCS

4.1. Définition des attributs

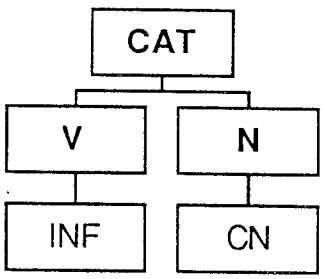
Il s'agit de déclarer formellement les informations devant être codées par les attributs et utilisées dans une GSCS écrite en LSCS. Le document de départ est la partie du modèle linguistique concernant les attributs.

Afin de faciliter la consultation des attributs lors de l'écriture des planches, nous mettons les attributs dans un dictionnaire. Un article correspond à un type d'attribut ou à un attribut. La clé d'un article commence toujours par le nom (du type ou de l'attribut). Le contenu d'un article donne la déclaration du type ou de l'attribut et des commentaires. Un attribut peut être déclaré directement par l'ensemble des valeurs qu'il peut posséder ou indirectement par un type. Un type est déclaré de la même manière qu'un attribut. L'occurrence d'un nom de type représente l'ensemble des valeurs déclarées. Un espace comme premier caractère d'une clé, pour les articles de type d'attribut, les place avant les articles d'attribut, dans le dictionnaire.

Il faut utiliser des identificateurs mnémoniques et succincts pour les noms et les valeurs d'attributs. Il faut commenter chaque attribut et ses valeurs. Des exemples pour les attributs et leurs valeurs sont importants pour faciliter la lecture. Les attributs composés hiérarchiquement CAT(N(CN)) (un mot de la CATégorie Nom et de la sous-catégorie Nom Commun) devraient être utilisés pour des attributs dépendants. Les attributs hiérarchiques, écrits sous forme indentée, peuvent être visualisés sous forme d'arbre par More :

```
+ CAT
   + V
      - INF
   + N
      - CN
```

Forme indentée



Forme d'arbre

Figure 8 : Attribut hiérarchique

Il faut veiller à ce que les attributs soient disponibles ou calculables. Les interfaces entre les composantes définies dans le modèle linguistique permettent de savoir quels sont les attributs qui doivent se trouver dans les arbres axiomes et terminaux.

Voici un extrait d'un dictionnaire d'attributs :

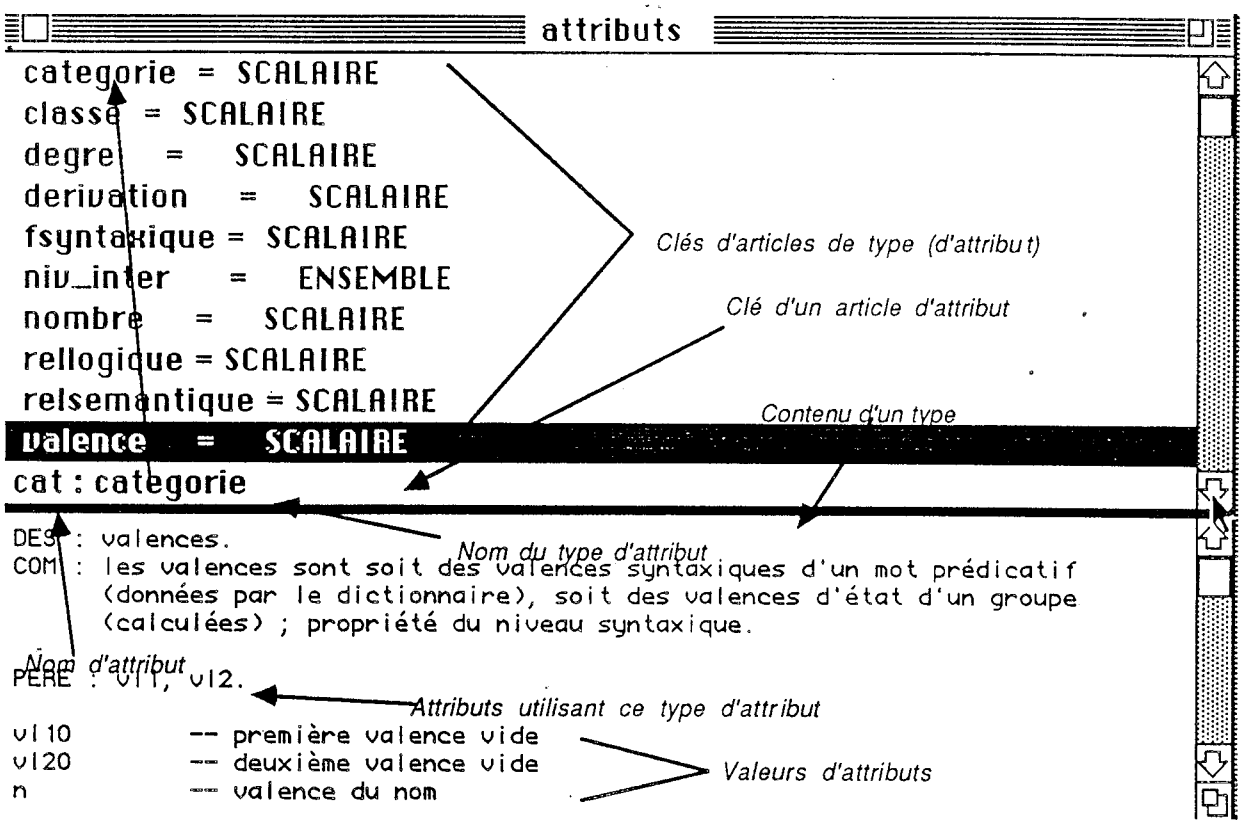


Figure 9 : Extrait d'un dictionnaire d'attributs

4.2. Définitions de groupes syntagmatiques

Nous vérifions la présence d'un exemple pour chaque phénomène qu'on veut traiter dans le modèle linguistique. S'il manque des exemples, on doit d'abord les trouver et les ajouter ensuite dans les corpus et la typologie retenue.

A partir du fichier 'Typologie retenue', nous organisons les informations autour des syntagmes en créant le dictionnaire 'Syntagmes et Formats' . Dans ce dictionnaire, chaque article correspond à un syntagme ou à un format. La clé de l'article du syntagme est le nom du syntagme. Ce type d'article contient :

1) un format (une liste d'attributs) caractérisant le syntagme,
2) une description des cas traités par ce syntagme,
3) des exemples,
4) le squelette de l'arbre représentant le syntagme.

Le choix des formes d'arbre est guidé par le modèle linguistique. Le programme More permet la visualisation de ces arbres.

Un groupe peut être divisé en plusieurs sous-groupes, par exemples groupes nominaux en groupes élémentaires, simples et complexes et les propositions verbales en propositions infinitives, participiales, etc. Pour chaque type, nous indiquons les sous-types, s'il y en a.

Nous définissons les axiomes et terminaux comme deux syntagmes spéciaux. Nous les mettons comme deux premiers articles du dictionnaire (en laissant un espace devant les clés "Terminaux" et "Axiomes").

Chaque syntagme a un format. La définition de ce format peut se trouver directement dans l'article du syntagme ou dans un article indépendant. Outre ces formats caractérisant les syntagmes, on peut aussi définir des formats constitués d'une liste d'attributs utilisés fréquemment ensemble et caractérisant des ensembles de sous-arbres ou de noeuds. Le nom d'un format commence toujours par F ; cela permet de grouper les formats dans le dictionnaire. Certains attributs doivent figurer dans les arbres terminaux et d'autres dans les arbres axiomes. Il est préférable de donner explicitement ces deux listes d'attributs pour connaître les informations portées par les forêts et par les arbres. Nous avons donc deux formats spéciaux : Fterminaux et Faxiomes.

La figure suivante est un extrait d'un dictionnaire de 'Syntagmes et Formats' :

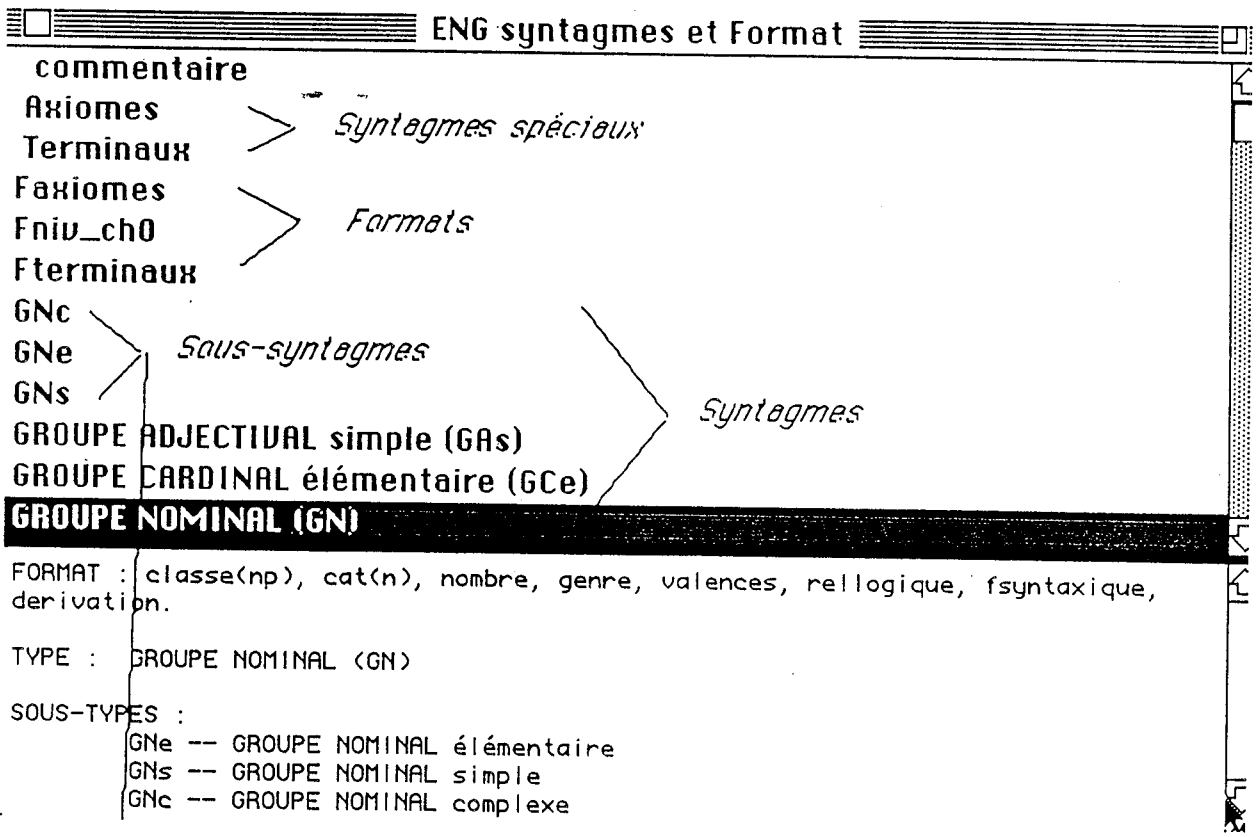


Figure 11 : Syntagmes et Formats

4. 3. Organisation des planches

L'organisation des planches est une conception descendante de la GSCS. On prépare des données : les dictionnaires 'Prédicats' et 'Formats, Types et Planches' et on décide quelles planches devront décrire tel ou tel phénomène.

4.3.1. 'Prédicats'

Des prédicats du 1er ordre sont utilisés dans les planches. Nous créons un dictionnaire 'Prédicats' pour des combinaisons de contraintes qui vont être utilisées lors de l'écriture des planches. Le nom d'un article est constitué du nom du prédicat et de ses paramètres. Le contenu est une expression booléenne.

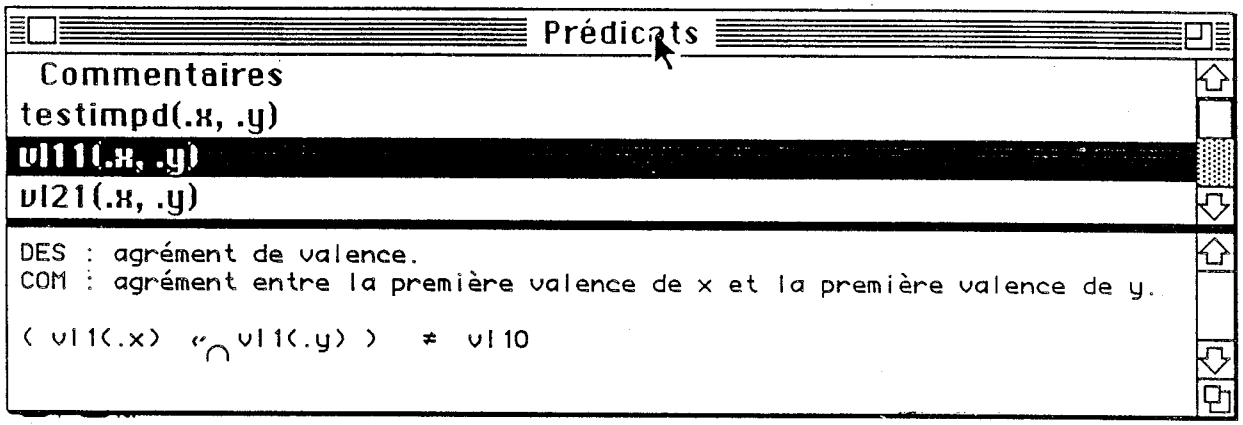


Figure 12 : 'Prédicats'

4.3.2. 'Formats, Types et Planches'

Ce dictionnaire est obtenu à partir de celui de 'Syntagmes et Formats' en gardant les articles de formats, en transformant les articles de syntagme en articles de type et en rajoutant les articles de planche.

Un groupe syntagmatique est en général décrit par une ou plusieurs planches. A chaque article de syntagme, nous ajoutons les noms des planches qui le décrivent. Les autres informations vont être données dans les articles des planches. Si le nombre de planches pour un groupe est trop important, nous divisons les planches en sous-types. La division est effectuée en fonction de la complexité du groupe ou des phénomènes linguistiques. Un groupe complexe est décrit par un ensemble de planches, chacune décrivant un groupe plus simple.

Comme un syntagme se décompose en d'autres syntagmes, on doit définir une hiérarchie entre les planches. La description de cette hiérarchie met en relief les relations entre les planches. Cette hiérarchie peut être exprimée par un arbre dont un noeud peut être une planche, un type ou un

sous-type de planche. La vue graphique de la hiérarchie est obtenue par More.

Ayant établi la hiérarchie entre les groupes, nous créons un article pour chaque planche. Cet article, dont la clé est le nom de la planche, contient :

1) le type ou le sous-type auquel la planche appartient,
2) les cas traités par la planche,
3) des exemples (au moins un pour chaque cas),
4) le squelette de l'arbre prévu pour le syntagme,
5) les planches auxquelles cette planche se réfère,
6) les planches qui se réfèrent à cette planche.

La plupart de ces informations peuvent être coupées depuis l'article contenant le syntagme correspondant. Les références de planches sont obtenues à partir de la hiérarchie. La figure suivante montre une partie des informations présentes dans l'article de planche :

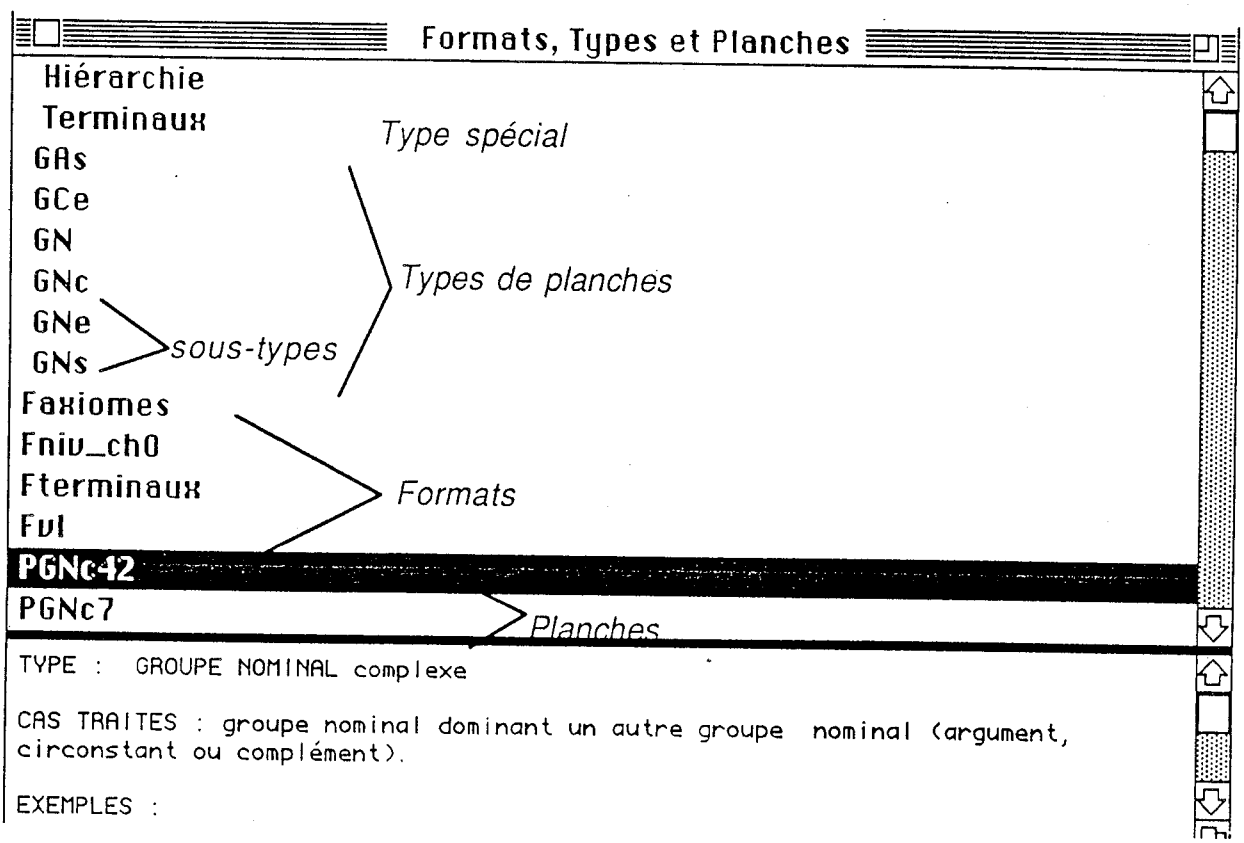


Figure 13 : Formats, Types et Syntagmes

4.3.3. Ordre d'écriture des planches

A cause des références entre les planches, un bon ordre de l'écriture des planches facilite le maintien de la cohérence. Nous donnons les planches dans l'ordre d'écriture :

1) les planches élémentaires,
2) les planches se référant à des planches déjà écrites,
3) s'il y a une boucle de référence (Pa -> Pb -> Pc -> Pa), les planches de la hiérarchie la plus basse ou celles ayant le plus de planches appelantes.

Selon cet ordre, nous choisissons les planches à écrire et mettons leurs noms dans l'article baptisé "<nom de GSCS> Planches à écrire" dans le 'bloc-note' :

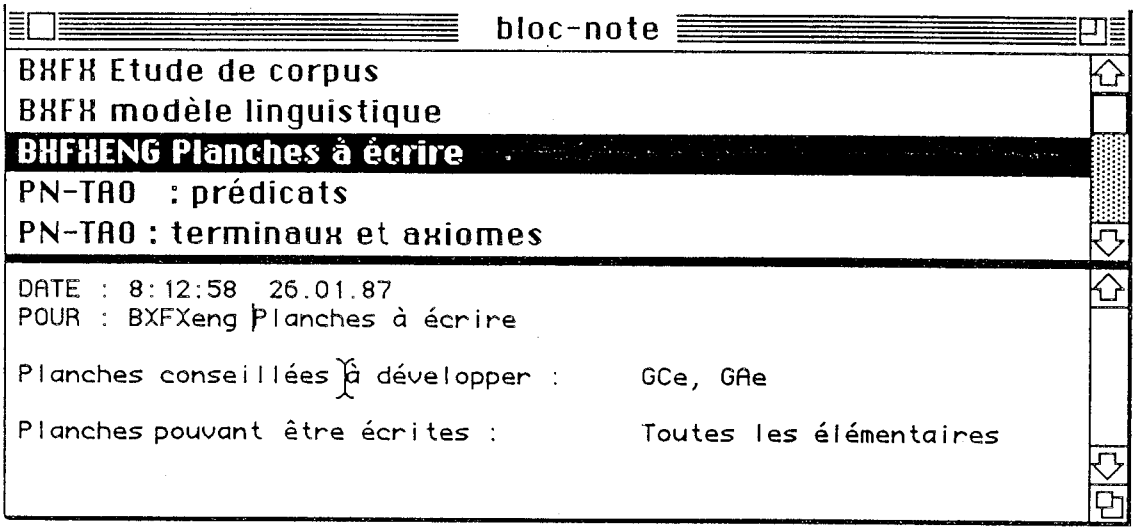


Figure 14 : Utilisation du bloc-note

La conception d'une GSCS se fait d'une manière descendante, tandis que l'écriture des planches est ascendante. Pour l'écriture des planches, la stratégie descendante paraît moins naturelle et plus difficile à suivre, parce que l'on ne sait pas bien préciser les contraintes portant sur des objets qui ne sont pas encore définies.

4.4. Ecriture d'une planche

L'écriture d'une planche est décomposée en sept étapes :

1) Choisir une planche à écrire ;
2) Remplir l'en-tête de la planche ;
3) Construire le schéma de forêt ;
4) Construire le schéma d'arbre ;
5) Vérifier les références ;
6) Etablir les contraintes fines ;
7) Vérifier la planche totale et mettre à jour.

Rappelons qu'en More, un noeud d'arbre a un titre et un texte, ou une figure. Le titre de la racine est le nom de la planche, le texte contient l'en-tête de la planche. La racine a six fils. Le premier contient les références de planches, le deuxième le schéma d'arbre, le troisième les contraintes sur le schéma d'arbre, le quatrième le schéma de forêt, le cinquième les contraintes sur le schéma de forêt et le sixième les contraintes fines. Les titres des racines de ces six fils portent les mots-clés correspondants. En général, nous mettons les informations importantes dans le titre du noeud et les commentaires ou les exemples dans le texte du noeud.

Chaque fils du mot-clé **REFERENCES** consiste en un sous-arbre (figurant dans le schéma) et en une liste de planches qui le définissent. Le mot-clé **ARBRE** domine au moins un fils qui est un schéma d'arbre et éventuellement d'autres pour les contextes. Chaque fils du mot-clé **FORET** est un élément du schéma de forêt. Les fils du mot-clé **CSCHEMA** (Contraintes sur ce SCHEMA) ou **CFINES** (Contraintes FINES) sont des expressions logiques liées par l'opérateur ET. Grâce à la visualisation graphique de More, une expression logique peut s'écrire en arbre ET-OU, plus lisible.

4.4.1. Choisir une planche à écrire

Pour choisir une planche à écrire, nous lisons l'article "BXFXeng Planches à écrire" dans le 'bloc-note' que nous avons mis à jour après chaque écriture d'une planche. Voici le contenu de l'article :

BXFXENG Planches à écrire
DATE : 8:12:58 26.01.87
POUR : BXFXENG Planches à écrire
Dans /GS/BXFXENG/Planches
Planches à développer : PGNs37

Nous copions le nom de la planche 'PGNs37' dans le presse-papiers avant de le fermer et passons dans le menu de More. Nous ouvrons le dictionnaire 'Formats, Types et Planches' et copions l'article de la planche choisie à écrire dans le presse-papiers. Voici son contenu :

PGNs37
TYPE : GNs -- GROUPE NOMINAL simple

CAS TRAITES : groupes gouvernés par un nom commun

EXEMPLES :
1) "THE MOST IMPORTANT COMPONENT"
REF. CORPUS BXFXENG, TEXTE IBM, PAGE 3
2) "UNUSUAL SLOW GAS REACTIONS"
REF. CORPUS BXFXENG, TEXTE EXEMPLE, PAGE 3
3) "THE FIVE MOST INTERESTING POINTS"
REF. CORPUS BXFXENG, TEXTE EXEMPLE, PAGE 3

ARBRE :
.4NPs
.1d?
à2CP?
à3AP*
$L
.cn
-- le mot suivant ne peut pas être un nom.

REFERENCES

.4NPs($L,.cn) : (PGNe5, PGNs7)
à2CP : (PGCe3, PGCs34)
à3AP : (PGAe4,PGAs20,PGAs21) V (PGAc36(..PGAc35))

PLANCHES APPELANTES : PGNs38, PGNc42

4.4.2. Remplir l'en-tête de planche

Nous ouvrons le fichier ' modèle', puis le texte associé à la racine **PLANCHE**. L'écran se présente comme ceci :

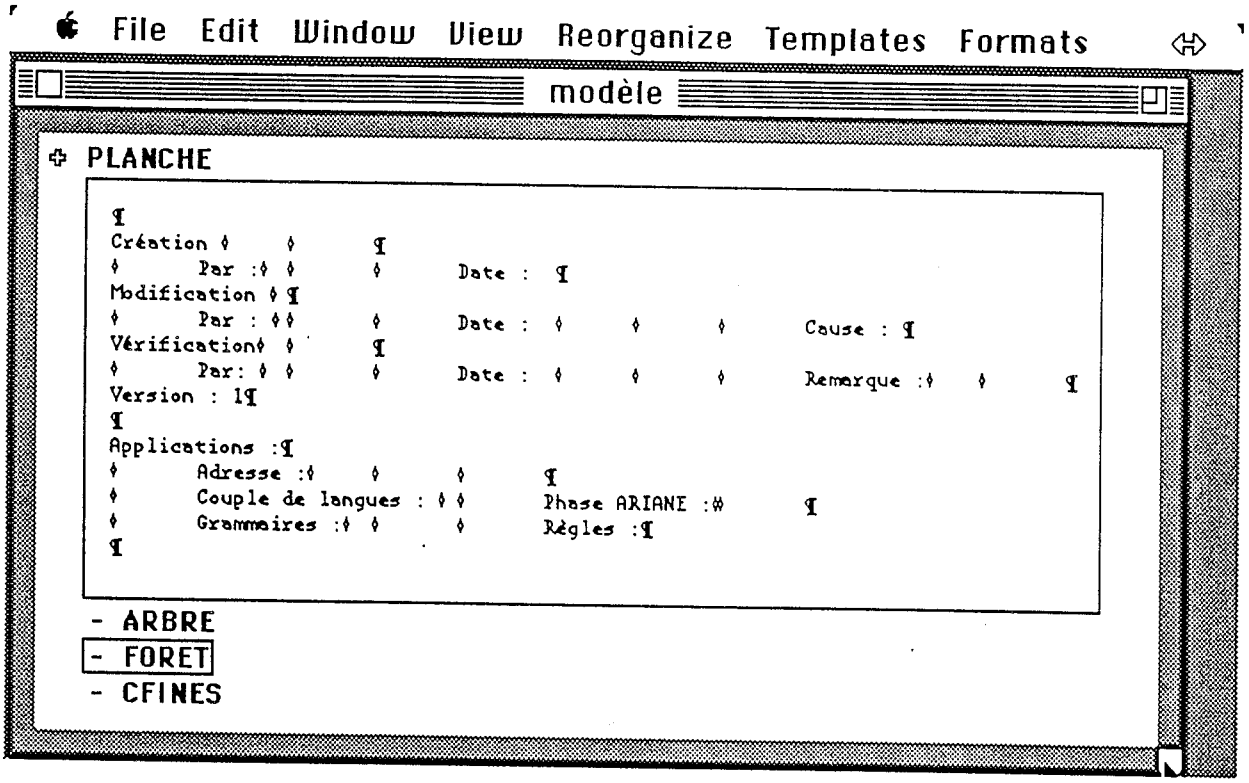


Figure 15 : Modèle d'une planche en More

Nous remplissons tout de suite le nom de l'auteur et la date de création. Nous collons ensuite le contenu du presse-papiers au début du texte de la racine. Nous coupons seulement le nom de la planche et le collons après le mot-clé **PLANCHE.**

4.4.3. Construire le schéma de forêt

Après avoir lu l'article copié, nous essayons de trouver un schéma de forêt pouvant représenter les exemples selon les cas traités, les références de la planche et le schéma d'arbre. Nous réglons la taille et la position de la fenêtre sur le texte de façon à voir l'exemple, l'arbre et le titre **FORET** en même temps.

Nous voyons que les deux exemples commencent avec l'article THE qui appartient à la catégorie Déterminant. Nous créons donc le désignateur de noeud .1d? représentant ce déterminant sous le mot-clé **FORET** (Le désignateur d'arbre commence avec à et le désignateur de forêt commence avec $). Le point d'interrogation signifie que cet élément est optionnel ; car dans l'exemple 3) il n'y a pas de déterminant. Nous copions "THE" l'exemple de déterminant et le collons comme commentaire dans le texte du noeud .1d?.

Après le déterminant, il peut y avoir un groupe cardinal. Nous ajoutons le deuxième élément optionnel : à2CP? (CP = Cardinal Phrase en anglais). Nous mettons toujours les exemples correspondants dans le texte associé au noeud, ici, "FIVE".

Il peut y avoir zéro ou plusieurs groupes adjectivaux (AP = Adjectival Phrase, en anglais) avant le nom commun gouvernant le groupe nominal. Nous les représentons par à3AP* (* et + notent l'itération libre et l'itération positive).

Le dernier élément du schéma de forêt est un groupe nominal ne contenant que des noms parmi lesquels il y a un nom commun qui est le gouverneur. Cette structure a déjà été définie par les planches référées. Nous la schématisons comme suit : .4NP($L, .cn).

Un commentaire dans le squelette dit que l'élément qui suit ne doit pas être un nom. Nous ajoutons donc un élément contextuel " CTXD : .nn : ¬cat(n)" et copions ce commentaire dans son texte.

Maintenant nous avons trouvé un schéma de forêt représentant tous les exemples des cas traités et l'avons mis sous le mot-clé **FORET**. L'écran paraît comme suit :

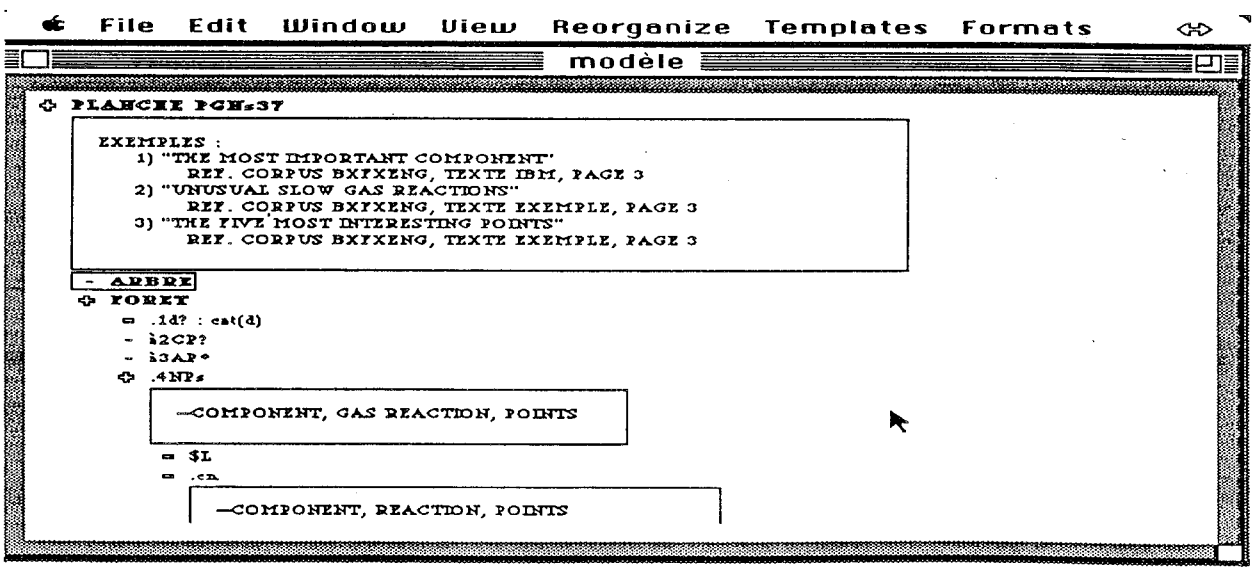


Figure 16 : Schéma d'une forêt

Contraintes propres aux éléments

Il faut encore donner des contraintes pour préciser les forêts pouvant s'instancier avec le schéma. Nous mettons directement les contraintes propres à chaque élément du schéma dans le titre du noeud. Par exemple, la catégorie de l'élément .1d? est déterminant, nous écrivons " .1d? : cat(d)" ainsi " .cn : cat(n) " pour .cn.

