Numéro d'ordre : 722

ANNÉE : 1991

USTL FLANDRES ARTOIS — LIFL — U.A. 369 du C.N.R.S.

LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE



# THÈSE

Présentée à

**L'UNIVERSITÉ DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS**

pour obtenir le grade de

**DOCTEUR en INFORMATIQUE**

selon l'arrêté ministériel du 23 novembre 1988

par

**PHAN Huy Khánh**



# CONTRIBUTION À L'INFORMATIQUE MULTILINGUE. EXTENSION D'UN ÉDITEUR DE DOCUMENTS STRUCTURÉS

## Đóng góp cho ứng dụng tin học trên nhiều ngôn ngữ. Mở rộng một hệ thống xử lý văn bản có tính cấu trúc

## 对多种语言计算科学的贡献. 结构文件编辑程序的扩展

**Thèse soutenue le 21 mai 1991 devant la commission d'examen**

| | |
|---|---|
| J. P. DELAHAYE | Président |
| M. NANARD | Rapporteur |
| V. QUINT | Rapporteur |
| F. TCHÉOU | Rapporteur |
| Ch. BOITET | Directeur |
| A. COUSQUER | Examinateur |
| J. M. GEIB | Examinateur |

UNIVERSITE DES SCIENCES
ET TECHNIQUES DE LILLE
FLANDRES ARTOIS

## DOYENS HONORAIRES DE L'ANCIENNE FACULTE DES SCIENCES

M.H. LEFEBVRE, M. PARREAU.

## PROFESSEURS HONORAIRES DES ANCIENNES FACULTES DE DROIT ET SCIENCES ECONOMIQUES, DES SCIENCES ET DES LETTRES

MM. ARNOULT, BONTE, BROCHARD, CHAPPELON, CHAUDRON, CORDONNIER, DECUYPER, DEHEUVELS, DEHORS, DION, FAUVEL, FLEURY, GERMAIN, GLACET, GONTIER, KOURGANOFF, LAMOTTE, LASSERRE, LELONG, LHOMME, LIEBAERT, MARTINOT-LAGARDE, MAZET, MICHEL, PEREZ, ROIG, ROSEAU, ROUELLE, SCHILTZ, SAVARD, ZAMANSKI, Mes BEAUJEU, LELONG.

## PROFESSEUR EMERITE

M. A. LEBRUN

## ANCIENS PRESIDENTS DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

MM. M. PAREAU, J. LOMBARD, M. MIGEON, J. CORTOIS.

## PRESIDENT DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

M. A. DUBRULLE.

## PROFESSEURS - CLASSE EXCEPTIONNELLE

| | |
|---|---|
| M. CONSTANT Eugène | Electronique |
| M. FOURET René | Physique du solide |
| M. GABILLARD Robert | Electronique |
| M. MONTREUIL Jean | Biochimie |
| M. PARREAU Michel | Analyse |
| M. TRIDOT Gabriel | Chimie Appliquée |

## PROFESSEURS - 1ère CLASSE

| | |
|---|---|
| M. BACCHUS Pierre | Astronomie |
| M. BIAYS Pierre | Géographie |
| M. BILLARD Jean | Physique du Solide |
| M. BOILLY Bénoni | Biologie |
| M. BONNELLE Jean-Pierre | Chimie-Physique |
| M. BOSCQ Denis | Probabilitéss |
| M. BOUGHON Pierre | Algèbre |
| M. BOURIQUET Robert | Biologie Végétale |
| M. BREZINSKI Claude | Analyse Numérique |

| | |
|---|---|
| M. BRIDOUX Michel | Chimie-Physique |
| M. CELET Paul | Géologie Générale |
| M. CHAMLEY Hervé | Géotechnique |
| M. COEURE Gérard | Analyse |
| M. CORDONNIER Vincent | Informatique |
| M. DAUCHET Max | Informatique |
| M. DEBOURSE Jean-Pierre | Gestion des Entreprises |
| M. DHAINAUT André | Biologie Animale |
| M. DOUKHAN Jean-Claude | Physique du Solide |
| M. DYMENT Arthur | Mécanique |
| M. ESCAIG Bertrand | Physique du Solide |
| M. FAURE Robert | Mécanique |
| M. FOCT Jacques | Métallurgie |
| M. FRONTIER Serge | Ecologie Numérique |
| M. GRANELLE Jean-Jacques | Sciences Economiques |
| M. GRUSON Laurent | Algèbre |
| M. GUILLAUME Jean | Microbiologie |
| M. HECTOR Joseph | Géométrie |
| M. LABLACHE-COMBIER Alain | Chimie Organique |
| M. LACOSTE Louis | Biologie Végétale |
| M. LAVEINE Jean-Pierre | Paléontologie |
| M. LEHMANN Daniel | Géométrie |
| Mme LENOBLE Jacqueline | Physique Atomique et Moléculaire |
| M. LEROY Jean-Marie | Spectrochimie |
| M. LHOMME Jean | Chimie Organique Biologique |
| M. LOMBARD Jacques | Sociologie |
| M. LOUCHEUX Claude | Chimie Physique |
| M. LUCQUIN Michel | Chimie Physique |
| M. MACKE Bruno | Physique Moléculaire et Rayonnements Atmosph. |
| M. MIGEON Michel | E.U.D.I.L. |
| M. PAQUET Jacques | Géologie Générale |
| M. PETIT Francis | Chimie Organique |
| M. POUZET Pierre | Modélisation - calcul Scientifique |
| M. PROUVOST Jean | Minéralogie |
| M. RACZY Ladislas | Electronique |
| M. SALMER Georges | Electronique |
| M. SCHAMPS Joel | Spectroscopie Moléculaire |
| M. SEGUIER Guy | Electrotechnique |
| M. SIMON Michel | Sociologie |
| Melle SPIK Geneviève | Biochimie |
| M. STANKIEWICZ François | Sciences Economiques |
| M. TILLIEU Jacques | Physique Théorique |
| M. TOULOTTE Jean-Marc | Automatique |
| M. VIDAL Pierre | Automatique |
| M. ZEYTOUNIAN Radyadour | Mécanique |

## PROFESSEURS - 2ème CLASSE

| | |
|---|---|
| M. ALLAMANDO Etienne | Composants Electroniques |
| M. ANDRIES Jean-Claude | Biologie des organismes |
| M. ANTOINE Philippe | Analyse |
| M. BART André | Biologie animale |
| M. BASSERY Louis | Génie des Procédés et Réactions Chimiques |

| | |
|---|---|
| Mme BATTIAU Yvonne | Géographie |
| M. BEGUIN Paul | Mécanique |
| M. BELLET Jean | Physique Atomique et Moléculaire |
| M. BERTRAND Hugues | Sciences Economiques et Sociales |
| M. BERZIN Robert | Analyse |
| M.BKOUCHE Rudolphe | Algèbre |
| M. BODARD Marcel | Biologie Végétale |
| M. BOIS Pierre | Mécanique |
| M. BOISSIER Daniel | Génie Civil |
| M. BOIVIN Jean-Claude | Spectroscopie |
| M. BOUQUELET Stéphane | Biologie Appliquée aux enzymes |
| M. BOUQUIN Henri | Gestion |
| M. BRASSELET Jean-Paul | Géométrie et Topologie |
| M. BRUYELLE Pierre | Géographie |
| M. CAPURON Alfred | Biologie Animale |
| M. CATTEAU Jean-Pierre | Chimie Organique |
| M. CAYATTE Jean-Louis | Sciences Economiques |
| M. CHAPOTON Alain | Electronique |
| M. CHARET Pierre | Biochimie Structurale |
| M. CHIVE Maurice | Composants Electroniques Optiques |
| M. COMYN Gérard | Informatique Théorique |
| M. COQUERY Jean-Marie | Psychophysiologie |
| M. CORIAT Benjamin | Sciences Economiques et Sociales |
| Mme CORSIN Paule | Paléontologie |
| M. CORTOIS Jean | Physique Nucléaire et Corpusculaire |
| M. COUTURIER Daniel | Chimie Organique |
| M. CRAMPON Norbert | Tectolique Géodynamique |
| M. CROSNIER Yves | Electronique |
| M. CURGY Jean-Jacques | Biologie |
| Melle DACHARRY Monique | Géographie |
| M. DEBRABANT Pierre | Géologie Appliquée |
| M. DEGAUQUE Pierre | Electronique |
| M. DEJAEGER Roger | Electrochimie et Cinétique |
| M. DELAHAYE Jean-Paul | Informatique |
| M. DELORME Pierre | Physioliogie Animale |
| M. DELORME Robert | Sciences Economiques |
| M. DEMUNTER Paul | Sociologie |
| M. DENEL Jacques | Informatique |
| M. DE PARIS Jean Claude | Analyse |
| M. DEPREZ Gilbert | Physique du Solide - Cristallographie |
| M. DERIEUX Jean-Claude | Microbiologie |
| Melle DESSAUX Odile | Spectroscopie de la réactivité Chimique |
| M. DEVRAINNE Pierre | Chimie Minérale |
| Mme DHAINAUT Nicole | Biologie Animale |
| M. DHAMELINCOURT Paul | Chimie Physique |
| M. DORMARD Serge | Sciences Economiques |
| M. DUBOIS Henri | Spectroscopie Hertzienne |
| M. DUBRULLE Alain | Spectroscopie Hertzienne |
| M. DUBUS Jean-Paul | Spectrométrie des Solides |
| M. DUPONT Christophe | Vie de la firme (I.A.E.) |
| Mme EVRARD Micheline | Génie des procédés et réactions chimiques |
| M. FAKIR Sabah | Algèbre |
| M. FAUQUAMBERGUE Renaud | Composants électroniques |

| | |
|---|---|
| M. FONTAINE Hubert | Dynamique des cristaux |
| M. FOUQUART Yves | Optique atmosphérique |
| M. FOURNET Bernard | Biochimie Sturcturale |
| M. GAMBLIN André | Géographie urbaine, industrielle et démog. |
| M. GLORIEUX Pierre | Physique moléculaire et rayonnements Atmos. |
| M. GOBLOT Rémi | Algèbre |
| M. GOSSELIN Gabriel | Sociologie |
| M. GOUDMAND Pierre | Chimie Physique |
| M. GOURIEROUX Christian | Probabilités et Statistiques |
| M. GREGORY Pierre | I.A.E. |
| M. GREMY Jean-Paul | Sociologie |
| M. GREVET Patrice | Sciences Economiques |
| M. GRIMBLOT Jean | Chimie Organique |
| M. GUILBAULT Pierre | Physiologie animale |
| M. HENRY Jean-Pierre | Génie Mécanique |
| M. HERMAN Maurice | Physique spatiale |
| M. HOUDART René | Physique atomique |
| M. JACOB Gérard | Informatique |
| M. JACOB Pierre | Probabilités et Statistiques |
| M. Jean Raymond | Biologie des populations végétales |
| M. JOFFRE Patrick | Vie de la firme (I.A.E.) |
| M. JOURNEL Gérard | Spectroscopie hertzienne |
| M. KREMBEL Jean | Biochimie |
| M. LANGRAND Claude | Probabilités et statistiques |
| M. LATTEUX Michel | Informatique |
| Mme LECLERCQ Ginette | Catalyse |
| M. LEFEBVRE Jacques | Physique |
| M. LEFEBVRE Christian | Pétrologie |
| Melle LEGRAND Denise | Algèbre |
| Melle LEGRAND Solange | Algèbre |
| M. LEGRAND Pierre | Chimie |
| Mme LEHMANN Josiane | Analyse |
| M. LEMAIRE Jean | Spectroscopie hertzienne |
| M. LE MAROIS Henri | Vie de la firme (I.A.E.) |
| M. LEROY Yves | Composants électroniques |
| M. LESENNE Jacques | Systèmes électroniques |
| M. LHENAFF René | Géographie |
| M. LOCQUENEUX Robert | Physique théorique |
| M. LOSFELD Joseph | Informatique |
| M. LOUAGE Francis | Electronique |
| M. MAHIEU Jean-Marie | Optique-Physique atomique |
| M. MAIZIERES Christian | Automatique |
| M. MAURISSON Patrick | Sciences Economiques et Sociales |
| M. MESMACQUE Gérard | Génie Mécanique |
| M. MESSELYN Jean | Physique atomique et moléculaire |
| M. MONTEL Marc | Physique du solide |
| M. MORCELLET Michel | Chimie Organique |
| M. MORTREUX André | Chimie Organique |
| Mme MOUNIER Yvonne | Physiologie des structures contractiles |
| Mme MOUYART-TASSIN Annie Françoise | Informatique |
| M. NICOLE Jacques | Spectrochimie |
| M. NOTELET Françis | Systèmes électroniques |
| M. PARSY Fernand | Mécanique |

| | |
|---|---|
| M. PECQUE Marcel | Chimie organique |
| M. PERROT Pierre | Chimie appliquée |
| M. STEEN Jean-Pierre | Informatique |

# *Remerciements*

*Depuis mon arrivée en France en 1987 dans le Groupe d'Étude pour la Traduction Automatique (GETA), Monsieur Christian BOITET, professeur à l'Université Joseph Fourier (UJF-Grenoble 1) n'a cessé de suivre et de diriger mes études dans le domaine des applications informatiques multilingues. Je lui dois de nombreux conseils précis et des critiques détaillées des rapports de travail sans lesquels je n'aurais pas pu mener à bien cette thèse.*

*C'est au sein du Groupe d'Étude des DIscours Scientifiques (GEDIS) du Laboratoire d'Informatique Fondamentale de Lille (LIFL) et du GETA que j'ai poursuivi les recherches qui font l'objet de cette thèse.*

*J'exprime ma profonde reconnaissance à Monsieur Alain COUSQUER, assistant à l'Université des Sciences et Techniques de Lille Flandres Artois (Lille 1) qui m'a accueilli dans le laboratoire GEDIS. Ce n'est que grâce aux outils et techniques d'informatisation du chinois qu'il a rassemblés ou créés que j'ai pu mener à bien ce travail.*

*Monsieur Vincent QUINT, Directeur de recherche à l'Institut National de la Recherche en Informatique et en Automatique (INRIA) et auteur principal de Grif, a bien voulu me fournir le sujet de cette thèse et m'a constamment aidé dans le travail que j'ai fait sur Grif. Grâce à lui, j'ai acquis des connaissances indispensables pour réaliser une extension multilingue de ce logiciel. Je lui en suis très reconnaissant.*

*Je tiens à remercier Monsieur François TCHÉOU, Maître de conférences à l'UJF, très grand connaisseur de la langue et de la calligraphie chinoise, pour son aide efficace et ses encouragements.*

*Je tiens aussi à exprimer ma profonde gratitude et ma reconnaissance à tous ceux et celles qui m'ont aidé et soutenu au quotidien, et en particulier*

*Madame Éliane COUSQUER et à Monsieur ZHANG Hua du laboratoire GEDIS, avec qui j'ai eu des discussions fructueuses sur la linguistique et la langue chinoises,*

*Madame Irène VATTON, réalisatrice de nombreux éléments de Grif, et particulier du Médiateur, qui m'a prodigué de nombreux conseils et une aide constante,*

*Messieurs Daniel BACHUT et René GERBER de Bernard Vauquois Informatique et Traitement Automatisé des Langues (B'VITAL) de Grenoble, Nicolas NEDOBEJKINE et Pierre GUILLAUME du GETA, qui m'ont éclairé sur les problèmes informatiques et linguistiques liés aux transcriptions,*

*Monsieur NGUYÊN Phú Phong, Docteur en Linguistique à l'Université de Paris 7, dont l'aide sur la langue vietnamienne m'a été précieuse,*

*Messieurs Bernard SZÉLAG, Gilles CARIN, Fred HEMERY du LIFL, qui m'ont aidé en ce qui concerne le matériel informatique.*

Imprimé par Grif le 20/3/91

*Je suis très heureux de remercier Monsieur Jean-Paul DELAHAYE, professeur à l'Université de Lille 1, qui me fait l'honneur de présider le jury de thèse et qui a manifesté son intérêt pour mon travail.*

*Je tiens à remercier Monsieur Jean-Marc GEIB, Maître de conférences à l'Université de Lille 1, spécialiste de la programmation par objets, pour sa présence en tant que membre du jury.*

*Que Monsieur Marc NANARD, professeur au Conservatoire National des Arts et Métiers (CNAM) et membre du Centre de Recherche en Informatique de Montpellier, qui a bien voulu rapporter sur ce travail, trouve ici tous mes remerciements.*

*Je tiens aussi à remercier et à affirmer ma grande gratitude*

*à Monsieur le professeur Jean-Charles PARIAUD, ex-Directeur de l'École Nationale Supérieure d'Électrochimie et d'Électrométalurgie de Grenoble (ENSEEG), qui m'a donné la chance de faire des études en France,*

*à Messieurs François ROBERT et Raymond BOUTTAZ, professeur et ingénieur à l'École Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble (ENSIMAG), pour leur aide désintéressée et leurs encouragements cordiaux au Viêtnam et depuis,*

*à Monsieur Christian VANHAECKE, responsable du bureau d'accueil des étudiants étrangers, et aux membres du service de CROUS de Lille qui m'ont donné les conditions financières et matérielles nécessaires, ainsi que pour leur sympathie,*

*à Monsieur le Professeur François PECCOUD, directeur du GETA, ainsi qu'à Nadine BACHUT, à Kiki LEVENBACH, à Christine MAIDA et à tous les membres du GETA, pour leur aide et le très bon accueil que j'ai constamment trouvé auprès d'eux.*

*Mes remerciements vont enfin à NGUYỄN văn Hiệp, à HOÀNG Ngọc Minh , à tous mes amis à Grenoble et à ceux et celles qui, par leurs conseils, leur aide, leurs encouragements et leur sympathie, ont contribué à faire aboutir mon travail en France.*

Imprimé par Grif le 20/3/91

*À ma femme et mon fils*

# TABLE DES MATIÈRES



Imprimé par Grif le 20/3/91

Imprimé par Grif le 20/3/91

# Introduction

## 1 De la localisation au véritable multilinguisme

Dans les premiers temps de l'informatique, la plupart des ordinateurs ne jouaient que le rôle de calculateurs scientifiques et n'offraient que de la typographie pauvre. Les exigences typographiques étaient limitées à la présentation des programmes et à l'impression des résultats.

Le traitement des documents n'a été pris comme sujet de recherche qu'à partir de 1965, au mieux, donc vingt ans après les débuts. Axe important de recherche, ce domaine en lui-même est devenu une application de l'informatique. On a d'abord rencontré le problème de normalisation des codages. Les systèmes de codage connus (ASCII, EBCDIC, etc.) ne traitaient que des textes en anglais et certaines langues européennes, mais de façon plus ou moins limitée. Sur le plan du matériel, les contraintes des arts graphiques et la faible variété des caractères disponibles en sortie (non latins, par exemple) posaient des difficultés insolubles. En fait, la notion même de document était mal cernée à cause d'une distinction non faite entre la présentation physique (typographie en particulier) et la structure logique d'un document.

Le traitement de différentes langues non latines a conduit à développer des systèmes spécifiques qui sont utilisés pour une langue donnée, comme les systèmes chinois, arabe... dans divers environnements (PC, Mac, grands systèmes). Ces systèmes sont encore peu répandus, pour des raisons de prix, ou existent de façon confidentielle. En général, ils ne sont pas susceptibles de généralisation à d'autres langues, et nécessitent soit l'usage de cartes additionnelles, comme le système HanMa [III.4 Asia 89], soit la modification du système d'exploitation (comme CCDOS, version chinoise de MS-DOS).

Le développement spectaculaire de la bureautique et l'évolution rapide de la technologie de fabrication des matériels informatiques ont permis d'améliorer la situation. Parallèlement à l'utilisation de normes internationalement reconnues pour le codage des alphabets et à l'introduction de modèles de documents (ODA, SGML par exemple) dans les systèmes de traitement de texte, on voit apparaître des outils informatiques adaptés au multilinguisme : présentation cohérente de textes et de nomenclatures multilingues, citation dans différentes langues des références et des notes, etc.

Imprimé par Grif le 20/3/91

Un autre aspect du multilinguisme est lié à l'étude des langues naturelles elles-mêmes, qu'il s'agisse de règles typographiques particulières (sur certains caractères ou groupements de caractères, problèmes de segmentations de mots ou phrases...), des règles d'usage (symboles, marques de ponctuation), ou d'applications spécifiques : correction orthographique, recherche de données textuelles, etc.

On en arrive alors, à vouloir construire des programmes de production et de manipulation de documents utilisables en conjonction avec diverses applications de traitement des langues naturelles. Nous parlerons alors de *Système de Traitement de Texte Multilingue* (STTM).

# 2 Intérêt et difficulté d'un véritable multilinguisme

La conception et la réalisation d'un tel STTM présentent un grand intérêt, non seulement en bureautique, pour la PAO (Publication Assistée par Ordinateur), mais encore dans l'enseignement, pour l'EAO (Enseignement Assisté par Ordinateur) des langues naturelles, ou dans d'autres applications de TALN (Traitement Automatique des Langues Naturelles), comme la lexicographie, l'indexation automatique, la recherche documentaire, etc.

Selon J. D. Becker [III.1 Becker 84], les problèmes du traitement de texte multilingue concernent essentiellement trois aspects : le *codage*, ou représentation interne d'un texte multilingue en mémoire, la *saisie* sur un dispositif d'entrée (clavier, souris), la *restitution* (affichage sur l'écran ou impression sur papier). Nous y ajoutons le problème de l'interaction homme-machine ou *le dialogue*.

Dans un STTM, le multilinguisme est plus que la possibilité d'afficher de multiples jeux de caractères : les particularités nationales propres à chaque système d'écriture doivent pouvoir être prises en compte dans les algorithmes de traitement (sens d'écriture, ou variations morphologiques par exemple).

L'introduction du multilinguisme dans la conception d'un STTM se heurte à deux grandes difficultés. La première est d'obtenir à la fois les quatre propriétés souhaitables suivantes :

1. normalisation du codage de tous les systèmes d'écriture du monde : ce code doit être universel, portable, et adapté aux traitements, tant classiques que spécifiquement linguistiques ;
2. commodité et disponibilité de la saisie : il faut pouvoir utiliser facilement un clavier alphanumérique disponible partout, avec des interfaces de saisie convenables ;

3. satisfaction de contraintes en restitution : sur l'écran ou sur le papier, on doit pouvoir obtenir toute variété de présentation ;
4. convivialité du dialogue : on doit pouvoir interagir avec le système dans la langue de son choix, qui peut ne pas être une des langues du document.

La deuxième difficulté est de concilier deux points de vue à partir des solutions proposées:

1. celui de l'utilisateur : l'augmentation du nombre de langues naturelles traitables et des performances d'un STTM dépend strictement des capacités du matériel disponible (espace mémoire, temps d'exécution, etc.) et des ressources correspondantes ;
   pourtant, il n'est pas facile de produire des polices différentes suivant les besoins divers quand on pense qu'il y a des systèmes d'écriture soit riches de caractères (chinois, japonais), soit complexes avec des signes diacritiques (thaï, vietnamien...), soit contenant des ligatures et/ou variantes morphologiques (arabe, grec, hébreu...), etc., parmi 3 000 langues vivantes du monde présentées dans [II Malherbe 83] ;
2. celui de l'implémenteur : les capacités intrinsèques d'un système multilingue (et la complexité qui en découle, du point de vue du programme lui-même comme de son interface avec l'utilisateur) peuvent être contradictoires avec les objectifs de diffusion ou de commercialisation (coût d'acquisition, facilité d'apprentissage, simplicité d'utilisation, etc.).

À cause de ces deux difficultés, les problèmes du multilinguisme sont toujours ouverts. Il n'existe pas vraiment de STTM à l'heure actuelle, il n'existe que des solutions partielles, qui répondent cependant à des besoins impérieux. On peut citer les diverses extensions multilingues de TEX de D. Knuth, le système de traitement de texte multilingue, appelé Star, intégré sur la 1165, une station de travail de la société Xerox [III.1 Becker 84], WinText pour les Mac de WinSoft [III.5 WinSoft 88], etc.

Les désavantages de ces systèmes sont d'abord dus à l'utilisation d'un codage trop limité et souvent local. Ce codage ne s'adapte qu'à une application particulière et il n'existe aucun moyen de l'améliorer pour d'autres applications. Pour la saisie, certains systèmes permettent de choisir entre différentes méthodes (dans le MacOS chinois par exemple), mais en général, l'utilisateur est obligé d'utiliser des méthodes de saisie fixées à l'avance. Il existe des systèmes où les méthodes d'entrée sont "paramétrables" (comme l'éditeur GNU-emacs dans l'environnement X-Window), mais leur manipulation n'est pas très commode. Pour la sortie, il n'y a pas d'indépendance entre les matériels disponibles et les ressources correspondantes. On ne peut utiliser que les ressources existantes et il est difficile d'en ajouter de nouvelles.

# 3 Résumé de l'approche suivie

Dans le contexte du traitement de texte multilingue, nous sommes parti d'un système de production de documents de haut niveau, Grif [III.2 Quint 83, 86, 87], avec comme objectif d'étudier les solutions possibles pour la conception d'un STTM en général, et de les appliquer concrètement à ce système. Le résultat est une extension de Grif, qui rend ce produit multilingue, d'une façon cohérente avec sa conception globale, fondée sur la généricité.

Grif est un éditeur de documents structurés commercialisé par GIPSI S. A., et issu du prototype Grif développé par Vincent Quint et Irène Vatton à l'INRIA et au LGI (Laboratoire de Génie Informatique de l'IMAG, INPG, UJF & CNRS). Conçu comme un système interactif, Grif est capable de gérer à la fois et indépendamment le découpage logique d'un document et sa présentation typographique. La production d'un document sous Grif présente un haut niveau d'abstraction. De plus, la conception avancée par langages assure la généricité et l'ouverture. Un autre avantage de travailler sur Grif était la disponibilité du code source et la possibilité de collaborer avec les auteurs.

Enfin, Grif associe à un document une "structure abstraite", sur laquelle il est assez facile d'ajouter des attributs concernant les systèmes d'écriture, et de les traiter de façon homogène à tous les autres. Dans la version prototype sur laquelle nous avons travaillé, Grif utilise principalement l'alphabet latin pour la production de ses documents et l'alphabet grec pour la manipulation des symboles mathématiques.

La première étape de notre travail sur le multilinguisme a consisté à étendre Grif au vietnamien. L'implémentation utilise les codes ASCII étendus et la méthode de "composition des touches" pour la saisie des caractères portant des signes diacritiques. Cette approche peut être utilisée pour des langues dont l'écriture utilise un ensemble de moins de 256 signes typographiques ; la limite de ce type de solution est son caractère partiel.

Dans une seconde étape, nous avons donc voulu proposer des solutions beaucoup plus générales. Nous sommes arrivé à une solution plus générique par le développement d'un langage de transcription d'entrée, appelé langage E, analogue aux autres langages de Grif. L'implémentation du langage E nous a permis de construire une version multilingue de Grif, dans laquelle un même document peut contenir à la fois du chinois, des langues utilisant les caractères latins (y compris le vietnamien), et des langues utilisant les alphabets grec et cyrillique.

Notre thèse a été entièrement réalisée avec cette version multilingue de Grif.

Nous avons divisé la présentation de ce travail en trois grandes parties qui reflètent les diverses étapes de notre travail et en tirent les conclusions essentielles :

## Partie A : Problèmes liés au traitement informatique de textes multilingues

### Chapitre I : État de l'art

Ce chapitre donne une vue générale du domaine d'application des outils informatiques pour le traitement de textes multilingues. Après une présentation des outils disponibles, nous discutons les limites des outils actuels.

### Chapitre II : Analyse du problème

Il est naturel de présenter précisément les différents aspects du multilinguisme et les problèmes que cela introduit dans la réalisation d'un STTM. La conception d'un STTM se heurte à des contraintes gênantes et sa réalisation comporte de nombreuses difficultés. Les principaux problèmes présentés et discutés dans cette partie sont ceux du codage, de la saisie, de la restitution et de l'interface utilisateur dans un environnement multilingue.

### Chapitre III : Problèmes spécifiques en édition de documents structurés

L'édition de documents structurés représente un progrès dans les outils informatiques. Nous présentons ce concept, encore peu répandu, et montrons l'intérêt qu'il y a à travailler sur un éditeur de documents structurés comme Grif plutôt que sur un éditeur ou un traitement de texte classique.

## Partie B : Une première approche et son application au vietnamien

### Chapitre IV : Grif : un éditeur de documents structurés

Très largement paramétrable, Grif est considéré comme un outil informatique puissant pour la production de documents scientifiques et techniques. Nous en présentons les caractéristiques les plus marquantes, comme sa souplesse et son ouverture, sur lesquelles repose l'extension.

### Chapitre V : Multilinguisme par "remplissage des vides" : l'exemple du vietnamien

Le vietnamien est ma langue maternelle. Il s'écrit aujourd'hui avec les caractères "latin" et deux niveaux de diacritiques. Son informatisation n'a été entreprise que récemment. Il était donc intéressant de commencer par étendre Grif au vietnamien, en quelque sorte prototype des langues utilisant un système d'écriture alphabétique.

Ce travail nous a conduit à une première méthode d'extension multilingue, dans laquelle on utilise des codes ASCII étendus pour le codage (méthode de "remplissage des vides"), et la méthode de "composition de touches" pour la saisie des caractères ayant un ou deux signes diacritiques.

### Chapitre VI : Évaluation de la méthode

L'exemple du vietnamien montre comment adapter Grif aux systèmes d'écriture en utilisant l'alphabet latin ou grec, ou de façon plus générale, un ensemble de moins de 256 signes typographiques. Cette application n'est qu'une solution partielle, qu'on ne peut étendre aux systèmes d'écriture à (très) grand nombre de signes, comme le chinois, ni même à plusieurs systèmes alphabétiques à la fois. Cependant, elle peut donner rapidement des versions "localisées" pour un grand nombre de langues.

## Partie C : Une approche générique : introduction du langage E dans Grif. Application au chinois

### Chapitre VII : Étude des stratégies possibles

Nous proposons trois stratégies possibles. La première consiste à introduire directement dans le code source de Grif des modules de traitement multilingue. Il s'agit d'utiliser un codage mixte et de mélanger des méthodes d'entrée-sortie convenables sur le codage intégré.

La deuxième stratégie consiste à étendre à la fois les trois langages S, P et T de Grif. Pour faciliter l'étude, l'extension porterait seulement sur l'ajout de nouveaux mots-clés, en conservant la sémantique, la syntaxe, et en respectant la nature de chaque langage.

La troisième stratégie consiste à introduire un nouveau langage analogue aux langages S, P et T, et spécifique au multilinguisme.

Après avoir fait une évaluation concernant les avantages et les inconvénients de chaque stratégie proposée, nous avons choisi la solution par langage et l'avons implémentée.

### Chapitre VIII : Le langage E, une solution générique

Nous définissons un langage de transcription d'entrée, appelé langage E, qui permet d'écrire des schémas de transcription. E est défini et traité comme les autres langages de Grif. Les schémas compilés sont des tables de transcription que Grif utilise pour le traitement multilingue.

La présentation montre les caractéristiques sémantiques, la syntaxe, ainsi que différents aspects de l'implémentation du langage E. Nous présentons aussi dans ce chapitre les problèmes restants, qui concernent la combinaison des langages et certains aspects à compléter ultérieurement.

### Chapitre IX : Application au chinois

Nous avons réalisé une adaptation de Grif à la langue chinoise dans un contexte multilingue. Avant de présenter le traitement des textes chinois, nous présentons quelques notions indispensables pour comprendre les problèmes liés à l'informatisation du chinois.

L'implémentation a été réalisée en utilisant des polices de caractères chinois de la norme GB 2312-80 disponibles actuellement au laboratoire GEDIS (LIFL, Université de Lille 1). Dans cette extension, le langage E est également utilisé pour les caractères cyrilliques (nous nous sommes limité à l'alphabet russe).

## Conclusion

Cette partie résume notre contribution à l'informatique multilingue, présente les problèmes encore ouverts et indique quelques perspectives de recherche et de développement.

Imprimé par Grif le 20/3/91

# Partie A
# Problèmes liés au traitement informatique de texte multilingue



Imprimé par Grif le 20/3/91

# Introduction

Les alphabets courants, comme les alphabets latin (ou romain), grec, cyrillique, etc. sont souvent utilisés dans la plupart des programmes de traitement de texte. Lorsque qu'on veut étendre ces traitements de texte à d'autres langues qui emploient d'autres signes diacritiques, ou des accents, ou des "alphabets" comme ceux utilisés en chinois, en arabe..., les solutions utilisées pour les alphabets courants ne sont plus applicables.

Les traitements de texte courants sont fortement dépendants de la représentation interne des caractères utilisés dans ces alphabets, à la fois pour la saisie et pour le stockage des informations. De plus, il n'y a pas de relation biunivoque entre les codes de saisie et la représentation interne des caractères.

Pour ne pas limiter l'ensemble de caractères traitables, notamment les caractères non latins, on a développé plusieurs systèmes de traitement de texte dont le codage utilise des normes internationales. À côté des normes ISO bien connues pour les langues latines, on peut citer des normes pour les langues à idéogrammes : JIS (Japan International Standard) au Japon, GB 2312-80 (国 标 pour "Norme Nationale") en République Populaire de Chine (RPC) [III.1 GB 81], BIG-5 à Taïwan, etc. L'utilisation de ces normes est suffisante dans certaines applications, mais, du point de vue de la conception d'un STTM, il y a d'autres problèmes à prendre en compte.

Il faut concilier deux niveaux : celui du traitement de texte et celui du traitement des langues naturelles. Le premier niveau ne s'intéresse qu'au résultat obtenu en répondant à la question : comment visualiser, sur l'écran ou sur le papier, un caractère d'un alphabet donné ? En revanche, le second doit répondre à une deuxième question : ce système est-il utilisable dans une application linguistique ? Pour cela, il est nécessaire d'apporter au contenu du texte traité, ou au niveau de chaque signe d'un système d'écriture donné, une information suffisante pour reprondre aux besoins de l'application envisagée.

À titre d'exemple, en s'appuyant sur un codage donné, dans le traitement de texte, les caractères đ et ô du vietnamien sont traités de la même manière : on ajoute des signes diacritiques (barre ou circonflexe) à d et o pour obtenir leur image, sur l'écran, ou sur le papier. Mais du point de vue du traitement des langues naturelles, on ne peut pas reconnaître à partir des codes utilisés en représentation interne que đ est une variante de d, et ô une variante de o, ce qui serait bien utile pour que les tris "système" produisent des résultats linguistiquement corrects. C'est aussi le même problème avec des variantes morphologiques ou avec la ligature (pour les caractères arabes, par exemple). On aboutit ainsi à la conclusion qu'il faut établir une distinction pertinente entre les codes des caractères et le traitement qu'on peut effectuer.

Suivant celui du codage, les problèmes concernent la saisie, la restitution et le dialogue multilingue. Pour chaque système d'écriture, on peut avoir différentes méthodes de saisie.

Mais deux systèmes d'écriture différents peuvent avoir éventuellement une méthode de saisie commune. La restitution pose aussi des problèmes spécifiques pour chaque système d'écriture, comme pour l'arabe (sens d'écriture), pour le chinois (nombre des idéogrammes), pour le thaï (position relative des signes)... Enfin, pour un système multilingue, il est nécessaire d'effectuer facilement le changement de langue des messages et des menus qui doivent s'adapter à l'utilisateur.

Dans le développement du traitement de documents, l'édition de documents structurés est l'aboutissement actuel de l'évolution des traitements de texte et des concepts qui ont été progressivement dégagés dans ce domaine. Cette approche distingue deux aspects différents : la structure logique et la présentation physique du document.

La structure logique concerne les entités figurant dans le document (chapitres, sections, sous-sections, paragraphes, texte...) et les relations entre ces entités en distinguant ce qui appartient au *modèle* (structure générique) et ce qui appartient à un document réel (structure spécifique).

La présentation physique concerne trois types d'objets différents : texte, hors-texte (formules mathématiques, tableaux, éléments graphiques, figures, etc.), et commandes (ou ordres de formatage) qui donnent des indications sur la présentation des entités pour la mise en page.

La manière de prendre en compte les commandes et de manipuler du texte sont des facteurs importants dans la conception d'un système de traitement de documents.

Cette première partie est organisée en trois chapitres. Le premier, qui est un état de l'art, précise les notions nécessaires en présentant l'histoire de l'évolution des systèmes de traitement de documents et les outils actuellement disponibles. Le deuxième analyse le problème du multilinguisme en s'appuyant sur quatre aspects essentiels : codage, saisie, restitution et dialogue. Le dernier donne une présentation générale de l'édition des documents structurés.

Imprimé par Grif le 20/3/91

# Chapitre 1 : État de l'art

# 1 Présentation

## 1.1 Évolution du traitement de documents

Les systèmes de traitement de documents proviennent des *systèmes de traitement de texte*, eux-mêmes dérivés des *éditeurs de texte*. Un éditeur de texte est un programme qui permet de manipuler un *texte* composé d'une suite de *caractères*, puis de le sauvegarder physiquement dans un fichier. Le type *caractère* est un type de données fondamental. Chaque caractère a un code informatique et est représenté par un signe typographique, ou dessin. Comme exemple de caractères, on peut citer les lettres majuscules et minuscules, les chiffres, les signes de ponctuation, etc.

Dans les premiers éditeurs, on traitait un texte simple ne comportant que des caractères anglais. Un éditeur de texte dispose de fonctions permettant la saisie du texte au clavier et son affichage avec un seul type de dessin typographique. Il existe aussi des fonctions plus ou moins perfectionnées qui permettent de corriger le texte. Les opérations les plus usuelles sont l'insertion et la suppression de lignes dans un texte, ou de caractères dans une ligne, la recherche et/ou le remplacement des occurrences d'une suite donnée de caractères par une autre suite, etc.

Plus élaborés, les systèmes de traitement de texte permettent de manipuler, et de formater sur plusieurs unités de sortie, des textes plus complexes et des hors-textes. Un texte contient non seulement des caractères, mais également des marques ou balises codées qui signalent des modifications typographiques ou de présentation (italiques, gras, paragraphes, sautes de pages, etc.), ainsi que les changements éventuels de *système d'écriture*.

Un système d'écriture est un ensemble de propriétés sur un *jeu de caractères*, le plus souvent lié à une langue naturelle. Un jeu de caractères est un ensemble de signes alphabétiques, syllabiques, idéographiques ou autres. Une langue peut avoir plusieurs systèmes d'écriture, comme le japonais, qui utilise les systèmes de caractères kanjis et kanas (hiraganas et katakanas). Inversement, il y a plusieurs systèmes d'écriture utilisant les caractères romains ou latins, comme ceux de l'anglais, du français, de l'allemand, etc.

Pour un système d'écriture, on distingue différentes fonctions d'un traitement de texte en entrée et en sortie. En entrée, il s'agit des descriptions de l'interface utilisateur, des capacités des moyens matériels disponibles : écrans alphanumériques ou graphiques, claviers (nombre de touches et fonctions disponibles), souris... En sortie, on veut produire un texte ayant des caractéristiques de présentation données (pages, polices, styles...) ; mais à cause de la variété des imprimantes, les commandes nécessaires pour obtenir cette

présentation sont spécifiques à chacune d'entre elles. Un pilote d'imprimante se charge de produire le flux de commandes requis.

À l'aide de fonctions complémentaires, le texte est saisi dans les jeux de caractères retenus, en utilisant le codage spécifique à chacun, et affiché avec le centrage, la justification, la numérotation et l'indentation de paragraphes. Ces fonctions permettent également d'effectuer, d'une part, le changement des attributs typographiques (tels que gras, italique ou souligné) de certaines parties du texte, et d'autre part, la mise en page du document (pagination, interlignage, fixation des marges, etc.).