Contraintes sur le schéma

Le nombre grammatical entre le déterminant, le goupe cardinal et le nom commun doit être cohérent, nous écrivons sous le mot-clé **CSCHEMA** : "num(.1d?)∩num(.2CP?)∩num(.cn)≠num0". C'est une contrainte concernant plusieurs désignateurs du schéma de forêt.

4.4.4. Construire le schéma d'arbre

Nous allons maintenant construire le schéma d'arbre correspondant au schéma de forêt. Ce schéma doit confirmer le squelette de l'arbre, prévu dans l'en-tête, sinon il faut modifier le squelette, commenter la raison de cette modification dans le fichier 'Formats, Types et Planches' et vérifier la compatibilité avec les planches appelantes. Si la compatibilité est violée, il faut modifier les articles des planches concernées et les planches déjà écrites si nécessaire. Cela n'est pas notre cas.

Au lieu de refaire le schéma d'arbre qui ressemble beaucoup au schéma de forêt, nous copions le schéma de forêt et le collons sous le mot-clé **ARBRE**. Nous le transformerons ensuite en un schéma d'arbre désiré.

En comparant le squelette avec le schéma de forêt, nous voyons qu'il faut monter la racine .4NPs comme racine du schéma d'arbre. Les commandes (**Demote, Up et Promote**) du menu Reorganize nous permettent de le réaliser en quelques frappes.

Nous transformons ensuite l'arbre sous forme indentée en arbre graphique :

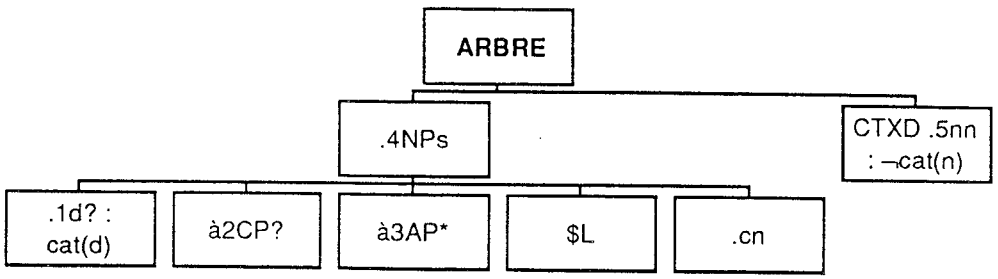


Figure 17 : Schéma d'arbre

Nous copions cet arbre graphique et le collons comme figure associée au mot-clé **ARBRE.** Nous effaçons le squelette de l'arbre dans l'en-tête qui n'est plus utile.

Nous cachons tout le sous-arbre dominé par **ARBRE** et ouvrons sa figure. Nous donnons maintenant des contraintes sur le schéma d'arbre. La relation sémantique (rs) du groupe cardinal à2CP par rapport au gouverneur .cn est quantificateur; nous écrivons donc " rs(à2CP?)=qfier " sous le mot-clé **CSCHEMA.**

La relation sémantique (rs) de à3AP* est 'qualificateur'. Nous écrivons " rs(à3AP*)=? ". Nous avons oublié le code de cette valeur. Nous copions "rs" le nom de l'attribut et ouvrons le dictionnaire des attributs et trouvons l'article rs :

---

rs : relsémantique
PERE : niv_inter
TYPE : relsémantique -- relation sématique

id	-- identique à la relation du noeud père
inst	-- instrument concret
qfier	-- quantificateur
qfobj	-- objet quantifié
qual	-- qualificateur
rs0	-- relation vide

Nous trouvons le code **'qual'** et le copions à la place de '?' .

4.4.5. Vérifier les références

Nous n'avons pas précisé la structure des éléments 2, 3 et 4 qui sont définis par les planches référées. Les noms de ces planches ont été prévus dans l'organisation des planches et se trouvent dans le texte de la racine. Maintenant, nous les rajoutons, via les commandes **Couper/Coller,** juste au-dessous de la racine **PLANCHE,** comme sous-arbre dominé par le mot-clé **REFERENCES.**

Nous vérifions que les noms utilisés dans la référence et dans le schéma de forêt sont identiques. Nous copions aussi des exemples pour chaque élément référant et les mettons dans le texte.

Il faut aussi s'assurer que les planches référées décrivent correctement les exemples. Nous pouvons lire les articles des planches référées dans 'Formats, Types et Planches' ou dans les planches elles-mêmes.

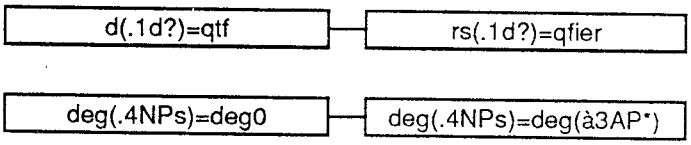
4.4.6. Etablir les contraintes fines

Dans le schéma d'arbre, les noeuds .1d? , .4NPs et la racine de à2CP? doivent avoir le même nombre grammatical qui est l'intersection de leurs valeurs dans le schéma de forêt. Ces trois contraintes se ressemblent. Nous en écrivons une : 'num(.4NPs) = num(F.1d?) & num(Fà2CP?) & num(F.4NPs)' et obtenons les deux autres par copie et modification. Les noms utilisés dans les contraintes fines désignent par défaut les sous-arbres dans le schéma d'arbre. Quand il s'agit d'un sous-arbre du schéma de forêt nous le préfixons par la lettre F.

Les deux expressions suivantes sont écrites en arbre ET-OU sous forme indentée :

```
-----------------------------------------------------------------------
        d(.1d?)=qtr
            rs(.1d?)=qfier
        deg(.4NPs)=deg0
            deg(.4NPs)=deg0
-----------------------------------------------------------------------
```

Voici leurs équivalents sous forme graphique :



Cette correspondance s'interprète ainsi :

"Quand le déterminant est un quantificateur, sa relation sémantique est quantificateur dans l'arbre et quand le groupe nominal dans la forêt n'a pas de degré, le groupe nominal défini par cet arbre a la valeur de degré des groupes adjectivaux et inversement".

Nous mettons les expressions sous forme graphique dans les figures et les interprétations dans les textes des noeuds.

Nous consultons le format dans l'article "GNs" du fichier 'Formats, Types et Planches' afin de vérifier que les contraintes sur la racine .4NPs sont complètes.

Voici cette planche entière imprimée :

PLANCHE PGNs37

TYPE : GNs -- GROUPE NOMINAL simple

CAS TRAITES : groupes gouvernés par un nom commun

EXEMPLES :

1) "THE MOST IMPORTANT COMPONENT"
REF. CORPUS BXFXENG, TEXTE IBM, PAGE 3

2) "UNUSUAL SLOW GAS REACTIONS"
REF. CORPUS BXFXENG, TEXTE EXEMPLE, PAGE 3

3) "THE FIVE MOST INTERESTING POINTS"
REF. CORPUS BXFXENG, TEXTE EXEMPLE, PAGE 3

PLANCHES APPELANTES : PGNs38, PGNc42

Création
Par : Y.YAN Date : Février 1987

Modification
Par : Date : Cause :

Vérification
Par: Date :

Version : 1

Application :

Adresse :

Code langue : Phase ARIANE :

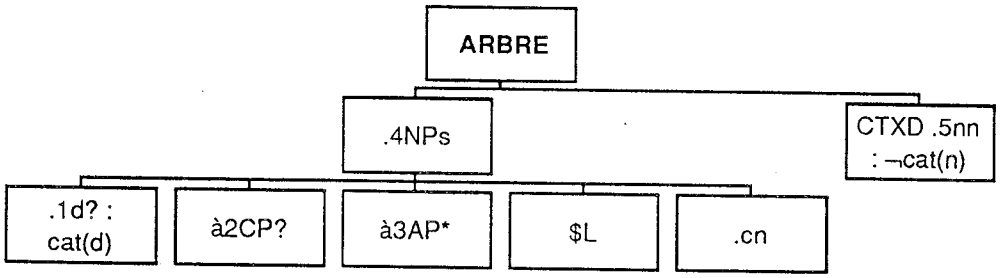
Grammaires : Règles :

REFERENCES

.4NPs($L, .cn) : (PGNe5, PGNs7)
-- points, gas reaction

à2CP? : (PGCe3, PGCs34)
-- five

à3AP* : (PGAe4, PGAs20, PGAs21, PGAc36 (..PGAc35))
-- most interesting

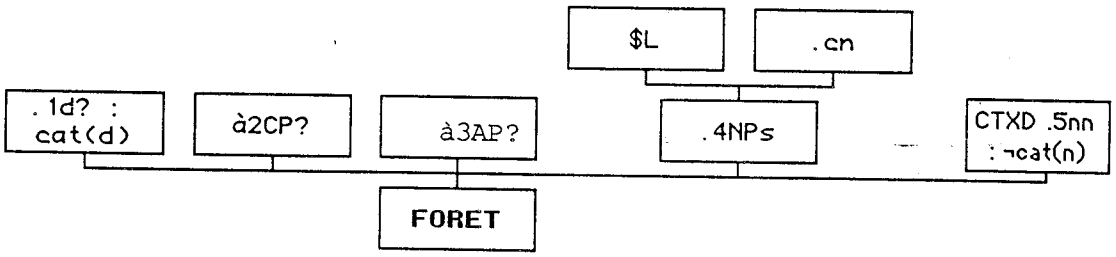


-- Le mot suivant ne peut pas être un nom.

CSCHEMA

rs(à3AP*)=qual -- la fin de ligne équivaut à l'opérateur logique ET.

rs(à2CP?)=qfier



CSCHEMA

num(.1d?) ∩ num(.2CP?) ∩ num(.cn) ≠ num0

CFINES

num(.4NPs)=num(F.1d?)∩num(FàCP*)∩num(F.4NPs)

num(.1d?)=num(F.1d?)∩num(FàCP*)∩num(F.4NPs)

num(à2CP*)=num(F.1d?)∩num(FàCP*)∩num(F.4NPs)

d(.1d?)=qtf — rs(.1d?)=qfier
-- Quand le déterminant est quantificateur, sa rs sera quantificateur.

deg(.4NPs)=deg0 — deg(.4NPs)=deg(à3AP*)
-- Quand le groupe nominal dans la forêt n'a pas de degré, le groupe nominal défini par cet arbre
-- dans cette planche aura la valeur de degré des groupes adjectivaux.

4.4.7. Vérifier la planche et mettre à jour

L'article "Vérification d'une planche" dans 'Plans de travail' définit les tâches et les instructions. Entre autres, pour chaque sous-arbre référent, il doit y avoir au moins une planche référée qui peut le définir. Dans la figure suivante : .4NPs ($L, .cn) est le sous-arbre référent et PGNe5 est la planche référée.

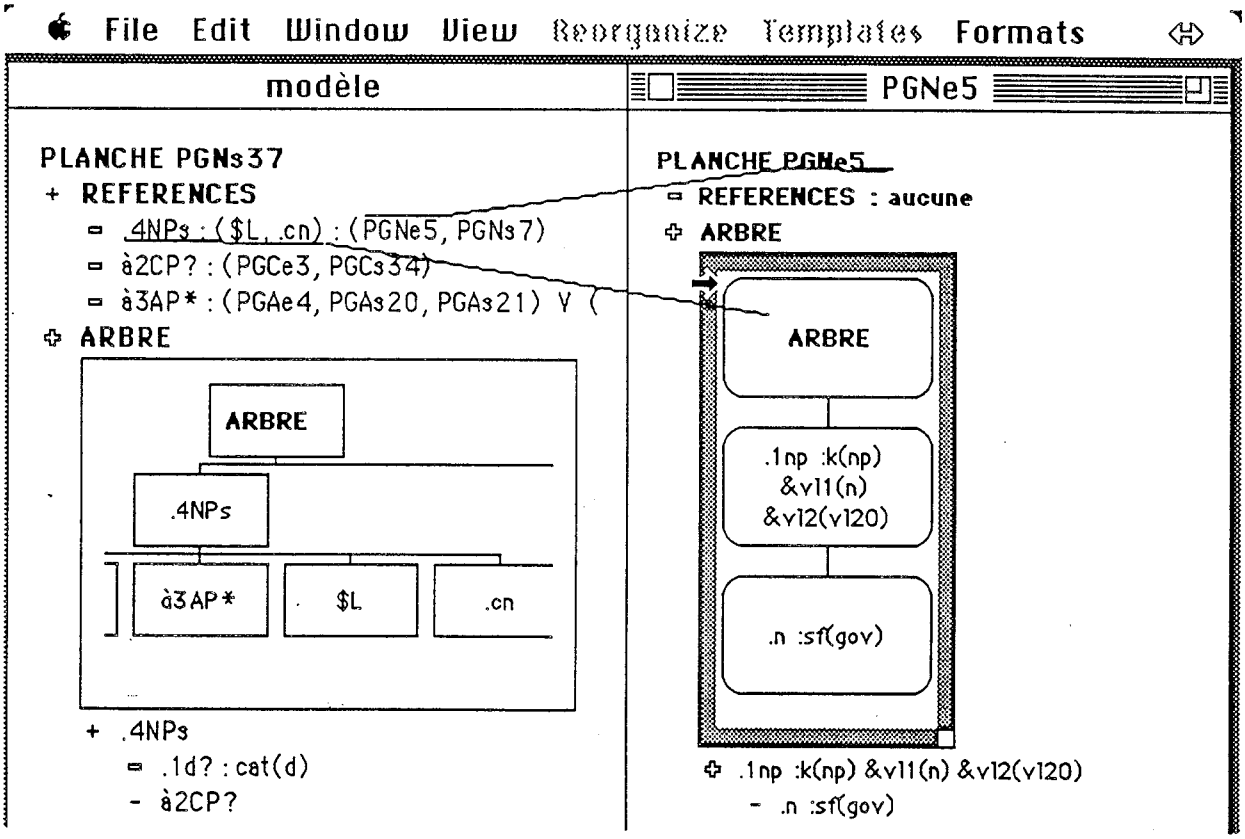


Figure 18 : Une planche et sa référence

Nous enregistrons cette planche dans le dossier de son type (GNs). Le fichier aura le nom de la planche. Dans le fichier 'Formats, Types et Planches', nous changeons l'état de la planche PGNs37 en achevé. Car la planche est la dernière achevée du sous-types GNs, l'état du sous-type GNs devient aussi achevé.

Dans l'article "BXFXeng Planches à écrire" du fichier 'bloc-note' nous rajoutons comme planches à écrire prochainement les planches appelantes de PGNs37 et du sous-type GNs.

4.5. Vérification d'une GSCS

La vérification d'une GSCS doit assurer que :

1) la syntaxe du LSCS est respectée, car l'utilisation de différentes conventions posera des problèmes aux lecteurs de la GSCS.

2) les exigences du cahier des charges et du modèle linguistique sont remplies (tous les phénomènes à traiter sont abordés dans la GSCS).

3) chaque planche a été révisée après la dernière modification, sinon il faut effectuer la révision des planches en question. La révision d'une planche doit vérifier que :

   - les prédicats et les attributs utilisés sont déclarés ;
   - les attributs sur la racine des arbres sont complets (des formats contiennent les attributs obligatoires des arbres) ;
   - les noms des éléments (noeuds et sous-arbres) sont cohérents dans l'arbre et la forêt ;
   - La planche répond aux demandes des planches appelantes et les planches référées existent et décrivent les sous-arbres référents ;
   - la date de vérification et le nom de l'auteur sont mis à jour, et les observations sous la rubrique 'remarque' et le nom de la planche présentant des problèmes non résolus, sont notés.

4) la GSCS s'applique au jeu d'essai afin de garantir la correction des correspondances spécifiées.

5) les problèmes rencontrés sont notés dans un fichier (WinTool, de préférence) .

4.6. Modification d'une GSCS

Il y a deux causes de modification : correction et extension. Dans les deux cas, avant de toucher à la GSCS, il convient de définir d'abord un plan de modification analogue à la conception d'une GSCS. Dans ce plan, noter ce qui doit être modifié et l'ordre des modifications. Il est indispensable de suivre le plan et de le mettre à jour, au fur et à mesure des modifications. L'établissement du plan de modification est fondé sur les remarques de la vérification de la GSCS, de l'écriture et des tests de la GD correspondante, ainsi que sur les notes de maintenance et d'exploitation.

4.6.1. Correction

Les erreurs les plus fréquentes nécessitant une correction sont l'incohérence entre la définition d'un objet et sa référence (son utilisation). Ces objets peuvent être une planche, un prédicat, un attribut ou sa valeur, un désignateur (de noeud, d'arbre ou de forêt). Certaines erreurs sont triviales : par exemple, une faute de frappe, et d'autres sont plus importantes : par exemple, une planche référée ne définit pas un arbre référent.

En corrigeant une erreur importante, il faut veiller à ne pas produire de nouvelles erreurs. Il faut réviser de nouveau la planche après une modification (Ref. révision d'une planche) avant de l'enregistrer.

4.6.2. Extension

L'extension peut nécessiter des actions allant de la construction de nouveaux corpus jusqu'au rajout de planches. Il faut modifier toutes les parties touchées d'une manière incrémentale et essayer de maintenir la modularité.

5. Ecriture d'une grammaire dynamique spécifiée par une GSCS

5.1. Conception d'une grammaire dynamique

La stratégie de construction d'une GD à partir d'une GS peut être décrite comme suit.

Il convient tout d'abord de définir le graphe de contrôle ROBRA, selon la hiérarchie des groupes syntagmatiques et les références entre les planches. Le graphe peut être dessiné en utilisant les facilités graphiques du MacIntosh.

Pour définir le graphe, il faut décrire les sous-systèmes transformationnels, les grammaires et les principales règles, c'est-à-dire préciser leur rôle dans le processus d'analyse ou de génération et, plus simplement, leur donner un nom.

Le parcours du graphe et les stratégies heuristiques employées font partie intégrante du graphe de contrôle et doivent y être décrits avec soin.

Pour faciliter la programmation et les corrections ultérieures, la correspondance entre les planches et les règles de grammaires ROBRA peut également être indiquée.

De plus, il est bon de prévoir un document annexe pour préciser les propriétés que l'arbre objet doit vérifier à la sortie des grammaires les plus importantes.

Enfin, il faut essayer, si possible, de décider du mode d'application de chaque grammaire dès la conception du graphe.

Il existe deux moyens de consulter une GSCS, afin de développer la GD correspondante : en ligne (local ou à distance) ou sur papier.

En général, une planche est réalisée par plusieurs règles ROBRA qui peuvent être appelées dans différentes grammaires ROBRA. Lors de la conception, on peut constater certains problèmes dans la GSCS et demander une justification ou une modification. Il est recommandé de noter les problèmes comme source de modification pour un éventuel contrôle ultérieur et pour aider à la correction de la GSCS. Il est important de modifier d'abord la GSCS avant de continuer, afin d'éviter la conception d'une GD infidèle à la GSCS.

5.2. Programmation d'une grammaire dynamique

Il est important de commenter chaque règle et d'indiquer les planches correspondantes afin que la GD soit compréhensible et puisse être tracée. Pendant l'écriture des règles ROBRA, on a besoin de consulter plusieurs planches en même temps.

5.3. Test d'une grammaire dynamique

Via un programme de communication, actuellement Kermit, le poste peut se connecter à une machine supportant le système ARIANE. Le poste peut simuler un terminal de la machine hôte. Ainsi, toutes les fonctions du système hôte en général et celles du système ARIANE en particulier sont accessibles depuis le poste.

Nous pouvons envoyer une GD à la machine hôte, la compiler et l'exécuter. La trace et les résultats intermédiaires peuvent être enregistrés dans un fichier.

Nous lançons la traduction (ou l'exécution pour une phase) sur les jeux de test. Nous demandons une trace détaillée pour les textes dont la traduction est douteuse. Nous pouvons étudier la trace en utilisant l'éditeur de la machine hôte ou en la récupérant sur le poste. Dans tous les cas, nous pouvons visualiser simultanément les textes sources et traduits, la trace de l'exécution, la GD source, sa spécification (GSCS). Ainsi nous voyons l'évolution du processus de traduction et le rôle de chaque planche.

Nous illustrons par la suite la façon d'utiliser le poste pour étudier une GSCS, sa GD correspondante, les résultats d'exécution et les traces.

Nous mettons dans le Switcher trois programmes d'application : More, MacWrite (DiskWrite) et Kermit.

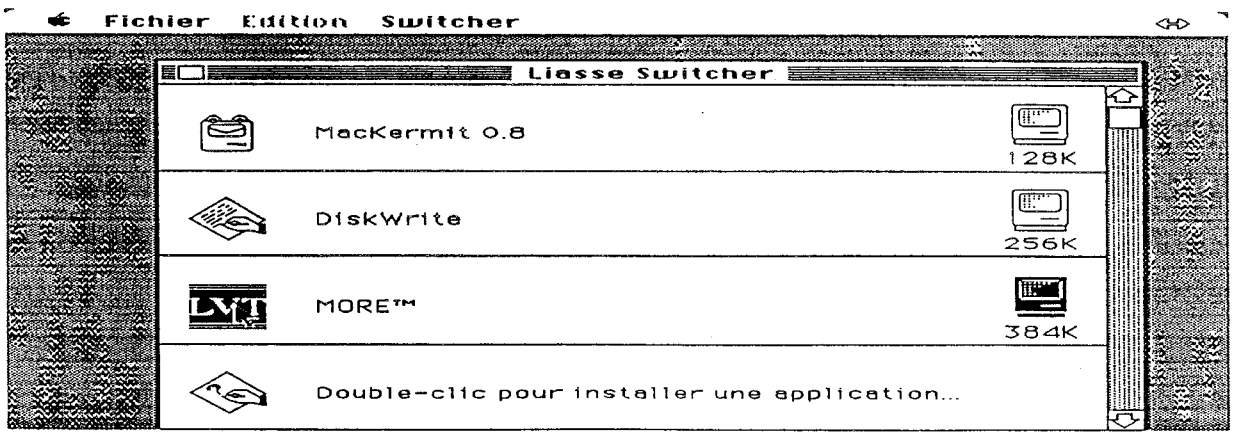


Figure 19 : Outils du poste de travail

Nous appelons Kermit, puis cherchons dans l'article BXFX du dictionnaire 'Index de GDs' des informations à propos des GDs du projets BXFX via WinTool. Cet article (voir la figure suivante) nous donne des renseignements pour l'accès au système hôte. Quand la liaison est établie, nous voyons s'afficher le message "VM/370 ONLINE" de la machine IBM4331.

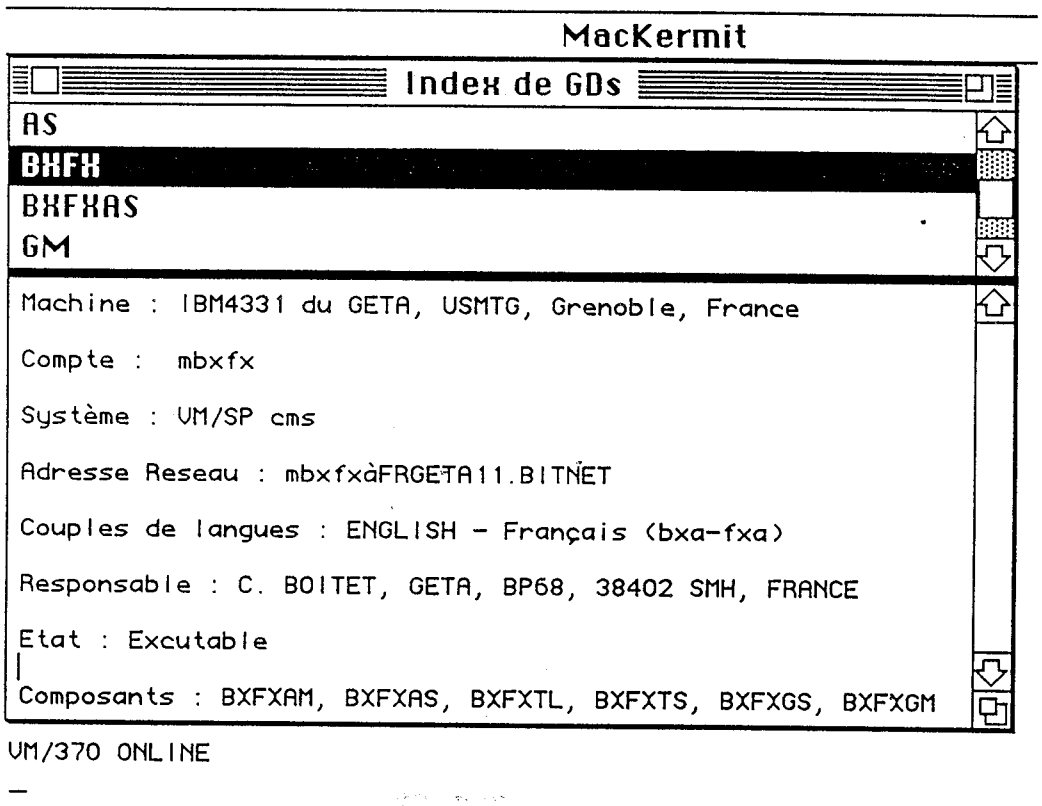


Figure 20 : Index de grammaire dynamique

Nous faisons "logon" dans la machine virtuelle MBXFX et appelons ARIANE. Pendant que la machine s'initialise, nous allons choisir le texte à traduire. Nous voulons tester les planches décrivant les groupes nominaux. Nous passons dans le menu de More et ouvrons la planche nominale complexe : PGNc42.

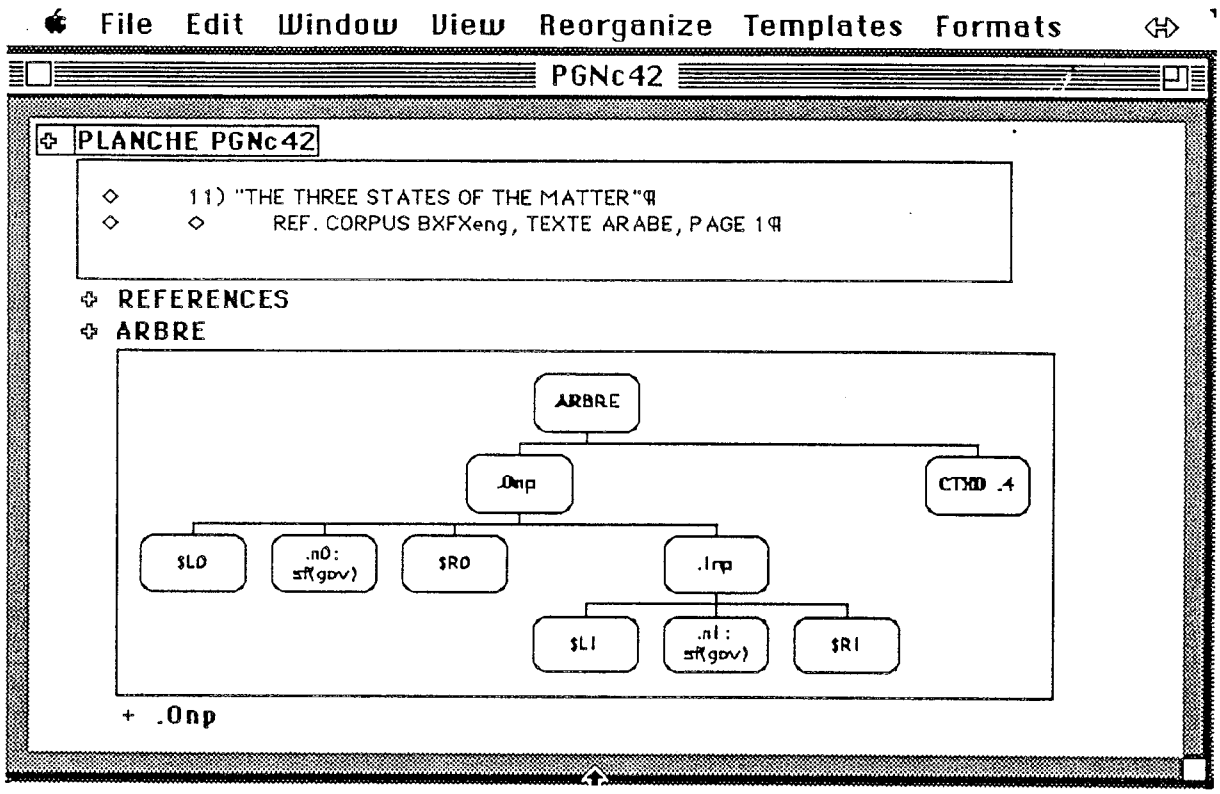


Figure 21 : Planche de groupe nominal complexe

Nous notons la référence de l'exemple "the three states of the matter." et retournons dans le menu Kermit où le système ARIANE est prêt.

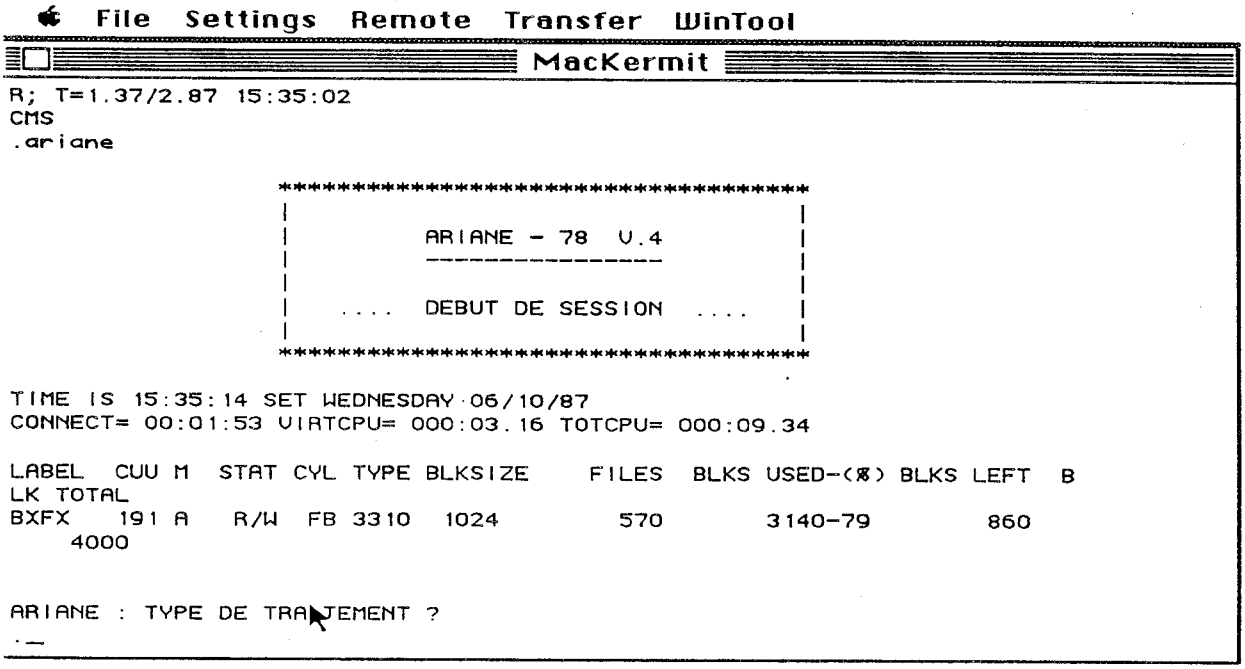


Figure 22 : Début d'une session ARIANE

Nous lançons une exécution partielle sur le texte EXMPL dans le corpus BXFXeng contenant la phrase choisie. Voici le dialogue avec ARIANE (les lignes commençant par un point sont les réponses de l'utilisateur) :

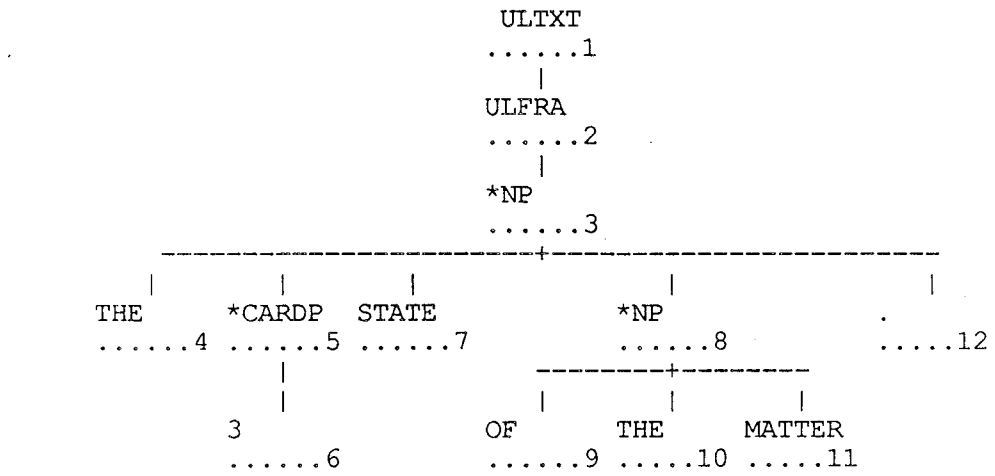
```
. ariane
                    ************************************
                    |                                  |
                    |          ARIANE - 78  V.4        |
                    |          --------------          |
                    |                                  |
                    |    ....  DEBUT DE SESSION  ....  |
                    |                                  |
                    ************************************

TIME IS 11:28:24 SET TUESDAY 04/07/87
CONNECT= 00:00:34 VIRTCPU= 000:01.74 TOTCPU= 000:06.51

LABEL CUU M STAT CYL TYPE BLKSIZE FILES BLKS USED-(%) BLKS LEFT BLK TOTAL
BXFX  191 A R/W  FB  3310 1024    569        3111-78         889      4000

ARIANE : TYPE DE TRAITEMENT ?
. exgm
LANGUE SOURCE ?
. bxa

   LANGUE CIBLE ?
. fxa

NOM DU CORPUS DES TEXTES ( INDICATIF GENERAL ) ?
. bxfxe

ESSAIS : TYPE DE TRAITEMENT ?
. execut ti ti es

NOM DU TEXTE OU NOM TYPE DE LISTE : NOM OU NOM X ?
. exmpl

TL : DONNER LE OU LES NUMEROS DES DICTIONNAIRES PAR ORDRE DE PRIORITE .
. 1 2 3

'EXGM' : NUMERO DE GRAMMAIRE ?
. 0

'EXGM' : RESULTATS . S / ES ?
. es
PARAMETRES INITIAUX DE TRAITEMENT :

'EXAM' : MODE INTERACTIF ? NON / X Y / OU DET .
. non
'EXAM' . IMPRESSION DES MOTS INCONNUS : T , I , TI , NON ?
. non
PARAMETRES POUR LES PHASES :
'EXAM' 'EXAS' 'EXTL' 'EXTS' 'EXGS' 'EXGM' ?
. n a ti a n n  n n  n n n ti

   EXECUTION BEGINS...
   EXECUTION BEGINS...
   EXECUTION BEGINS...
   EXECUTION BEGINS...
```

Nous avons demandé la traduction de la phrase "The three states of the matter" avec la trace de la phase AS. Pendant que la machine traduit le texte, nous repassons dans le menu de More pour étudier les planches. La planche PGNs37 nous intéresse particulièrement. Nous notons les règles APNP et DNP dans la GD spécifiée par cette planche.

La traduction est terminée. Voici le résultat.

```
      BXFXE     EXMPL                              7 AVRIL 1987  11H 35MN 44S

                          LANGUES DE TRAITEMENT: BXA-FXA

                                -- TEXTE ORIGINE --

  the three states of the matter.

      BXFXE     EXMPL

                                -- TEXTE TRADUIT --

   ----- ( TRADUCTION DU  7 AVRIL 1987       11H 35MN 06S ) -----
   VERSIONS : ( A : 10/05/85 ; T : 10/05/85 ; G : 10/05/85 )

  Les trois états de la matière.
```

Resultat de l'Analyse Structurale




```
SOMMET 1   ' ': UL('ULTXT').
SOMMET 2   ' ': UL('ULFRA').
SOMMET 3   ' ': UL('*NP'),K(NP),CAT(N),SUBN(CN),NUM(PLU),SEM(ABST),VL1(N).
SOMMET 4   '*THE': UL('THE'),SF(DES),TYPOG(CAP),CAT(D),SUBD(ART),NUM(PLU).
SOMMET 5   ' ': UL('*CARDP'),
           RS(QUAL),K(CARDP),SF(ATG),CAT(A),SUBA(CARD),NUM(PLU).
SOMMET 6   'THREE': UL('3'),SF(GOV),CAT(A),SUBA(CARD),NUM(PLU).
SOMMET 7   'STATES': UL('STATE'),SF(GOV),CAT(N),SUBN(CN),NUM(PLU),SEM(ABST).
SOMMET 8   ' ': UL('*NP'), RS(QUAL),UNSAFE(RS),K(NP),
           SF(COMP),CAT(N),SUBN(CN),NUM(SIN),SEM(CONC),SEMCO(SUBST), VL1(OF).
SOMMET 9   'OF': UL('OF'),RS(QUAL),UNSAFE(RS),SF(REG),CAT(S),SUBS(PREP),VL1(OF).
SOMMET 10   'THE': UL('THE'),SF(DES),CAT(D),SUBD(ART),NUM(SIN).
SOMMET 11   'MATTER': UL('MATTER'),
            SF(GOV),CAT(N),SUBN(CN),NUM(SIN),SEM(CONC),SEMCO(SUBST).
SOMMET 12   '.': UL('.'),CAT(P).
```

Nous avons mis la trace dans un fichier et la commande de recherche d'éditeur nous permet de trouver dans ce fichier la partie qui nous intéresse. Nous voulons voir l'effet de la planche PGNs37.

Voici l'arbre objet avant et après l'application des règles spécifiées par PGNs37.

ARBRE.4 (0; 1 A, 0 R)

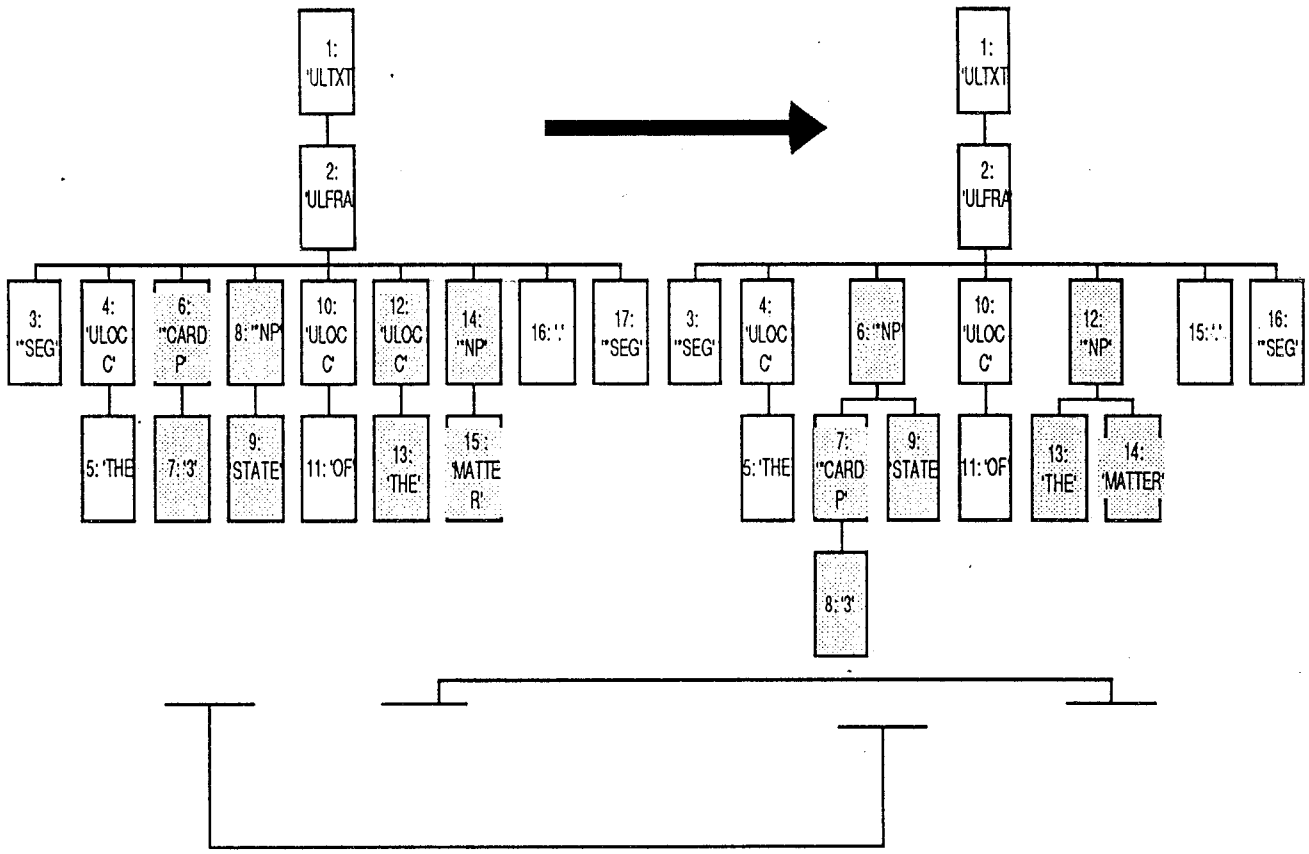
1: 'ULTXT' (2: 'ULFRA' (3: '*SEG', 4: 'ULOCC' (5: 'THE'), 6: '*CARDP' (7: '3'), 8: '*NP' (9: 'STATE'), 10: 'ULOCC' (11: 'OF'), 12: 'ULOCC' (13: 'THE'), 14: '*NP' (15: 'MATTER'), 16: '.', 17: '*SEG'))

.38. GRAMMAIRE NOUNP1 (UB / O) : APPEL EFFECTIF
ARBORESCENCE ARGUMENT:
ARBRE.4 (0; 0 A, 0 R)

APNP : *<<2: 'ULFRA'>> (6: '*CARDP', 8: '*NP' (9: 'STATE'))*
DNP : *<<2: 'ULFRA'>> (12: 'ULOCC' (13: 'THE'), 14: '*NP' (15: 'MATTER'))*

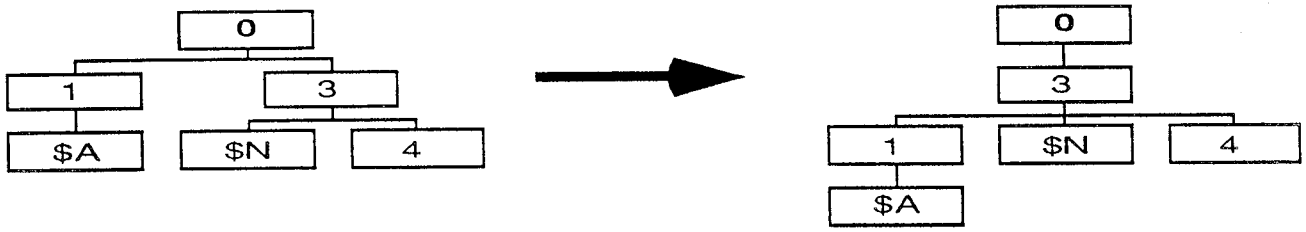
ARBORESCENCE TRANSFORMEE:
ARBRE.5 (0; 1 A, 1 R)

1: 'ULTXT' (2: 'ULFRA' (3: '*SEG', 4: 'ULOCC' (5: 'THE'), 6: '*NP' (7: '*CARDP' (8: '3'), 9: 'STATE'), 10: 'ULOCC' (11: 'OF'), 12: '*NP' (13: 'THE', 14: 'MATTER'), 15: '.', 16: '*SEG'))

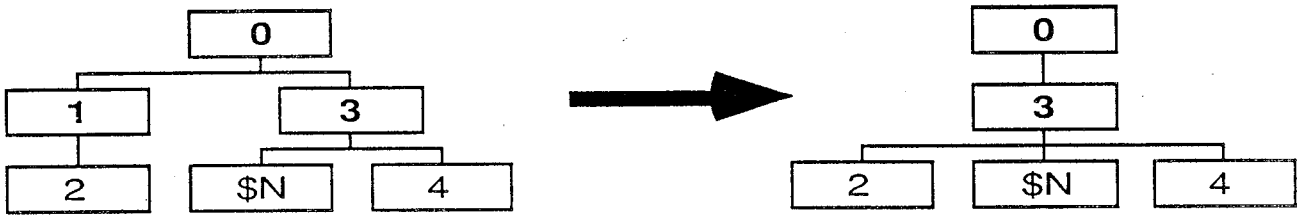


Nous pouvons nous déplacer entre l'écran d'ARIANE et celui de More pour comparer les règles et la planche. Voici les règles de ROBRA (la planche est celle que nous utilisons comme exemple dans la sous-section **Ecriture des planches**) :

```
** AP + NP: AP INCORPORATED IN NP:  ACIDIC OXIDES     REC: 0(3) ON APNP & DNP.

APNP:  (*0,&NIV<3)  0(1($A),*,3($N,4)) /
       1: $AP -OU- $CP;  3: $NP;  $N: SF-NE-DES;
      4: $GOV -ET- (- CAT-NE-R -OU- UL-E-'ONES' -)
      == 0(3(1($A),$N,4)) / /
       1:1, SF:=ATG, RS:=QUAL;
       3:3,  -SI- DEG(3)-E-DEG0 -ALORS- DEG(3):=DEG(1) -SNFSI- .
```




```
** DEICTOR + NP : DEICTOR INCORPORATED IN NP.
**    THIS REACTION;  ONE ATOM;  THE EXTRACTION;  ITS ELEMENTS.

DNP:   (*0,&NIV<3)  0(1(2),*,3(SN,4)) /
       1: $ULOCC;  2: CAT-INC-D -ET- UL-NE-THAT';    3: SNP;
       4: $GOV -ET- (- CAT-NE-R -OU- UL-E-'ONES' -OU- UL-E-'OTHER' -)
      == 0(3(2,$N,4))  /*<--1 /
       2:2, SF:=DES, NUM:=NUM(2)-I-NUM(3),
          -SI- DEG(2)-E-COMP -ALORS- DEG(3):=COMP -SNFSI- .
```




```
** AP + ING --> NP.
```

Nous mettons une partie de la trace et de la GDs qui nous intéresse particulièrement dans des fichiers WinTool afin de les voir simultanément à l'écran. Nous voulons savoir si le résultat de l'analyse structurale est le même que ce que la GSCS a spécifié. Nous allons retrouver dans l'arbre sorti par AS les sous-arbres spécifiés par les planches. Nous commençons avec la planche complexe PGNc42.

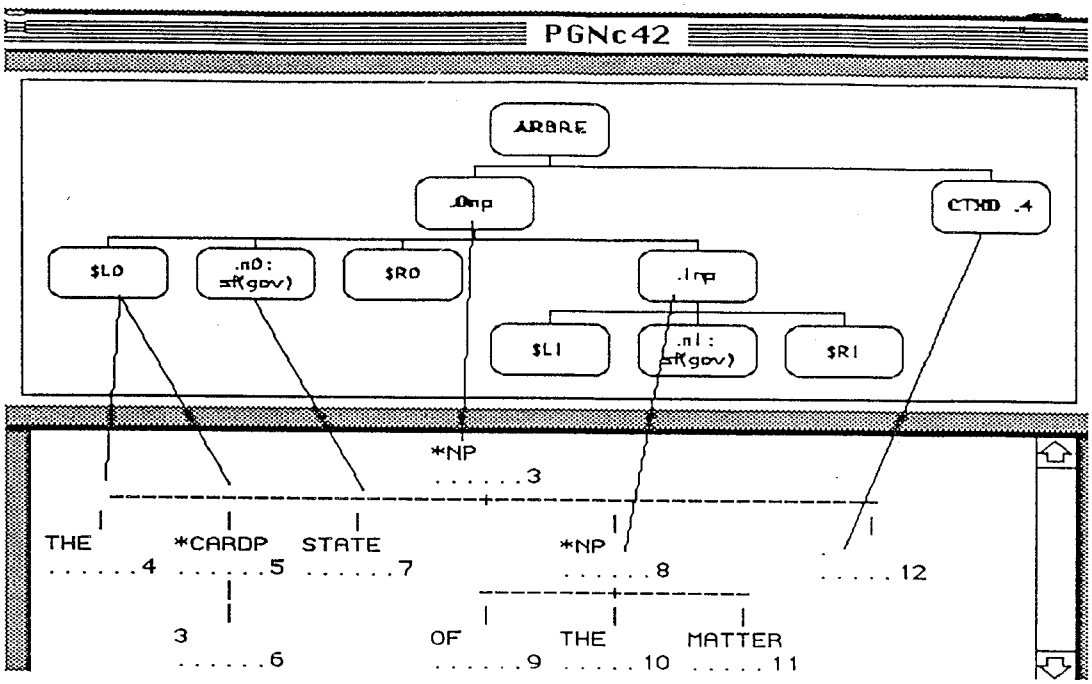
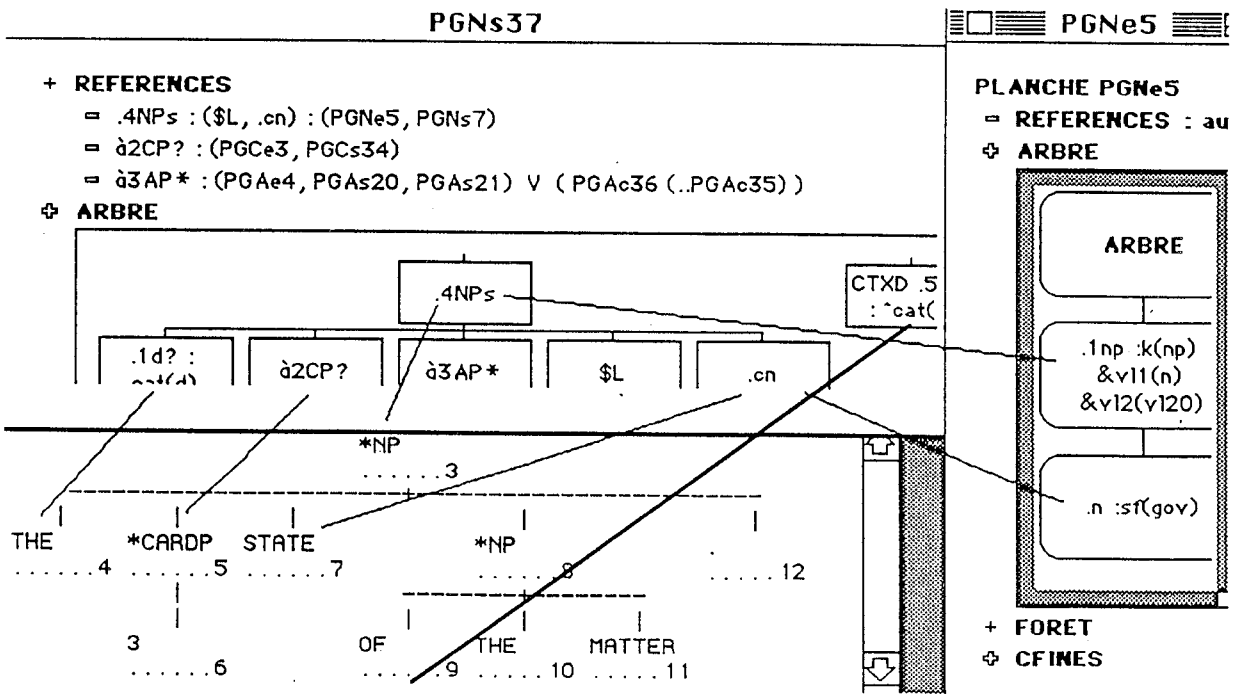


Figure 23 : Arbres de la planche PGNc42 et de la sortie de l'AS

Nous voyons que l'arbre de la planche et celui du résultat de l'analyse structurale sont cohérents. Le sous-arbre .0np ($L0, .n0, $R0) est défini par une autre planche PGNs37 qui se réfère encore à la planche PGNe5.



Nous avons vérifié jusqu'au niveau élémentaire (la planche PGNe5 décrit le mot "state") que l'arbre produit par l'analyse structurale respecte les spécifications des planches.

CONCLUSION

1. Bilan

Après une étude de l'historique et des techniques dominantes en TAO, nous avons dégagé deux axes principaux de recherche : le transfert de l'état de l'art vers l'industrie, et le perfectionnement de la technologie au moyen de techniques issues de l'intelligence artificielle.

Cette thèse a été consacrée au premier axe, car il est temps d'automatiser la production de linguiciels. En nous fondant sur les deux idées fondamentales du génie logiciel, une méthodologie et un ensemble d'outils mettant en œuvre la méthodologie, nous avons étudié le processus de développement d'un système de TAO et en avons déduit une méthodologie. Cette méthodologie a été illustrée par une première version du poste de travail linguistique et mise en pratique à titre expérimental. Malgré la simplicité de ce poste de travail, il est déjà possible d'améliorer de façon significative la productivité des linguistes et la qualité des linguiciels.

2. Développements du système actuel

Nous espérons améliorer le système actuel en fonction des réactions des utilisateurs (pour l'instant en petit nombre).

Nous envisageons plus particulièrement d'intégrer au système un analyseur incrémental de LSCS, ce qui permettrait d'offrir les fonctions supplémentaires suivantes :

- vérification de la syntaxe d'une GSCS ou de parties d'une GSCS (déclarations d'attributs ou de fonctions booléennes, planches ou parties de planches,...) ;
- vérification de type dans les expressions booléennes sur des attributs ;
- génération de références croisées ;
- etc.

Nous envisageons également de passer sur un matériel plus puissant offrant le même type de fonctionnalités (Macintosh II, XEROX 1186, ou analogue).

3. Quelques axes de recherche

Terminons par une brève réflexion sur l'atelier linguiciel idéal du futur.

Les fonctions principales du poste de travail concernent essentiellement l'étape de *spécification fonctionnelle* d'un linguiciel. Ce premier pas vers l'automatisation de la production de linguiciels nous paraît encourageant, bien que nous ayons été limités par le temps et le matériel.

Beaucoup d'environnements existants sont concentrés principalement sur une seule phase du cycle de vie du logiciel (par exemple, la spécification [Ross 77]), avec un seul type d'activité (par exemple, le développement de grammaires [Evan 85, Haugeneder 86]) ou une seule fonctionnalité (par exemple, l'édition dirigée par la syntaxe [Mélèse 83]). La tendance actuelle est d'intégrer des outils complémentaires en fabriquant des systèmes ouverts [Lamsweerde et col. 86, André, Moreau et Rougeot 86].

Il nous semble difficile d'utiliser un atelier universel pour toutes sortes de logiciels. Un atelier linguiciel intégrerait divers outils informatiques (une base de données, un éditeur structural, du graphisme, des moyens de communication, etc.) et couvrirait l'ensemble du cycle de vie d'un linguiciel. Il devrait être conçu pour des objectifs bien définis. L'approche objectale ("orientée vers les objets") nous paraît prometteuse pour réaliser l'architecture d'un atelier ayant une telle complexité [Robson 81, IEEEsoft 84].

Voici quelques fonctions et ressources que devrait posséder un tel atelier.

i) Grammaires

- création et édition d'une GSCS guidées par un éditeur structural. L'éditeur permettrait d'éditer indifféremment une GSCS, au niveau de la structure ou au niveau du texte, en acceptant temporairement des objets mal formés.
- interprétation d'une GSCS sur une chaîne, un texte, un corpus pour produire les arbres décorés correspondants ou sur des arbres décorés pour produire les chaînes correspondantes.
- comparaison d'un arbre produit par une grammaire dynamique avec le schéma d'arbre d'une planche.

ii) Dictionnaires

Accès à :

- des dictionnaires naturels (CD-ROM);
- des bases de données lexicales informatisées (contenant des informations linguistiques);
- des dictionnaires terminologiques (contenant des listes d'équivalents);
- des dictionnaires automatiques (contenant des codes pour tel ou tel linguiciel).

iii) Textes

Outils d'étude de textes opérant sur :

- des chaînes (JEUDEMO [Ouelette 80]).
- des formes lemmatisées.
- des structures (TOSCA [Arts & Heuvel 85], RABOR [Durand 87]). Le mécanisme de recherche de schéma de l'éditeur transformationnel d'arbres RABOR, implémenté par J.C. Durand au GETA permettrait d'implémenter des fonctions telles que la recherche de chaînes dont la structure correspond à un schéma donné.

iv) Ressources de caractère général

- grand écran plat pour la présentation de données sous forme graphique (arbres décorés,...), l'édition structurée (schémas d'arbre des planches,...) ou la manipulation de l'arbre des objets sous le moniteur.
- communication (messagerie et réseaux).
- SGBD évolué, interfacé avec le moniteur, et permettant une gestion unifiée et cohérente des grammaires, des dictionnaires, des textes et de la documentation. Il nous semble que le modèle entité-relation [Chen 76], qui peut s'unifier avec l'approche objectale [Zdonik 86], pourrait être bien adapté à cet objectif.
- logiciel de base réellement multilingue et multicaractère.

Il est clair que la construction d'un tel atelier nécessiterait l'effort de toute une équipe sur une durée assez longue (au moins trois ans à trois personnes). Toutefois, il ne faut pas oublier qu'un tel projet ne sera viable que si certaines conditions sont réunies, et en particulier la disponibilité (commerciale) des matériels et surtout des logiciels cités plus haut, dans un environnement de programmation unique et, bien sûr, à un prix abordable pour un groupe de recherche.

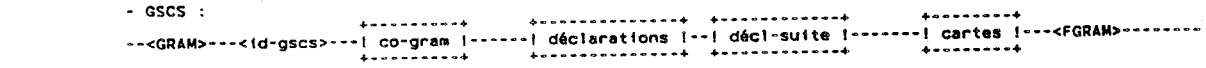
ANNEXE I. SYNTAXE DE LSCS

Dans cette annexe, nous présentons la syntaxe de LSCS. La définition du langage LSCS était une étape préliminaire de cette thèse, à laquelle J.C. Durand et R. Zajac ont aussi participé. La syntaxe de LSCS présentée est une modification de celle de [Zajac 86], due principalement à l'utilisation de l'éditeur d'arbre **More**. *Les diagrammes modifiés sont marqués par un signe '*'.*

1. DIAGRAMMES DE FLOYD

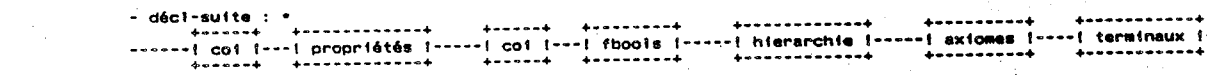
Convention : les unités lexicales sont notées entre crochets ("<" et ">") et sont considérées comme des terminaux pour cette grammaire. Les unités lexicales sont décrites en BNF dans la partie lexicale.

- GSCS :

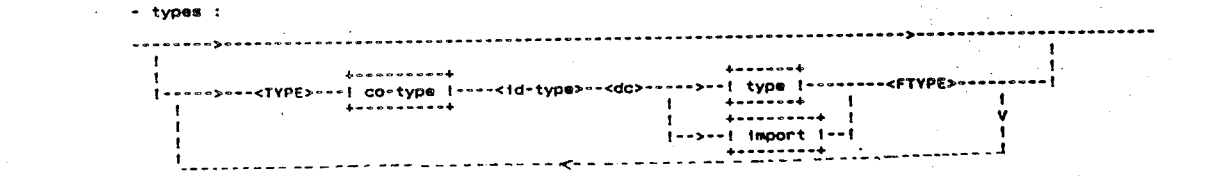


- déclarations :

- décl-suite : *



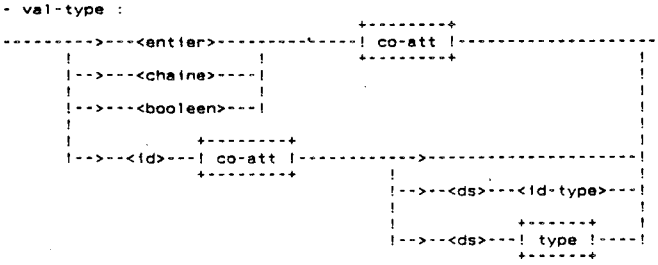
- types :



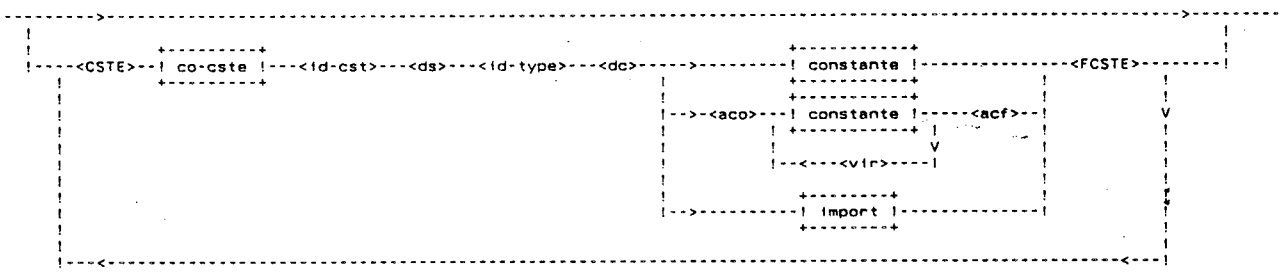
- import :

- type :

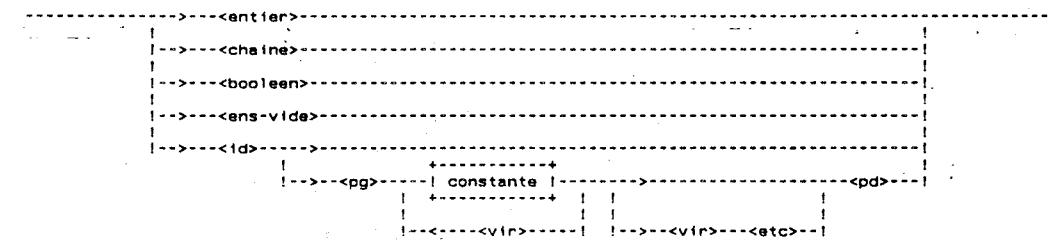
- val-type :



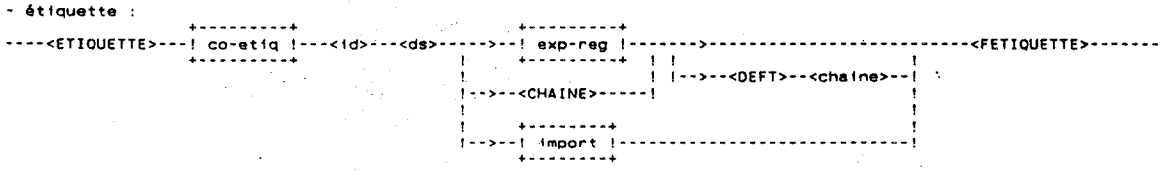
- constantes :



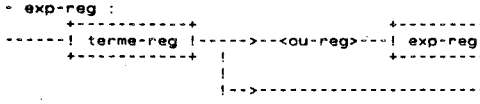
- constante :



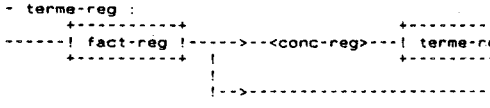
- étiquette :



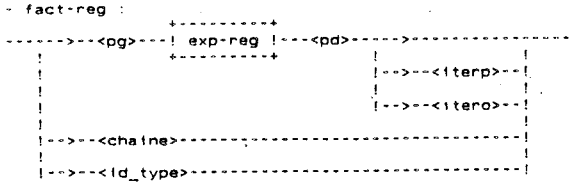
- exp-reg :



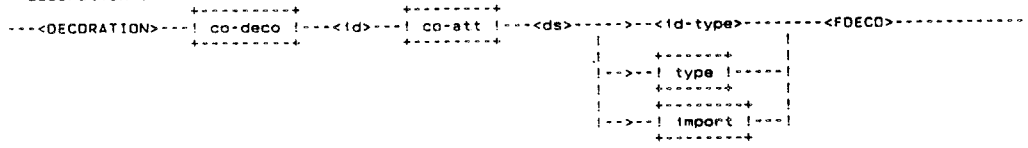
- terme-reg :



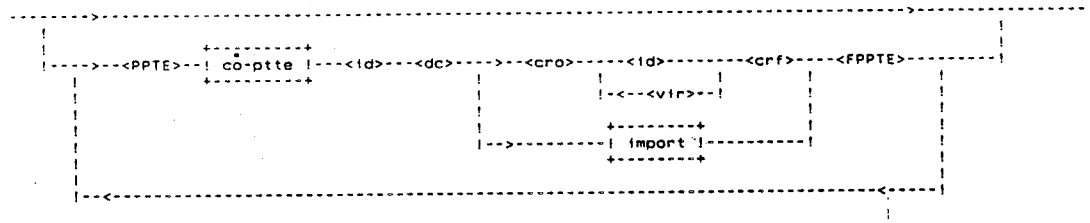
- fact-reg :