Aujourd'hui, les systèmes de traitement de texte ont été conçus pour être simples d'utilisation et capables de supporter le traitement de documents. De nombreux systèmes de ce type ont été développés et sont disponibles commercialement. Ils intègrent des outils ou fonctions pour la saisie ou pour la restitution sur différents dispositifs d'entrée et de sortie.

## 1.2 Deux types de conceptions

Il existe actuellement deux types de conception pour les systèmes de traitement de documents : la conception *non-interactive* et la conception *interactive*.

La conception des systèmes non-interactifs est fondée sur la *compilation* : on construit un fichier source contenant le texte et les commandes de formatage à l'aide d'un éditeur. Un programme, appelé *formateur*, analyse les commandes dans ce fichier et les exécute automatiquement sur le texte source. L'utilisateur obtient le texte cible lisible sur l'écran ou imprimable et il ne peut pas intervenir pendant le déroulement du programme. S'il n'est pas satisfait du résultat, il doit modifier le fichier source décrivant ce qu'il veut obtenir et relancer l'exécution de l'ensemble.

Sur ce concept, Troff (de la Société ATT) et un certain nombre d'autres, comme TEX (de D. Knuth [III.5 Knuth 86]), Scribe (développé par B. Reid à l'université Carnegie-Mellon) et Mint (de P. Hibbard), présentés par [III.2 Quint 87b], ont été développés et implémentés. Ces systèmes fonctionnent sur différents matériels et systèmes d'exploitation.

La conception des systèmes interactifs est fondée sur l'*interprétation* : on interprète au fur et à mesure qu'on entre des commandes au clavier. Dans ce cas, la forme finale du texte est visualisée directement à l'écran durant la saisie. L'utilisateur peut réagir rapidement sur les résultats intermédiaires à l'aide de commandes. Il peut manipuler simultanément des images textes ou hors-textes (schémas, formules, tableaux, etc.) grâce aux écrans à haute résolution. Ces images sont représentées de la même manière que sur papier avec une différence éventuelle due à la résolution de l'écran disponible.

C'est cette possibilité d'utiliser un modèle d'image unique pour la visualisation et l'impression qu'on appelle WYSIWYG (en anglais : "what you see is what you get", ou "à l'écran comme sur le papier"). Ce concept est à la base de nombreux systèmes disponibles sur le marché, comme, pour les PC, Words, WordStar, Write, Multiscribe... ou TianMa

[III.4 Asia 89] (un système de saisie du chinois qui a des capacités WYSIWYG restreintes), MacWrite, Word... pour les Macintosh, "The Publisher" sous Unix, ainsi que de projets de recherche, comme Grif dans notre travail, etc.

Il existe des logiciels conciliant ces deux conceptions. Ce sont des programmes permettant l'échange de documents stockés dans des fichiers selon deux possibilités : la transmission sur support informatique d'une machine à une autre et la traduction vers d'autres systèmes (le cas de PostScript). La transmission est nécessaire parce que l'utilisateur peut travailler sur plusieurs types d'ordinateurs avec le même document. La traduction assure, en revanche, le traitement de document de façon cohérente selon l'outil d'édition choisi.

Il est difficile de faire une comparaison absolue sur les performances entre un système non-interactif et un système interactif. Pour les systèmes non-interactifs, l'utilisateur doit posséder une certaine compétence en programmation, mais il peut travailler sur n'importe quel terminal. Ces systèmes demandent plutôt une analyse des composantes d'un texte (notion de blocs logiques, d'attributs locaux...) et une pratique constante (pour ne pas oublier le sens des balises utilisées).

Par contre, les systèmes interactifs sont plus populaires, car faciles à comprendre et simples d'emploi, mais l'utilisateur doit disposer de matériels onéreux ou spécifiques (écrans à haute résolution, souris, etc.). Il y a cependant deux limites à noter :

1. Ce sont souvent des systèmes fermés, dont on ne connait pas les codes, ce qui pose des problèmes lors des "traductions" vers d'autres systèmes (filtres pour l'importation de documents, compatibilité dans le temps avec les versions antérieures...).
2. Le traitement est souvent local : une modification quelque part doit parfois être répercutée "à la main" dans tout le reste du texte ; c'est un gros problème pour la production industrielles de documents.

## 1.3 Informatique multilingue

Les problèmes du multilinguisme ont été d'abord ignorés : tout devait se faire en anglais. Depuis une vingtaine d'années, les nécessités commerciales ont amené les constructeurs et les éditeurs de logiciels à se soucier de localisation : pour chaque langue assez importante, on fournit ainsi la possibilité de traiter des documents utilisant deux systèmes d'écriture, l'un fondé sur celui de l'anglais et l'autre sur celui de la langue en question.

Xerox a été un précurseur, dans ce domaine comme dans bien d'autres, en reconnaissant, il y a déjà plus de dix ans, l'intérêt d'une informatique réellement multilingue. Il apparait en effet aujourd'hui que l'informatique "localisée" coûte finalement très cher, en obligeant à maintenir des versions parallèles, et est un frein aux échanges électroniques et à la normalisation. À cet égard, le cas d'Unix, soit disant universel, est presque ubuesque.

Au niveau des applications, on voit aussi de plus en plus la nécessité de bases lexicales et textuelles multilingues, associées à la production et au traitement de documents qui peuvent être écrits à l'aide de plus de deux systèmes d'écriture (notes techniques, touristiques, éditions bilingues ou multilingues avec citations, etc.).

# 2 Outils disponibles

Nous présentons maintenant trois systèmes de traitement de documents : TEX de D. Knuth, Star de Xerox [III.1 Becker 84] et WinText de WinSoft [III.5 WinSoft 88] dont les deux derniers ont la particularité d'être multilingues.

## 2.1 TEX

### Présentation

TEX, créé par Donald E. Knuth en 1977, est un système non-interactif où on effectue le traitement de texte avec *balisage* : l'image finale du document est produite par compilation d'un fichier source contenant le texte et des commandes de formatage. Ce système produit des documents d'une qualité typographique exceptionnelle, avec notamment des formules mathématiques et comporte des caractéristiques intéressantes.

Ainsi, certains aspects de la restitution sont automatiquement résolus, comme les ligatures (fi, fl, œ, æ...), la coupure des mots en bout de ligne et la justification, la mise en page, etc. Pour imprimer un document complet contenant toutes sortes d'images (texte complexe, hors-texte), TEX fournit les descriptions de pilotage d'imprimante sous forme d'un fichier particulier, appelé `dvi` (device independent).

La deuxième caractéristique est la portabilité absolue. Un document TEX peut être composé sur n'importe quelle station de travail, sur un Macintosh ou sur un compatible PC, sur un CRAY ou sur un Atari portable, etc.

TEX est essentiellement un outil typographique. Or, un utilisateur pense (et écrit) avec d'autres catégories : paragraphes, sections, chapitres, etc. TEX fournit donc à l'utilisateur la possibilité de créer de nouvelles commandes complexes (gérant des changements de polices, de taille, des positionnements...), ou des mises en page (taille, multi-colonnage, en-tête et pieds de page, numérotation...). Dans ce sens, c'est aussi un langage de programmation. Les commandes ainsi créées sont appelées des macros. Comme ensemble cohérent de macros, on peut citer $\mathcal{AMS}$-TEX (pour les mathématiques) et LATEX. On aboutit alors à une tâche plus performante.

Il y a deux étapes de manipulation d'un document TEX : la saisie et la compilation. Lors de l'étape de saisie, l'utilisateur crée un fichier source sur disque à l'aide d'un éditeur de texte. Cette étape ne permet pas encore la mise en page. TEX compile ensuite le fichier source pour créer un fichier de sortie, le fichier `dvi` qui est une description des pages du document final (ou plutôt des opérations à faire pour composer ces pages). Dans ce fichier,

les caractères du fichier source, qui composent le texte final, sont remplacés par leurs dimensions à l'aide d'un fichier spécial `tfm` (text font metrics), fichier qui contient toutes les informations de taille et de déplacement nécessaire au positionnement des caractères dans la page.

À partir du fichier `dvi`, un programme de pilotage (de visualisation sur l'écran ou d'impression) dessine le texte voulu en utilisant les fichiers de polices. La figure 1.1 ci-dessous résume ce processus.

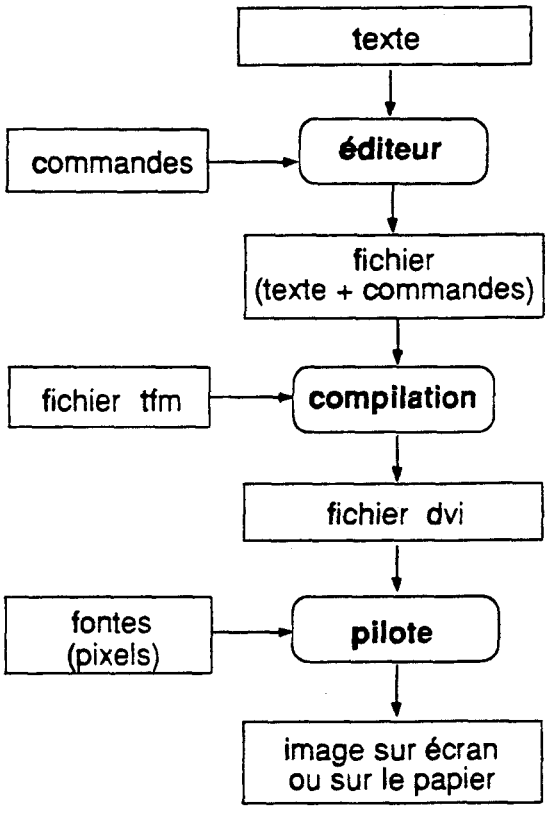



Figure 1.1 : Architecture de TEX

## Codage

Jusqu'à la version 3.0, TEX n'utilise qu'un ensemble limité de caractères ASCII, à savoir les caractères standard de codes inférieurs à 127 (sur 7 bits). Cette restriction ne permet pas de manipuler facilement les autres caractères.

Cependant, le problème des lettres accentuées n'était pas très difficile à résoudre, car TEX dispose d'un ensemble restreint de commandes qui permettent de construire, à la composition, les lettres accentuées. Il existe actuellement des versions étendues de TEX adaptées particulièrement au traitement de texte multilingue, comme TEX multilingue [III.4 Ferguson 86], JTEX pour le japonais, l'insertion dans TEX des caractères chinois [III.4 Cousquer 90], ou arabes, etc. La nouvelle version (3.0 et suivantes) permet de gérer

des alphabets de 256 caractères (sur 8 bits), mais les polices correspondantes ne sont pas encore disponibles.

TEX étant essentiellement un outil typographique, le multilinguisme y apparaît d'abord comme un changement de police de caractères. C'était le cas avant la nouvelle version 3.0. À titre d'exemple, avec cette version, une commande spéciale `\langage` permet de préciser le fonctionnement de certains paramètres (règles de césure par exemple). Ces règles peuvent ne s'appliquer qu'à une partie du texte. La restitution multilingue est toujours fondée sur le changement de polices qui sont limitées à 256 caractères.

Pour noter les alphabets utilisés dans un texte source multilingue, on peut utiliser deux méthodes, *balisage* et *marquage*. La méthode de balisage consiste à insérer, à chaque changement d'alphabet, l'indication de l'alphabet suivant. La méthode de marquage consiste à marquer l'alphabet sur chaque caractère, soit par utilisation alternative d'un octet (l'indication au premier bit), ou de deux octets (au premier bit du premier octet), soit par codage de chaque caractère sur deux octets (un signe spécial au premier octet pour les caractères latins, etc.).

Dans les deux cas, il est nécessaire d'utiliser un filtre permettant de transcrire le texte source multilingue en un texte source "à la TEX" (c'est-à-dire gérant les changements de polices).

### Saisie

Comme TEX est un langage de programmation, la saisie d'un texte est guidée par des règles. Il existe des commandes qui permettent de traiter les lettres accentuées d'une façon simple. On utilise le symbole `'\'` (antislash), qui introduit un nom de commande. Ainsi, `\^`, `\'`, `\c`, etc. sont des commandes de TEX qui agissent sur les entités `a`, `e`, `c`, etc. pour donner le symbole cherché.

À titre d'exemple, voici comment saisir quelques lettres accentuées :

| *Composition :* | *Produit :* |
|---|---|
| `\^a` | â |
| `\'a` | á |
| `\c c` ou `\c{c}` | ç |
| `\^E` | Ê |
| `\'E` | É |
| `\c C` ou `\c{C}` | Ç |
| etc. | etc. |

Pour améliorer cette méthode de combinaison, un préprocesseur pour la saisie, appelé Stratec, a été développé par [III.5 Seroul 89]. Il utilise les caractères accentués du clavier. L'alphabet comporte la plupart des symboles mathématiques courants et toutes les lettres

grecques. Les avantages de la méthode sont la possibilité d'utiliser un correcteur orthographique et la lisibilité des fichiers de saisie.

Pour un texte comportant des caractères idéographiques (japonais, ou chinois), il faut disposer d'un éditeur spécialisé. La figure 1.2 présente une application de TEX pour traiter un texte chinois.

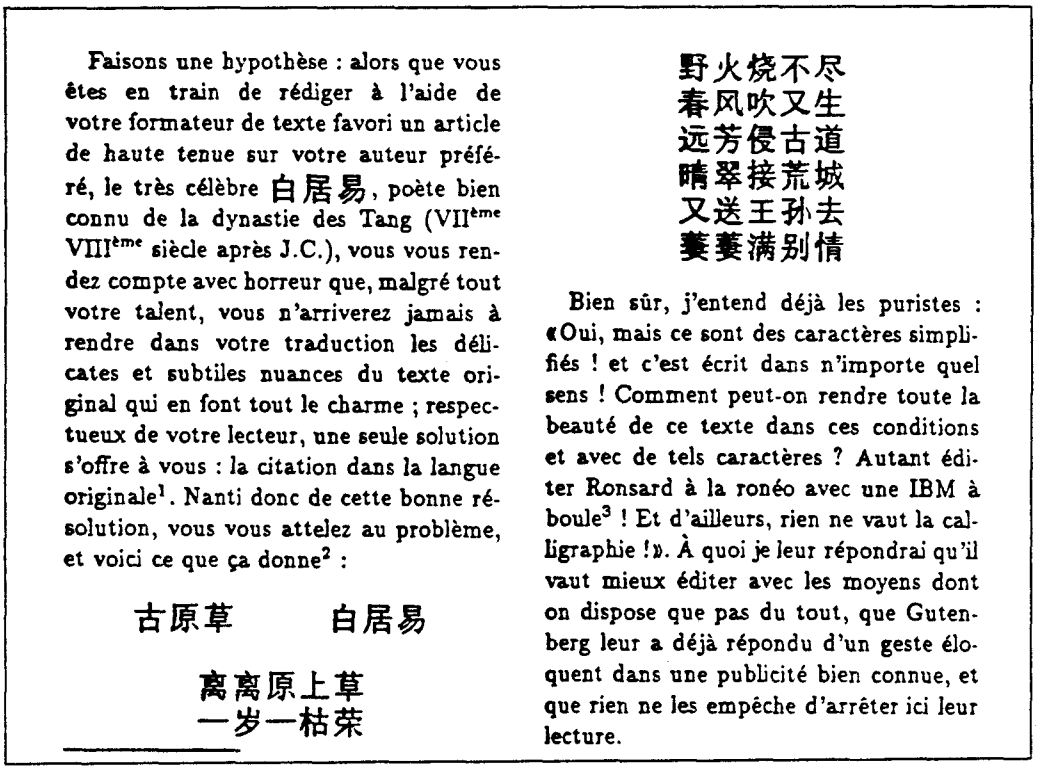



Figure 1.2 : Insertion d'un texte chinois dans TEX

## Restitution

Tous les aspects de restitution d'un document TEX sont prévus et introduits dans le fichier source. À côté des fonctionnalités usuelles d'un système de traitement de texte, comme la mise en page, la mise en paragraphes, la justification, les tabulations, les enrichissements typographiques (taille, style...), il existe des utilitaires de TEX qui permettent de traiter les problèmes de césure et de correction orthographique.

La césure à chaque passage à la ligne était un gros problème pour les textes multilingues (avec lettres accentuées) avant la version 3.0 de TEX. Pour chaque langue, il faut utiliser un algorithme particulier, car les règles de coupure des mots varient selon les langues. Par exemple, le français décompose le mot *continue* en *con-ti-nue*, alors que l'anglais le décompose en *cont-in-ue*.

### Dialogue

Dans l'étape de saisie du document, l'utilisateur travaille avec un éditeur de texte et il est guidé par un dialogue spécifique à cet éditeur.

Quand TEX compile, l'utilisateur obtient des messages d'erreur sauvegardés dans un fichier de dialogue. À l'aide de ces messages, il peut corriger son document. Il arrive aussi que TEX s'arrête pour annoncer une erreur et demander une réponse immédiate à l'utilisateur.

Il y a une seule langue de dialogue par version.

## 2.2 Star de Xerox

### Présentation

Star est un logiciel de traitement de texte multilingue intégré sur une station de travail "Documenteur ™" fabriqué par Xerox (machine Xerox 1165, ou maintenant Sun plus ajouts Xerox). Avec Star, on peut manipuler des textes non-latins japonais, chinois, arabes... et les mélanger avec les textes latins dans un même document. Le fonctionnement de Star demande un équipement propriétaire : la station de rédaction, appelée aussi Star, dispose d'un écran graphique à haute résolution, d'un clavier anglais et d'une souris. Le clavier contient des touches particulières, comme la touche de renvoi, la touche d'espacement qui est divisée en quatre segments correspondant aux trois alphabets japonais (katakana, hiragana et kanji) et l'alphabet latin, etc.

### Codage

Star utilise un système de codage "souple" mis au point par G. Curry (service de bureautique de la Société Xerox). Le principe de la méthode "souple" est l'utilisation d'octets d'identification et de marquage du changement d'alphabet à coder. Il y a deux types d'alphabets : les *alphabets ordinaires* (ou phonétiques) et les *superalphabets* (idéogrammes). Les caractères des alphabets ordinaires utilisent chacun un seul octet par code. Par contre, les superalphabets utilisent deux octets pour représenter leurs caractères et signaler le passage à un alphabet à un autre superalphabet.

Pour les caractères des alphabets ordinaires, l'alphabet latin (considéré comme l'alphabet courant) est codé 00, le grec 46, le russe 27, l'arabe E0, etc., en hexadécimal. Le code FF est le signal de changement d'alphabet.

Pour les caractères chinois, on considère chaque bloc de 65 536 caractères comme un superalphabet, et deux octets consécutifs FF comme un signal de changement de superalphabet. Le superalphabet principal, codé 00, est pratiquement le seul utilisé, sauf pour quelques caractères chinois très peu courants.

Le texte est une suite d'octets rangée dans l'ordre de lecture (ou l'ordre phonétique). Au départ, l'alphabet par défaut est l'alphabet latin. Chaque fois que le signal de changement d'alphabet ordinaire (un octet FF), ou de superalphabet (deux octets consécutifs FF) apparaît, l'octet suivant est interprété comme le nom d'un nouvel alphabet ordinaire, ou

le nom d'un nouveau superalphabet respectivement. Les octets qui suivent sont les codes des caractères de cet alphabet (un ou deux octets par caractère).

### Saisie

La plupart des alphabets traitables par Star sont saisis par interprétation des touches. Le principe de la méthode est très simple. Par exemple, en alphabet russe, la frappe de la touche "F" sur le clavier latin donne la lettre Ф (ef). De même, en alphabet grec, la frappe de cette touche donne la lettre Φ(phi).

Pour rappeler à l'utilisateur l'alphabet associé au clavier et la correspondance entre les touches et les caractères, Star peut afficher sur son écran un clavier virtuel, ou un petit tableau suivant le cas. La saisie, ou le choix d'un caractère peut donc aussi être effectuée par sélection grâce à la souris.

Star utilise la méthode de conversion phonétique pour la saisie des caractères d'un superalphabet. La saisie s'effectue en trois étapes. La première étape est l'entrée au clavier de la transcription phonétique du caractère. Par exemple, le japonais utilise la conversion *romaji*, le chinois utilise la conversion *pinyin*, le coréen utilise la conversion *hangul*.