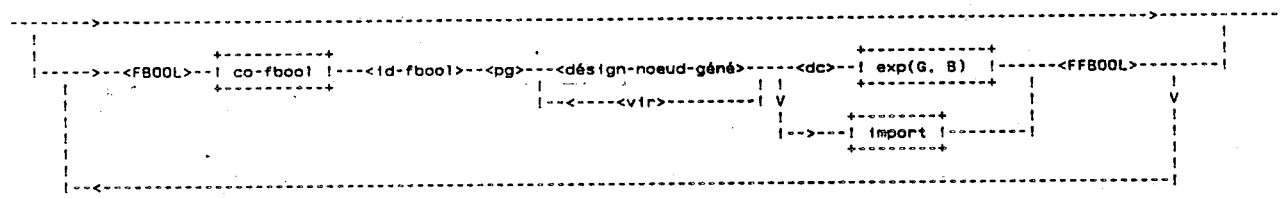
- décoration :



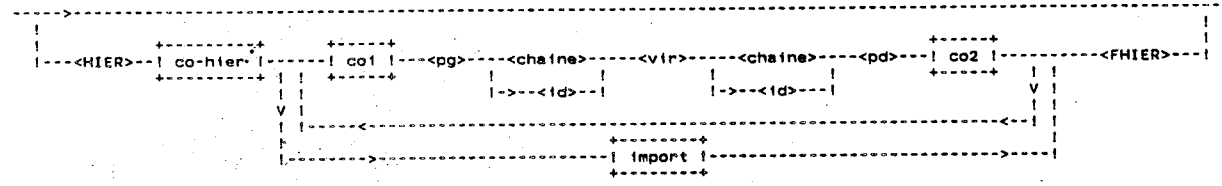
- propriétés :



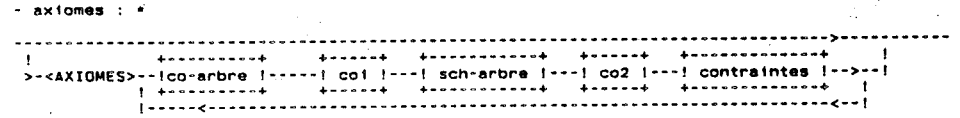
- fbools :



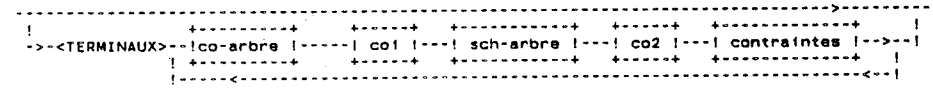
- hiérarchie :



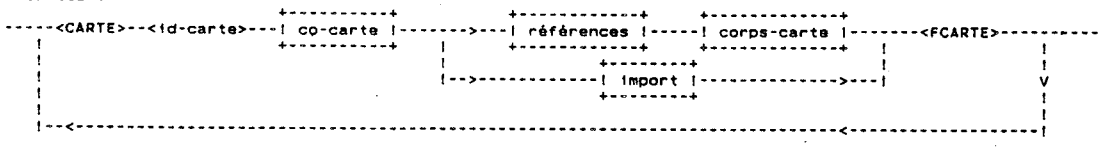
- axiomes : *



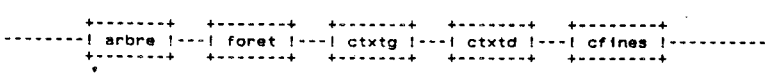
- terminaux : *



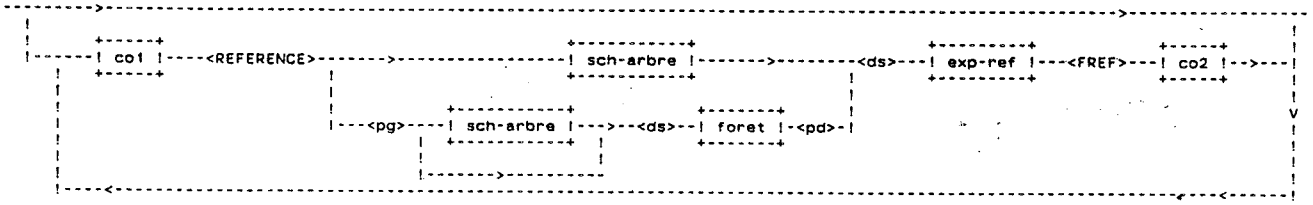
- cartes :



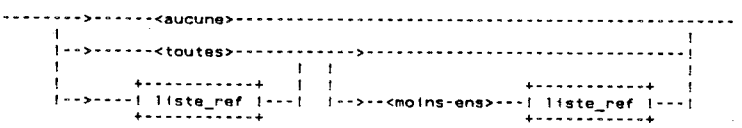
- corps-carte :



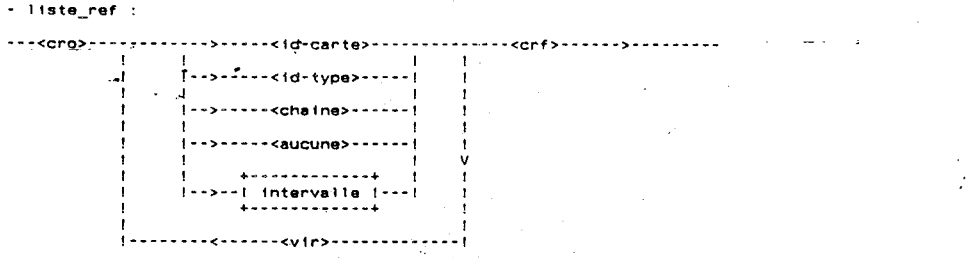
- références : *



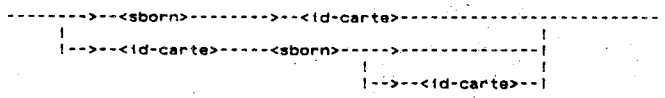
- exp-ref :



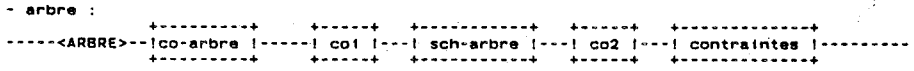
- liste_ref :



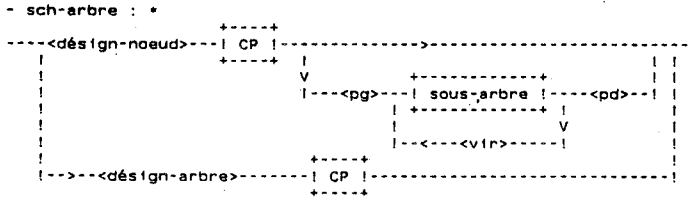
- intervalle :



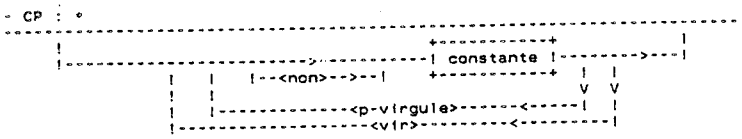
- arbre :



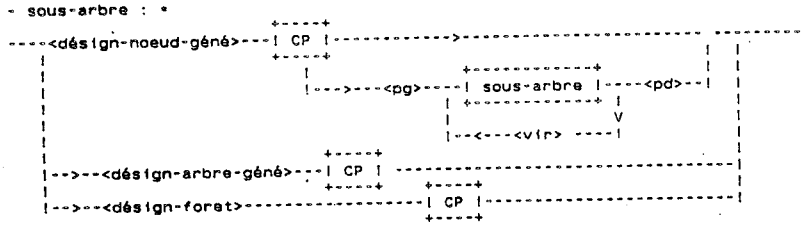
- sch-arbre : *



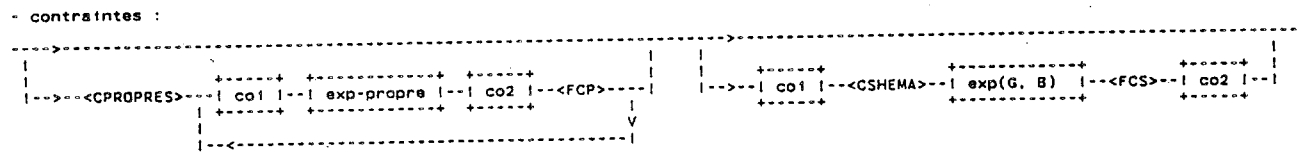
- CP : *



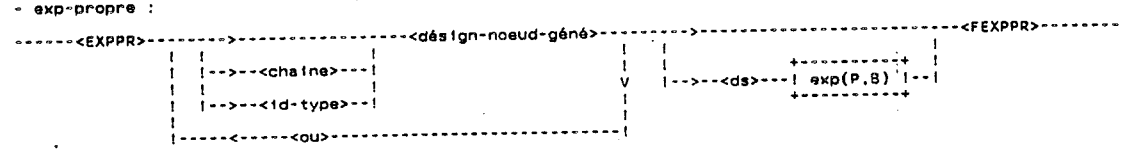
- sous-arbre : *



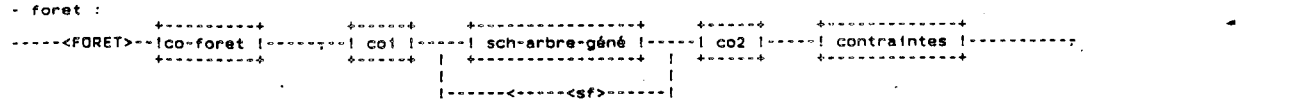
- contraintes :



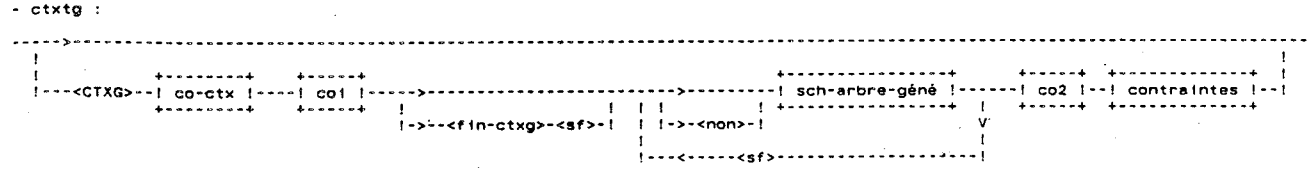
- exp-propre :



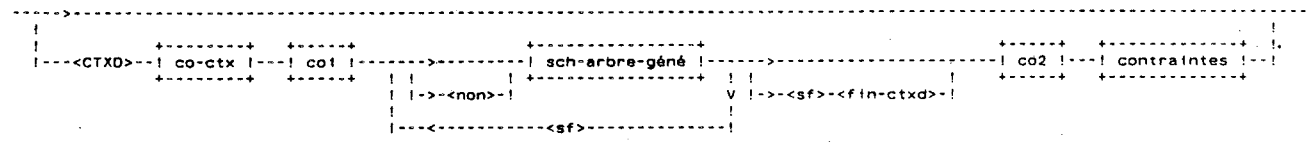
- foret :



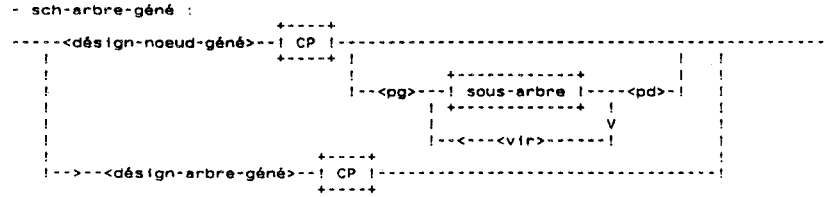
- ctxtg :



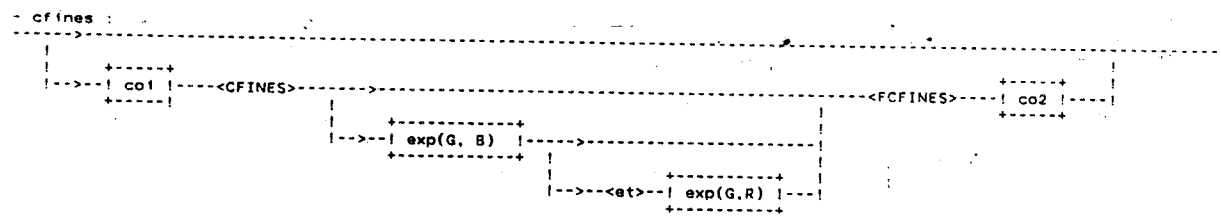
- ctxtd :



- sch-arbre-géné :



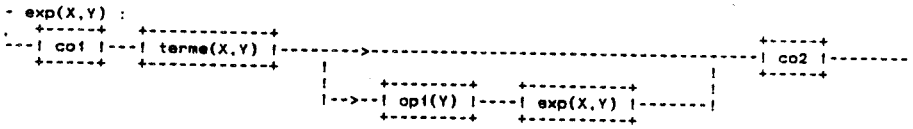
- cfines :



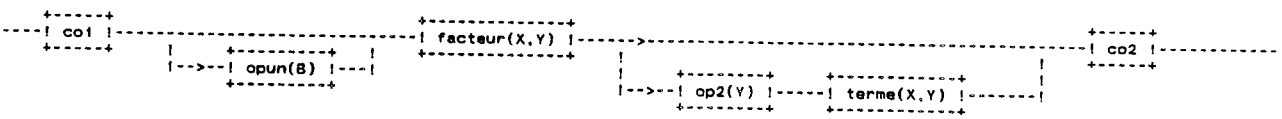
CODES DES TYPES D'EXPRESSIONS		
X	Y	
général : G propre : P	entier : N chaine : C graphe : R	booleen : B ensemble : E

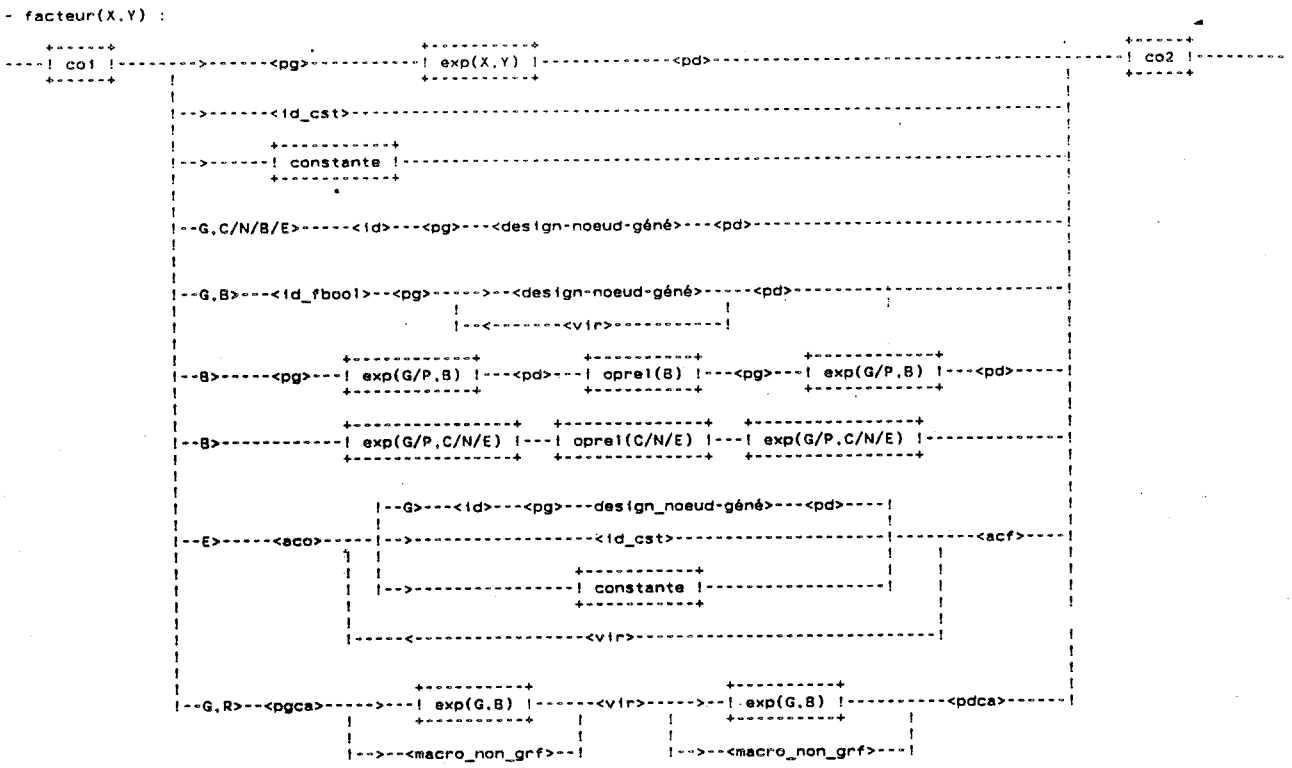
- Les codes indiqués sur les arcs indiquent les restrictions acceptées des codes du non-terminal défini.
- Un arc sans code ne comporte pas de restrictions.
- Un non-terminal qualifié par X,Y doit avoir les memes restrictions que l'appelant.
- Le code graphe n'existe qu'avec le code général.

- exp(X,Y) :

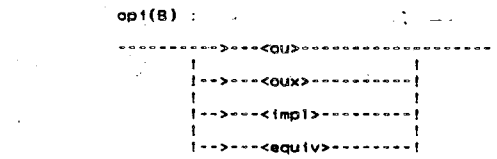


- terme(X,Y) :

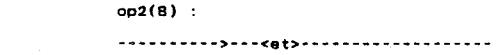




op1(B) :

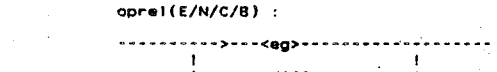


op2(B) :

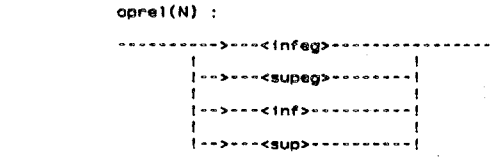


opun(B) :

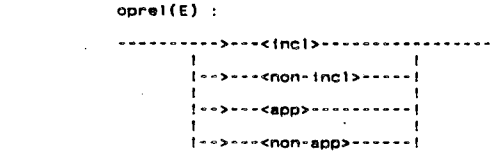
oprel(E/N/C/B) :



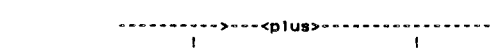
oprel(N) :



oprel(E) :



op1(N) :



op2(N) :

op1(E) :

op2(E) :

op1(C) :

op2(C) :

op1(R) :

op2(R) :

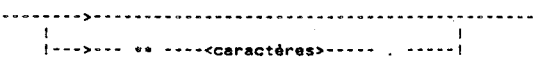
2. COMMENTAIRES

En plus des deux types de commentaires, nous proposons d'attacher à toute unité importante des informations ayant une syntaxe et une sémantique :

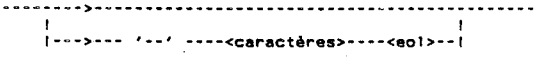
- descripteur : description de l'unité utilisée pour les traces, la visualisation partielle;
- commentaire : commentaire libre de l'unité;

- exemples : couples de valeurs d'entrées/sorties, valeurs vérifiées par l'éditeur;
- index : références croisées des objets, dans un premier temps écrites par l'auteur comme commentaire, calculées si la grammaire est éditée avec un éditeur structural;
- gestion : contient l'auteur, la date et la raison de toute modification;

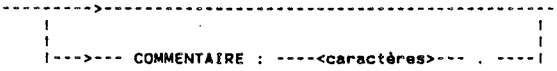
- co1 :



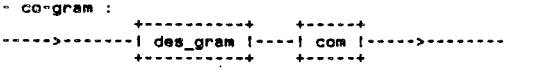
- co2 :



- com :

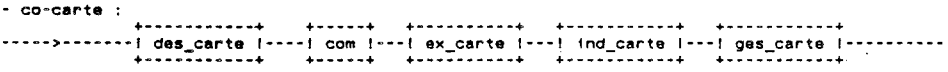


- co-gram :



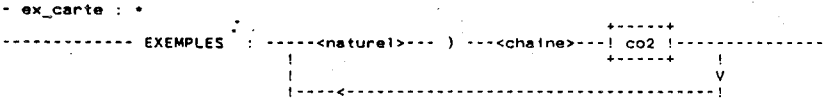
- des_gram :

- co-carte :

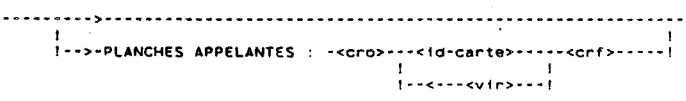


- des_carte : *

- ex_carte : *

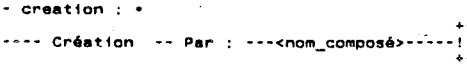


- ind_carte : *

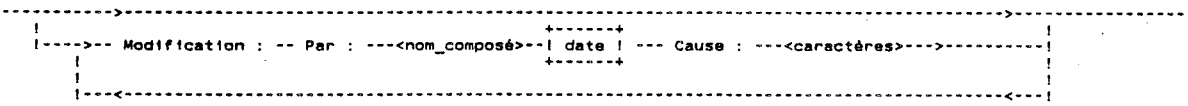


- ges_carte : *

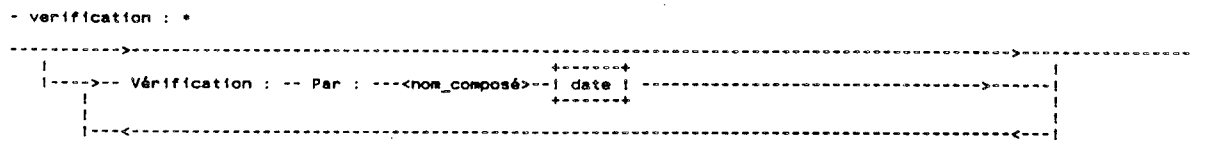
- creation : *



- modifications : *

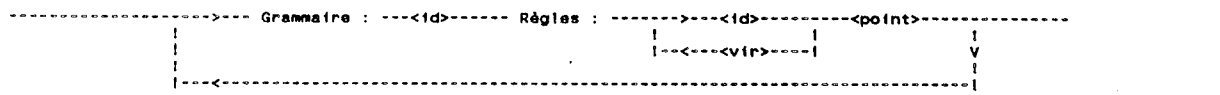


- verification : *



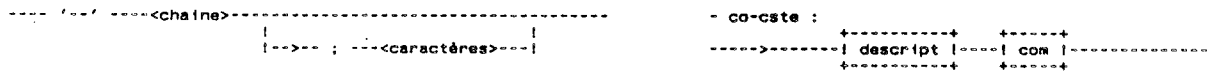
- applications : *

- dans_grammaire : *



- date : *

- co-att :



- co-type :

- descript :

- co-csts :

- co-étiq :

- co-déco :

- co-ppte :

- co-fbool :

- co-hier :

- co-arbre :

- co-foret :

- co-ctx :

3. Unites lexicales

Cette partie est dépendante des alphabets, des polices et des jeux de caratères disponibles. Le séparateur de la notation BNF '|' représente l'alternative.

3.1. Constantes

```
<entier>        ::= <naturel> | - <naturel> | + <naturel>
<naturel>       ::= <chiffre> | <chiffre><naturel>
<booleen>       ::= VRAI | FAUX
<ens-vide>      ::= Ø                                          -- ensemble VIDE
<chaine>        ::= "<suite-car>"
<suite-car>     ::= | <caractère><suite-car> | ""<suite-car>
<caractère>     ::= <lettre> | <chiffre>
                    | ! | $ | % | + | / | ( | ) | = | ? | ¨ | * | ° | £ | _ | >
                    | < | ' | ^ | & | ù | - | , | . | ; | : | à | ç

<caractères>    ::= <caractère> | <caractère> <caractères>
<chiffre>       ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
<lettre>        ::= <maj> | <min>
<min>           ::= a | b | c | d | e | f | g | h | i | j | k | l | m
                    | n | o | p | q | r | s | t | u | v | w | x | y | z
<maj>           ::= A | B | C | D | E | F | G | H | I | J | K | L | M
                    | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
<toutes>        ::= TOUTES
<aucune>        ::= AUCUNE
<nom_composé>   ::= <nom> | <nom> <nom_composé>
<nom>           ::= <lettre> | <lettre><nom>
```

3.2. Identificateurs

```
<id>            ::= <min> | <min><suite-alphanum>
<suite-alphanum>::= <min> | <min><suite-alphanum>
                    | <chiffre> | <chiffre><suite-alphanum>
                    | _<suite-alphanum>
<id-gscs>       ::= <id>
<id-carte       ::= <id>
<id-cst>        ::= <id>
<id-fbool>      ::= <id>
<id-type>       ::= <id>
```

3.3. Désignateurs

```
<désign>        ::= <désign-noeud> | <désign-arbre> | <désign-foret>
<désign-noeud>  ::= .<des> | A.<des> | F.<des>
<désign-arbre>  ::= à<des> | Aà<des> | Fà<des>
<désign-foret>  ::= $<des> | A$<des> | F$<des>
<des>           ::= <chiffre> | <chiffre><suite-alphanum>
<désign-géné>   ::= <désign-noeud-géné> | <désign-arbre-géné> |<désign-foret>
<désign-noeud-géné>  ::= <désign-noeud>  | <désign-noeud>?
                    |  <désign-noeud>+ | <désign-noeud>*
<désign-arbre-géné>  ::= <désign-arbre> | <désign-arbre>?
                    | <désign-arbre>+ | <désign-arbre>*
```

3.4. Opérateurs

Les opérateurs sont classés selon la nature de leurs domaines.

3.4.1. Domaine booléen

```
<eg>    ::=  =
<diff>  ::=  ≠                                                 -- ^=
<ou>    ::=  V
<et>    ::=  &
```

<oux> ::= ⊕ -- W
<non> ::= ¬ -- ^
<impl> ::= =>
<equiv> ::= <=>

3.4.2. Domaine entier (relatifs)

<eg> ::= =
<diff> ::= ≠ -- ^=
<plus> ::= +
<moins> ::= -
<mult> ::= *
 ::= /
<mod> ::= MOD
<inf> ::= <
<sup> ::= >
<infeg> ::= ≤ -- <=
<supeg> ::= ≥ -- >=

3.4.3. Domaine chaîne

<eg> ::= =
<diff> ::= ≠ -- ^=
<conc> ::= .
<ou-reg> ::= +
<conc-reg> ::= .
<iterp> ::= +
<itero> ::= *

3.4.4. Domaine de décoration : ensemble et scalaire

<eg> ::= =<suff>
<diff> ::= ≠<suff>
<union> ::= ∪<suff> -- v
<inter> ::= ∩<suff> -- &
<moins-ens> ::= -<suff>
<incl> ::= ⊃<suff> -- =>
<non-incl> ::= ¬⊃ <suff> -- ^=>
<app> ::= ∈ <suff> -- E
<non-app> ::= ∉ <suff> -- ^E

3.5. Mots-clés et symboles spéciaux

3.5.1. Mots-clés

<ARBRE> ::= ARBRE
<AXIOMES> ::= AXIOMES -- Nouveau
<BOOLEEN> ::= BOOLEEN
<CARTE> ::= CARTE
<FCARTE> ::= -- Fin de carte
<CFINES> ::= CFINES -- Contraintes fines
<FCFINES> ::=
<CHAINE> ::= CHAINE
<CSTE> ::= CSTE -- Constante
<FCSTE> ::=
<CTXD> ::= CTXD -- Contexte droit
<CTXG> ::= CTXG -- Contexte gauche
<DECORATION> ::= DECORATION
<FDECO> ::=
<ENSEMBLE> ::= ENSEMBLE

```
<ENTIER>        ::= ENTIER
<ETIQUETTE>     ::= ETIQUETTE
<FETIQUETTE>    ::=
<FBOOL>         ::= FBOOL                    -- Fonction booléenne
<FFBOOL>        ::=
<FORET>         ::= FORET
<GRAM>          ::= GRAMMAIRE
<FGRAM>         ::= FGRAMMAIRE
<HIER>          ::= HIERARCHIE
<FHIER>         ::=
<DEFT>          ::= DEFT                     -- Définition
<EXPPR>         ::=                          -- Expression
<FEXPPR>        ::=
<PPTE>          ::= PPTE                     -- Propriété
<FPPTE>         ::=
<CPROPRES>      ::= CPROPRES                 -- Contrainte propre
<FCP>           ::=
<REFERENCE>     ::= REFERENCE
<FREF>          ::=
<SCALAIRE>      ::= SCALAIRE
<CSCHEMA>       ::= CSCHEMA                  -- Contrainte de schéma
<FCS>           ::=                          -- Fin contrainte de schéma
<TERMINAUX>     ::= TERMINAUX                -- Nouveau
<TYPE>          ::= TYPE
<FTYPE>         ::=
```

3.5.2. Symboles spéciaux

```
<cro>           ::= [                        -- (
<crf>           ::= ]                        -- )
<aco>           ::= {                        -- (
<acf>           ::= }                        -- )
<pg>            ::= (
<pd>            ::= )
<sborn>         ::= ..
<vir>           ::= ,
<etc>           ::= ...
<pgca>          ::= <                        -- Caractère de partie gauche
<pdca>          ::= >                        -- Caractère de partie droite
<dc>            ::= =                        -- Déclarer comme
<ds>            ::= :                        -- Déclarer sur
<sf>            ::= ,                        -- Séparateur
<point>         ::= .
<p-virgule>     ::= ;
<fin-ctxg>      ::= !
<fin-ctxd>      ::= !
<suff>          ::=  | <entier> | *          -- Suffixe
<macro_non_grf> ::= ¬PREC<suff> | ¬SUIV<suff>  -- Macro de négation en graphe
```

4. Explication des modifications

La modification de la syntaxe garde la compatiblité par rapport à la version précédente (sauf le vocabulaire). C'est-à-dire que la nouvelle syntaxe permet des possibilités nouvelles et que les GSCS écrites dans l'ancienne syntaxe sont correctes à condition de remplacer les terminaux modifiés par leurs correspondants.

4.0. Vocabulaire

Dans la thèse, nous utilisons un vocabulaire légèrement différent.

La liste suivante donne les correspondances pour certains non-terminaux :

Nouveaux	**Anciens**
format	propriété
planche	carte
prédicat	fonction booléenne

Non-terminaux ajoutés

axiomes
terminaux
CP -- Contraintes propres

La liste suivante donne les correspondances pour certains non-terminaux :

Nouveaux	**Anciens**	
Ø	VIDE	-- ensemble vide
≠	^=	-- différent
⊕	W	-- ou exclusif
¬	^	-- négation
≤	<=	-- inférieur ou égal
≥	>=	-- suppérieur ou égal
∪	v	-- union
∩	&	-- intersection
⊃	=>	-- contenant
¬⊃	^=>	-- ne contenant pas
∈	E	-- appartenant à
∉	^E	-- n'appartenant pas à
[	(	-- crochet ouvrant
]	)	-- crochet fermant
{	(	-- accolade ouvrante
}	)	-- accolade fermante

Terminaux ajoutés

;	-- point-virgule : séparateur et ou logique
AXIOMES	-- précède la déclaration des axiomes
TERMINAUX	-- précède la déclaration des terminaux

4.1. Ecriture des schémas d'arbre et des expressions booléennes

Les schémas d'arbre (unité syntaxique **sch-arbre**) sont utilisés dans la déclaration des axiomes et des terminaux (**axiomes** et **terminaux**) et dans les schémas d'arbre et de forêt des planches (**cartes**). Les changements par rapport à l'ancienne syntaxe sont que :

1) l'arbre peut être écrit sous forme indentée ;
2 les contraintes propres peuvent être écrites juste après leurs désignateurs.

4.1.1. Arbre sous forme indentée

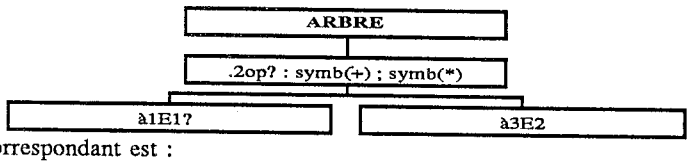
Les parenthèses et la virgule sont remplacées par l'indentation et la fin de ligne.