Quand l'utilisateur frappe la touche de renvoi (spécifique au clavier Star), la deuxième étape consiste à produire la liste des caractères correspondant à la transcription entrée. Pour le chinois, on tient aussi compte du dialecte.

La dernière étape est le choix du caractère désiré.

### Restitution

La restitution multilingue dans Star est bien résolue pour les caractères accentués, les ligatures, les variations morphologiques, et le traitement des textes mixtes. Pour les caractères accentués (comme à, è, é... du français, ü de l'allemand, etc.), Star stocke le code du signe diacritique, suivi immédiatement du code du caractère auquel il s'applique.

Les ligatures se rencontrent très souvent dans l'écriture de l'arabe, du latin (comme æ, œ, fi, ffi...), etc. Par exemple, Star affiche le mot *cœur* en quatre caractères en conservant en mémoire la séquence de cinq caractères *c o e u r*.

Les variations morphologiques se rencontrent dans certains systèmes d'écriture (grec, arabe, hébreu...) où il n'existe pas de correspondance biunivoque entre le code interne d'un caractère et son image graphique. Par exemple, Star sait afficher le caractère grec sigma (σ) sous la forme particulière (ς) lorsqu'il se trouve à la fin d'un mot, ou les caractères arabes qui ont quatre formes différentes suivant que leur position est isolée, au début, au milieu ou à la fin d'un mot.


Figure 1.3 La consonne ba' arabe a 4 formes différentes

Le traitement des textes mixtes dans Star repose sur les codes de changement d'alphabet et sur le contrôle de fin de ligne pour afficher correctement les textes suivant leur sens d'écriture. Par exemple, en conservant en mémoire l'ordre phonétique, on peut insérer un texte arabe dans un texte latin, et inversement, un texte latin dans un texte arabe. La figure 1.4 présente l'insertion d'un texte latin dans un texte arabe.

Dans le schéma, le texte arabe est représenté par la boîte grise et les caractères de cette langue apparaissent donc sur l'écran de la droite vers la gauche (a). Le curseur se déplace chaque fois vers la gauche pour indiquer l'endroit où apparaît le caractère suivant (b).

Il s'agit d'entrer un texte latin qui est représenté par la boîte transparente (c). Le curseur devient immobile et, à chaque entrée d'un nouveau caractère, les autres doivent se déplacer d'un cran vers la gauche pour lui laisser une place (d). Quand la ligne est pleine, le curseur saute à droite au début de la ligne suivante.

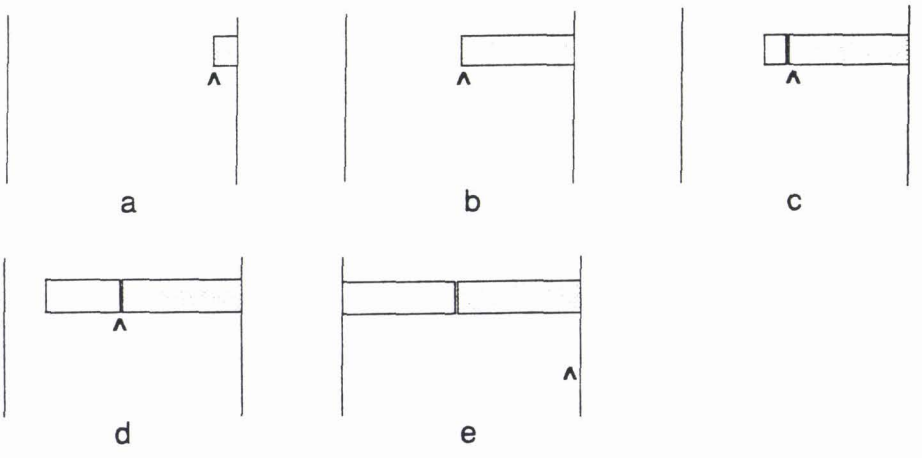



Figure 1.4 : Combinaison de deux sens d'écriture dans Star

## Dialogue

En utilisant la souris et le clavier physique, l'utilisateur peut effectuer un dialogue interactif avec Star pour produire un document fortement multilingue. Sur l'écran, on trouve des "touches de fonction" qui représentent des commandes disponibles, des fenêtres de documents, des claviers virtuels, ou des tables. Ces intitulés peuvent être affichés en même temps en anglais et en quelques langues disponibles. Par exemple, pour travailler avec le japonais, il y a une touche de fonction qui est affichée en japonais et en anglais, etc.

## 2.3 WinText de WinSoft

### Présentation

Le logiciel WinText, produit par WinSoft, est particulièrement adapté au traitement de texte multilingue et multi-documents pour Macintosh. Avec WinText, l'utilisateur peut travailler sur les langues non romaines, comme l'arabe, l'hébreu, le japonais, le chinois, etc.

Fondé sur le principe interactif, WinText permet la combinaison dans un même document, sans restriction, des langues disponibles sous le MacOS utilisé. De plus, WinText dispose, non seulement de la possibilité d'agrémenter le texte édité de dessins et graphiques élaborés à l'aide d'autres applications (comme MacPaint ou MacDraw), mais encore d'un éditeur spécialisé ÉditMath (développé par V. Quint et I. Vatton) pour les formules mathématiques.

### Codage

Dans WinText, on utilise le codage du "SCRIPT monitor" du MacOS. C'est un codage mixte, utilisant un octet ou deux octets, selon la version du MacOS. Il s'agit donc d'un multilinguisme restreint, de type "localisation". Les systèmes MacOS suivants sont actuellement disponibles : AIS pour l'arabe, HEIS pour l'hébreu, HanzeTalk pour le chinois, KanjiTalk pour le japonais, etc.

Par exemple, le système KanjiTalk utilise le standard S-JIS (Shift-Japanese Industrial Standard) [III.1 Kleinberg 87]. C'est un ensemble de caractères divisé en deux niveaux. Le premier niveau contient 3 800 caractères usuels. Le second contient 3 005 caractères additionnels. La représentation d'un caractère utilise deux cotets.

### Saisie

WinText permet la saisie directe d'un texte à partir du clavier ; le texte frappé s'affiche immédiatement sur l'écran. Ainsi, la saisie de WinText est effectuée par la correspondance et par le contexte. La correspondance donne à l'utilisateur le clavier correspondant au système choisi parmi les ressources disponibles (arabe, chinois, hébreu, japonais, etc.). Ce système, qui est mixte, permet d'éditer également les caractères latins. Par contre, le contexte de saisie autorise l'échange entre les systèmes avec cohérence. L'utilisateur sélectionne un alphabet à l'aide d'un signe particulier, puis il travaille avec les règles et le sens d'écriture propres à cet alphabet.

### Restitution

La restitution d'un texte dans WinText est effectuée par les règles de formatage. Il y a deux types de règle : règle de texte et règle de tableau. Une règle de texte permet de choisir le sens dominant d'écriture (de gauche à droite ou de droite à gauche), le cadrage du texte entre les marges (à droite, à gauche, centré, justifié), l'espacement entre les lignes, et de placer des tabulations (gauches, droites, centrées, décimales).

Une règle de tableau permet de définir les colonnes et l'encadrement d'un tableau contenant du texte ou des formules.

L'interactivité de WinText permet à l'utilisateur d'obtenir sur le papier une image de son document identique à celle affichée à l'écran.

### Dialogue

WinText existe en à peu près autant de versions que le MacOS. Chaque version n'a qu'une langue de dialogue. Cependant, on peut utiliser n'importe quelle version de WinText sous n'importe quel MacOS, car la différence se situe essentiellement au niveau des messages. Toutefois, tout ce qui concerne la présentation des nombres, des montants financiers, des dates et des heures est particulier à chaque version.

La figure 1.5 ci-dessous montre une fenêtre de travail en japonais du système WinText japonais.

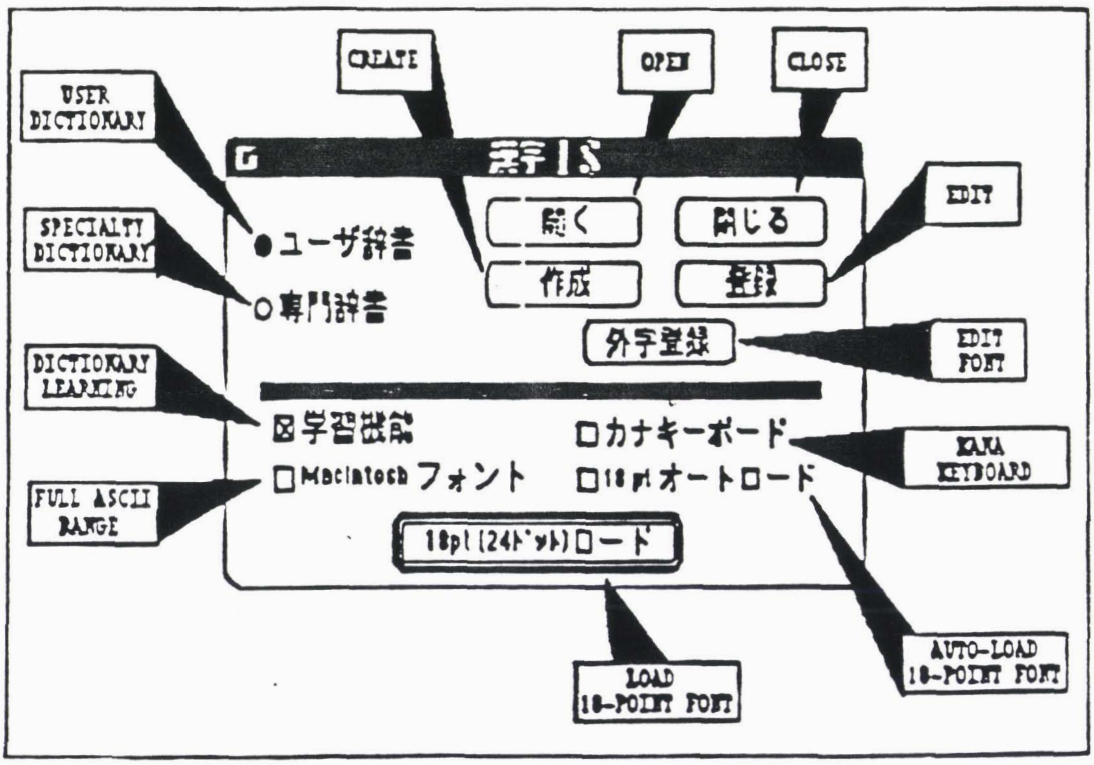



Figure 1.5 Fenêtre de travail du système WinText japonais

# 3 Limites des outils actuels

À part le Star de Xerox, les systèmes "multilingues" actuels ne sont que des systèmes "localisés". Tous présentent des problèmes pour arriver à un réel multilinguisme, et à la portabilité souhaitable.

## 3.1 Codage

Bien que les codages utilisés dans les systèmes présentés (sur un octet, sur deux octets, ou sur un mélange de deux types) répondent aux besoins d'un éditeur, ils ne répondent pas aux exigences de portabilité et surtout de facilité d'utilisation par des applications de TALN.

Notons que ce problème de codage des textes est actuellement l'objet d'un gros travail au sein de la TEI [III.2 Sperberg 90], pour l'instant surtout au niveau logique et typographique, l'aspect multilingue étant seulement envisagé.

## 3.2 Saisie

Les méthodes de saisie utilisées dans Star, ou WinText... sont intégrées à l'avance par les constructeurs. Pour une langue donnée, le système utilisé peut proposer à l'utilisateur un certain nombre de méthodes différentes (le cas du chinois dans WinSoft par exemple), toutes les méthodes de saisie sont dépendantes des matériels disponibles et des ressources (polices de caractères et tableaux de données correspondantes). Il est difficile ou impossible d'y intervenir pour modifier selon les besoins de commodité ou d'expérience de l'utilisateur. En outre, comme nous l'avons vu, la saisie multilingue dans TEX (pour les versions étendues qui traitent le chinois, le japonais, ou l'arabe...) n'est pas toujours facile.

## 3.3 Restitution

Les systèmes que nous avons présentés ne permettent pas encore d'effectuer facilement et complètement certains aspects de la restitution.

D'une part, ces systèmes, comme beaucoup d'autres systèmes non multilingues que nous verrons dans le chapitre suivant, ne disposent pas de la notion de modèle de document. Cette lacune sera comblée dans l'édition de documents structurés où on construit un modèle de document de haut niveau et où on distingue la structure logique de la présentation physique du document.

D'autre part, comme la restitution multilingue dépend strictement des polices de caractères disponibles, dans Star, ainsi que dans WinText, les traitement sont partiels : l'utilisateur n'a aucun moyen d'utiliser une autre police, si le système n'en dispose pas encore, pour modifier la présentation de son document.

Étant un outil typographique,TEX a par exemple des possibilités multilingues en ce qui concerne la gestion des polices, mais il y a des limitations :

* nombre de caractères par police de 128 ou 256 (version 3.0) ;
* algorithmes de mise en page suivant la disposition des lignes en colonnes : l'écriture horizontale est seule prise en compte.

* absence de la notion de langue : des traitements linguistiques comme la correction orthographique sont donc difficiles.

Dans le traitement multilingue, il faut mettre en évidence ce qui ressort de la présentation et ce qui appartient à la langue écrite. Pour la présentation, c'est celle de la structure du document et celle de la typographie (y compris la gestion des polices pour les mises en évidence, par exemple) liée aux alphabets utilisés et à l'usage. Pour la langue écrite, ce sont des aspects comme le sens d'écriture, les unités lexicales (mots, locutions...), la ou les normes de codage ou de représentation.

## 3.4 Dialogue

Les outils actuels ne permettent pas encore de changer facilement les langues de dialogue. Dans tous les cas, l'utilisateur est obligé de travailler avec un dialogue fixé à l'avance. Il a ainsi des difficultés inévitables quand il produit un document dans sa langue, mais avec une langue de dialogue qu'il connait mal ou qu'il ne connait pas.

On peut remarquer que la notion de dialogue n'appartient pas en propre à un logiciel donné, mais devrait dépendre de l'environnement de travail. C'est ce que procure par exemple l'environnement OSF-Motif (Open Software Foundation) de X-Window [IV O'Reilly et al. 88] [IV Young 90] où des ressources particulières à chaque langue peuvent être définies et utilisées par les logiciels selon la valeur d'une variable d'environnement (LANG). C'est sans doute cette solution qui est la plus souple d'utilisation.

# Chapitre 2 : Analyse du problème



Imprimé par Grif le 20/3/91

# 1 Codage

La présentation des textes dans la plupart des systèmes de traitement de texte utilise des codes sur huit bits, comme les codes ASCII (en norme ISO), EBCDIC... Avec cette représentation, on ne peut coder que 256 caractères différents au maximum. Ce n'est pas suffisant quand on pense que le nombre de caractères utilisés dans l'écriture de toutes les langues naturelles du monde est bien supérieur (cas du japonais, ou chinois).

Suivant les besoins des utilisateurs, la manipulation de signes quelconques des systèmes d'écriture demande non seulement un codage contenant tous les caractères, mais aussi un codage unique prenant en compte des informations telles que l'ordre du caractère dans le jeu utilisé, ou la nature typographique (taille, style, mise en évidence, distinction MAJ/min, etc.). Par exemple, le code du caractère 'A' n'est pas simplement représenté par un 'A', mais peut aussi être représenté par un '*A*' italique ou 'A' gras.

De plus, en utilisant le codage défini, il faut pouvoir effectuer simplement divers traitements, comme la recherche et la substitution de chaînes de caractères, le tri alphabétique, etc. Certains systèmes de codage sont construits en s'appuyant sur la relation entre le codage et le traitement des caractères de ce codage. Par exemple, en norme ISO ou en code EBCDIC, la relation est basée sur la distinction MAJ/min, l'ordre alphabétique des lettres, l'ordre des chiffres et des signes spéciaux, ainsi que le groupement des majuscules, des minuscules et des chiffres.

Quoi qu'il en soit, il est nécessaire de proposer de nouvelles méthodes de codage plus élaborées et plus efficaces pour plusieurs applications. On aboutit soit à des codages en format fixe (2, 3, 4 octets) [III.1 Anderson 84], soit à des codages en format variable [III.1 Becker 84]. Dans tous ces cas, le code interne est illisible sur les matériels usuels et non portables. Une deuxième solution consiste à définir des transcriptions fondées sur un ensemble universel de caractères et visant à la lisibilité et la portabilité. On aboutit à des codages en format variable sur plusieurs octets [III.1 Boitet & Tchéou 90].

La mise en œuvre d'un système de codage commun et unique pour tous les systèmes d'écriture est un problème important du point de vue linguistique. Il est clair que le problème devient de plus en plus difficile quand on passe d'un alphabet latin à un alphabet non latin. Cela demande une transcription, ou une translittération, convenable pour chaque système d'écriture considéré, et adaptable à tous.

Une transcription est une représentation de l'écriture d'une langue donnée qui utilise les caractères graphiques conventionnels d'un autre système d'écriture. On peut ainsi disposer d'une transcription phonétique qui permet de retrouver la prononciation réelle d'une langue. On peut définir également plusieurs types de transcription avec des qualités adaptées au traitement de texte ou au traitement des langues naturelles.

Dans ce qui suit, on examine quelques systèmes de codage, allant de deux octets (méthode d'Anderson, codages mixtes) à plusieurs octets (grands caractères CARIX). Le problème de codage du chinois sera présenté au chapitre IX, partie C.

## 1.1 Méthode d'Anderson

L. Anderson a proposé un codage en format fixe, où cahque caractère est codé sur deux octets, et de les analyser suivant trois critères pour le traitement de texte multilingue [III.1 Anderson 84]. Ces critères sont établis pour assurer l'indépendance de chaque alphabet et faciliter la translittération :

1. **Totalité** : chaque caractère indépendant est représenté par deux octets (lettre, ligature, diacritique indépendant, combinaison lettre-diacritique, idéogramme, syllabique), chacun doit avoir une position d'affichage ou d'impression indépendante.
2. **Identification** : à partir d'un code donné, on peut identifier l'alphabet. Il est possible que deux alphabets distincts contiennent un même caractère, même si sa transcription, venant de la prononciation, est différente dans chacun.

   Par exemple, le caractère 'c' en tchèque représente le son [ts], ce qui n'est pas le cas en anglais. La translittération anglaise de ce caractère devrait écrite 'ts' et non 'c'.
3. **Optimisation** : le codage doit maximiser la possibilité d'utiliser des réductions de codes à un octet (utile dans les longs textes monolingues) et minimiser les changements entre les différents blocs de 256 codes.

Il est difficile de trouver des solutions satisfaisant entièrement les trois critères. Anderson a proposé trois solutions dont le principe est de coder l'alphabet dans chaque caractère.

La première est d'utiliser un octet pour le nom d'alphabet et le deuxième octet pour introduire les codages standard existants, normes ISO nationales (7 bits ou 8 bits). La solution satisfait les trois critères, mais il n'y existe pas de transcription correspondante pour les alphabets non latins. Il est nécessaire de disposer de programmes de conversion, soit Nx(N-1) programmes pour N alphabets.

La deuxième est de prendre en compte systématiquement tous les alphabets utilisant des caractères latins par l'énumération de toutes les formes de lettres, et de toutes les combinaisons lettre-diacritiques. On obtient un tableau mixte de tous les caractères de tous les alphabets. Cette solution ne satisfait pas les trois critères. On peut l'appliquer effectivement aux alphabets latins, y compris l'alphabet vietnamien, mais pour les alphabets non latins, cette méthode devient inacceptable pour des raisons de taille.

Par exemple, en utilisant la norme ISO/TC97/SC2 N 1255 1982-11-01, on peut grouper toutes les lettres et tous les diacritiques possibles de 20 alphabets latins (y compris

l'alphabet vietnamien) dans un tableau. Ce codage considère *a, â, ă...* comme des lettres simples.