Les règles de la présentation de l'arbre par indentation sont :
- chaque noeud se trouve sur une ligne,
- les noeuds d'un même niveau (frères) commencent sur la même colonne, plus à droite par rapport à leur ancêtre (père).

Donc, pour transformer la présentation d'un arbre de la forme parenthésée en forme indentée, il suffit de :
- remplacer toute parenthèse ouvrante par une fin de ligne, et commencer la nouvelle ligne décalée d'un espace plus à droite que la ligne précédente;
- remplacer toute virgule par une fin de ligne et commencer la nouvelle ligne K espaces plus à gauche que la ligne précédente, K étant le nombre de parenthèses fermantes immédiatement avant la virgule ;
- supprimer les parenthèses fermantes restantes.

Dans le schéma d'une planche, nous suggérons de numéroter, via un préfixe de désignateur, les désignateurs correspondant aux racines des éléments du schéma de forêt. Cela permet de retrouver à partir du schéma d'arbre celui de forêt, donc la présence de ce dernier n'est plus nécessaire.

Prenons comme exemple le schéma d'arbre suivant :



Le schéma de forêt correspondant est :
à1E1?, .2op? : symb(+) ; symb(*), à3E2

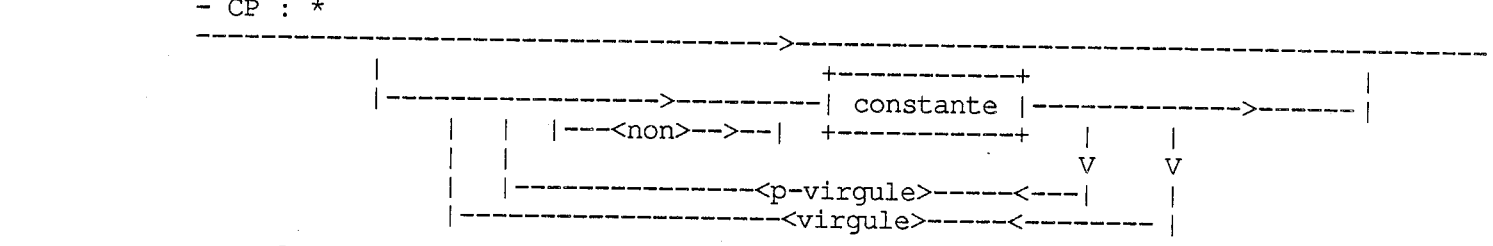
La règle générale pour retrouver le schéma de forêt à partir de celui d'arbre correspondant est simplement de couper récursivement dans le schéma d'arbre de bas en haut les feuilles numérotées et les sous-arbres dont les racines sont numérotées et dont aucun fils n'est numéroté, jusqu'à ce que l'on ne puisse plus couper. Les feuilles et les sous-arbres coupés dans l'ordre de leur numération constituent le schéma de forêt.

Si les relations géométriques à l'intérieur d'un élément sont différentes dans le schéma de forêt et dans celui d'arbre, dans ce cas-là, cet élément doit figurer dans le schéma de forêt et se référer à des planches. Nous verrons plus tard que la précision sur les références de planches (§ 4.2.) permet de décrire complètement cet élément.

4.1.2. Contraintes propres suivant leurs désignateurs

Nous avons ajouté la possibilité d'écrire les contraintes propres à un désignateur immédiatement après le désignateur. L'avantage de mettre les contraintes propres avec leur désignateur est que l'on peut les voir en même temps. Cela est surtout intéressant quand l'arbre est sous forme graphique.

Voici la nouvelle syntaxe de ces contraintes propres :



Les contraintes propres placées après leurs désignateurs portent sur les attributs des noeuds instanciés avec les désignateurs. La virgule joue ici le rôle de l'opérateur logique ET et le point-virgule celui de l'opérateur logique OU et ¬ pour négation. Notons .R un désignateur de noeud, symb un nom d'attribut et E, T des valeurs possibles de symb, voici l'équivalence entre les contraintes propres et les expressions booléennes correspondantes dans des contraintes de schéma (CSCHEMA) :

Contraintes propres	Contraintes de schéma
.R : symb(E)	sybm(.R) = E
.R : symb(E) ; symb(T)	symb(.R) = E V symb(.R) = T
.R : ¬symb(E), ¬symb(T)	¬(symb(.R) = E) & ¬symb(.R) = T

Les contraintes propres à un désignateur d'arbre portent sur les racines des arbres instanciés avec ce désignateur et

celles propres à un désignateur de forêt portent sur chaque arbre d'une forêt instanciée.

L'écriture d'une expression booléenne comme un arbre ET-OU sous forme graphique est plus lisible. Le principe consiste à remplacer les opérateurs ET et OU par les relations père-fils et frère-frère. Chaque noeud est une expression booléenne écrite sous forme usuelle dans laquelle on peut avoir l'opérateur ¬ (négation). Si un noeud ne contient que l'opérateur ¬, il est interprété comme la négation de la conjonction de tous ses fils.

Une exception est que dans une planche, si on considère le mot-clé CSCHEMA comme racine de l'expression booléenne, tous ses fils (chacun étant une expression booléenne) sont liés implicitement par l'opérateur logique ET.

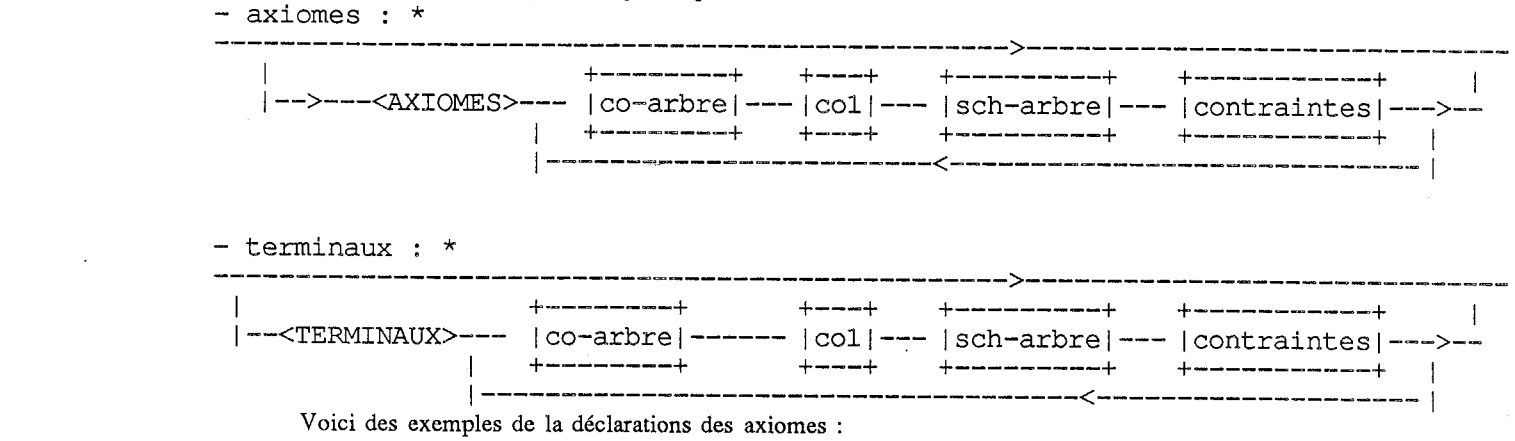
Les arbres ET-OU peuvent être écrits de la même manière que les schémas d'arbre.

4.2. Déclaration des axiomes et des terminaux

La possibilité de déclarer des axiomes et des terminaux permet, d'une part, de mieux préciser les arbres pouvant être dans les deux domaines de la relation définie par une GSCS et, d'autre part, de rendre les deux domaines de la relation spécifiée par la GSCS plus clairs, aussi bien pour l'auteur que pour les lecteurs.

Sans déclaration des axiomes et des terminaux, un arbre formé selon la déclaration des attributs peut être un arbre axiome ou terminal. La déclaration des axiomes est une liste de schémas d'arbre (un schéma géométrique et des contraintes sur le schéma comme ceux dans les planches). Après la déclaration des axiomes, seuls les arbres pouvant s'instancier avec un des ces schémas d'arbre constituent l'ensemble d'arbres axiomes. C'est aussi valable pour la déclaration des terminaux.

Voici la partie de la syntaxe rajoutée pour les déclarations des axiomes et des terminaux :



Voici des exemples de la déclarations des axiomes :

-- Deux schémas d'axiomes : soit un seul noeud dont la décoration est symb(a)
-- soit un arbre qui a deux fils et dont la racine est décorée par symb(+) ou symb(*).

TERMINAUX : .T -- un terminal est un noeud quelconque.

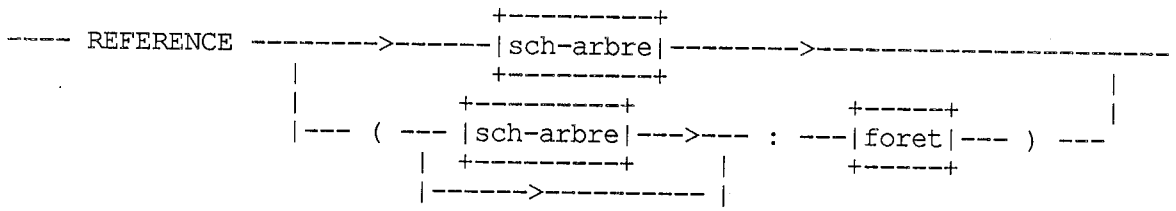
4.3. Références des planches

Rappelons que les références permettent d'utiliser dans une planche des arbres définis dans d'autres planches. Dans l'ancienne syntaxe, on donne seulement le désignateur racine de sous-arbre référent (l'arbre qui se réfère à une planche qui le définit). Cela peut provoquer des ambiguïtés lorsque le désignateur ne domine pas les mêmes fils dans le schéma d'arbre que dans celui de forêt. Jusqu'à présent, on suppose que ce désignateur représente le sous-arbre du schéma de forêt. Il y a encore un fait implicite : quand un arbre est défini par une planche, cet arbre correspond au moins à une forêt qui est aussi définie par la planche. Mais on ne s'intéresse pas souvent à ces forêts et donc on ne les écrit pas.

Dans la nouvelle syntaxe, nous permettons d'écrire d'une part la totalité du schéma d'arbre référent, et d'autre part le schéma de forêt correspondant à l'arbre référent. Cela a trois avantages :

1) les référents sont explicites (entièrement écrits) ;
2) les référents peuvent être dans les schémas d'arbre **et** de forêt ;
3) les référents peuvent se référer aux schémas d'arbre **et** de forêt d'une planche référée.

Voici la partie de la syntaxe concernant les références :



Prennons comme exemple la référence de la planche PGNs37 :

Ancienne écriture	Nouvelle écriture
.4NPs	.4NPs ($L, .cn) (.4NPs ($L, .cn) : $X)

Les exemples suivants illustrent l'utilisation de nouvelles références.

5. Exemples illustrant les moficications

Prennons comme exemple une grammaire des expressions arithmétiques simplifiées. La grammaire hors-contexte est $G_0=(V_t, V_n, E, P)$:

V_t = {a, +, *}, V_n = {E, T, F},

P = { E -> E + T | T, T -> T * F | F, F -> a}.

Nous allons écrire plusieurs GSCS. La première Expa, avec une seule planche, définit le même langage en associant l'arbre abstrait comme structure correspondante. La deuxième spécifie les correspondances entre les arbres syntaxiques et les arbres abstraits du même langage. Enfin , la troisième qui permet d'obtenir les mêmes arbres syntaxiques dans une "double-dérivation" montrant l'utilisation de la référence de forêt (le référent étant une sous-forêt faisant partie de l'ensemble de forêts définies par une planche).

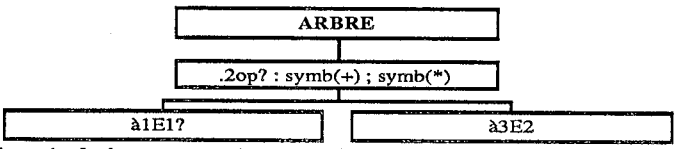
GSCS : Exxa

ATTRIBUT : symb SCALAIRE (+, *, a)
-- cette GSCS utilise un seul attribut

-- Deux schémas d'axiomes : soit un seul noeud dont la décoration est symb(a)
-- soit un arbre ayant 2 fils, dont la racine est décorée par symb(+) ou symb(*)

TERMINAUX : .T
-- Les terminaux sont des arbres triviaux ayant un seul noeud
-- qui peut être symb(a), symb(+) ou symb(*).

PLANCHE Pa

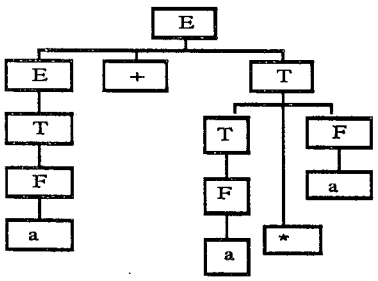


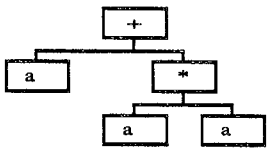
-- Le schéma de forêt correspondant est : à1E1?, .2op? : symb(+) ; symb(*), à1E2

CSCHEMA

EXISTE(à1E1?) <=> EXISTE(.2op?)
-- Les deux éléments optionnels figurent ensemble.
symb(.2op?) = * => (symb(à1E1?) ≠ + & symb(à1E2?) ≠ +)
-- Priorité à la multiplication
-- ET logique entre les deux termes ci-dessus.

Pour la même chaîne : "a + a * a", on a l'arbre syntaxique et l'arbre abstrait :



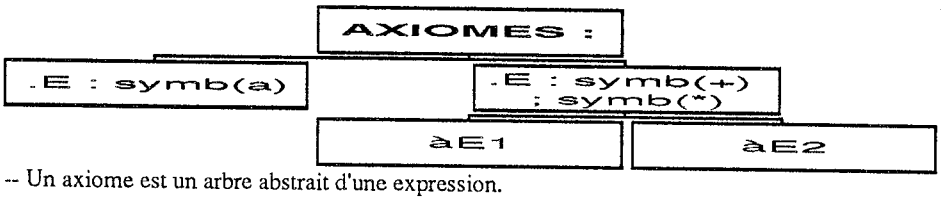


Arbre syntaxique (grammaire hors-contexte) *Arbre abstrait (GSCS)*

La GSCS Corr spécifie les correspondances entre les arbres syntaxiques et les arbres abstraits des mêmes chaînes du langage.

GSCS : Corr

ATTRIBUT : symb SCALAIRE (E, T, F, +, *, a)



-- Un axiome est un arbre abstrait d'une expression.

TERMINAUX : àE : symb(E).

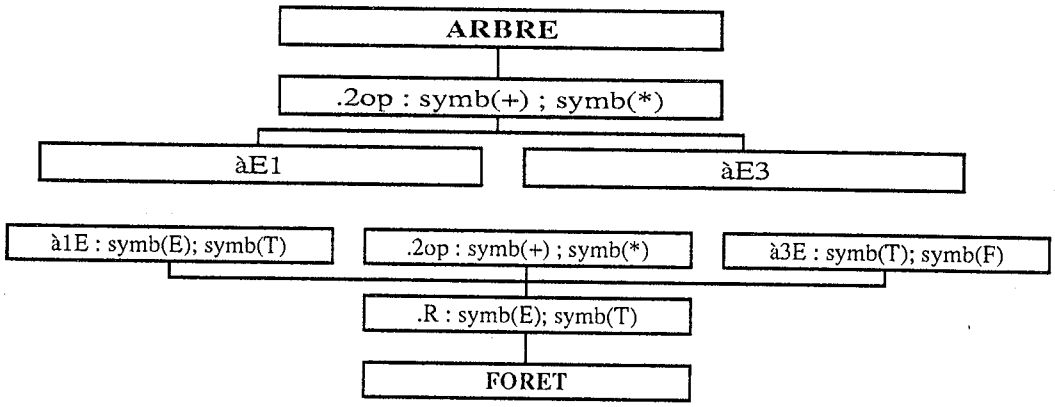
-- Un terminal est un arbre syntaxique d'une expression.

PLANCHE Pa1

REFERENCES

(àE1 : à1E) : TOUTES

(àE3 : à3E) : TOUTES



CSCHEMA

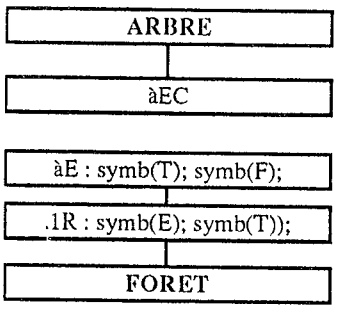
symb(.R) = E => (symb(à1E)=E & symb(.2op)=+ & symb(à3E?) =T)

symb(.R) = * => (symb(à1E)=T & symb(.2op)=* & symb(à1E?) =F)

PLANCHE Pa2

REFERENCES

- (àEC : àE) : TOUTES



CSCHEMA

symb(.1R) = E => (symb(àE) = T)

symb(.1R) = T => (symb(àE) = F)

symb(.1R) = F => (symb(àE) = a)

La dernière GSCS Expb permet d'avoir un arbre syntaxique au moyen de "double-dérivation" et utilise la référence de forêt.

GSCS : Expb

ATTRIBUT : symb SCALAIRE (E ,F , T, +, *, a)

AXIOMEX : .E : symb(E)
TERMINAUX : .T : symb(+) ; symb(*) ; symb(a)

PLANCHE Pbe
REFERENCES
(: $1) : Pbe
(: $2) : Pbt

ARBRE
.E : symb(E)

FORET
$1
.2op : symb(+)
$3 : ¬symb(+)

PLANCHE Pbt

REFERENCES
(: $1) : Pbt
(: .A) : Pbf

ARBRE
.T : symb(T)

FORET
$1
.2op : symb(*)
.3A

PLANCHE Pbf

REFERENCES : AUCUNE

ARBRE
.F : symb(F)

FORET
.1A : symb(a)

ANNEXE II. STRUCTURE DE BSCS

Cette annexe définit la structure générique des bases de données linguistiques pour la spécification de correspondances structurales.

Introduction

Arbre des objets. L'ensemble des objets gérés par ESCS est une Base de Spécifications de Correspondances Structurales – BSCS. Cet ensemble d'objets est organisé hiérarchiquement pour former un *arbre des objets* dont la structure générique est schématisée ci-dessous (l'ordre des fils d'un nœud n'étant pas significatif).

Objet. Un objet est un sous-arbre.

Un *nœud interne* est matérialisé par un dossier géré par Finder (système d'exploitation du MacIntosh). Le contenu d'un nœud interne est l'ensemble de ses sous-arbres.

Une *feuille* est matérialisée selon son type par :
- un fichier de texte (manipulé au moyen du programme MacWrite),
- une structure (manipulée au moyen du programme More),
- un dictionnaire (manipulé au moyen du programme WinTool).

Le contenu d'une feuille est le contenu du fichier, de la structure ou du dictionnaire.

Un objet est désigné par le chemin de la racine (BSCS par défaut) à la racine du sous-arbre. Par exemple, /GSCS/BXFXeng et BSCS/GSCS/BXFXeng désignent l'objet BXFXeng.

Les informations attachées à chaque objet sont contenues dans un fichier accessible, pour l'objet courant sélectionné sous Finder, dans le menu Fichier, article Lire information.

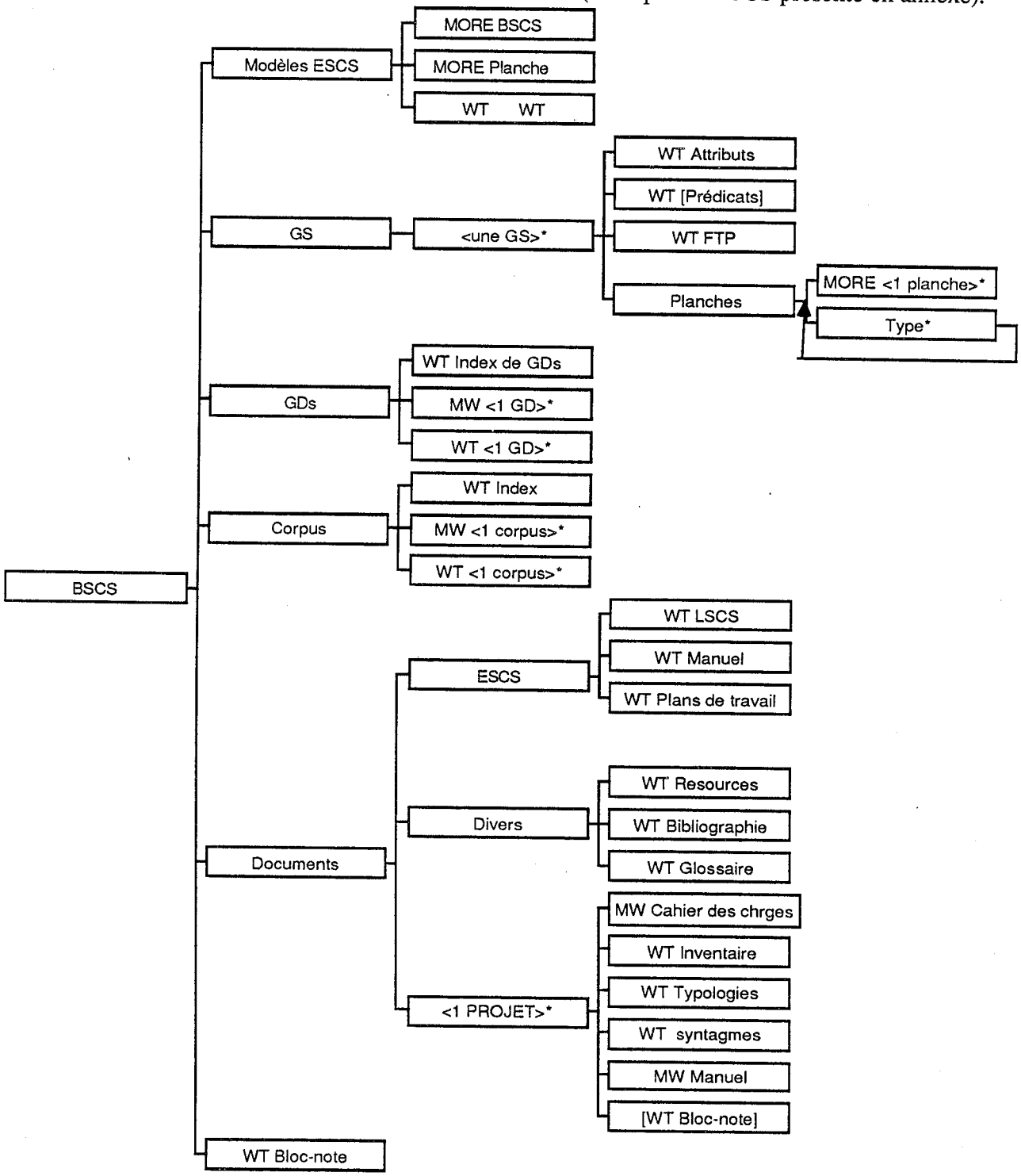
A chaque nœud sont attachés 3 types d'informations :
- une description en clair de la nature de l'objet (champ DESCRIPTION),
- un ensemble d'attributs qui le caractérise (champ ATTRIBUTS),
- ses liens avec d'autres objets (champ LIENS).

Définition de la structure de l'arbre des objets.

On présente la définition de chaque objet ou classe d'objets à partir de la racine en suivant l'ordre donné par le schéma ci-dessous. Nous définissons récursivement d'abord la racine et ensuite les sous-arbres d'un objet. La relation entre la racine et les sous-arbres d'un objet est définie par le schéma d'arbres des objets ci-dessous. Il nous reste donc à définir seulement les racines des objets.

Pour chaque objet, on trouvera 6 rubriques :

- son *nom* fixe ou défini par l'utilisateur,
- sa *description* définissant sa nature et les opérations associées (correspond au champ DESCRIPTION),
- ses *attributs* qui le caractérisent (correspond au champ ATTRIBUTS),
- ses *liens* éventuels avec d'autres objets (correspond au champ LIENS),
- la *syntaxe* de son contenu,
- un *exemple* de contenu extrait de BexFex (exemple de BSCS présenté en annexe).



Vue graphique de la structure générique d'une BSCS

Pour les définitions, les convention suivantes sont utilisées :

1) Un terme entre chevrons < > décrit la nature de l'élément. Par exemple <entier> est un entier : 3 ou 70 ; <idf> est un identificateur : BXFX, PGNs37.

2) Un élément entre crochets [ ] est optionnel. S'il est suivi par +, il peut être répété, par * répété et optionnel. Le symbole | note l'alternative.

3) Les exemples ou les commentaires de la définition commencent avec -- et se terminent par la fin de ligne.

4) Les attributs suivants sont communs à tous les noeuds :

```
Type: DOSS | More | WT | MW.          -- DOSS : dossier, WT : WinTool, MW : MacWrite
Créateur du noeud : <idf>            -- ex : S.Chappuy
Date de création : <date>            -- ex : Mardi 3 mars 1987 (gérée par le système)
Date de dernière modification : <date>  -- ex : Samedi 14 mars 1987 (gérée par le système)
Taille : <entier>                    -- nombre d'octets (gérée par le système)
Etat : <chaîne>                      -- achevé (ou autre état).
Droits d'accès :
        Lecture : système | auteur | tous
        Ecriture : système | auteur | tous
        Exécution : système | auteur | tous
        -- par défaut, lecture pour tous et écriture et exécution par l'auteur.
```

La syntaxe d'un attribut est <nom-de-l'attribut> : <valeur-d'attribut> <commentaire ou exemple>. La rubrique attribut est omise s'il n'y a que des attributs communs.

5) La rubrique syntaxe est omise s'il n'y a pas d'exigence particulière (ce qui est souvent le cas pour les noeuds internes). La rubrique exemple est omise si l'objet est invariant.

6) Pour un objet de type WinTool, le premier article du dictionnaire est toujours nommé " commentaire" et contient le champ description du noeud et des exemples d'articles. Un article a une clé et un ou plusieurs champs. La clé est notée entre guillemets.

7) Le fichier 'bloc-note' peut être ajouté dans tous les dossier pour une utilisation locale.

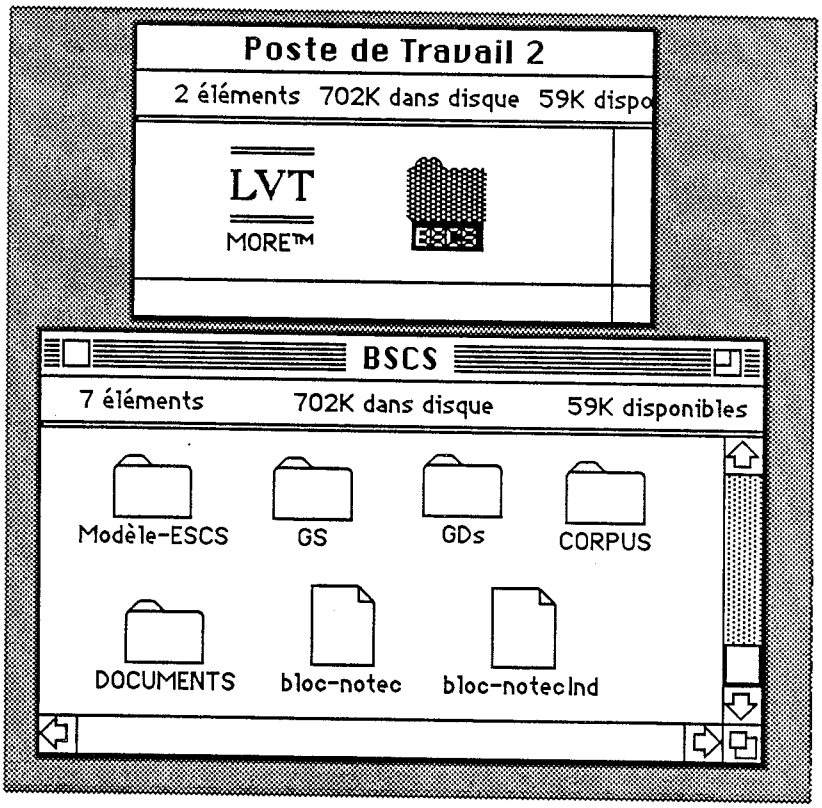
0. BSCS [Dossier]

Description Racine de la base. A la livraison du système, il doit y avoir au moins une instance pour chaque noeud de la base. Ce nœud ne peut être modifié.

Attributs

Noms de projets : <idf>* -- ex : PNTAO, GC-chi, BXFX.
Langages décrits : (<idf> : <idf>*)*
-- Exemple :
-- Français : PN-TAO, BXFX
-- Chinois : GCchi
-- Anglais : BXFX

Exemple



Vue de l'objet BSCS sous Finder (système d'exploitation du MacIntosh).

1. Modèle ESCS [Dossier]

Description:. Ce dossier contient des modèles d'objets pour ESCS (structure de BSCS, squelette de planche et d'article de dictionnaire).

1.1. Structure de BSCS [More]

Description Ce fichier contient la description de la structure de BSCS (cette annexe). Il ne peut

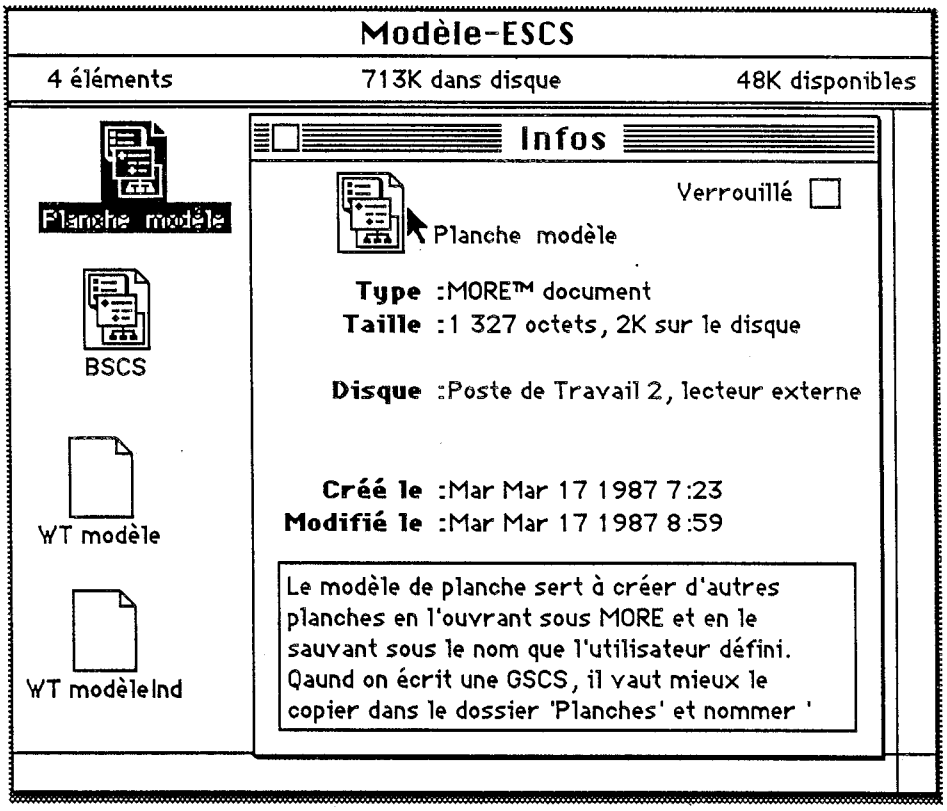
être modifié.

Liens Chaque objet possède un lien vers la définition correspondante dans ce fichier.

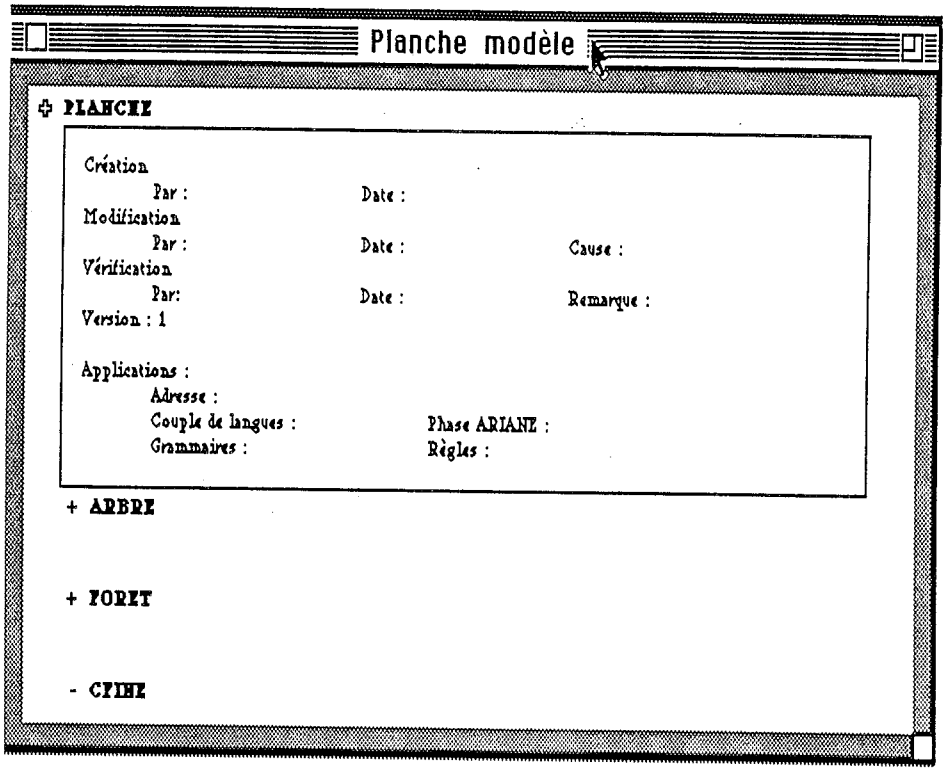
1.2. Planche [More]

Description Le modèle de planche est un squelette à compléter par l'utilisateur. Il ne peut pas être modifié, mais seulement copié : pour créer une planche, il suffit de compléter ce modèle et de le sauver sous un autre nom.

Exemple



Vue externe.

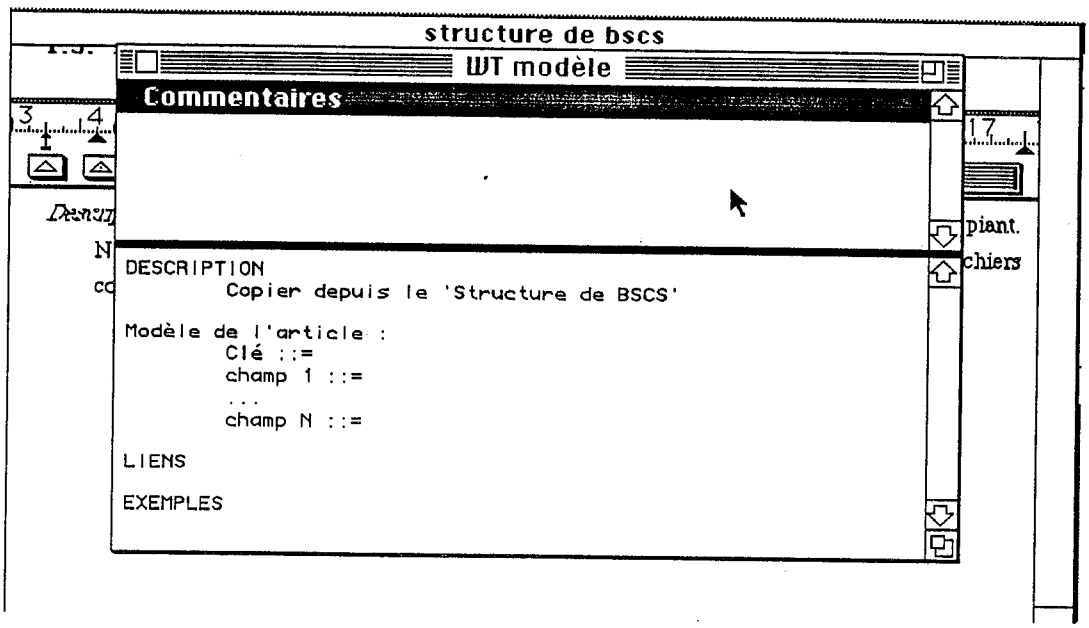


Vue interne. Seul le premier niveau du modèle est affiché.

1.3. Dictionnaire [WinTool]

Description. Modèle de fichier WinTool. Ne peut être modifié mais seulement copié : pour créer un nouveau fichier WinTool, il faut faire une copie du modèle et du fichier de même nom suffixé par Ind. Il faut nommer les 2 fichiers copiés de la même manière (toto et totoInd).

Exemple



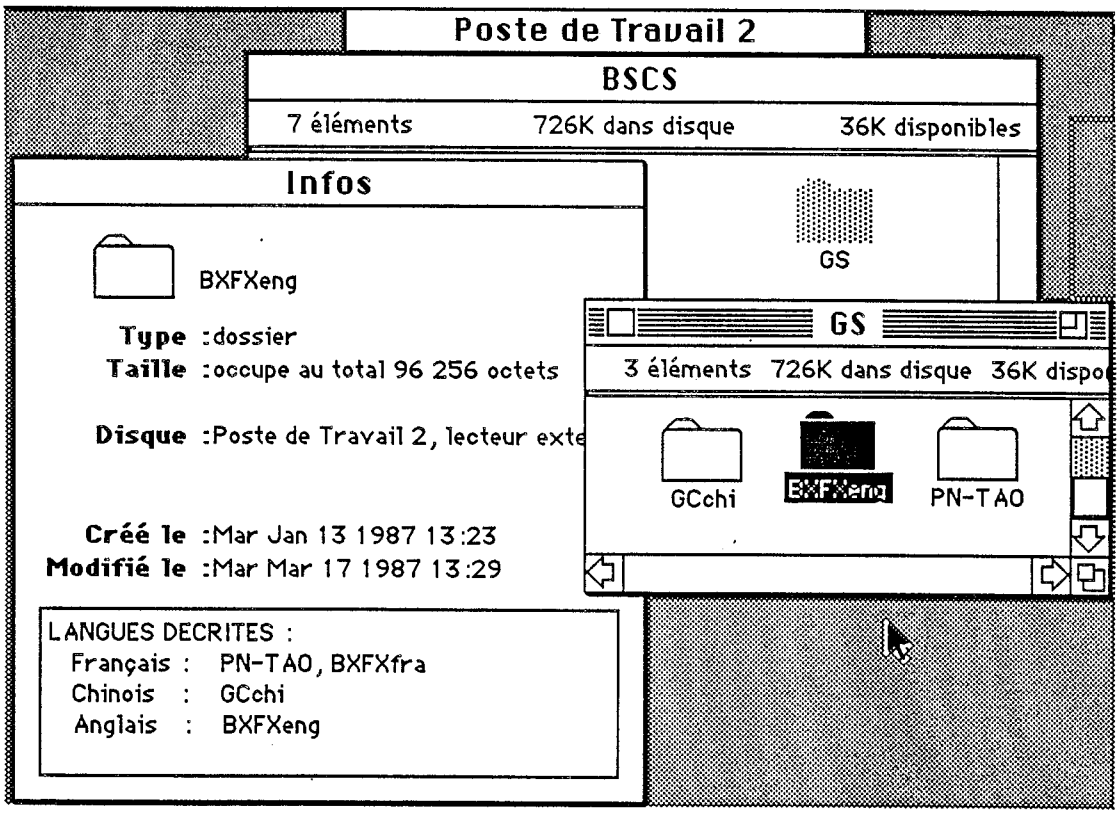
2. GSCS [dossier]

Description. Ce dossier contient l'ensemble des GSCS de la BSCS, avec une GSCS par sous-dossier. Ce nœud ne peut être modifié.

Attributs

LANGUES DECRITES :
-- Français : PN-TAO, BXFXfra
-- Chinois : GCchi
-- Anglais : BXFXeng

Exemple.



2.1. GSCSxxxx [dossier]

Description. Ce dossier contient les différents objets qui constituent la GSCS (qui sont écrits en LSCS). Une nouvelle GSCS par l'utilisateur est créée en copiant le squelette de GSCS avec le nom de la nouvelle GSCS.

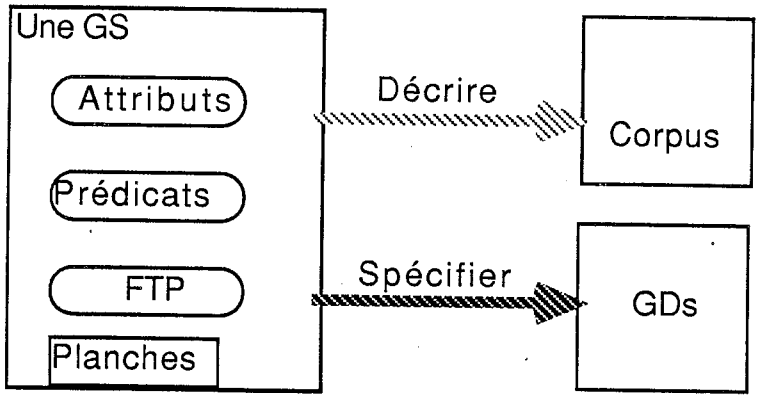
Attributs

LANGUE :	<idf>	-- english
DOMAINE :	<idf>	-- chimie
TYPOLOGIE :	<idf>	-- manuel de cours
CORPUS :	<idf>	-- BXFXENG

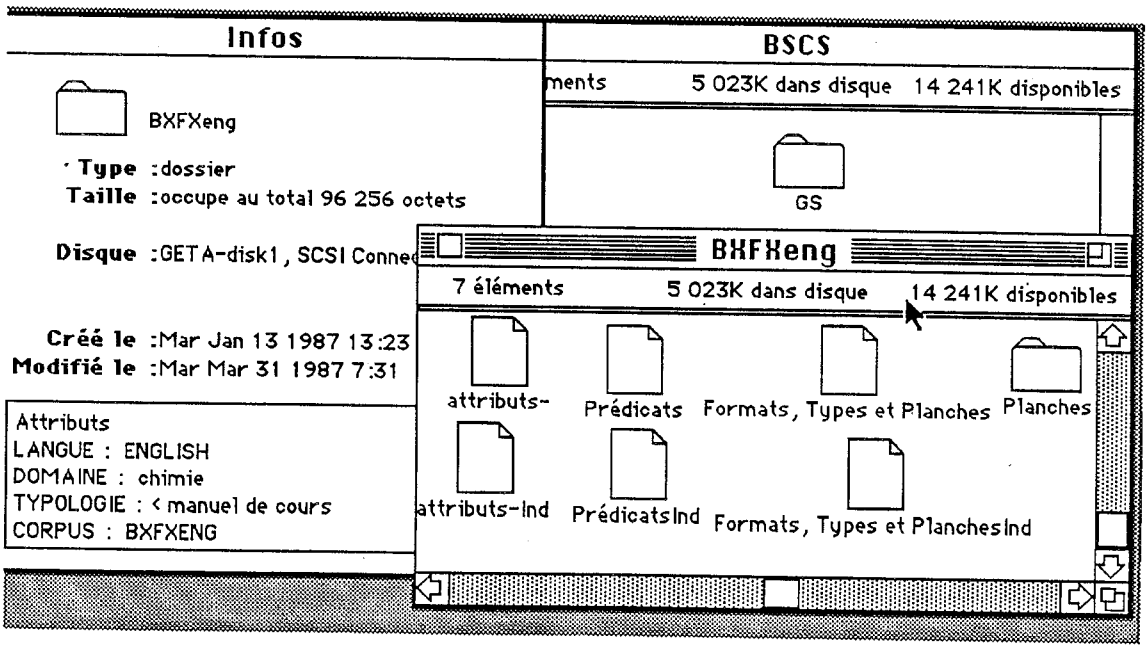
Syntaxe Le nom du dossier est le nom de la GSCS. La longueur du nom ne doit pas dépasser 8 caractères. Ce nom doit respecter les conventions suivantes :

<nom de GSCS> ::= <nom du projet> <3 lettres pour la langue> [<suffixe>].

Liens



Exemple



2.1.1. Attributs [WinTool]

Description. Ce dictionnaire contient les définitions des attributs (types, étiquette et décoration) de la GSCS. Ces définitions sont établies à partir du modèle linguistique mis en place après l'analyse de corpus.

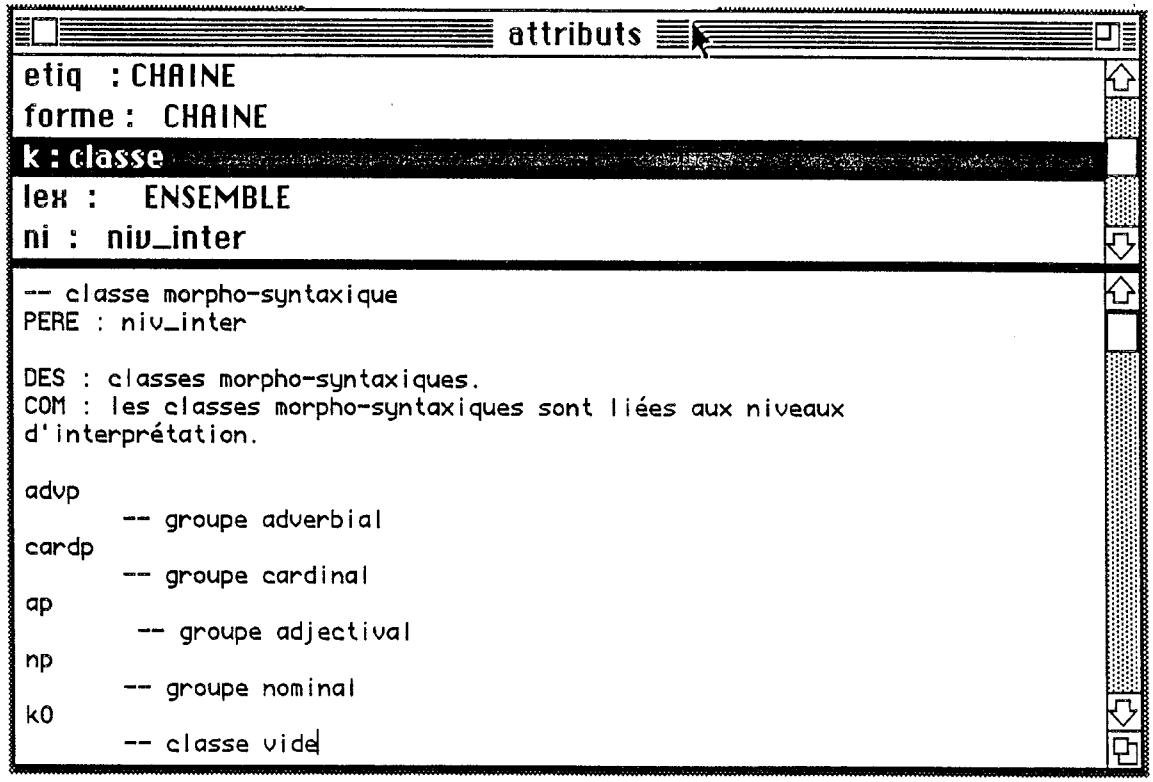
Syntaxe. Un article du dictionnaire contient une définition de type, d'étiquette ou de décoration. Ces définitions sont rédigées en LSCS. Leur syntaxe est présentée sous forme de diagrammes de Floyd dans /DOCUMENTS/ESCS/LSCS/Attributs. Une définition est décomposée en une clé et un corps selon les conventions suivantes :

<clé> ::= <nom d'attribut> : <type d'attribut> |
<type d'attribut> = (SCALAIRE | ENSEMBLE | CHAINE | ENTIER | BOOLEEN)

<corps> ::= <commentaire et exemple>
PERE : <attribut père>
<valeurs>*

Liens Les attributs sont utilisés dans les planches, les formats et les prédicats. Ils ont été définis à partir du modèle linguistique.

Exemple



2.1.2. Prédicats [WinTool]

Description. Ce dictionnaire contient les définitions des prédicats de la GSCS. Les prédicats portent sur les attributs de la GSCS. Ils sont définis à partir des attributs et de la définition des syntagmes.

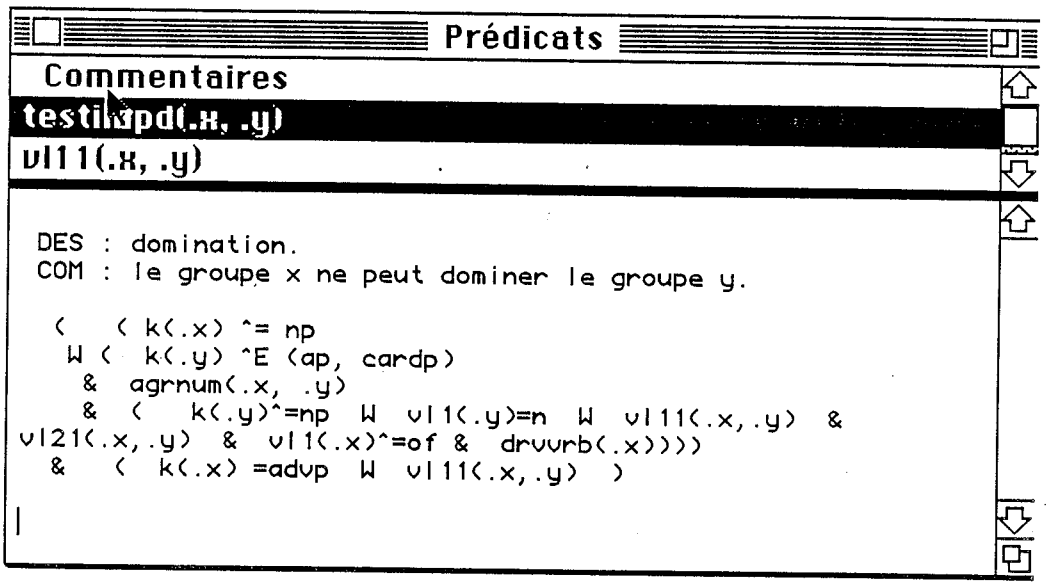
Syntaxe. Les définitions des prédicats sont rédigés en LSCS. Leur syntaxe est donnée sous la forme de diagramme de Flyod dans /DOCUMENT/ECSC/LSCS/Prédicat. Une définition est décomposée dans un article selon les conventions suivantes :

<clé> ::= <nom du prédicat> <paramètres formels>

<corps> ::= <commentaire> -- chaque paramètre doit être commenté.
<définition>

Liens Les prédicats sont utilisés dans des planches. Ils portent sur les attributs et les propriétés.

Exemple



2.1.3. Formats, Types et Planches [WinTool]

Description. Ce dictionnaire contient les propriétés (listes d'attributs), les classes de planches (une classe correspondant à un syntagme établi lors de l'analyse de corpus) et la liste des planches de la GSCS (avec une description de leurs relations, des syntagmes qu'elles doivent décrire, des exemples de ces syntagmes et de la structure générale associée à ces syntagmes). Il contient donc 3 types d'articles : PROPRIETE, CLASSE, PLANCHE.

Syntaxe. Les propriétés sont écrites en LSCS. Les 3 types d'articles respectent les conventions suivantes :

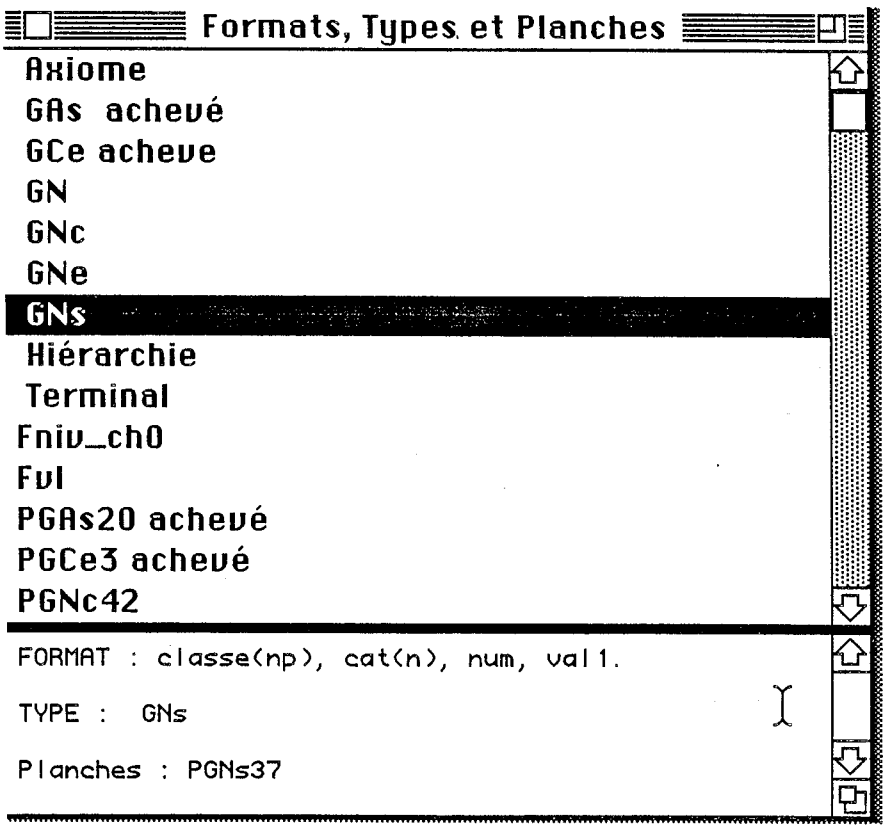
PROPRIETE :
<clé> ::= <nom de la propriété>
<corps> ::= <commentaire> <liste des attributs>

CLASSE :
<clé> ::= <nom de la classe>
<corps> ::= CLASSE : <nom de la classe>
(SOUSCLASSE : (<nom de classe>* |
PLANCHES : <nom de planche>*)
PPTE : (<nom de propriété> | <liste d'attributs>)
<commentaires>

DPLANCHE :
<clé> ::= <nom de planche>
<corps> ::= CLASSE : <nom de classe> -- à laquelle appartient la planche.
CAS TRAITES : <description des cas traités>
EXEMPLES : (<exemple> <référence dans le corpus>)$^{+}$
ARBRE : <structure de la classe>
REFERENCES : (AUCUNE | (<schéma d'arbre : <nom de planche>$^{+}$))
REFERRES : <nom de planche>$^{+}$

Liens. Les définitions sont produites à partir des attributs et des définitions des syntagmes (qui contiennent eux-même des liens vers le corpus). Elles sont utilisées dans les planches.

Exemple

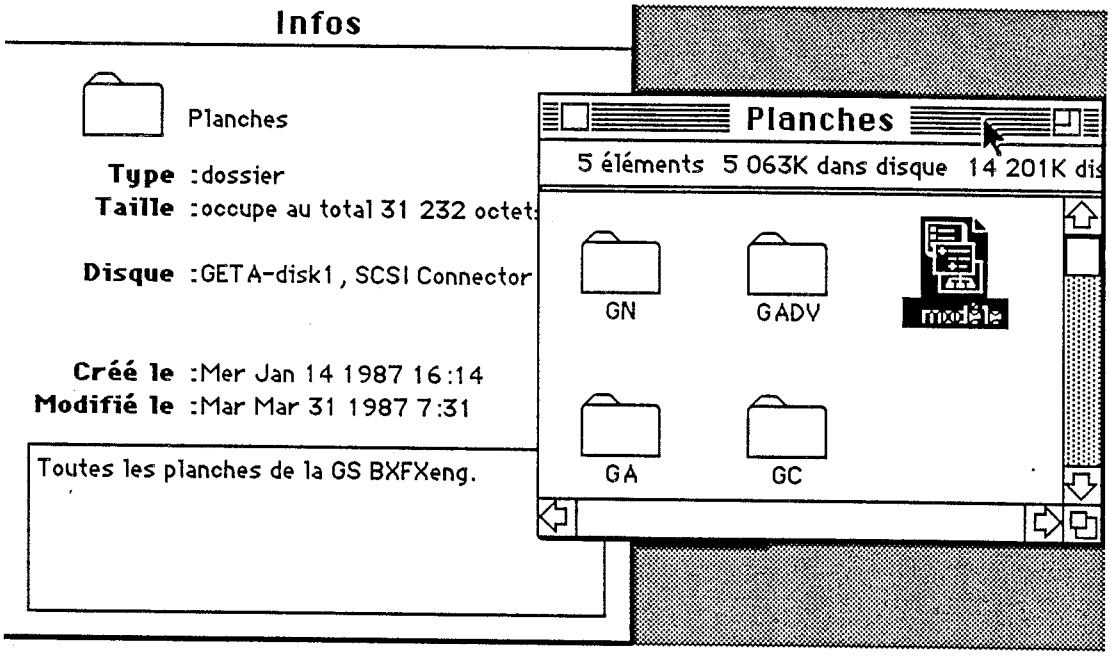


Exemple de classe.

2.1.4. Planches

Description. Ce dossier contient les planches de la GSCS, chaque planche se trouvant définie dans une structure More. Les planches peuvent être regroupées dans des dossiers, par classe et sous-classe. Ce dossier contient également un modèle de planche. Les planches sont écrites à partir de la liste des planches contenue dans le dictionnaire 'Propriétés, classes et liste des planches'.

Exemple



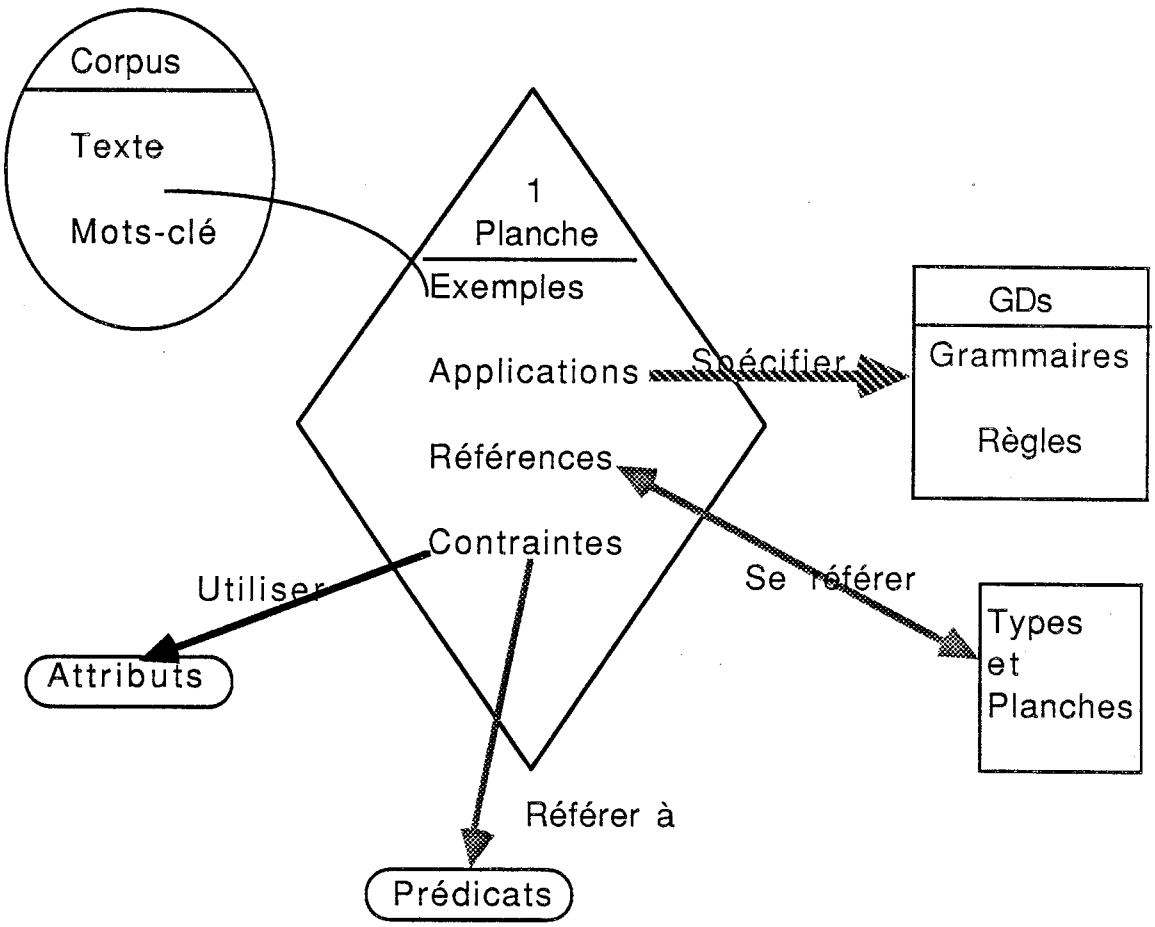
2.1.4.1. Planche [More]

Description. Cette structure More contient la définition d'une planche de la GSCS. Une planche est écrite à partir de sa description contenue dans la liste des planches (dictionnaire 'Propriétés, classes et liste des planches').

Syntaxe.

Le nom d'une planche doit respecter les conventions suivantes : il débute par la lettre P avec le nom de la sous-classe ou de la classe à laquelle la planche appartient et un numéro.
La planche est écrite en LSCS. La syntaxe est donnée sous forme de diagramme de Flyod dans /BSCS/DOCUNMENT/ESCS/LSCS/"PLANCHE". Un squelette de planche se trouve dans /BSCS/ESCS/PLANCHE.

Liens. Ce sont les liens déjà établis dans la description qui se trouvent dans la liste des planches plus les liens vers les définitions d'attributs, de prédicats et de propriétés utilisés dans la planche.



Exemple La planche suivante se trouve également dans le menu de Template du programme More.

PLANCHE PGNs37

TYPE : GNs -- GROUPE NOMINAL simple

CAS TRAITES : groupes gouvernés par un nom commun

EXEMPLES :

1) "THE MOST IMPORTANT COMPONENT"
REF. CORPUS BXFXENG, TEXTE IBM, PAGE 3

2) "UNUSUAL SLOW GAS REACTIONS"
REF. CORPUS BXFXENG, TEXTE EXEMPLE, PAGE 3

3) "THE FIVE MOST INTERESTING POINTS"
REF. CORPUS BXFXENG, TEXTE EXEMPLE, PAGE 3

PLANCHES APPELANTES : PGNs38, PGNc42

Création
Par : Y.YAN Date : Février 1987

Modification
Par : Date : Cause :

Vérification
Par: Date :

Version : 1

Application :

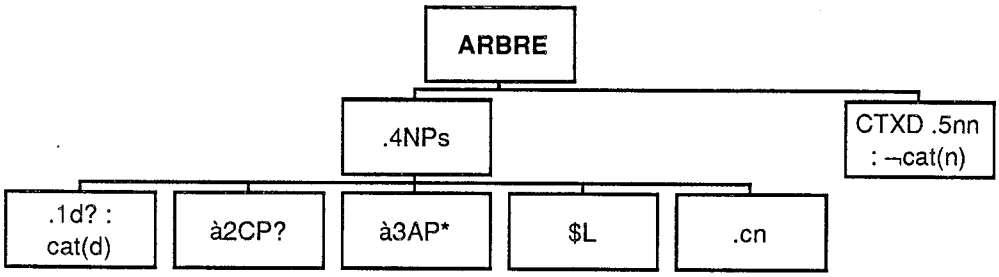
Adresse :
Code langue : Phase ARIANE :
Grammaires : Règles :

REFERENCES

.4NPs ($L, .cn) : (PGNe5, PGNs7)
-- points, gas reaction

à2CP? : (PGCe3, PGCs34)
-- five

à3AP* : (PGAe4, PGAs20, PGAs21, PGAc36 (..PGAc35))
-- most interesting

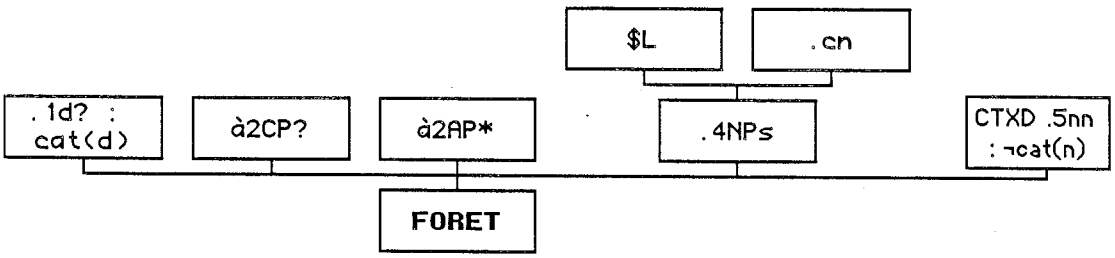


-- Le mot suivant ne peut pas être un nom, par exemple une préposition : of.