Dans la dernière solution, le premier octet représente le nom d'alphabet. Le deuxième octet correspond à un bloc de 256 codes, chacun d'entre eux respectant la translittération équivalente dans l'alphabet indiqué par le premier octet au caractère considéré. Ainsi, pour certains alphabets non latins, on a besoin de programmes de translittération.

D'après L. Anderson, la troisième solution satisfait les trois critères. Elle est utilisable pour le traitement de texte multilingue et peut donner un système de codage international unique. Pour les caractères occidentaux, il n'y a pas de translittération automatique.

Cependant, au niveau de traitement des langues naturelles (pour toutes les langues), celle-ci n'est pas encore satisfaisante. Il faut avoir plus d'informations pour chaque caractère codé (nature linguistique et graphique, comme les idéogrammes chinois).

## 1.2 Codages mixtes

Ces codages ont été rencontrés dans Star (codage "souple") de Xerox, ou dans WinText de WinSoft. Suivant le cas, le code est représenté sur un, ou sur deux octets.

Par exemple, dans Star, on code des pièces d'un texte dans lesquelles les caractères héritent (au sens informatique) des caractéristiques de ces pièces. La souplesse de cette méthode permet de s'appliquer à tous les systèmes d'écriture, tout en minimisant la longueur des suites de bits utilisées pour le stockage de textes en mémoire.

Comme les méthodes d'Anderson, les codages mixtes ne s'adaptent qu'au niveau de traitement de texte multilingue. Pour s'adapter mieux aux différentes applications linguistiques (notamment pour les caractères chinois, japonais), il est nécessaire d'utiliser des codes représentés sur plusieurs octets (des transcriptions, par exemple). Il s'agit plutôt de portabilité, de lisibilité et de "débogabilité".

## 1.3 La transcription et les CARIX

Les concepts de construction et réalisations des *grands caractères*, ou les *CARIX*, ont été développés au GETA lors de travaux du PN-TAO (Projet National de Traduction Assistée par Ordrdinateur) [II Vauquois & Boitet 84] [II Boitet 85] [III.1 Boitet et al. 86] [III.1 Vauquois & Robert 87].

Dans ces travaux, on utilise les CARIX dans un environnement qui est adapté notamment au TALN et qui repose sur trois types de transcription : *maximale, minimale* et *intermédiaire*.

S'il existe plusieurs types de transcription, c'est parce que les qualités souhaitables pour une transcription sont souvent contradictoires. Voici cinq contraintes qu'on rencontre :

1. **disponibilité** des signes utilisés sur la plupart des unités d'entrée-sortie ;
2. **minimalité** du nombre moyen de signes codant un caractère et du coût de saisie et de stockage ;
3. **lisibilité** dans l'utilisation des signes transcrits (accents, signes spéciaux...) et **facilité** d'apprentissage des transcriptions ;
4. **adaptation** à un système de traitement de texte ;
5. **maximalité** des informations codées.

### Transcription maximale

La transcription maximale utilise plusieurs octets, si nécessaire, pour coder le maximum de propriétés utiles sur chaque caractère. Les propriétés sont décrites par les attributs suivants :

1. *marque* : indique si le caractère appartient au texte ou au métatexte (ordre de formatage, élément d'un programme contenant le texte...) ;
2. *langue* : indique la langue naturelle, indépendamment du jeu de caractères utilisé pour sa représentation ;
3. *jeu* de caractères : ensemble ordonné de signes, alphabétiques, syllabiques, idéographiques, ou autres ;
4. *caractère* : identificateur dans le jeu de caractères correspondant à un numéro dans l'ordre défini sur le jeu ;

   Suivant le jeu, cet attribut peut s'accompagner des deux sous-attributs :

   1. *diacritique* : tous les diacritiques possibles du jeu donné ;
   2. *taille* : distinction MAJ/min ;
5. *relief* : avec trois sous-attributs *soulignement, gras, italique*.

On constate que l'attribut relief est pris en compte dans la transcription, tandis que l'attribut de police de caractères n'est pas défini. Ce choix est dû à ce que la police ne semble pas avoir de valeur linguistique, alors que le relief, à la limite, peut en avoir une.

### Transcription minimale

La transcription minimale utilisait un jeu de caractères (romains) très réduit (majuscules, chiffres et certains signes spéciaux), sous-ensemble de l'ISO 646, pour la disponibilité, la lisibilité et la portabilité. La définition de cette transcription a pour rôle de traduire la transcription maximale.

Pour transcrire une phrase complète, on utilise des conventions d'écriture, des ordres pour indiquer la taille (MAJ/min), le relief et les séquences d'échappement, et des commandes de formatage. À titre d'exemple, les lettres diacritées sont écrites par des combinaisons `lettre '!' nombre` (la juxtaposition des symboles indique la concaténation de ces symboles) :

```
&&FRA.*A!2 *BESANC!5ON, *MME. **PE!1LE!1 EU!3T ACHETE!1 UNE
BOI!3TE A!2 BIJOUX CHEZ **MATY POUR *NOE!4L.
```

pour

*À Besançon, Mme. PÉLÉ eût acheté une boîte à bijoux chez MATY pour Noël.*

La commande `&&FRA.` indique que la langue en cours est le français, écrit dans le jeu romain ; les signes `*` et `**` sont des ordres pour mettre le ou les caractères suivants en majuscules, `A!2` transcrit *À*, `C!5` transcrit *ç*, etc.

On remarque que quelquefois, pour éviter l'ambiguïté, on doit utiliser des signes d'échappement. Par exemple, pour obtenir *côté0,* il faut transcrire par `CO!3TE!1/0` et non `CO!3TE!10` qui donnerait côté, car `E!10` donne **é.**

### Transcription intermédiaire

La transcription intermédiaire considérée dans ces travaux utilisait les caractères de la norme ISO-25, adaptée aux claviers français. On pourrait en définir d'autres à partir d'autres tables ISO normalisées, nationales ou non. Comme la transcription minimale, cette transcription a encore pour rôle de traduire la transcription maximale.

Dans la définition, on utilise les conventions d'écriture sur le jeu de caractères utilisé. Par exemple, un diacritique peut avoir trois méthodes de représentation, soit directement dans le caractère (â, é, ç...), soit suivant la lettre considérée (a^ pour â, o´ pour ó...), soit comme dans la transcription minimale (y!4 pour ÿ...). Ainsi, l'opposition MAJ/min est notée dans le caractère lui-même. Avec l'exemple précédent :

```
&&FRA. A!2 Besançon, Mme. PE!1LE!1 eu!3t acheté une boi!3te à
bijoux chez MATY pour Noe!41.
```

### Construction de CARIX

On associe à chaque CARIX toutes les propriétés définies dans la transcription maximale. En principe, tous les caractères usuels, pour toutes les langues écrites à tradition, scientifique, littéraire ou religieuse, sont prévus. Le concept de construction des CARIX a fait l'objet d'une première implémentation en Pascal pour une version spéciale de Prolog-CRISS par Ph. Vauquois et S. Robert [III.1 Vauquois & Robert 87], et d'une seconde en Le_Lisp sur SM-90 par C. Boitet, D. Bachut et R. Gerber [III.1 Boitet et al. 86].

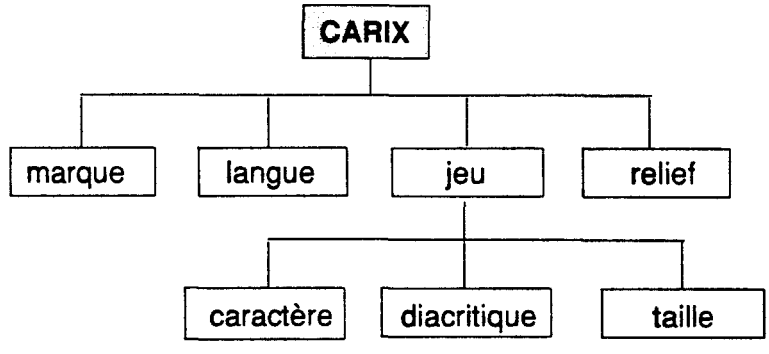



Figure 2.1 Structure interne d'un CARIX

### Implémentation de CARIX en Pascal

En Pascal, un CARIX est représenté sur un mot de 32 bits divisés en champs correspondant aux attributs définis. Ainsi, chaque CARIX se voit comme un entier long du type prédéfini integer. Voici la disposition des attributs :

*marque* : 1 bit
*langue* : 6 bits
*jeu* : 5 bits
*caractère* : 16 bits
(*caractère* + *diacritique* + *taille* : 7 bits + 6 bits + 1 bit
idéogramme : 16 bits
etc.)
*relief* : 3 bits
(*soulignement* + *gras* + *italique* : 1 bit + 1 bit + 1 bit)

**Exemples** : Le caractère $\hat{A}$, A avec circonflexe (code 3) majuscule italique en français (code 1) du jeu romain (code 4), est représenté par les champs suivants :

| 0 | 000001 | 00100 | 0001000001 | 000011 | 1 | 001 |
|---|---|---|---|---|---|---|
| marque | langue | jeu | caractère | diacritique | taille | relief |

Figure 2.2 Un CARIX dans un mot de 32 bits en Pascal

On constate que la représentation d'un CARIX en Pascal ne donne pas la lisibilité, mais permet un coût faible en taille mémoire.

### Implémentation de CARIX en Le_Lisp

En Le_Lisp, on associe à chaque CARIX un symbole Lisp représenté mnémoniquement sur au moins 13 caractères ASCII. À ce symbole est attachée une représentation interne qui est une liste de propriétés (ou d'attributs CARIX). La figure suivante présente la structure d'un symbole.

| 0 | 1 | 4 | 7 | 8 | 9 | 10 | 11 | 12 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| I/- | LLL | JJJ | U/L/- | I/- | G/- | S/- | < | CCC | DDD | |
| marque | langue | jeu | taille | ital. | gras | soul. | | carac. | diacr. | |

Figure 2.3 Structure d'un symbole CARIX en Le_Lisp

Dans la figure 2.3 :

/ signifie "ou" indiquant un choix alternatif ; par exemple, si la valeur de l'attribut *marque* est `I`, ce symbole est un identificateur d'un langage

spécifique aux transcriptions définies, au contraire, sa valeur est - si ce n'est pas le cas ;

LLL indique l'une des 55 langues naturelles prévue (voir Annexe 1) ;

JJJ indique un jeu de caractères correspondant à la langue ;

< est un séparateur pour la lisibilité ;

CCC et DDD indique le nom (de longueur de 1 à 3) du caractère et le code (de longueur de 1 à 4) de son diacritique, si c'est le cas.

**Exemples :**

| | |
|---|---|
| `-FRAROMUI--<A--3` | représente le caractère majuscule italique *Â* du jeu romain, écrit en français ; |
| `-FRASPE--G-<%` | représente le caractère gras % du jeu spécial ; |
| `-FRAGRCL---<A--2` | représente le caractère minuscule "alpha" α du jeu grec, écrit en français. |

Avec cette technique, on a implémenté un certain nombre de manipulations sur les CARIX, comme la gestion de l'environnement des CARIX, la construction des chaînes avec calcul de la longueur, la comparaison, la coupure des sous-chaînes, le tri alphabétique, et l'adaptation du traitement de texte en CARIX [III.1 Phan 88] à l'aide d'un éditeur.

### Éditeur PÉPÉ

PÉPÉ, qui est un éditeur minimum EMACS de Le_Lisp version 15.2 [IV Chailloux 86], permet d'éditer des fichiers ou n'importe quelle expression Lisp [IV Meyrowitz & van Dam 82]. En modifiant PÉPÉ, j'ai pu fournir un environnement de manipulation des CARIX avec l'entrée-sortie sur un terminal virtuel de Le_Lisp [III.1 Phan 88]. L'éditeur permet la saisie et l'affichage des CARIX sous forme de caractères diacrités, de ligatures et de commandes spécifiques aux transcriptions retenues.

Une première application de cette extension de PÉPÉ a été le traitement de textes vietnamiens sur SM-90.

### Performances de CARIX

L'implémentation des CARIX offre plusieurs avantages pour le TALN. D'abord, toutes les informations linguistiques sont complètement portées par chaque CARIX. Ensuite, on arrive à un système de codage universel de l'ensemble des caractères de toutes les langues naturelles. Enfin, on peut appliquer les CARIX dans un STTM.

Cependant, la réalisation pratique des CARIX nécessite l'utilisation d'un langage de programmation de bas niveau pour obtenir des temps d'exécution et des tailles mémoires acceptables. D'autres problèmes se posent :

1. stockage des CARIX sur périphérique (bandes, disques) ;
2. définition d'une méthodologie générique d'utilisation des CARIX et de leurs fonctions d'accès en vue du multilinguisme ;
3. possibilité de rendre multilingues des applications classiques existantes (compilateurs, éditeurs, etc.).

# 2 Saisie

Le problème suivant du codage est la conception des logiciels de saisie qui permettent d'entrer des caractères sur le dispositif d'entrée. À partir de la représentation interne d'un caractère, on construit une entrée, ou forme de saisie, pour l'obtention d'une correspondance. Cette correspondance n'est pas biunivoque, car une entrée ne donne qu'une représentation interne ; celle-ci peut, en revanche, avoir plusieurs entrées différentes (dans le cas où le dispositif d'entrée est un clavier "minimal"). La construction d'une entrée est basée sur la désignation d'un type de clavier et sur l'utilisation manuelle de ses touches.

Dans les systèmes non-interactifs, la saisie d'un texte est effectuée usuellement sur les claviers normalisés en alphabet latin. Sauf certaines touches de contrôle, ces claviers disposent des touches fréquentes dont chacune a un code propre. Suivant le cas, ce code de touche peut être différent du code interne transmis. Sur le clavier, on trouve les majuscules et minuscules, les chiffres, les signes de ponctuation et les opérateurs mathématiques usuels. Les signes moins fréquents, comme @, #, $..., ne sont souvent pas normalisés et leur disposition varie suivant les claviers.

Ainsi, on effectue la saisie directe pour entrer les caractères disponibles sur les claviers normalisés ; pour entrer d'autres caractères, latins ou non latins, on peut utiliser la méthode de combinaison : l'entrée d'un caractère se compose maintenant d'une suite de caractères disponibles. Ce principe peut évidemment être applicable à la saisie multilingue dans les systèmes interactifs à condition de l'appliquer avec simplicité et efficacité.

Dans l'application multilingue, la saisie directe donne lieu à l'interprétation des touches. Cependant, cette méthode n'est pas utilisable avec certains alphabets qui emploient plusieurs signes diacritiques (vietnamien, thaï, etc.). Pour ceux-là, on utilise la méthode de saisie par composition de touches.

Il est souhaitable de développer et d'adapter plusieurs méthodes de saisie pour être utilisées simultanément au cours de saisie d'un texte multilingue. On arrive à d'autres méthodes : saisie par transcription et saisie par choix. La dernière est spécifique aux systèmes interactifs.

## 2.1 Saisie par composition de touches

Cette méthode est bien adaptée pour entrer au clavier des caractères accentués ou avec signes diacritiques. Suivant la décomposition graphique du caractère, de la forme `lettre + signe`, on a une suite de frappes. Les lettres et les signes diacritiques, comme circonflexe (^), l'apostrophe gauche (‘), l'apostrophe droite (’), etc., sont, soit disponibles sur le clavier, soit créés par composition des touches. Dans le dernier cas, les lettres elles-mêmes peuvent être entrées par interprétation des touches et les signes diacritiques par composition de touches. Par exemple, la composition d'un signe diacritique se fait en combinant une touche de contrôle et de la touche portant ce diacritique (ou un autre signe).

Selon l'ordre de frappe, on distingue deux types de composition, à gauche et à droite. Dans la composition à gauche, les diacritiques sont entrés avant la lettre. La frappe de la lettre termine l'entrée et le résultat est affiché directement sur l'écran pour les systèmes interactifs. Si un caractère porte plusieurs signes diacritiques, l'ordre d'insertion de ces signes peut être quelconque ou fixé en fonction de l'ordre d'écriture usuel.

En revanche, dans la composition à droite, la lettre est entrée avant les diacritiques. Selon les signes possibles on a des caractères différents affichés jusqu'au moment de terminaison de l'entrée.

## 2.2 Saisie par transcription

Pour les langues non latines, deux avantages frappants de la méthode de saisie par transcription sont la simplicité et la non-injectivité. Si la transcription est portable, le travail de programmation est simplifié. La non-injectivité, par contre, est importante. En effet, on peut fort bien désirer avoir plusieurs transcriptions possibles pour un même caractère : c'est le cas des idéogrammes en chinois, en coréen, ou en japonais, ou de façon à tenir compte de la prononciation, qui varie souvent selon le contexte (début, milieu, fin).

L'inconvénient de la méthode est que selon la transcription utilisée, le nombre de frappes peut être considérable et l'utilisateur rencontre parfois des difficultés de manipulation des expressions complexes de transcription.

**Exemple** : On utilise la transcription minimale pour la saisie :

| *Entrée :* | *Affichage :* | *Entrée :* | *Affichage :* |
|---|---|---|---|
| A!2 | À | a!2 | à |
| C!5 | Ç | c!5 | ç |
| E!1 | É | e!1 | é |
| E!4 | Ë | e!4 | ë |
| etc. | | | |

La transcription des signes diacritiques latins est donnée dans l'annexe.

## 2.3 Saisie par choix

Pour de grands jeux de caractères, la méthode de saisie par choix est en fait mixte : frappe au clavier d'une l'entrée définissant un sous-ensemble de caractères, puis sélection sur l'écran du caractère voulu. L'équilibre de ces méthodes va de 0% de frappe (en utilisant la souris) et 100% de sélection à 100% de frappe et 0% de sélection.

Ainsi, en cours de saisie, on peut disposer de tableaux et/ou de claviers virtuels (images de claviers physiques) affichés sur l'écran. Suivant le cas, l'utilisateur peut prendre des informations nécessaires sur les tableaux pour manipuler correctement les entrées. Si une entrée correspond à plusieurs caractères, ces caractères sont affichés temporairement sur un tableau. Puis, l'utilisateur peut désigner, soit par la souris, soit par une touche de sélection, un caractère convenable sur ce tableau. Il peut également ne choisir qu'un caractère sur un clavier virtuel. Par exemple, le système Star de Xerox utilise un clavier virtuel représentant le clavier physique pour manipuler l'alphabet arabe.

La saisie par choix contient donc par les trois étapes suivantes :

*entrée* — L'utilisateur entre une entrée qui est, soit une suite de frappes sur le clavier physique, soit en cliquant sur le clavier virtuel par un bouton de la souris, soit un élément de tablettes.

*renvoi* — Après la fin d'une entrée, les caractères correspondants sont affichés sur un tableau prévu. Il y a deux types de fin d'entrée : soit la longueur de l'entrée atteint une borne fixée (à l'avance), soit l'utilisateur entre un signe particulier (comme un blanc).

*sélection* — À l'aide du tableau intermédiaire ou du clavier virtuel, l'utilisateur choisit le caractère.

Pour une bonne visualisation des différents types de caractères, en particulier chinois, la saisie par choix demande éventuellement des écrans graphiques à haute résolution. La figure 2.4 explique comment une expression d'entrée peut être interprétée pour donner le caractère voulu.

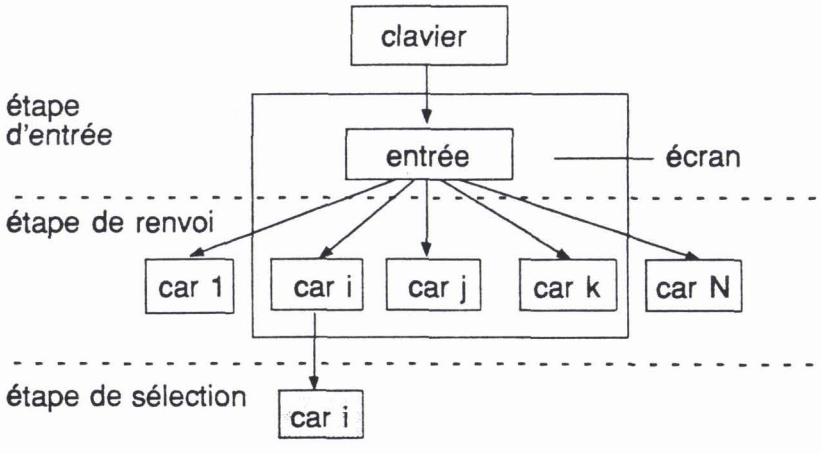



Figure 2.4 : Trois étapes d'interprétation d'une entrée

# 3 Restitution

La restitution (sur l'écran ou sur le papier) d'un texte multilingue codé et mémorisé pose des problèmes très difficiles et imparfaitement résolus à l'heure actuelle. Bien que chaque problème soit résolu partiellement, on n'arrive pas à regrouper toutes les solutions partielles dans un outil de traitement de texte multilingue.