CSCHEMA

rs(à3AP*)=qual -- fin de ligne équivalant à l'opérateur logique ET.

rs(à2CP?)=qfier



CSCHEMA

num(.1d?) ∩ num(.2CP?) ∩ num(.cn) ≠ num0

CFINES

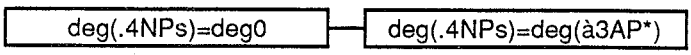
num(.4NPs)=num(F.1d?)∩num(FàCP*)∩num(F.4NPs)

num(.1d?)=num(F.1d?)∩num(FàCP*)∩num(F.4NPs)

num(à2CP*)=num(F.1d?)∩num(FàCP*)∩num(F.4NPs)

-- Quand le déterminant est quantificateur, sa rs sera quantificateur.

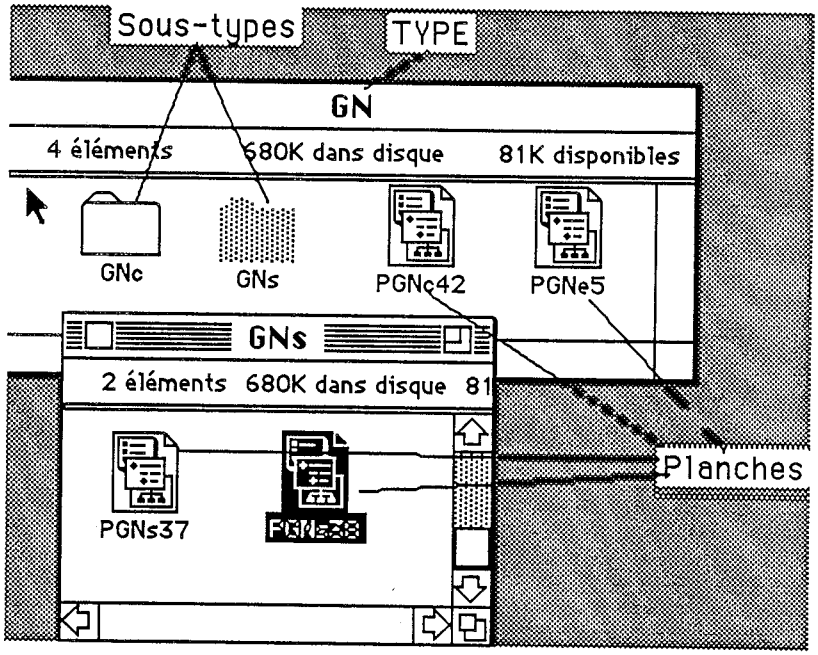
-- Quand le groupe nominal dans la forêt n'a pas de degré, le groupe nominal défini par cet arbre
-- dans cette planche aura la valeur de degré des groupes adjectivaux.



2.1.4.2. Type [dossier]

Description Ce dossier contient les planches décrivant une même classe de syntagmes, par exemple, toutes les planches décrivant les groupes nominaux. L'ensemble des groupes nominaux peut être lui-même divisé en sous-classes, les groupes nominaux élémentaires, simples et complexes. A chaque sous-classe correspond alors un sous-dossier.

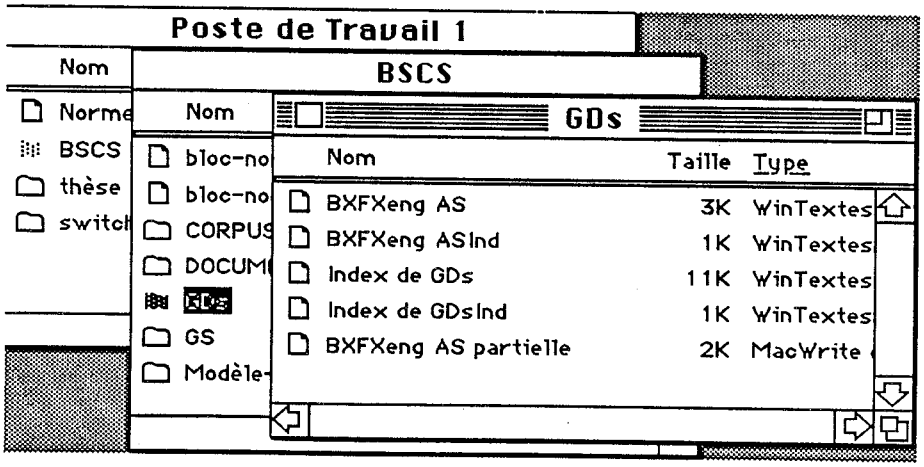
Exemple



3. GD [dossier]

Description Ce dossier contient les grammaires dynamiques –programmes linguistiques écrits dans l'un des LSPL d'ARIANE– qui réalisent l'analyse ou la génération du langage spécifié par les GSCS correspondantes.

Exemple



3.1. Index [WinTool]

Description Le fichier Index contient la liste des grammaires dynamiques et les liens avec les GSCS qui les spécifient. Les grammaires dynamiques ne peuvent être exécutées que sur une machine IBM : le fichier contient donc le descriptif des grammaires dynamiques qui se trouvent sur cette machine.

Syntaxe Les articles du dictionnaire doivent respecter l'une des 3 conventions suivantes :

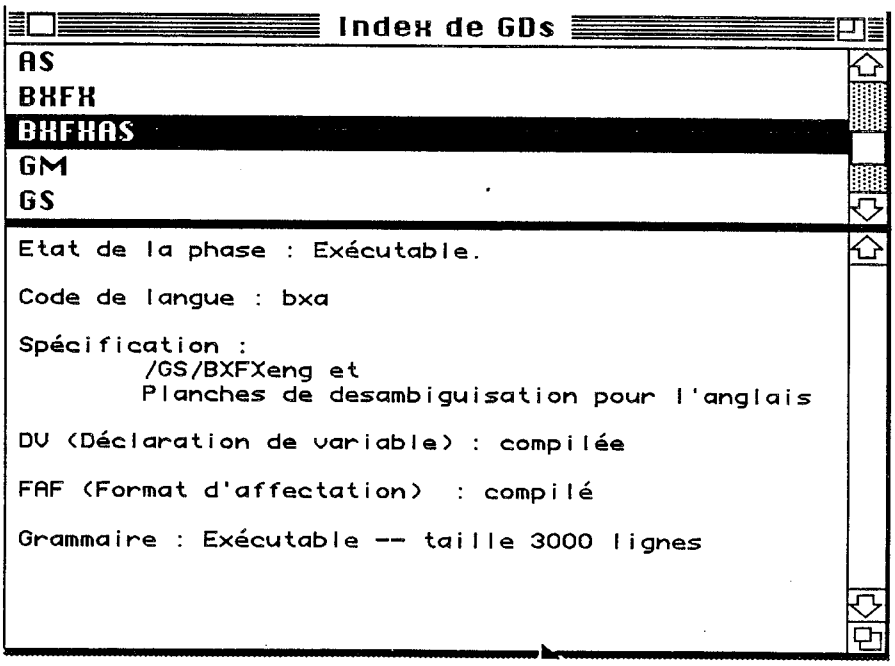
1. <clé> ::= <nom de projet>
   <corps> ::= MACHINE : <nom de machine> -- où le projet est implanté.
   COMPTE : <nom d'utilisateur>
   RESEAU : <Uid>@<NoeudID>
   LANGUES : <couple de langues>* -- traitées dans le projet
   RESPONSABLE : <nom du responsable du projet>
   ETAT : (exécutable | compilable | en développement)
   COMPOSANTS : <nom de composant>*

2. <clé> ::= <nom de projet>.(analyse | transfert | génération) -- phase traitée
   <corps> ::= ETAT : (exécutable | compilable | en développement) -- de la phase
   LANGUES : <codes langues>* -- traitées dans la phase
   SPECIFICATION : <idf>
   (<composant> : (exécutable | compilable | en développement))+

3. <clé> ::= (analyse | transfert | génération)
   <corps> ::= <nom de projet>* -- dont la phase existe.

Liens Les GD sources se trouvent sous BSCS/GD ou dans la machine IBM accessible par un réseau.

Exemple



3.2. GD : grammaire dynamique [MacWrite]

Description Ce fichier contient une grammaire dynamique qui peut être envoyée sur la machine IBM pour compilation et exécution.

Syntaxe Une grammaire dynamique est écrite dans un Langage Spécialisé pour la Programmation Linguistique dont elle doit, bien sûr, respecter la syntaxe.

Liens Chaque grammaire dynamique est spécifiée par une GSCS, une règle de la grammaire correspondant généralement à une planche (mais une planche peut spécifier plusieurs règles différentes).

3.3. GD : grammaire dynamique [WinTool]

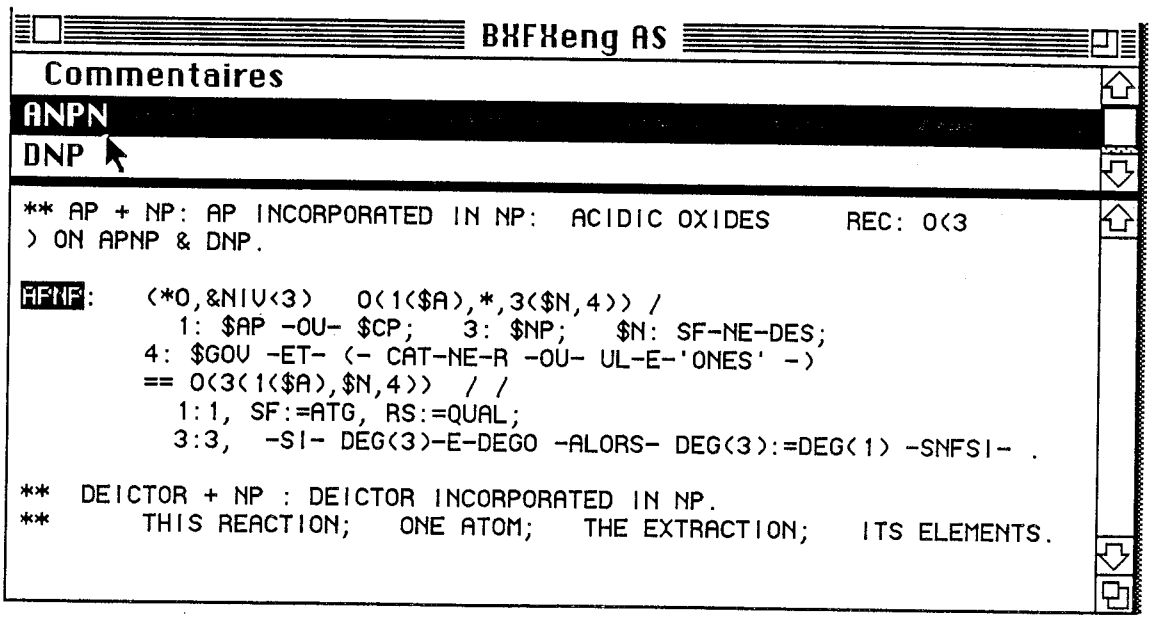
Description Ce dictionnaire contient des parties d'une grammaire dynamique qui peuvent être envoyées sur la machine IBM pour compilation et exécution.

Syntaxe

<clé> ::= <nom de la règle ou de la GD>
<corps> ::= <corps de règle ou de GD>

Liens Chaque grammaire dynamique est spécifiée par une GSCS, une règle de grammaire correspondant généralement à une planche (mais une planche peut spécifier plusieurs règles différentes).

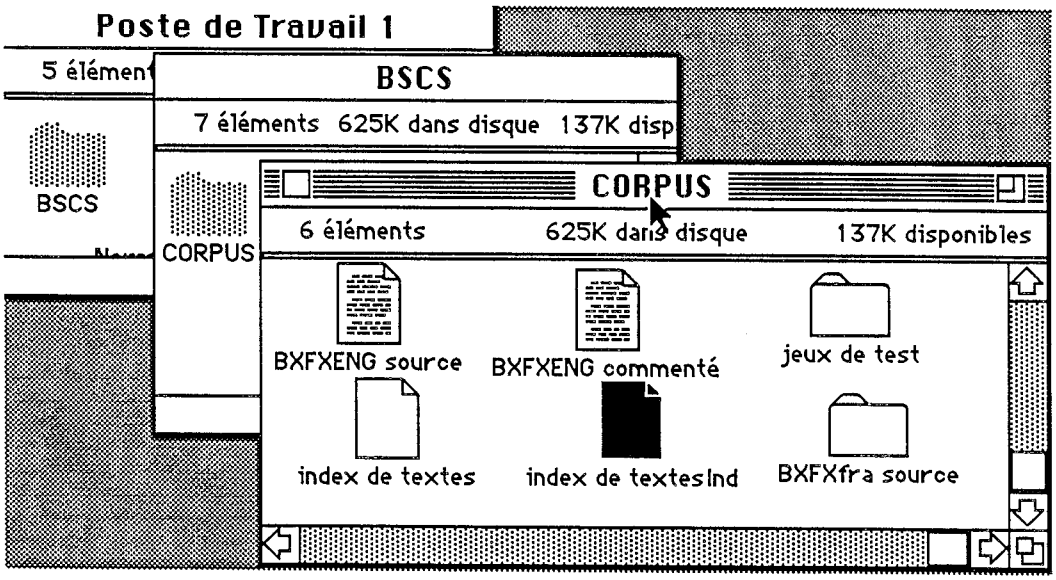
Exemple



4. Corpus

Description Dossier contenant les corpus de la BSCS, ainsi qu'un index de ces corpus.

Exemple



4.1. Index [WinTool]

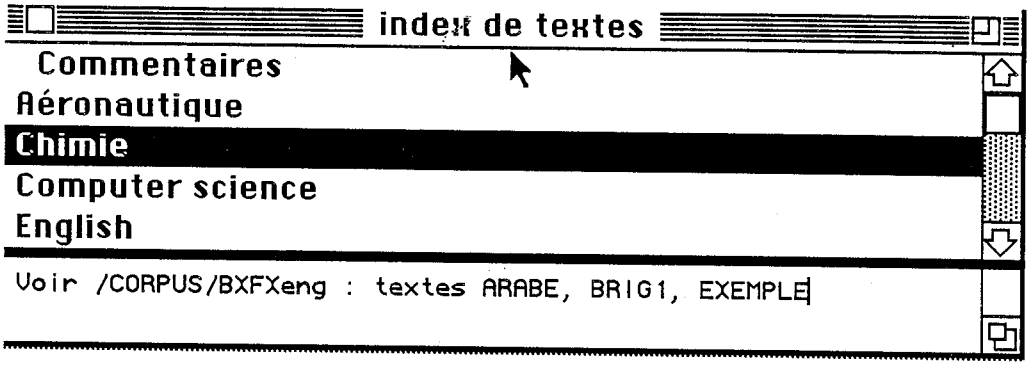
Description Index des textes de corpus.

Syntaxe

```
<clé>    ::= <id>          -- langue, domaine, syntagme etc.
<corps> ::= <référence>  -- corpus, texte et page où se trouve le texte correspondant.
```

Liens Vers les fichiers du corpus.

Exemple



4.2. Corpus [MacWrite]

Description Corpus sources ou commentés.

Attributs

LANGUE	: <id>	-- exemple: ENGLISH
DOMAINE	: <id>	-- exemple: chimie
TYPOLOGIE	: <id>+	-- exemple: manuel de cours

Syntaxe Un nom de corpus doit être préfixé par le nom du projet. Un suffixe (source/commenté) peut s'ajouter au nom du corpus afin d'indiquer le type du corpus. Un corpus est composé de textes. Tous les textes d'un corpus sont paginés. Dans un corpus commenté, les exemples de phénomènes linguistiques particuliers sont soulignés et un mot-clé indiquant la nature du phénomène est rajouté en index ou en exposant.

Liens Les corpus sources sont utilisés pour tester les GD. Les exemples des GSCS sont tirés des corpus commentés.

Exemple

---------- Corpus commenté ----------

*A SUBSTANCE CAN BE EITHER A SOLID , A LIQUID OR A GAS .

*THESE ARE CALLED THE GAe THREE STATES OF THE MATTER GNc.

*A SOLID HAS A DEFINITE SHAPE WITH DISTINCT BOUNDARIES .

*ITS VOLUME GNe IS HARDLY AFFECTED BY CHANGES IN TEMPERATURE AND PRESSURE .

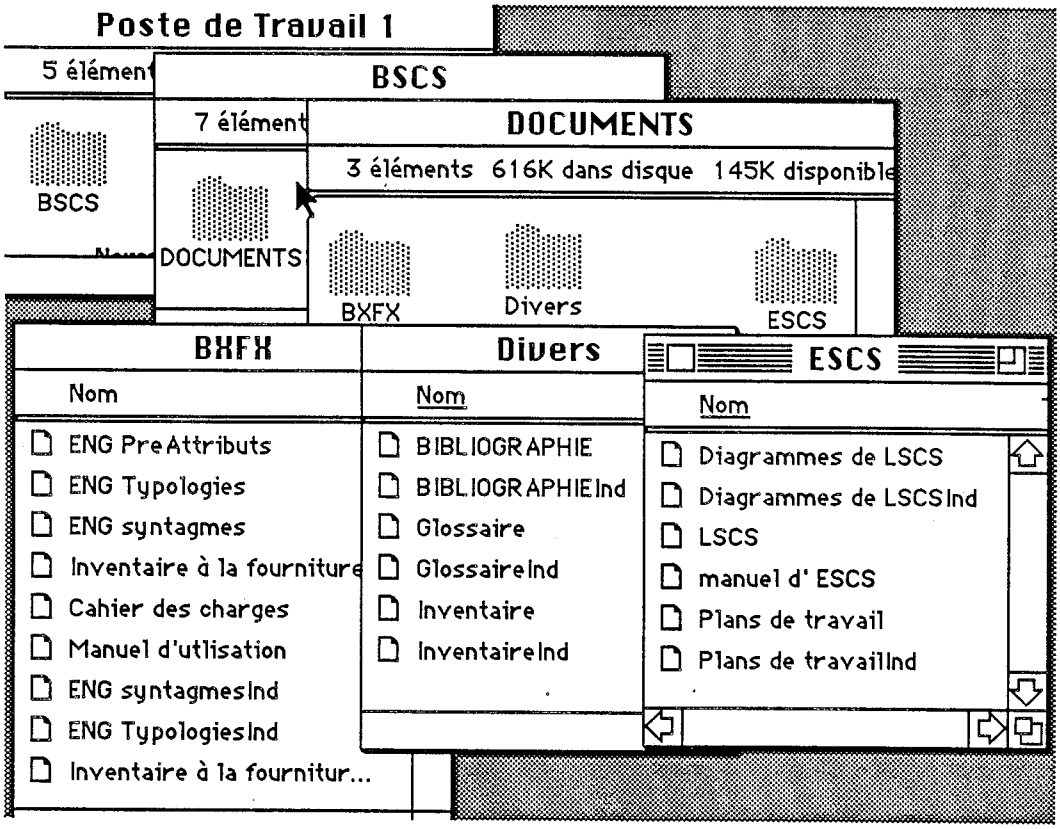
*A GAS HAS NEITHER A DEFINITE SHAPE NOR VOLUME .

---

5. Documents [dossier]

Description Documentation de la BSCS.

Exemple



5.1. ESCS [dossier]

Description Documentation du système ESCS.

5.1.1. LSCS [WinTool]

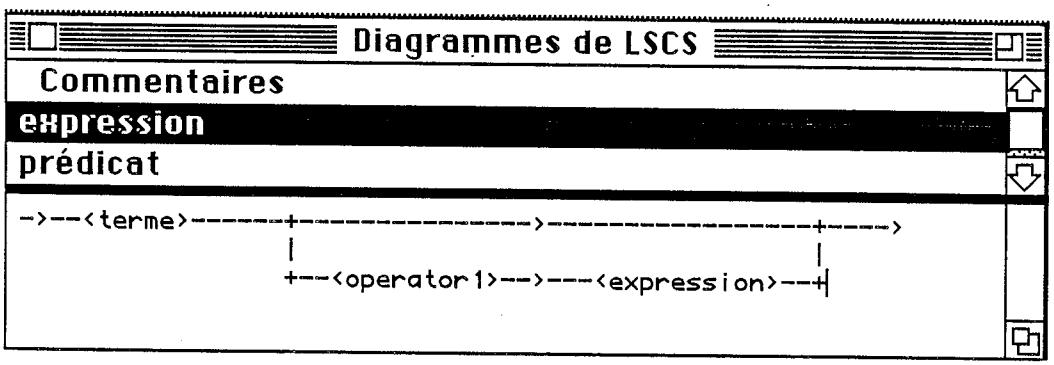
Description Syntaxe de LSCS sous forme de diagrammes de Flyod.

Syntaxe

<clé> ::= <nom d'unité syntaxique>
<corps> ::= <diagramme de l'unité>
<commentaire et Exemple>

Liens Le dictionnaire peut être utilisé comme aide en ligne lors de l'écriture de GSCS.

Exemple



5.1.2. Manuel [WinTool]

Descriptiion Manuel de l'utilisateur de ESCS : outils et méthodologie.

5.1.3. Plans de travail [WinTool]

Description Inventaire de tâches pour la réalisation d'un système TAO et manuel en ligne. Ce dictionnaire contient les articles LISTE DES TACHES et LISTE DES OPERATIONS.

Syntaxe

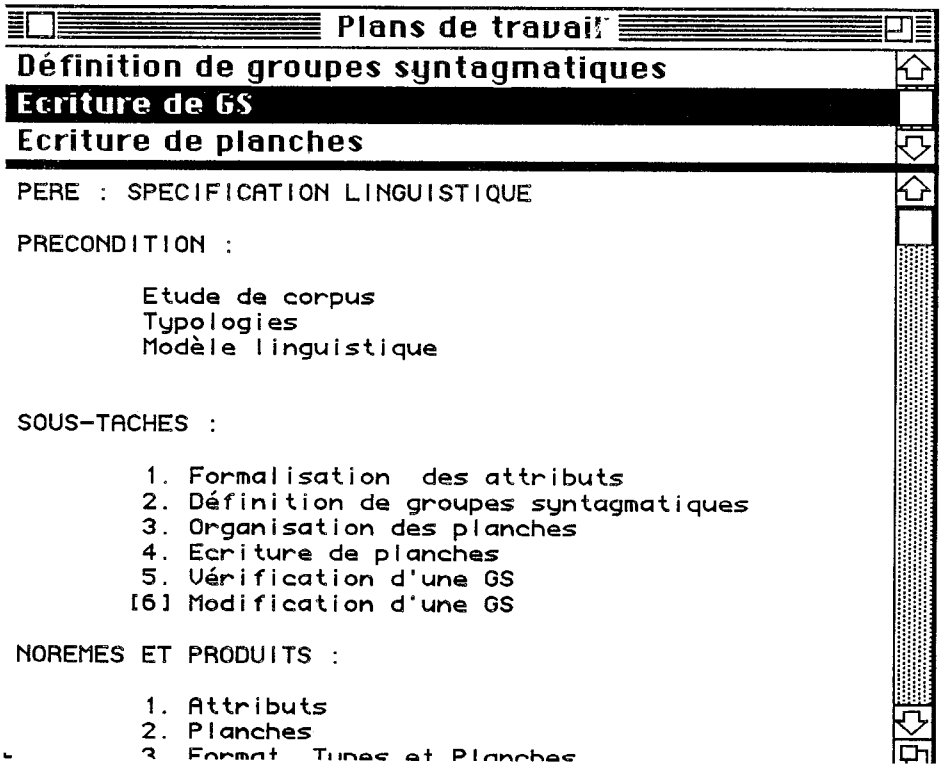
Tâche

<clé> ::= <nom de la tâche>
<corps>::= PERE : <clé de la tâche père>
PRECONDITIONS: <lignes> -- Conditions à respecter avant de commencer la tâche.
SOUS-TACHES : (<texte> | <clé>) -- Indique les sous-tâches pouvant s'exécuter parallèlement.
PRODUIT : <texte> -- résultats de la tâche.
EXEMPLE : <texte> -- démonstration de la tâche sur des exemples.

Opération

<clé> ::= <nom de l'opération>
<corps> ::= <texte> -- Instruction de l'opération ou référence au manuel.

Exemple



5.2. Divers [dossier]

Description Documentation diverse : glossaire, références bibliographiques, inventaires des ressources,...

5.2.1. Inventaire des ressources [WinTool]

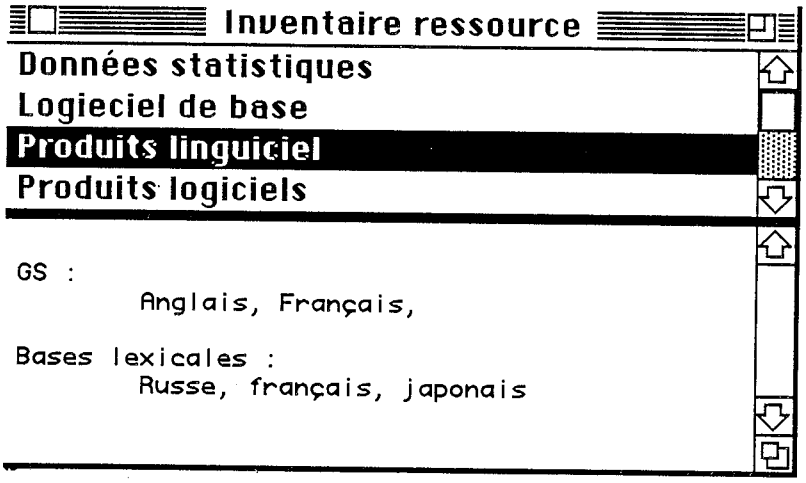
Description Inventaire des ressources de l'atelier logiciel spécialisé dans la production de systèmes de TAO.

Syntaxe

```
<clé>    ::= <nom de la ressource>
<corps>  ::= FONCTION : <texte>          -- description du produit.
             PRIX : <texte>              -- en H/M du développement
             DELAI : <texte>             -- temps minimal de réalisation et d'installation.
             SOURCE : <texte>            -- où se trouve le produit.
             [ CONDITIONS : <texte> ]    -- d'utilisation.
```

Liens Vers les ressources.

Exemple



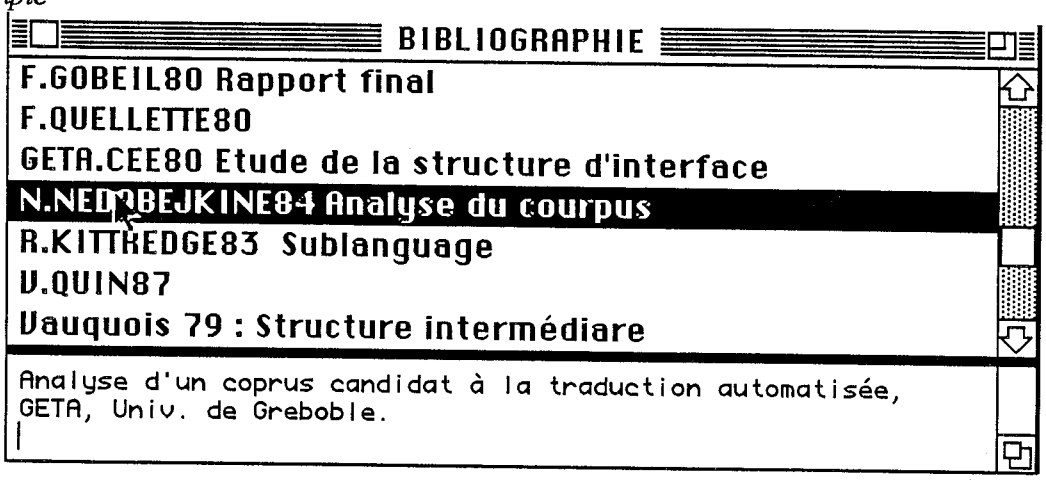
5.2.2. Bibliographie [WinTool]

Description Références bibliographiques pour l'utilisation de ESCS.

Syntaxe

<clé ::= <NOM> <Année>
<corps> ::= <texte< -- corps de la référence.

Exemple



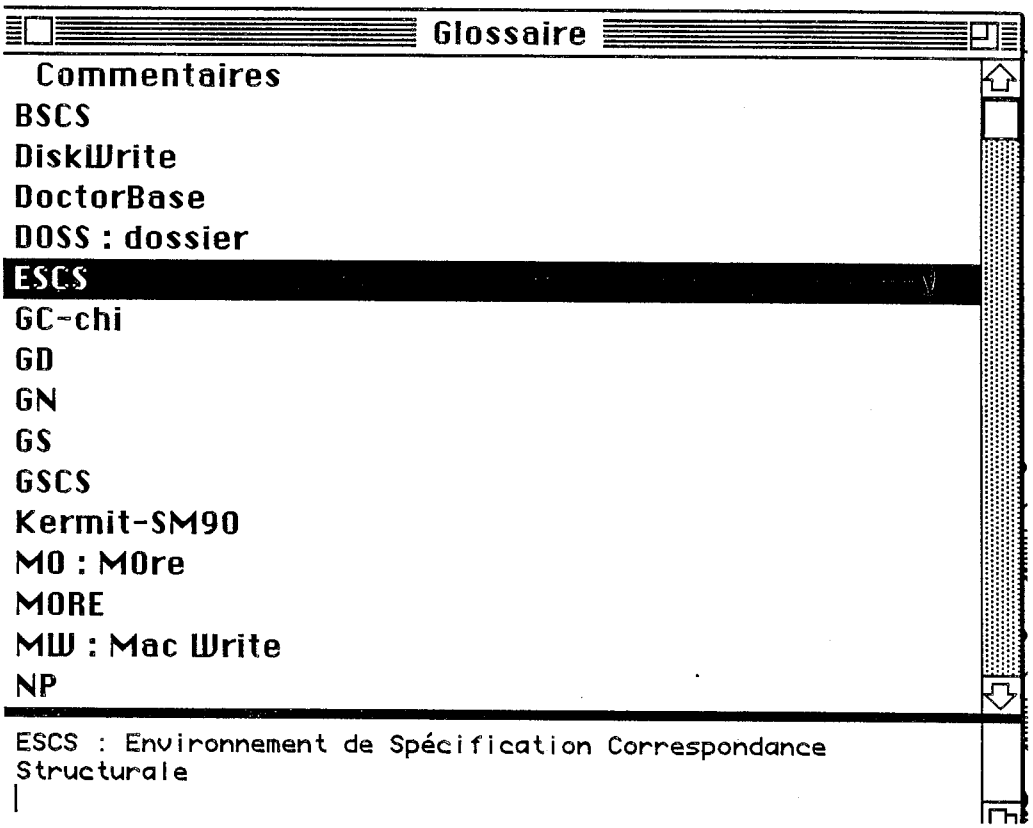
5.2.3. Glossaire [WinTool]

Description Glossaire de ESCS. Il peut être augmenté par l'utilisateur (un article ne peut être détruit que par l'auteur de l'article).

Syntaxe

<clé ::= <terme>
<corps> ::= <texte>

Exemple



5.3. PROJET [dossier]

Description La documentation à propos d'un projet : cahier des charges, fournitures,...

5.3.1. Cahier des charges [MacWrite]

Description Résultat de l'étude d'opportunité définissant le cadre du projet.

5.3.2. Liste des fournitures [WinTool]

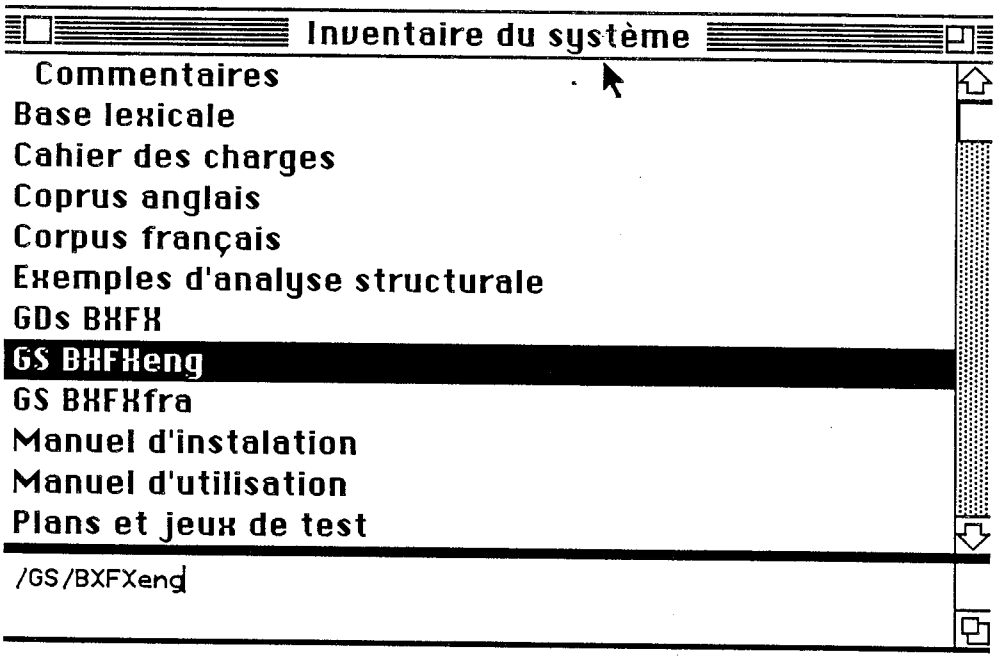
Description Liste des fournitures établie par le cahier des charges.

Syntaxe

Clé ::= <nom du document>
<corps> ::= <source du produit> -- Forme hiérarchique possible : un produit est composés de sous-produits.

Liens Vers chaque fourniture.

Exemple



5.3.3. Typologies [WinTool]

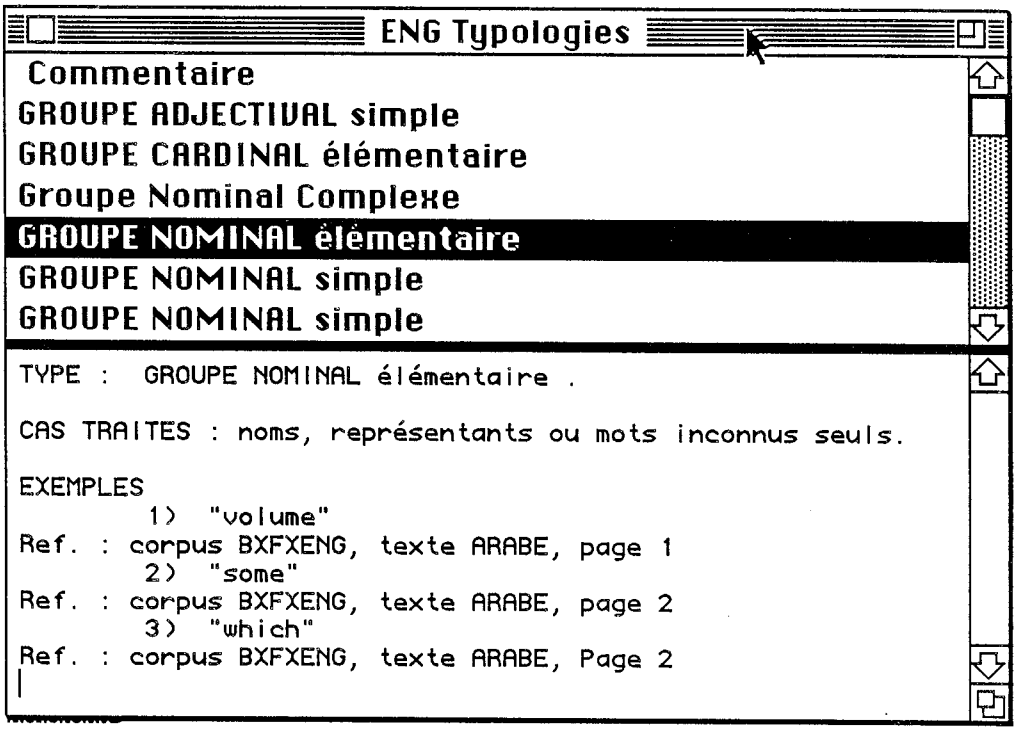
Description Typologie retenue dans un corpus. Le nom du fichier commence par 3 lettres indiquant la langue.

Syntaxe

<clé> :: = <nom>
<corps> ::= TYPE : <idf> -- type du phénomène décrit.
CAS TRAITES : <texte>
EXEMPLES : (<texte> [<ref-corpus>])*
<commentaires>

Liens Des exemples vers le corpus.

Exemple



5.3.4. Syntagmes [WinTool]

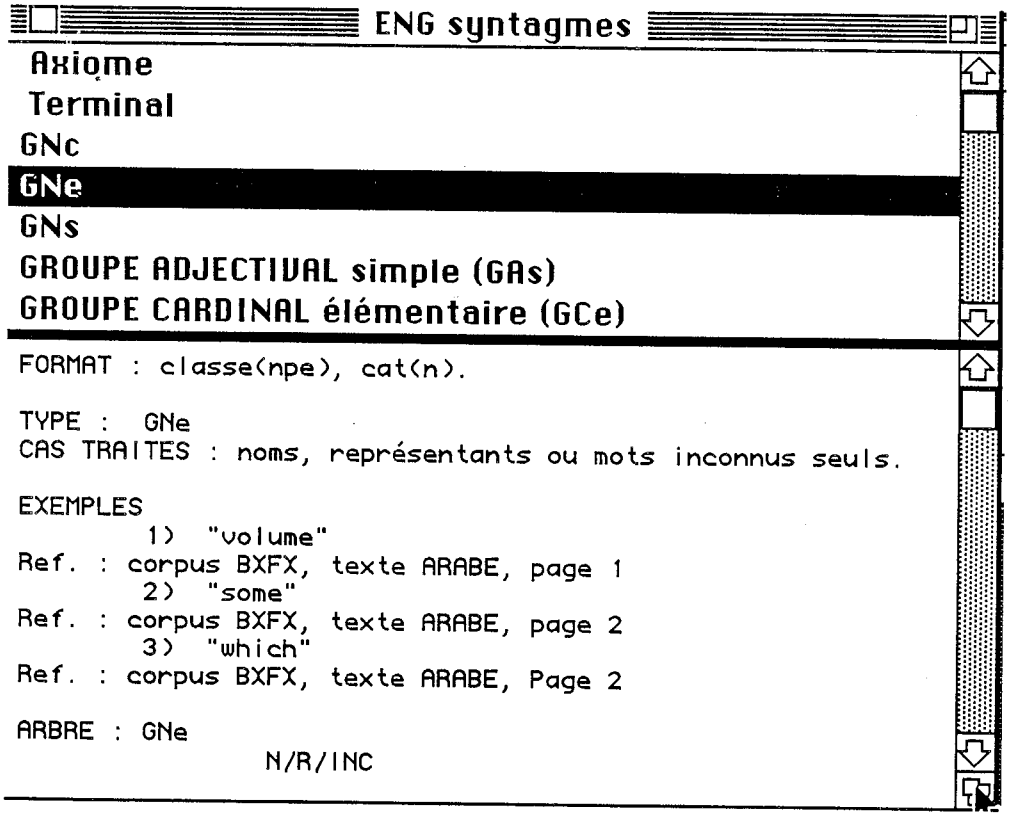
Description Ensembles des syntagmes du corpus. Ce fichier est dérivé de l'étude typologique.

Syntaxe Le nom du fichier commence par trois lettres indiquant la langue.

```
<clé> ::= <nom de syntagme>
<corps> ::=  FORMAT : <attributs>                    -- carctérisant le syntagme
             TYPE : <idf>                            -- du syntagme
             SOUS-TYPE : <nom de syntagme>

<corps> ::=  FORMAT : <attributs>                    -- caractérisant le syntagme
             TYPE : <idf>                            -- du syntagme
             CAS TRAITES : <texte>                   -- description des cas représentés par les exemples.
             EXEMPLES : ( <texte> [<ref-corpus>])*
             ARBRE : <arbre>                         -- structure décrivant le syntagme.
```

Liens Vers le corpus.

Exemple



5.3.5. Manuel d'utilisation [MacWrite]

Description Manuel d'utilisation pour le système produit.

6. Bloc-note [WinTool]

Description Aide-mémoire d'un projet. Un article ne peut être détruit que par son auteur.

Syntaxe

```
<clé> ::=     <chaîne>
<corps> ::=   DATE : <date>          -- de ccréation.
              <texte>
```

REFERENCES BIBILIOGRAPHIQUES
(citées)

Abréviations

AI	Artificial Intelligence.
CACM	Communications of the ACM.
CETA	Centre d'Etudes pour la Traduction Automatique, CNRS.
CNRS	Centre National de la Recherche Scentifique.
COLING-80	Proceedings of COLING 1980, Tokyo.
COLING-82	Proceedings of COLING 1982, Prague.
COLING-84	Proceedings of COLING 1984, ACL, Stanford University.
COLING-86	Proceedings of COLING 1986, IKS & ACL, Bonn.
COLGATE-85	Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, COLGATE University, Hamilton, N.-Y., U.S.A, August 14-16, 1985.
GETA	Groupe d'Etudes pour la Traduction Automatique, USTMG-CNRS.
INPG	Institut National Polytechnique de Grenoble.
LUGANO-84	ISSCO Tutorial on Machine Translation, Lugano, Switzerland, April 2-6, 1984.
TSI	Technique et Science Informatique.
USTMG	Université Scientifique, Technologique et Médicale de Grenoble.

Aarts J., Heuvel T.V.D. (1985) : *Computational Tools for the Syntactic Analysis of Corpora.* Linguistics, 23 : 2, 303-336.

Andre E., Moreau B., Rougeot B. (1986) : *Vers un atelier flexible et intégré de logiciel : le projet CONCERTO.* In CNET 86.

Apple (1985) : *Macintosh Plus.* Apple France (Filiale d'Apple Computer Inc).

Boehn R.W. (1976) : *Software Engineering.* IEEE Transactions on Computer. 25 : 12, 1227-1241.

Boehn R.W. (1982) : *Les facteurs du coût de logiciel,* TSI, 1 : 1, 5-25.

Boitet C., Guillaume P., Quézel-Ambrunaz M. (1978) : *Manipulation d'arborescences et parallélisme : le système ROBRA.* COLING-78.

Boitet C., Gerber R. (1984) : *Expert Systems and Other New Techniques in MT.* COLING-84, 468-471.

Boitet C. (1986) : *The French National MT-Project: Technical Organization and Translation Results of CALLIOPE-AERO.* Conference on Translation Mechanization, Copenhagen.

Caine S., Gordon K. (1975) : *PDL-- A Tool for Software Design.* In : Proceedings of the National Computer Conference. AFIPS Press, 271-276.

Caplain M. (1978) : *Langage de spécifications.* Thèse d'état INPG, Grenoble.

Catchings B., Cruz F., Schilit B. (1985) : *Kermit Version 0.8.* Columbia Uviversity.

Chandioux J. (1978) : *METEO: un système opérationnel pour la traduction automatique des bulletins météorologiques destinés au grand public.* Groupe TAUM, Université de Montréal, 1976.

Chappuy S. (1983) : *Formalisation de la description des niveaux d'interprétation des langues naturelles. Etude menée en vue de l'analyse et de la génération au moyen de transducteurs.* Thèse de 3ième cycle en informatique, INPG.

Chauché J., Guillaume P., Quézel-Ambrunaz M. (1972) : *Le système A.T.E.F. (Analyseur de Textes en Etats Finis).* Document CETA, N° G2600A.

Chauché J. (1975) : *Présentation du système CETA.* Document CETA N° G3100A.

Chen P. P. (1976) : *The Entity-Relationship Model -- Towards a Unified View of Data.* ACM Transactions on Database Systems, 1 : 1, 9 -- 36.

Chevalier & al (1978) :*TAUM-METEO, description du système,* janvier 1978, Groupe TAUM, Université de Montréal.

Chomsky N. (1957) : *Syntactic Structures.* Mouton, La Hague.

Chomsky N. (1975) : *Reflections on Language.* Panthear Books, Edition française : *Réflexions sur le langage,* Flammarion.

CNET (1986) : *Atelier de logiciels ; Présentation technique.* CNET, Lannion, France.

Colmerauer A. (1970) : *Les systèmes-Q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur.* Publication interne N° 43, Université de Montréal.

Cox B.(1984) : *Message/Object : an Evolutionary Change.* IEEE Software Engineering,pp50-61.

CRISS (1985) : *Prolog-Criss, une extension du langage PROLOG.* CRISS, Univ. II de Grenoble, Version 4.0.

Donzeau-Gouge V., Huet G., Kahn G., Lang B., Levy J.J. (1979) : *A Structure Oriented Program Editor : a First Step Towards Computer Assisted Programming.* International Computing Symposium, North Holland Publ. Co.

Donzeau-Gouge V., Huet G., Kahn G., Lang B. (1980) : *Programming Environments Based on Structured Editors: the MENTOR Experience.* Rapport de Recherche N° 20, INRIA.

Durand J.C. (1987) : *TTEDIT : Tree Transformational EDITor.* Document GETA.

Evan R. (1985) : *ProGram : a Developement Tool for GPSG Grammars.* Linguistics 23 : 2, 213-243.

Gaddas K. (1986) : *WinTool.* WinSoft, Inc.

Gazdar G., Pullum G.K. (1982) : *Generalized Phrase Structure Grammars : a theoretical synopsis.* Indiana University Linguistics Club, Bloomington, Indiana.

Gazdar G. (1985) : *Computational Tools for Doing Linguistics : Introduction.* Linguistics 1985, 23 : 2, 185-187.

GEC (General Electronic Company) (1986) : *Software Engineering Handbook.* McGRAW-HILL Book Company.

GETA-CEE (1980) : *Etude de la validité de formalisme choisi pour présenter la structure linguistique d'interface.* GETA, Contrat CEE.

GETA-DSE1 (1982) : *Le point sur ARIANE-78 début 1982. Partie I: le logiciel.* Convention ADI/CAP-SOGETI-LOGICIEL N° 81/423.

GETA-DSE2 (1982) : *Spécifications du système ARIANE-X.* Convention ADI/CAP-SOGETI-LOGICIEL N° 81/423.

GETA (1983) : *Maquette pédagogique BEX-FEX: grammaires statiques de l'anglais (morphologique et structurale).* Contrat ADI N° 83/333, GETA.

Grosz et al (1984) : *The Formalism and Implementation of PATR-II.* COLING-84.

Haugeneder H., Gehrke M. (1986) : *A User friendly ATN Programming Environment (APE).* COLING-86, 399-401.

Hertzfeld A. (1986) : *MacIntosh Switcher.* Apple France (Filiale d'Apple Computer Inc).

Isabelle P. (1980) : *Description linguistique du système de traduction informatisée TAUM-AVIATION.* Groupe TAUM, Université de Montréal.

Isabelle P. (1984) : *TAUM-METEO and TAUM-AVIATION.* LUGANO-84.

Karttunen L. (1986) : *D-PART : A development Environment for Unification-Based Grammars.* COLING-86, 74-80.

Kay M. (1979) : *Functional Grammar.* Xerox Palo Alto Research Center.

Kay M. (1982) : *Unification Grammar.* Xerox Palo Alto Research Center.

Kay M. (1984) : *Functional Unification Grammar: a formalism for Machine Translation.* COLING-84.

Knuth D. (1968) : *Semantics of Context-free Languages.* Math Systems Theory, Vol. 22, 127-147.

Krakowiak S. (1986) : *Formation au génie logiciel.* Rapport USMG-ADI.

Lamsweerde A. v., Buyse M., Delcourt B. et al. (1986) : *The Kernel of a Generic Software Development Environment.* ACM proceedings Software engineering.

Larousse (1980) : *Dictionnaire en six volumes.* Librairie LAROUSSE, Paris.

Liu Y.H. et al. (1983) : *Grammaire pragmatique du chinois moderne.* Waiyu Jiaoxue Yu Yuyan Chubansi.

Mélèse B. (1982) : *METAL, un langage de spécification pour le système MENTOR.* TSI 1 : 4.

Mélèse B. (1983) : *Mentor Rapport. Manipulation de textes structurés sous Mentor.* Rapport Technique N° 23, INRIA.

Minkley (1986) : *More Version 1.0.* Living Videotext, Inc.

Mossière J. et al (1982) : *Représentation interne et manipulation des programmes dans l'atelier de logiciel Adèle.* Rapport de Recherche N° 299, IMAG.

Nagao M. et al. (1985) : *The Japanese Government Project for Machine Translation.* Computational linguistics, 11 : 2-3,.

Ouelette F. (1980) : *JEUDEMO.* Bulletin N° 96-01a, Centre de Calcul, Université de Montréal.

Pressman Roger S. (1982) : *Software Engineering : a practitioner's approach.* McGRAW-HILL International Book Company.

Quint, V. (1987) : *Une approche de l'édition structurée des documents.* Thèse d'Etat, USTMG, Grenoble.

Robson, Daniel (1981) : *Object-oriented Software Systems.* Byte, 6 : 8, 74-90

Ross, D.T., Schoman K.E. (1977) : *Structured Analysis for Requirement Specification.* IEEE Transactions Software Engineering, Jan. 6-15.

Schank, R. (1975) : *Conceptual Informations Processing.* North-Holland/American Elsevier.

Slocum J. (1984a) : *METAL : the LRC Machine Translation System.* LUGANO-84.

Slocum J. (1984b) : *Machine Translation : its History, Current Status and Future Prospects.* COLING-84.

Stay J. F. (1976) : *HIPO and Integrated Program Design.* IBM System Journal. 15 : 2, 143-154.

Teichroew D. and Hershey E. (1977) : *PSL/PSA : A Computer Aided Technique for Structured Documentation and Analysis of Information Systems.* IEEE Transactions Software Engineering, 3 : 1, 41-48.

Vauquois B. (1978) : *Description de la structure intermédiaire.* Communication présentée au colloque de Luxembourg.

Vauquois B. (1979) : *Aspects of Automatic Translation in 1979.* IBM-Japan, Scientific Program.

Vauquois B., Chappuy S. (1985) : *Static Grammars : a formalism for the description of linguistic models.* COLGATE-85.

Wigginton R., Ruder E. et Bruener D. (1983) : *MacWrite.* Apple France (Filiale d'Apple Computer Inc.).

Wilks Y. (1975) : *Preference Semantics.* In Keenan E.L.(ed) "The Formal Semantics of Natural Language" C.U.P., Cambridge, 329-350.

Woods W. (1970) : *Transition Network Grammars for Natural Languages.* CACM, 13 : 10, 591-606.

Yan Y. (1983) : *Traduction automatique et ses applications au chinois.* Rapport de DEA, GETA.

Yan Y. (1986) : *ESCS : Environment for Structural Correspondence Specification.* COLING-86.

Zaharin Y. (1987a) : *String-Tree Correspondence Grammar : a declarative grammar formalism for defining the correspondence between strings of terms and tree structures.* Third Conference of the European Chapter of the Association for Computational Linguisticis, Copenhagen, Danmark.

Zaharin Y. (1987b) : *The Linguistic Approach at GETA : a synopsis.* In Technologos (Special Issue on Natural Language Processing), Paris, France.

Zajac R. (1986) : *Etude des possibilités d'interaction HM dans un processus de TA.* Thèse de Doctorat de l'INPG, Grenoble.

Zdonik S. B. (1986) : *Version Management in an Object-Oriented Database.* IFIP WG2.4 International Workshop on Advanced Programming Environments, Trondheim, Norway, June, 1986.

BIBLIOGRAPHIE
(non citée)

ALPAC (1966) : *Languages and Machines. Computers in Translation and Linguistics.* A report by the Automatic Language Processing Advisory Committee, Publication 1416, National Academy of Science, National Research Council, Washington D.C.

Bachut, Verastegui (1984) : *Software Tools for the Environment of a Computer-aided Translation System.* COLING-84.

Barr A., Feigenbaum E.A. (1982) : *The Handbook of Artificial Intelligence, Volume I.* W. Kaufman Inc.

Beaudoin-Lafon M., Gresse C. (1984) : *Caty : un environnement de programmation pour une construction graphique et interactive de programmes.* TSI, 3 : 4, 261-273.

Beesley K., Hefner D. (1986) : *PeriPhrase: Lingware for Parsing and Structural Transfer.* COLING-86, 390-392.

Bennett W., Slocum J. (1984) : *METAL : The LRC Machine Translation System.* Linguistic Research Center, Austin, Texas, USA.

Bennett W., Slocum J. (1985) : *The LRC Machine Translation System.* Computational Linguistics N° 11/2-3.

Bider D. (1985) : *Investigating Macroscopic Textual Variation Through Multifeature/Multidimensional Analyses.* Linguistics, 23 : 2, 337-360.

Boitet C. (1976) : *Un essai de réponse à quelques questions théoriques et pratiques liées à la traduction automatique. Définition d'un système prototype.* Thèse d'Etat, Grenoble.

Boitet C. (1979) : *Automatic Production of CF and CS Analysers.* II° Internationales Kolloquium über Maschinelle Ubersetzung, Lexikographie und Analyse, SFB 100, Universität des Saarlandes, Saarbrücken.

Boitet C., Nédobejkine N. (1981) : *Recent Developments in Russian-French Machine Translation at Grenoble.* Linguistics N° 19, 199-271.

Boitet C., Guillaume P., Quézel-Ambrunaz M. (1982) : *ARIANE-78 : an Integrated Environment for Automated Translation and Human Revision.* COLING-82.

Boitet C., Nedobejkine N. (1983) : *Illustration sur le développement d'un atelier de traduction automatisée .* Colloque "L'informatique au service de la linguistique", Université de Metz.

Boitet C. (1984) : *Research and Development on MT and Related Techniques at Grenoble University (GETA).* LUGANO-84.

Boitet C., Guillaume P., Quézel-Ambrunaz M. (1985) : *A Case Study in Software Evolution : from ARIANE 78.4 to ARIANE 85.* COLGATE-85.

Boitet C., Nedobejkine N. (1985) : *Vers une base lexicale intégrée pour la T(A)O : motivations et organisation linguistique .* 8ièmes Journées francophones sur l'informatique : Bases de Données et Bases de Connaissances, Grenoble.

Boitet C. (1985) : *Traduction (assistée) par ordinateur: ingénierie logicielle et linguicielle.* Colloque AFCET sur la reconnaissance des formes et IA, Grenoble.

Boitet C. (1987) : *Software and Lingware Engineering in Modern M(A)T Systems.* A paraître dans "Handbook of Mechanical Translation" ed NIEMEYER.

Colmerauer A. (1978) : *Metamorphosis Grammars in : "Natural Language Communication with Computers.* Computer Science N° 63, 133-189, Springer-verlag.

Colmerauer A. (1978) : *Metamorphosis Grammars*, in *Natural Language Communication with Computers.* Lecture Notes in : Computer Science N° 63, 133-189, Springer-Verlag.

Colmerauer A. (1982) : *PROLOG II Manuel de référence et modèle théorique.* Groupe IA, Univ. Aix-Marseille.

Could J.C., Clayton Lewis (1985) : *Designing for Usability: Key Principles and What Designers Think.* CACM, 28: 3, 300-311.

Cutter Mark (1985) : *MacDraw.* Apple France (Filiale d'Apple Computer Inc).

Domenig M., Shann P. (1986) : *Towards a Dedicated Database Management System for Dictionaries.* COLING-86, 91-96.

Ducrot J. M. (1973) : *Le système TITUS II.* Institut Textile de France.

Feng Z. (1982) : *Mémoire pour une tentative de traduction multilingue du chinois en français, anglais, japonais, russe et allemand.* COLING-82.

Gerber R. (1984) : *Etude des possibilités de coopération entre un système fondé sur des techniques de compréhension implicite (systèmes logico-syntaxiques) et un système fondé sur des techniques de compréhension explicite (système expert).* Thèse de 3ième cycle en informatique, INPG.

Grishman R., Nhan N.T., Marsh E., & al (1984) : *Automated Determination of Sublanguage Syntactic Range.* COLING-84.

Johnson M. (1985) : *Computer Aids for Comparative Dictionaries.* Linguistics, 23 : 2, 285-302.

Johnson R., Whitelock P. (1985) : *Machine Translation as an Expert Task.* COLGATE-85.

Kay M. (1982) : *Machine Translation.* AJCL N° 8/2, 74-78.

Kittredge R. (1978) : *Textual Cohesion within Sublanguages : implications for automatic analysis and synthesis.* COLING-78.

Kittredge R. (1983) : *Sublanguage Specific Computer Aids to Translation : a survey of the most promising application areas*, Contract N° 2-52Z3, Translation Bureau, Secretary of State Dept., Canada.

Kittredge R. (1985) : *The Significance of Sublanguage for Automatic Translation.* COLGATE-85.

Lepage Y. (1985) : *Ambiguïtés et traduction automatique, les méthodes du GETA.* Brises.

Lytinen S., Schank R. (1982) : *Representation and Translation.* Research Report N ° 234, Yale Univ., Dep. of Computer Science.

Maas H.-D. (1984) : *The MT System SUSY.* LUGANO-84.

Meyrowitz N., Van Dam A. (1982) : *Interactive Editing Systems.* ACM Computing Surveys, 14 : 3.

Melby A.K. (1980) : *ITS : Interactive Translation System.* COLING-80, Tokyo.

Melby A.K.(1982) : *Multi-level Translation Aids in a Distributed System.* COLING-82.

Nakamura J., Tsuji J., Nagao M. (1984) : *Grammar Writing System (GRADE) of Mu-Machine Translation Project and its characteristics.* COLING-84.

N.Nedobejkine (1984) : *Analyse d'un corpus candidat à la traduction automatisée.* GETA, Univ. de Grenoble.

Pansiot J. (1983) : *Les éditeurs dirigés par la syntaxe.* Séminaire d'Informatique CCES, Univ. Louis Pasteur, Strasbourg.

Pereira F.C.N., Warren D.H.D. (1980) : *Definite Clause Grammars for language analysis. A survey of the formalism and a comparison with Augmented Transition Networks..* AI N°13, 231-278.

Pereira F., Shieber S. (1984) : *The Semantics of Grammar Formalisms seen as Computer Language.* COLING-84.

Philipps J.D., Thompson H.S. (1985) : *GPSGP - A Parser for Generalized Phrase Structure Grammars.* Linguistics, 23 : 2, 245-262.

Quézel-Ambrunaz M. (1978) : *ARIANE-78, Système interactif pour la traduction automatique multilingue.* GETA.

Schank R.C. (1973) : *Identification of Conceptualisations Underlying Natural Language.* In : [Schank, Colby 73].

Schank R.C. (1980) : *Language and Memory.* Cognitive Science N° 4, 243-284.

Schank R.C. (1982) : *Remind and Memory Organization : an introduction to MOPs.* In : [Lehnert & Rigle 82].

Schank R.C., Colby K.M. (1973) : *Computer Models of Thought and Language.* W.H. Freeman and Company.

Schank R.C., Lebovitz M., Birbaum L. (1980) : *An Integrated Understander.* AJCL 6 : 1.

Shieber, S. (1984) : *The Design of a Computer Language for Linguistic Information.* COLING-84.

Shieber, S. (1985) : *Criteria for Designing Computer Facilities for Linguistic Analysis.* Linguistics, 23 : 2, 181-211.

Teitelbaum T. et al (1981a) : *The Why and Wherefore of the Cornell Program Synthesizer.* ACM SIGPLAN NOTICES, Vol. 16.

Teitelbaum T. et al (1981b) : *The Cornell Program Synthesizer : a syntax directed programming environment.* CACM, 24 : 9.

TITUS (1984) : *TITUS.* Tutorial on Machine Translation, Lugano.

Tong L. (1986) : *English-Malay Translation System : A Laboratory Prototype.* COLING-86, 639-642.

Vauquois B. (1975) : *La traduction automatique à Grenoble.* Document de Linguistique Quantitative, N° 29, Dunod, Paris.

Vauquois B. (1977) : *L'évolution des logiciels et des modèles linguistiques pour la traduction automatisée.* GETA.

Vauquois B. (1983) : *Automatic Computer-aided Translation and the Arabic Languages.* Proceedings of the Arab School on Science and Technology, Rabat.

Vauquois B. (1984) : *The Organisation of an Automated Translation System for Multilingual Translations at GETA.* IBM Europe Institute, Natural Language Processing, Davos, Suisse.

Vauquois B., Boitet C. (1985) : *Automated Translation at Grenoble University.* Computational Linguistics 11 : 1, 28-36.

Verastegui C.J.N. (1982) : *Etude du parallélisme appliqué à la traduction automatisée. STAR-PALE : un système parallèle.* Thèse de Docteur-Ingénieur, INPG.

Vercoustre A-M. (1983) : *Minerve, un méta-éditeur syntaxique.* Rapport de Recherche N° 229, INRIA.

Wang L. (1983) : *Hanyu yufa gangyao.* Edition de Shanghaï Jiaoyu Chubanshe.

Wilks Y. (1973) : *An Artificial Intelligence Approach to Machine Translation.* In : [Schank & Colby 73].

Wilks Y. (1975) : *A Preferential, Pattern-seeking, Semantics for Natural Language Inference.* AI 6 : 1, 53-74.

Wilks Y. (1979) : *Machine Translation and Artificial Intelligence.* In : [Snell 79].

Winograd T. (1973) : *A Procedural Model of Language Understanding.* In : [Schank/Colby 73].

Winograd T. (1980) : *What Does It Mean to Understand Language ?* Cognitive Science N°4, 209-241.

Winograd T. (1983) : *Language as a Cognitive Process.Volume I : syntax.* Addison Wesley.

Yang P. (1982) : *Un essai sur la génération du chinois.* T.A. Informations N ° 1, 40-68.

Zaharin Y. (1986) : *Strategies and Heuristics in the Analysis of Natural Language in MT.* Ph.D. Thesis, University Sains Malaysia, Penang.

Zajac R. (1986) : *SCSL : a Linguistic Specification Language for MT.* COLING-86.

GLOSSAIRE

AM	Analyse Morphologique
AS	Analyse Structurale
BNF	Backus Normal Form
BSCS	Base de Spécification de Correspondances Structurales
ESCS	Environnement pour la Spécification de Correspondances Structurales
DOSS	Dossier : sous-répertoire en MacIntosh
GETA	Groupe d'Etudes pour la Traduction Automatique
GD	Grammaire Dynamique : grammaires programmées
GSCS	Grammaire Statique de Correspondances Structurales
GM	Génération Morphologique
GS	Génération Structurale
Kermit	Logiciel de communication entre ordinateurs
LSPL	Langage Spécialisé pour la Programmation Linguistique
LSCS	Langage de Spécification de Correspondances Structurales
MacIntosh	Micro-ordinateur d'Apple Inc.
MD	MacDraw : logiciel de dessin sur MacIntosh
More	Logiciel d'édition structurale
MW	MacWrite, traitement de textes sur MacIntosh
PN-TAO	Projet National de TAO
TAO	Traduction Assistée par Ordinateur
TL	Transfert Lexical
TS	Transfert Structural
WT	WinTool, logiciel de gestion de dictionnaires (produit de WinSoft)

INDEX

Thèse de Doctorat en informatique

AUTEUR : Yongfeng YAN

TITRE : VERS UNE INGENIERIE DE LA PRODUCTION DE LINGUICIELS

DATE : 29 juin 1987

MOTS-CLES : Traduction automatique, Linguistique informatique, Génie logiciel, Linguiciel, Langage de spécification, Edition structurée, Base de données linguistiques.

RESUME : Après une étude de l'historique et des techniques dominantes en TAO, nous avons abordé deux axes principaux de recherche : le transfert de l'état de l'art vers l'industrie, et le perfectionnement de la technologie au moyen de techniques issues de l'intelligence artificielle.

Cette thèse est essentiellement consacrée au premier axe, l'ingénierie du linguiciel. Partant des idées fondamentales du génie logiciel, nous avons étudié le processus de construction de systèmes de TAO et en avons déduit une méthodologie de développement. Cette méthodologie a été illustrée par une première version d'un poste de travail linguistique, et mise en pratique dans plusieurs développements. Malgré la simplicité du poste de travail, on peut déjà améliorer de façon significative la productivité des linguistes et la qualité des linguiciels produits. En guise de conclusion, nous proposons une architecture pour un atelier linguiciel idéal du futur, intégrant une base de données linguistiques et un éditeur structural.

AUTORISATION DE SOUTENANCE

DOCTORAT 3ème CYCLE, DOCTORAT-INGENIEUR, DOCTORAT USTMG

Vu les dispositions de l'Arrêté du 16 avril 1974,

Vu les dispositions de l'Arrêté du 5 juillet 1984,

Vu les rapports de M. Jacques COURTIN

M. Patrice POGNAN

M. Jacques CHAUCHE

M. YAN Yongfeng est autorisé

à présenter une thèse en vue de l'obtention du Doctorat de l'USTMG

(spécialité : INFORMATIQUE)

Grenoble, le. 24 JUIN 1987

Le Président de l'Université Scientifique
Technologique et Médicale

UNIVERSITÉ SCIENTIFIQUE ET MÉDICALE DE GRENOBLE

J.J PAYAN