Il y a deux grandes difficultés. La première est l'homogénéité : il n'existe pas en effet de correspondance biunivoque entre la forme de représentation interne et l'image restituée d'un caractère. Un caractère, qui a une représentation interne unique, peut avoir plusieurs images différentes.

La seconde est la compatibilité : pour avoir exactement le même dessin à l'imprimante qu'à l'écran, on doit traiter différemment. Il arrive souvent qu'un texte d'un certain alphabet puisse être affiché immédiatement sur l'écran à l'aide des polices disponibles, mais qu'on n'ait aucun moyen de l'imprimer, par manque de polices sur l'imprimante.

Dans ce qui suit, nous présentons trois problèmes principaux de restitution multilingue : la création des ressources de restitution, la composition des caractères, et la mise en lignes et en paragraphes.

## 3.1 Création des ressources de restitution

La restitution multilingue nécessite des ressources qui sont des polices de caractères. Une police, ou fonte ("Font" en anglais), est un ensemble d'images de caractères. Chaque image du caractère utilise un numéro jouant le rôle d'adresse de ce caractère. Il en résulte que tous les aspects de la restitution d'un texte multilingue dépendent strictement des ressources disponibles et des traitements correspondants.

Dans l'environnement X Window, une police est désignée par un nom générique et des propriétés typographiques [III.3 Dardailler 89]. Le protocole de nommage de polices est devenu partie intégrante et sert un but multiple : la description unique, l'indépendance du matériel utilisé et la généricité (possibilité d'extension, encodage multiple...) pour la distribution.

**Exemple** : Le nom de police

```
-adobe-courier-bold-o-normal--10-100-75-75-m-60-iso8859-1
```

désigne la police Courier normale des caractères latins gras (norme ISO8859), digitalisée par Adobe en corps 8 pixels, 75 dpi (approximativement égale au nombre de points typographiques par pouce, 72), etc.

On peut utiliser l'alias pour sélectionner une police dans l'ensemble présent suivant l'application.

Dans une police, un caractère est défini par son image (le dessin à afficher) et par sa métrique (les mesures permettant de le positionner par rapport aux autres). Il y a deux formats de définition de l'image : bitmap ("Bitmap" en anglais pour "matrice de points") et contour.

En format bitmap, l'image d'un caractère est formée par des points auxquels sont associés des valeurs suivant la couleur (noirs et blanc par exemple) dans une grille rectangulaire. Il existe actuellement plusieurs méthodes de production de polices selon ce format, comme la création avec un éditeur de polices [III.3 Dardailler 89b], la génération de polices par numérisation d'images de caractères, ou l'utilisation de PostScript [III.3 Adobe 85, 87].

En format contour, l'image d'un caractère est déterminée en fixant des points de contrôle et des paramètres de courbes mathématiques. Plusieurs recherches pour la génération de polices ont été menées avec METAFONT [III.3 Knuth 86], Pandora (N. Billawala), technique de l'intélligence artificielle DAFFODIL (M. Nanard et al.)... [III.3 André & Mlouka 87]). D'ailleurs, PostScript donne aussi un moyen de création des polices en format contour.

Il y a trois difficultés pour enrichir les ressources de restitution :

1. diversité des propriétés typographiques ;
2. nombre de caractères dans une police (très nombreux, comme le chinois, le japonais,...) ;
3. ligature et/ou variation morphologique des caractères selon leur position dans les mots (caractères arabes, grecs, etc.).

## 3.2 Composition des caractères

Il y a trois problèmes concernant la composition des caractères. Le premier est le traitement des règles pour former les mots et les phrases. C'est en fait le découpage du texte en unités textuelles de plus bas niveau (et donc le regroupement d'ensembles de caractères). Ces unités sont de plusieurs natures : grammaticales ("mot"), syntaxiques... Dans la formation d'un mot, selon le contexte de restitution, on peut rencontrer des variations morphologiques, des ligatures et le crénage (pour évider la partie de l'œil d'une lettre débordant le corps, comme AV, VA, AY...).

Le problème suivant est la césure, ou la coupure des mots en fin de ligne. Certaines langues n'ont pas de césure, comme des langues asiatiques. La césure peut être effectuée automatiquement au niveau du paragraphe, ou pour tout le document. Mais on peut également désactiver la césure de certains mots dans un paragraphe, ou la placer à un endroit précis dans un mot. La résolution de coupure constitue une application particulière en TALN.

Il y a encore un problème concernant la typographie fine qui donne des règles de mise des signes de ponctuation. Ces règles diffèrent selon les langues. Par exemple, la façon d'écrire des majuscules en bout de ligne et des noms particuliers, ou d'utiliser un blanc, ou non, avant et/ou après les signes point, virgule, point virgule, point d'interrogation, etc. Cela sert à corriger des fautes orthographiques et à effectuer une bonne typographie automatiquement.

## 3.3 Mise en lignes et en paragraphes

### Diversité de l'écriture

Les langues naturelles s'écrivent de différentes manières. Voici les sortes de mise en lignes fréquentes qu'on peut retenir :

* GDHB : le texte s'écrit de gauche à droite et de haut en bas. Cet ordre est très usuel dans l'écriture de la plupart des langues, comme les langues utilisant l'alphabet latin (y compris le vietnamien), le chinois et le japonais actuels, etc. ;
* BHGD : le texte s'écrit de bas en haut et de gauche à droite ou en diagonale (particulièrement dans les tableaux, schémas, figures...) ;
* DGHB : le texte s'écrit de droite à gauche et de haut en bas (arabe, hébreu, farsi, hindi...) ;
* HBDG : le texte s'écrit de haut en bas et droite à gauche (chinois, japonais traditionnels...) ;
* Mélange des formes DGHB et GDHB.

De plus, il existe des lignes dont le texte est écrit le long d'une courbe, en rotation... avec les sortes de mise en ligne retenues. Cela se rencontre dans l'écriture ancienne de certaines langues ou actuellement dans la décoration, la publication, etc.

On constate que si un texte écrit par la combinaison de trois directions GD, DG et HB (texte tri-directionnel) est possible, chaque insertion de deux directions (des textes bi-directionnels GDDG, GDHB et DGHB) est également possible. En utilisant `troff`, un logiciel de mise en page sous UNIX, Z. Becker et D. Berry dans [III.6 Becker & Berry 89] ont développé un outil, appelé `triroff`, pour formater d'un texte tri-directionnel.

### Fractionnement d'un paragraphe

Ce traitement est effectué par les éléments de base qui sont des boîtes. Une boîte est une surface rectangulaie englobant l'élément d'une police de caractères sur le support d'affichage. On suppose que les côtés de la boîte sont normalement invisibles et parallèles aux bords de l'écran, ou de la feuille de papier (on ne regarde pas encore ici quelles solutions seront utilisables dans le cas contraire).

Par exemple, dans TEX, une boîte est désignée simplement par sa dimension (largeur, hauteur et profondeur) et par sa position (point de référence). Dans Grif, une boîte est délimitée par ses quatre côtés (gauche, droite, supérieur et inférieur), et décrite par son origine (repérée par les coordonnées du coin gauche supérieur), sa dimention (largeur, hauteur) et ses axes de référence (un horizontal et un vertical).

Dans l'affichage, suivant l'application, on utilise des métriques plus sophistiquées des polices [III.3 Dardailler 89], [III.3 André 89] et [IV Adobe 85].

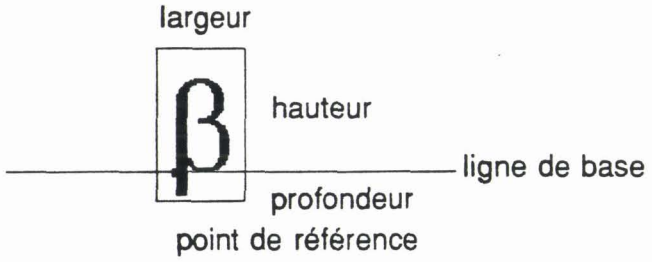



Figure 2.5 Désignation simple d'une boîte

Pour former un paragraphe quelconque, les boîtes de caractères sont placées en une liste, les unes à la suite des autres, de gauche à droite (direction GD), de droite à gauche (direction DG), ou de haut en bas (direction HB). Leurs points de référence sont ainsi alignés horizontalement, ou empilés verticalement. Le traitement est impossible dans le cas de mélange de textes horizontaux et de textes verticaux dans un même paragraphe.

Pour les paragraphes bi-directionnels (GDDG, GDHB et DGHB), les espaces entre deux boîtes adjacentes sont élastiques et leurs valeurs (en points typographiques) varient d'une certaine quantité de telle sorte que la liste horizontale de boîtes soit fractionnée en lignes physiques. Il y a deux modes de fractionnement, local, et global.

En mode local, le traitement se fait ligne par ligne. Lorsque la longueur de la ligne physique construite dépasse la largeur déterminée et fixée à l'avance, une nouvelle ligne est créée au-dessous (direction HB) pour les boîtes suivantes. C'est aussi à ce moment que le traitement de composition des caractères (coupure de mots, crénage...) est effectué en utilisant les règles grammaticales et typographiques. Ce mode est utilisable pendant la saisie.

En mode global (analogue à celui de TEX), le paragraphe entier est considéré en tenant compte de la composition des caractères (localiser les endroits possibles de coupure, par exemple). Le paragraphe est donc décomposé en lignes logiques d'une façon optimale. L'étape suivante est la composition des lignes logiques dans les lignes physiques. Ce mode ne peut être utilisé qu'à la fin de chaque paragraphe.

Pour les paragraphes verticaux (HBDG ou HBGD), il y a des problèmes difficiles concernant la pagination et le niveau de composition des caractères. Dans une fenêtre sur l'écran, ou pratiquement dans une feuille, au lieu de déterminer la longueur d'une ligne (ou largeur de paragraphe) et le nombre de colonnes d'affichage, on doit déterminer à l'avance la hauteur du paragraphe et le nombre effectif de paragraphes verticaux. Dans ce cas, l'interlignage est la juxtaposition des caractères pour former un mot, une phrase, et la juxtaposition des colonnes est vue comme l'interlignage.

Au niveau de composition des caractères, on peut avoir des colonnes de caractères ou des colonnes de mots. Donc on a le problème d'alignement à gauche, à droite, ou de centrage pour chaque colonne.

### Traitement des paragraphes

Le fractionnement des paragraphes concerne encore les problèmes du renfoncement, de l'interlignage, et de la pagination. Dans le contexte multilingue, chaque problème demande des solutions spécifiques à la langue traitée. En effet, un texte latin et un texte chinois n'ont pas le même renfoncement. Par exemple, la première ligne d'un paragraphe chinois commence éventuellement à partir du troisième caractère par rapport à la deuxième ligne.

Dans un paragraphe combiné de plusieurs écritures, l'interlignage dépend de la taille des caractères. En fait, la bonne présentation nécessite des ressources de polices convenables avec lesquelles le changement unique de langue ne donne pas des résultats inattendus (un caractère caché ou éloigné par l'autre) provenant de changement de taille. Par exemple, les polices par défaut des différentes langues utilisées doivent avoir la même taille relative.

Pour la pagination, il y a aussi deux problèmes dans la manipulation des paragraphes. Le premier est le contrôle des orphelins et des veuves d'un paragraphe, car parfois, l'utilisateur ne veut pas laisser en bas de la première page une première ligne d'un paragraphe (orphelin) dont le reste est en haut de la page suivante (veuve). Le second est le choix de placement d'un paragraphe entier avant ou après la saute de page, car ce contrôle permet d'éviter un changement de page entre certains paragraphes et de donner ainsi une présentation plus cohérente du document.

# 4 Dialogue

Au cours de la saisie et de la manipulation d'un document en mode interactif, on utilise très classiquement des menus et des boîtes de dialogue. Le seul problème apparent est de changer de langue. Il faut donc changer les messages ce qui est assez facile. Cela peut mener à changer la forme des menus et boîtes de dialogue. Certains messages peuvent être créés dynamiquement, il faut un mécanisme de substitution. En fin, certains messages ou menus peuvent être fabriqués à partir d'une description dans un certain métalangage (par exemple, des menus de Grif pour le type d'élément suivant à créer).

Le changement et la création dynamique des messages, de la forme des menus sont traitables de façon classique. C'est fait dans les logiciels existants du type multilingue "localisé". Par contre, la création de certains messages ou menus par l'utilisation d'un métalangage pose des problèmes plus profonds.

# Chapitre 3 : Problèmes spécifiques en édition de documents structurés



Imprimé par Grif le 20/3/91

# 1 Présentation

Dans notre objectif de conception d'un STTM, non seulement l'ensemble de problèmes posés par les aspects retenus (codage, saisie, restitution et dialogue) doit être résolu dans l'approche multilingue que nous avons analysée, mais également le système conçu doit être réalisé dans les deux approches, structurelle et interactive.

Au début des années 80, il n'existait pas encore, en effet, de systèmes offrant à la fois l'interactivité et la structuration, sauf dans quelques domaines particuliers, comme la conception assistée par ordinateur (CAO). Édimath, éditeur interactif de formules mathématiques, suivi de Grif, un éditeur interactif de documents structurés [III.2 Quint 87] ont été parmi les premiers travaux menés dans ce domaine.

L'approche structurelle, qui est apparue dans la fin des années 70 avec les systèmes comme Scribe, inaugure une nouvelle étape dans le traitement des documents complexes. Mais ce dernier système n'est pas interactif. De même, l'applicabilité de formateurs comme Troff, ou TEX de D. Knuth, bien connus dans les universités, est limitée, au moins dans le domaine de la bureautique, par la complexité d'utilisation et par la difficulté de représentation logique et physique de documents.

La tendance a donc été de décrire une représentation abstaite des documents traités. On trouve cette solution dans plusieurs systèmes disponibles actuellement pour le formatage de documents, comme Scribe, Mint, LATEX, et particulièrement, SGML (Standard Generalized Markup Language), et ODA (Office Document Architecture) présentés dans [III.2 Quint 87b].

Cependant, ces outils sont peu répandus sur le marché. Mis à part "The Publisher", l'utilisateur ne peut pas encore agir directement et interactivement sur une image du document de bonne qualité. De plus, selon les systèmes, la manipulation ou la modification des documents dans leur structure logique et leur présentation physique présente des limitations.

Par exemple, SGML, devenu une norme ISO, est un langage de marquage logique du document. Un document SGML est décrit comme une structure abstraite formée de plusieurs types d'objet divers : articles, chapitres, paragraphes, figures... Le créateur du document peut utiliser un simple éditeur de texte pour définir ces objets dans une entité particulière appelée DTD (Document Type Definition). La déclaration est lisible par l'utilisateur et traitée par des programmes spécifiques capables d'interpréter les balises SGML ainsi définies. Sur la même description, on peut exécuter plusieurs applications. Cependant, ce langage ne permet pas de décrire complètement des images et certains

objets. On ne trouve pas encore avec SGML des outils confortables pour créer des documents dans des formats plus génériques, ni la description physique des objets [III.2 André et al. 89] [III.2 Girard 89].

La représentation abstraite des documents a été développée d'une façon plus générique dans Édimath et dans Grif. Cette approche vise à construire un modèle de document par sa structure logique et par sa présentation physique. Le document et ses objets (section, sous-section, paragraphe, formule...) sont des entités et les objets sont reliés logiquement. L'édition d'un document est vue comme la manipulation sur un arbre abstrait qui représente l'organisation des objets définis.

Dans ce chapitre, nous présentons d'une façon générale les problèmes rencontrés dans l'approche structurelle et l'intérêt de travailler sur un éditeur interactif de ce type pour le multilinguisme.

# 2 Problèmes spécifiques

### Structure logique

La structure logique reflète l'organisation des composants du document. Comme il y a plusieurs catégories de composants et comme l'organisation de ces composants varie selon les besoins ou les idées de l'utilisateur, on introduit la notion de *classe de documents*. Une classe de documents est un ensemble de documents ayant des structures logiques semblables. On peut construire des classes de documents telles que les livres, les articles, les rapport scientifiques d'un laboratoire, les factures de vente d'une société, etc.

Une classe de document est caractérisée par une *structure générique*. Cette structure définit le mode de construction générale et hiérarchique des composants du document.

Par exemple, on construit une classe *rapport scientifique* en indiquant l'apparition successive des composants *titre, auteurs, résumé, mots-clés* et une *suite de sections*. Une *section* à son tour, comporte son *numéro*, un titre, une *suite* (peut-être vide) *de paragraphes* suivie d'une *suite* (peut être vide) *de sous-sections*, et ainsi de suite. Un *paragraphe* peut être une chaîne de caractères ou comporter d'autres composants hors-textes structurés, comme des images, des formules...

La figure 3.1 présente la structure générique de la classe *rapport scientifique*.

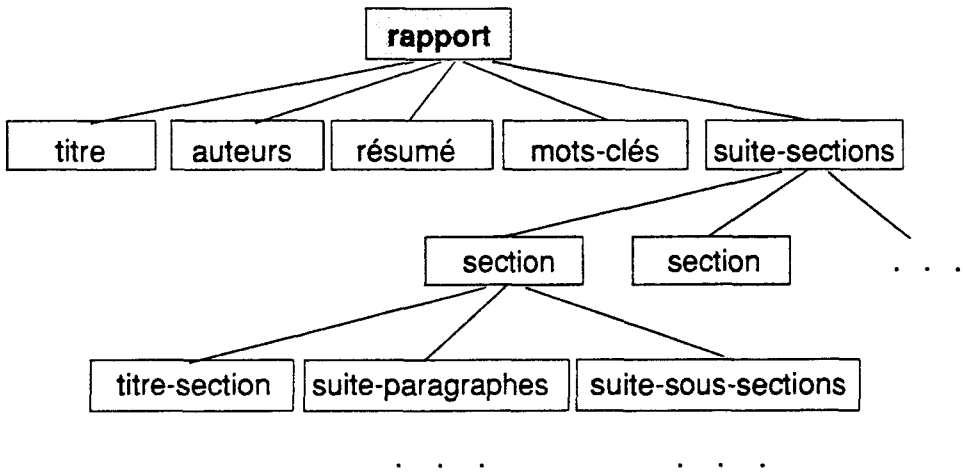



Figure 3.1 : Structure générique de la classe rapport scientifique

À partir de la structure générique, chaque document d'une classe est décrit par une *structure spécifique* qui organise les différents composants qui le constituent. Pour une même structure générique, chaque document a une structure spécifique différente comme dans l'exemple ci-desous des rapports A et B de la classe *rapport scientifique* :

rapport A

Titre
Auteurs
Résumé
Mots-clés
Section 1
Paragraphes
Sous-section1.1
Sous-section 1.2
Section 2
Paragraphes
Sous-section2.1
Sous-section 2.2
Sous-section 2.3

rapport B

Titre
Auteurs
Résumé
Mots-clés
Section 1
Sous-section1.1
Sous-section 1.2
Sous-section 1.3
Section 2
Paragraphes
Section 3
Paragraphes
Sous-section3.1
Sous-section 3.2
Sous-section 3.3

Figure 3.2 : Deux structures spécifiques de la classe rapport scientifique

Ainsi, la construction des structures spécifiques est déterminée précisément par la structure générique de la classe. On considère que la structure générique est le modèle d'une description abstraite pour une classe.

### Présentation physique

Après l'organisation logique de document, on définit une présentation physique pour la restitution. Cette présentation n'est pas unique, car on peut créer plusieurs présentations suivant les caractéristiques typographiques et les besoins esthétiques.

Ainsi, pour chaque composant défini dans la structure logique, on introduit un type de présentation. On peut distinguer un type de l'autre par l'organisation des composants : la taille (ou le corps), le style, la distance entre les caractères, le centrage ou le saut de page, etc. En modifiant ces facteurs, on peut créer différentes présentations. L'ensemble des règles qui permettent la description typographique des éléments définis par la structure logique générique forme la *présentation générique*.

À titre d'exemple, si l'on a des caractères de 6 tailles différentes (de 1 à 6) avec les styles romain, italique et gras, on peut définir la présentation d'un document dans la classe rapport scientifique comme suit : le titre est formé par les caractères romains de taille 4, les noms des auteurs sont en italiques de taille 2, le mot "Résumé :" en gras de taille 2, et le paragraphe précédé de ce mot ainsi que tous les paragraphes qui apparaissent dans le document sont aussi en romain de taille 2. On peut définir encore le numéro et le titre de section en romain de taille 3, le numéro et le titre de sous-section en gras de taille 2, etc.

Définir la présentation physique présente deux avantages. Le premier est l'homogénéité de présentation qui s'applique à tous les documents d'une classe et également à l'intérieur d'un document : les composants (titre, auteurs, résumé...) auront le même renforcement, tous les titres de section seront affichés de la même manière, etc. Le second est la facilité de changement de l'aspect graphique des documents de la classe. Dans ce cas, la modification de la seule présentation générique permet de modifier la présentation de tous les documents de cette classe.

### Documents et objets structurés

Par la complexité des documents qui contiennent non seulement le texte, mais également des hors-textes, le modèle retenu doit permettre de représenter différents objets. On arrive à définir la structure logique générique de tous les objets de la même façon qu'une classe de documents.

Cependant, les objets sont de natures différentes. Par exemple, l'ensemble des tableaux, des formules mathématiques, etc. est considéré comme un ensemble d'objets structurés, mais ce n'est pas le cas pour les dessins ou images. Il est nécessaire de définir une méta-structure générique non seulement pour les classes de documents, mais également pour les objets. Ainsi, selon la nature et le mode de construction à partir de la structure logique des objets considérés, on peut définir des classes d'objets convenables au même niveau de représentation logique.

Comme pour les documents, la définition de classes d'objets présente deux avantages. Le premier est la possibilité de présentation uniforme des objets de même nature et la possibilité de changement global de la présentation générique de tous les objets d'une classe donnée. Le second est l'indépendance entre la description du modèle de document (en aspect logique du document et des objets) et les règles de présentation générique.

# 3 Intérêt de travailler sur un éditeur structural multilingue

Si le Star de Xerox constitue un "documenteur" multilingue, on n'y trouve pas la possibilité de définir des classes de documents de façon générique, et de séparer une telle définition de celle d'une ou de plusieurs présentations. D'autre part, le problème du multilinguisme n'avait pas encore été abordé dans le cadre des éditeurs de documents structurés, comme Grif.

L'intérêt pratique de la "multilinguisation" de tels éditeurs est évident. Mais c'est surtout l'intérêt théorique qui a motivé notre étude. Il s'agit en effet de commencer à structurer les documents, non plus seulement au niveau formel, celui de leur structure, mais à celui de leur contenu. La première étape, objet de ce travail, ne va pas très loin dans l'analyse linguistique, mais constitue un préliminaire indispensable.

D'autre part, il est intéressant de voir comme "multilingualiser" des outils informatiques indépendamment de toute version spécifique du système d'exploitation, et sans attendre l'arrivée de systèmes d'exploitation réellement multilingues, et non pas "localisés". Peut être d'ailleurs les concepteurs de tels systèmes pourront-ils s'inspirer de certaines des techniques que nous avons développées dans le cadre de ce prototype.

# Partie B

# Une première approche et son application au vietnamien

Introduction

Chapitre 4 Grif : un éditeur de documents structurés

Chapitre 5 Multilinguisme par "remplissage des vides" : l'exemple du vietnamien

Chapitre 6 Évaluation de la méthode

Imprimé par Grif le 20/3/91

# Introduction

Dans la partie précédente, nous avons vu certain nombre de problèmes posés par les approches multilingue et structurelle. Maintenant, nous allons voir comment on a résolu les problèmes d'édition interactive de documents structurés dans Grif, avant de proposer des solutions possibles pour l'extension multilingue de cet éditeur.

Nous avons commencé par une étude sur le traitement des caractères diacrités. Cela est nécessaire, car il existe plusieurs langues naturelles dont l'écriture utilise des signes diacritiques. L'approche permet de résoudre rapidement et simplement les aspects spécifiques retenus du multilinguisme pour pouvoir aller plus loin dans la généricité et la globalité.

Selon Y. Haralambous [III.4 Haralambous 89], parmi 44 langues utilisant les caractères latins, y compris le vietnamien, il y a au total 190 caractères qui peuvent avoir des signes diacritiques. Sauf le vietnamien, dont un caractère peut comporter deux signes diacritiques, les caractères des autres langues ne portent qu'un seul signe. Ainsi, une adaptation de Grif à la langue vietnamienne est un bon exemple pour le traitement des caractères diacrités.

Dans cette application, comme l'alphabet vietnamien contient au total 178 signes typographiques, nous utilisons les codes ASCII étendus (sur 8 bits). Avec ce codage, nous pouvons appliquer et développer la méthode de saisie par composition des touches dans Grif. Il s'agit de construire des tables de données de saisie et d'imprimer un texte vietnamien avec un mélange de textes latin et grec en langage PostScript.

Nous remarquons que la création des polices des caractères diacrités vietnamiens pose les problèmes de restitution, notamment la mise en ligne et la justification, parce qu'un caractère diacrité vietnamien est simplement formé par l'ajout des signes diacritiques convenables à un caractère latin correspondant. Cela conduit à une différence inévitable des hauteurs des majuscules vietnamiennes et latines et de leurs polices correspondantes. Nous avons aussi la possibilité de traduire ces caractères vietnamiens dans un formateur connu, comme TEX.

L'organisation de la deuxième partie est la suivante : le chapitre 4 présente les caractéristiques de Grif sur lesquelles repose l'extension. Le chapitre 5 est consacré à l'application de Grif au vietnamien. Enfin, l'évaluation de la méthode est discutée dans le chapitre 6.

# Chapitre 4 : Grif, un éditeur de documents structurés

# 1 Le projet Grif

## 1.1 Présentation

Le projet Grif vise à construire un éditeur structuré dans un environnement interactif [III.3 Quint & Vatton 86] [III.3 Quint et al. 86]. Très largement paramétrable, l'éditeur Grif permet à l'utilisateur de définir de nouvelles classes de documents (livre, article, lettre, etc.), d'objets structurés (tableaux, schémas, formules mathématiques, etc.), ainsi que leur aspect de restitution dans un état de l'art de la typographie.

Grif fonctionne dans l'environnement X-Window, dont il utilise le système de multifenêtrage et la gestion des polices de caractères.

## 1.2 Modèle de document

Grif est fondé sur un modèle de très haut niveau : un document est représenté comme un arbre abstrait spécifié par sa structure logique et sa présentation physique. Grâce à l'approche par langages, la structure logique et la présentation physique sont décrites par le langage S et le langage P sous forme des *schémas*. Pour traduire un document Grif dans d'autres formalismes comme PostScript, TEX, Grif utilise un autre langage, le langage T, pour définir les règles de cette traduction.

À l'aide du modèle retenu, l'utilisateur de Grif peut travailler sur les composants du documents. Par exemple, les sections, les sous-sections, ou les figures, etc. sont numérotées, ou renumérotées automatiquement au moment de leur création, et elles sont organisées et présentées de la même façon. Ces composants héritent de leur modèle de définition un certain nombre de caractéristiques que l'utilisateur peut modifier interactivement. L'utilisateur peut également utiliser les produits de Grif dans les systèmes documentaires ou des bases de données.

En outre, le mécanisme de traduction peut être étendu pour transformer les documents dans différents formalismes, et obtenir ainsi la compatibilité avec des normes de représentation de documents, comme SGML, ou ODA.

## 1.3 Perspectives

Actuellement, en appuyant sur les résultats satisfaisants déjà acquis avec Grif et sa version industrielle développée par Gipsi SA, on continue des études fondamentales pour élaborer des concepts et des outils d'usage général.

Parmi plusieurs problèmes encore ouverts, on peut citer la production et l'échange de documents, l'introduction de l'image et du son (hypermédia), le génie éditorial

("document engineering" en anglais) [III.3 Quint et al. 90], les applications fondées sur les documents dans les aspects pluridisciplinaires, et bien sûr le multilinguisme.

# 2 Caractéristiques et architecture de Grif

## 2.1 Les schémas

Grif dispose de deux sortes de schémas différents pour décrire le modèle du document, schéma de structure en langage S et schéma de présentation en langage P, et d'une sorte de schéma de traduction en langage T. [III.3 Quint 87b]. Chaque schéma est écrit sous forme d'un programme. Pour sauvegarder un document édité, Grif utilise une représentation propre de ce document appelée la représentation pivot.

La description des langages de Grif utilise la grammaire du méta-langage M, dérivé de la forme de Backus-Naur (BNF).

**Exemple** : La grammaire du méta-langage M est spécifiée par une suite de règles. Chaque règle est constituée d'un symbole de la grammaire suivi du signe = et de la partie droite. Dans la partie droite, le formalisme utilise les conventions suivantes :

| | |
|---|---|
| ' ' | délimite les mots-clés du langage ; |
| [ ] | délimite une partie optionnelle ; |
| < > | délimite une répétition (de zéro à n fois) ; |
| / | délimite un choix ; |
| { } | délimite un commentaire ; |
| NAME | suite de lettres MAJ/min, de chiffres et du caractère de soulignement _, commençant par une lettre ; |
| STRING | suite de caractères quelconques délimités par des apostrophes ; |
| NUMBER | entier positif ou nul, sans signe, c'est-à-dire une suite de chiffres décimaux ; |

les lettres accentuées, dont les codes sont compris entre 1 et 37 en octal, sont écrites sous forme \nn ;

la juxtaposition des symboles indique la concaténation de ces symboles ;

la fin d'une règle est marquée par un point.

### Schéma de structure

Un schéma écrit en langage S décrit la structure générique d'une classe de documents ou d'objets. À partir de cette organisation logique, un document est représenté sous forme d'un arbre abstrait dont les feuilles représentent les types de base (texte, symbole mathématique, élément graphique...) et dont les nœuds correspondent aux objets complexes.

**Exemple** : La grammaire du langage S décrit la construction des types d'élément comme suit :

```
Constr = 'LIST' [ '[' min '..' max ']' ]
                                        'OF' '(' DefAvecAtt ')' /
        { constructeur liste constitué d'une liste d'éléments du même type }
        'BEGIN' SuiteDef 'END' /
        { constructeur agrégat constitué d'un ensemble d'éléments du type fixé }
        'CASE' 'OF' SuiteDef 'END /
        { constructeur choix choisit un type parmi l'ensemble de types possibles }
        'REFERENCE' '(' IdentElem ')' .
        { constructeur référence pour référer à un élément d'un type donné }
SuiteDef  = DefAvecAtt ';' < DefAvecAtt ';' > .
DefAvecAtt = Definition [ 'WITH' SuiteAttrFixes ] .
min = Entier / '*' .  { nombre minimum d'éléments d'une liste }
Entier = NUMBER .     { un entier sans signe }
etc.
```

Voici une partie du schéma de structure écrit en langage S pour la classe des rapports scientifiques :

```
Rapport = BEGIN  { type d'élément construit est Rapport }
     Titre = LIST OF (Unit\37 = UNIT)
     { liste d'une ou de plusieurs unités (texte, référence...) ;
      dans les polices utilisées par Grif, le code octal du caractère é est 37 }
     Auteurs = LIST OF (Auteur = BEGIN
           Nom = TEXT;
           Adresse = TEXT;
           END);
     R\37sum\37 = LIST OF (Paragraphe);
     Suite_section;
     END;
Suite_section = LIST [2..*] OF (Section); { au moins 2 sections }
etc.
```

## Schéma de présentation

Un schéma écrit en langage P définit une restitution physique, éventuellement parmi plusieurs, suivant le schéma de structure d'une classe de documents. Pour l'indépendance des matériels utilisés, on distingue deux niveaux de présentation : présentation abstraite qui peut être concrétisée sur des différents dispositifs et présentation générique qui décrit l'aspect physique d'une classe de documents ou d'objets.

L'homogénéité de la présentation des documents et des objets dans Grif est assurée par l'introduction de la notion de *boîte*. Il y a trois types de boîtes : boîtes associées aux éléments structurés du document, boîtes de présentation et boîtes de mise en page. L'organisation des boîtes pour afficher le document donne l'*image abstraite*.

On introduit aussi la notion de *vue* et pour chaque vue, on définit la *visibilité*. Les vues permettent à l'utilisateur de regarder à l'écran le contenu principal de son document et les éléments associés (table des matières, références, citations bibliographiques, etc.), chacune dans une fenêtre selon le niveau de la visibilité. Normalement, l'utilisateur travaille avec la vue principale, mais il peut en utiliser d'autres.

Parallèlement à la définition des pages et la façon de numéroter des éléments, on définit aussi les paramètres de présentation qui déterminent les *positions* et *dimensions* des boîtes, la *police*, le *style* et la *taille* de leurs contenus.

Exemple : Dans la grammaire du langage P, les règles de présentation d'un élément affiché dans les vues sont décrites ainsi :

```
SuiteRegleVues = 'BEGIN' < Regle > < RegleVue > 'END' ';' /
                 RegleVue / Regle .
RegleVue = 'IN' IdentVue SuiteRegles
SuiteRegles = 'BEGIN' Regle < Regle > 'END' ';' / Regle .
IdentVue = NAME .
Regle = ParamPres ';' / FonctPres ';' .
ParamPres = 'VertPos' ':' PosV /
            'HorizPos' ':' PosH /
            'Font' ':' NomHerit /
            'Style' ':' NomHerit /
            ... { autres paramètres de présentation }
```

À partir du schéma de structure, on écrit la présentation graphique du titre de la classe des rapports scientifiques par les règles suivantes :

```
Titre : { dans la vue principale }
    BEGIN
    Font : Times; { titre écrit dans la police Times }
    Style : Roman; { en romain }
    Size : Enclosing + 3; { taille par défaut de l'objet
                            contenant le titre + 3 }
    Width : Enclosing . Width * 80 %; { largeur de la boîte titre }
    VertPos : Top = Enclosing . Top + 1.5; { position verticale }
    HorizPos : VMiddle = Enclosing . VMiddle; { position horiz. }
    Line (VMiddle); { construction de ligne }
    Justify : No;   { pas de justification }
```

```
IN Table_Mati\11res; { titre écrit dans la vue "table des matières" }
        BEGIN
        Visibility : 3;     { niveau de la lisibilité }
        Size : Enclosing + 1; { plus petit que précédemment }
        VertPos : Top = Enclosing . Top + 0.5;
        END;
END;
```

On constate qu'il faut encore ajouter des méthodes de manipulation directe pour décrire l'aspect graphique des objets du document, plutôt que d'utiliser seulement un langage purement déclaratif.

### Schéma de traduction

Un schéma écrit en langage T définit le passage de la forme de représentation pivot d'un document de Grif à un autre formalisme tel que LaTeX, Scribe, Troff ou SGML.

**Exemple** : Dans la grammaire du langage T, les règles de traduction des caractères accentués sont décrites :

```
SchemaTrad =

      ... { autres règles de la grammaire }

      < 'TEXTTRANSLATE' SuiteTradTexte >

      ... { autres règles de la grammaire }

SuiteTradTexte = [ Alphabet ] SuiteTrad .
Alphabet = NAME . { alphabet est un nom }
SuiteTrad = 'BEGIN' < Traduction > 'END ';' / Traduction .
Traduction = Source [ '->' Cible ] ';' .
Source = STRING .
Cible = STRING .
```

Pour traduire les deux caractères accentués latins 'è' et 'é' dans leur représentation en LaTeX, on peut écrire dans un schéma de traduction les règles suivantes :

```
TEXTTRANSLATE Latin
     BEGIN
     '0\11' -> '\`{e}';   {pour è}
     '0\37' -> '\''{e}'; {pour é}
     END;
```

### Représentation pivot

La représentation pivot sert à stocker un document Grif dans un fichier. À partir de cette forme, on peut sortir un document par sa traduction, soit en un programme PostScript pour l'impression directe, soit en un formalisme donné.

Dans la représentation pivot, le contenu du document ainsi que sa structure arborescente des composants (arbre abstrait) sont décrits de façon linéaire. On utilise aussi la grammaire du méta-langage M pour cette description.

En effet, un document est représenté par la définition des élément structurés. Chaque élément commence par une indication de son type. Après, apparaissent ou non, suivant le cas, les composants de l'élément (propriété de référence, attributs, règles de présentation appliquées), le commentaire associé et le contenu.

Voici quelques règles de représentation pivot du document :

```
LeDocument = Element .
Element =
     TypeElement
     [ LabelReference ]
     < Attributs >
     < ReglePresentation >
     [ Commentaire ]
     [ Contenu ] .
```

etc.

La forme de représentation pivot est incompréhensible par l'utilisateur. Cependant, Grif permet de regarder l'organisation des composants du document sauvegardée dans des fichiers en mode "lister" : l'arbre abstrait, l'image abstaite, les textes des feuilles, etc.

## 2.2 Les phases de production de documents

On distingue trois phases différentes dans la production d'un document : compilation des schémas, édition et impression.

La phase de compilation traduit les schémas du modèle de document de Grif sous forme de tables de données utilisables par l'éditeur. Il y a trois sortes de tables : tables de structure, tables de présentation et tables de traduction. Pour la cohérence d'édition des documents produits, toutes les modifications des schémas entraînent leur recompilation.

Dans la phase d'édition, l'utilisateur crée et modifie interactivement son document en s'appuyant sur les schémas prédéfinis d'une classe de documents de Grif. Cette phase produit des fichiers sauvegardés sous forme de représentation pivot pour l'impression.

La troisième phase donne l'image souhaitée du document sur papier selon trois modes d'échange différents : exportation, importation et sortie directe. À l'aide du schéma de

traduction, qui détermine les formatages concernés, l'exportation permet de transformer la représentation pivot du document en description dans le langage d'entrée du formateur donné. En revanche, l'importation permet de lire un document externe sauvegardé dans un simple fichier ASCII, puis d'engendrer un document Grif composé d'une liste de paragraphes. Enfin, dans le même environnement que Grif, l'utilisateur peut obtenir directement un fichier PostScript pour l'imprimer.

Grif connait deux catégories d'utilisateurs : les superutilisateurs ou administrateurs et les utilisateurs finals. Les superutilisateurs s'occupent de créer et de modifier des schémas de documents pour la phase de compilation. En revanche, les utilisateurs finals ne peuvent que créer et modifier leurs documents en s'appuyant sur les schémas prédéfinis (phases édition et impression).

La figure 4.1 présente le schéma général de l'architecture de Grif avec les trois phases de production de documents :

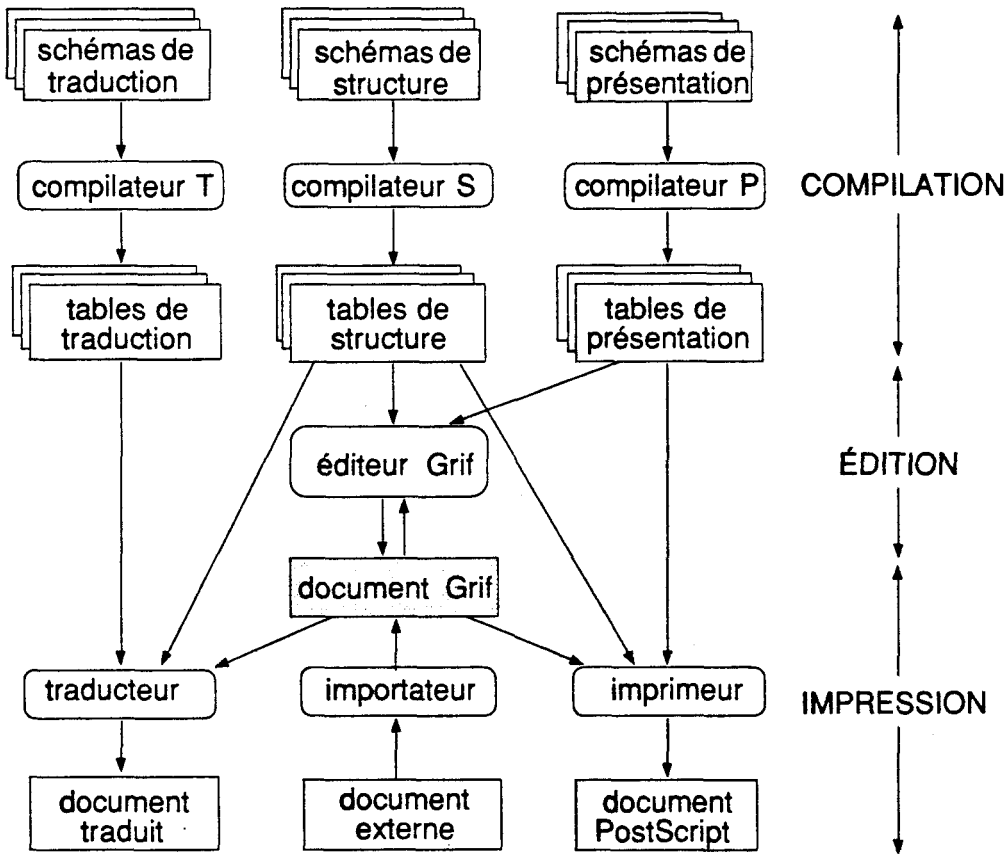



Figure 4.1 Architecture de Grif

## 2.3 Les composants logiciels

L'application Grif se compose de quatre composants logiciels essentiels : les compilateurs les traducteurs, l'`Éditeur` et le `Médiateur`.

Les compilateurs, qui compilent les schémas, fournissent les tables de données utilisées par l'`Éditeur` et les traducteurs.

Les traducteurs adaptent un document Grif (en format pivot) au langage d'entrée d'un formateur, ou lisent un fichier ASCII externe pour créer un document Grif.

L'`Éditeur`, guidé par les schémas de structure et de présentation, effectue les manipulations sur la structure logique du document et de ses objets dans un arbre abstrait, puis prépare l'affichage en décrivant l'image abstraite. Cette image définit un ensemble de contraintes de présentation pour chacun des éléments à afficher.

Le `Médiateur` gère l'interface utilisateur et l'affichage sur l'écran des images abstraites décrites par l'éditeur. Il fournit à l'éditeur des services d'entrée-sortie et de dialogue : saisie du contenu, gestion des fenêtres, des menus et des commandes, l'affichage des messages et des images de document.

L'`Éditeur` et le `Médiateur` réalisent séquentiellement tous les traitements. Ils communiquent entre eux par appels de procédures. Dans ce processus, les paramètres ainsi que les images abstraites sont partagés. Grif fonctionne en mode WYSIWYG : le texte et les éléments hors-texte sont entrés directement à leur place dans l'image concrète du document.

La figure 4.2 présente les structures de données pour le fonctionnement d'Éditeur et de Médiateur.

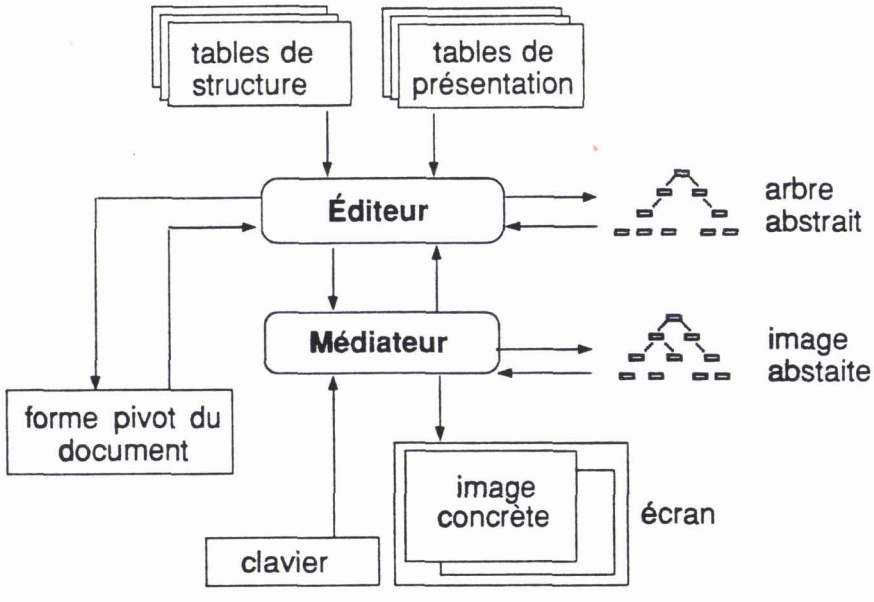



Figure 4.2 : Fonctionnement d'Éditeur et de Médiateur de Grif

# 3 Grif : un outil puissant pour la production de documents

## 3.1 Représentation des données

Grif utilise des codes ASCII de 7 bits, de 1 à 126 en décimal, pour représenter les types de base des composants du document (textes ou chaînes de caractères, éléments graphiques, symboles mathématiques et images). Les codes de contrôles étendus (8 bits), de 145 à 159 en décimal, sont réservés aux commandes et aux clés pour la saisie. La disposition des codes disponibles de Grif est présentée par les boîtes grises dans la figure 4.3 :

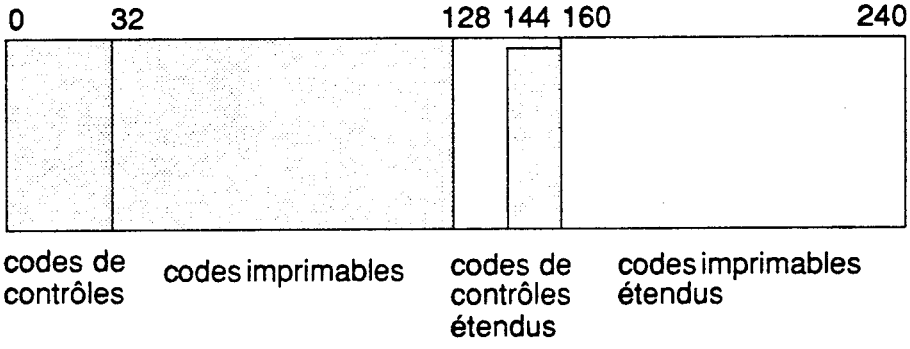



Figure 4.3 : Utilisation des codes ASCII dans Grif

Dans sa version prototype, Grif utilise deux alphabets (latin et grec) pour produire ses documents. L'alphabet latin contient tous les caractères ASCII imprimables (de 32 à 126 en décimal) et un ensemble de 28 caractères minuscule accentués dans la partie des codes de contrôles (de 1 à 31 en décimal). Ce sont : à, á, â, ä, å, æ, ç, è, ê, é, ë, ì, í, î, ï, ñ, ò, ó, ô, õ, ö, œ, ø, ù, ú, û, ü, ÿ.

L'alphabet grec ne contient que les caractères majuscules et minuscules donc sans diérésis, iota souscrit, ni anciens accents et esprits : Α α, Β β, Γ γ, Δ δ, Ε ε, Ζ ζ, Η η, Θ θ, Ι ι, Κ κ, Λ λ, Μ μ, Ν ν, Ξ ξ, Ο ο, Π π, Ρ ρ, Σ σ ς, Τ τ, Υ υ, Φ φ, Χ χ, Ψ ψ, Ω ω. L'alphabet grec contient aussi certains symboles mathématiques usuels dans la partie des codes de contrôles.

## 3.2 Saisie par codage du clavier

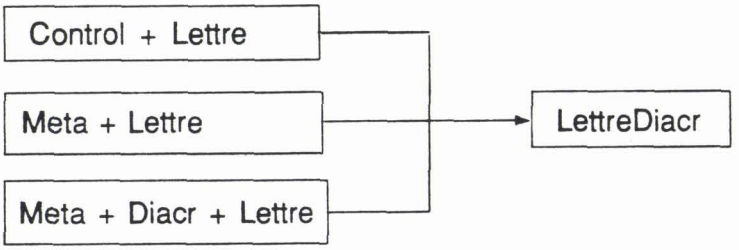
L'application actuelle Grif tourne sous Unix et sous X-Window. Ce système multifenêtre possède, entre autres caractéristiques intéressantes, celle de posséder un ensemble de touches virtuelles courantes (a, b, c...) et des touches `Control`, `Meta` (`Left et Right`), `Shift`, etc., et d'effectuer lui-même (par des tables) l'interface avec le matériel (Sun, IBM, Dec...). Une application (un "client" dans la terminologie

X-Window) n'a plus à se préoccuper du matériel (et des codes envoyés) et peut faire référence uniquement à des "touches virtuelles".

Dans la version que nous implémentons, Grif définit le codage des clés du clavier Sun sur une station Sun-3 sous X-Window V11R3. Il y a deux conventions retenues pour le codage des clés : traduction et composition. La traduction traduit les touches alphanumériques en codes ASCII correspondants et donc celles-ci sont interprétées de la façon habituelle. En revanche, la composition donne la correspondance entre les touches de fonction (`F1` à `F9`) et de contrôle (`Back Space`, `Control`, `Return`...) avec les clés de commande et de saisie.

Ainsi, Grif dispose de deux méthodes de saisie : saisie directe et saisie par composition des touches. Dans la méthode de saisie directe, les caractères alphanumériques du clavier sont entrés dans leur sens habituel. Mais si l'on veut insérer un caractère grec, on effectue d'abord une commande de changement d'alphabets, traitée comme un changement d'attributs, pour obtenir un clavier grec. L'étape suivante est l'entrée directe sur ce clavier du caractère grec voulu.

Dans la méthode de saisie par composition des touches, les caractères accentués sont insérés en utilisant une des trois manières présentées dans la figure 4.3. On utilise aussi cette méthode pour entrer les symboles mathématiques disponibles dont le code se trouve entre 1 et 31 en décimal.




Meta = Left | Right .
Diacr = Circonflexe | Grave | Aigu | Trema .
Lettre = a | c | e | i | n | o | u | y .

Figure 4.3 Trois méthodes de composition des touches

**Exemple :**

| *Entrée par composition (+) :* | *Affichage :* |
|---|---|
| `Control + a` | â |
| `Control + a` | signe ∪ (en alphabet grec) |
| `Meta + a` | â |
| `Meta + ^ + a` | â |
| `Meta + [` | é |
| `Meta + ' + e` | é |

## 3.3 Mise en page et polices de caractères

La mise en page d'un document Grif est définie par les schémas de présentation. L'utilisateur obtient une image identique sur papier et sur écran avec les aspects de restitution : alignement, justification, renforcement, interlignage. Les traitements plus fins, comme la coupure des mots, le crénage, l'insertion des images complexes, n'existent pas encore dans la version actuelle de Grif.

Pour l'affichage, Grif utilise des fichiers de polices de caractères dans le format SNF (Server Natural Format) de X-Window. Le nom de chaque fichier reflète la police selon l'alphabet (latin ou grec), la famille (Times, Helvetica, Courier), le style (romain, gras, italique) et la taille en valeur entière par des lettres symboliques. Par exemple, le fichier intitulé `lti13.snf` désigne la police de caractères en latin (`l`), utilisant la famille Times (`t`), avec le style italique (`i`) et de taille `13`.

En impression directe, Grif utilise les polices standard de PostScript pour restituer les caractères accentués. Ces caractères se trouvent dans la partie haute du code ASCII (code étendu sur 8 bits). Pour la compatibilité avec son codage interne, Grif utilise la méthode de modification du vecteur de codage de PostScript : les caractères accentués retenus sont remis, suivant à la police d'affichage, dans les emplacemets de rang 1 à 31 en décimal.

## 3.4 Interface utilisateur

L'utilisateur de Grif travaille interactivement à l'écran, grâce à des menus qui fournissent la liste des commandes à exécuter, les messages, ainsi que les informations d'entrée-sortie. Les menus sont affichés, soit en permanence pendant le processus de la création du document dans une fenêtre éditeur (voir figure 4.4), soit à l'initiative de l'éditeur ou à la demande. L'utilisateur peut sélectionner une entrée avec la souris, ou en utilisant le clavier.

La création de différents composants du document de Grif est faite automatiquement avec héritage. Selon les schémas de structure et de présentation prédéfinis, une fois qu'un nouveau document est créé, sur l'écran, sa vue principale est affichée sur une fenêtre sous forme de "squelette". Ce sont des boîtes grises représentant les éléments vides et organisées hiérarchiquement suivant la structure logique de document décrite dans le schéma. L'aspect physique de la présentation se trouve dans ces boîtes, c'est-à-dire qu'on dispose des boîtes correspondant aux règles de présentation dans le schéma concernant la dimention, la position, le déplacement ou la modification relative, ainsi que les attributs typographiques.

L'utilisateur n'a plus qu'à entrer le contenu de chaque composant : il sélectionne une boîte, ou un élément par la souris, puis entre le texte au clavier en remplissant partiellement les boîtes. Il peut à tout moment créer des composants supplémentaires (paragraphes, sections, sous-sections...) ou détruire des parties inutiles. Toutes ces modifications des composants sont prises en charge sur le "squelette" de façon cohérente : mise à jour des numéros, changement des renvois sur ces numéros, etc.

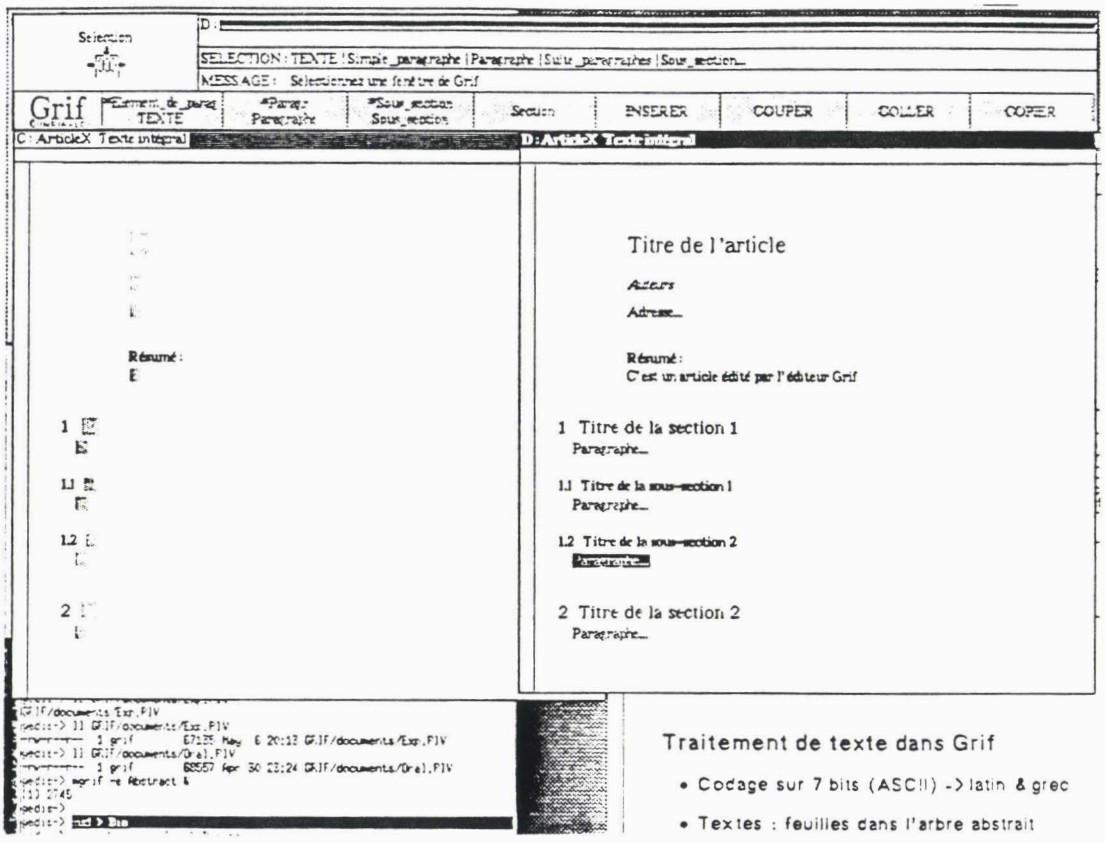



Figure 4.4 : Fenêtre éditeur de Grif

Pendant l'édition du document (par défaut en français), l'utilisateur peut travailler avec les caractères grecs par une commande de changement d'attributs. En effet, un alphabet est considéré comme un attribut. Ainsi, un alphabet, ou un attribut de la partie éventuellement textuelle sélectionnée par l'utilisateur (en inversion vidéo) est changé selon le choix. Les attributs possibles sont typographiques (famille, style et corps dans les polices de caractères).

La figure 4.5 présente un changement d'alphabet. Le morceau de texte en français en inversion vidéo ("texte" dans la boîte grise) est devenu en grec "τεξτε ".

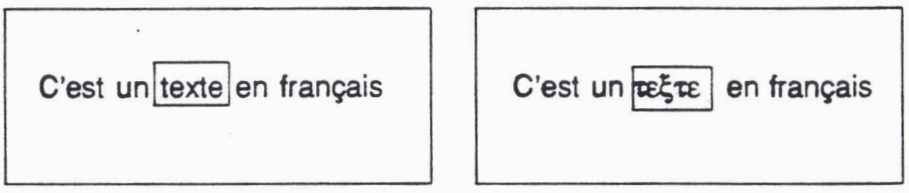



Figure 4.5 : Changement des alphabets

En ce qui concerne la recherche et le remplacement, le traitement les réalise sur les alphabets latin et grec en utilisant les codes ASCII. La méthode considère que les caractères *a* en latin et α en grec sont identiques dans la recherche ainsi que dans le remplacement, car ils ont le même code. Dans la sous-fenêtre de la commande de recherche, c'est le caractère *a* en latin qui est affiché, quel que soit l'alphabet recherché.

La recherche peut être effectuée sur un type donné de composants ou sur les références (par les renvois). À l'aide du parcours à travers toutes les vues, l'utilisateur peut créer, détruire ou déplacer n'importe quelle partie.

Le prototype de Grif existe en deux versions, anglaise et française (`Egrif` et `grif`). En version anglaise, les messages, les intitulés de menus, de commandes... dans les sous-fenêtres de dialogue sont en anglais.

# Chapitre 5 : Multilinguisme par "remplissage des vides", l'exemple du vietnamien



Imprimé par Grif le 20/3/91

# 1 Problème du vietnamien

## 1.1 Introduction à l'écriture vietnamienne

Le vietnamien s'écrit avec 17 consonnes simples et 12 voyelles. Voici l'alphabet dans l'ordre alphabétique des lettres :

| | | |
|---|---|---|
| consonnes | MAJ | *B C D Đ G H K L M N P Q R S T V X* |
| | min | *b c d đ g h k l m n p q r s t v x* |
| voyelles | MAJ | *A Ă Â E Ê I O Ô Ơ U Ư Y* |
| | min | *a ă â e ê i o ô ơ u ư y* |

Figure 5.1 : Alphabet vietnamien

L'alphabet contient aussi 11 consonnes composées : *CH ch, GH gh, GI gi, KH kh, NG ng, NGH ngh, NH nh, PH ph, QU qu, TH th, TR tr*. Il y a une consonne particulière *Đ đ* (D ou d barrée) et les voyelles accentuées *Ă ă* (A ou a brève), *Â â* (A ou a circonflexe), *Ê ê* (E ou e circonflexe), *Ô ô* (O ou o circonflexe), *Ơ ơ* (O ou o crochet) et *Ư ư* (U ou u crochet).

Le vietnamien possède six tons différents. Les tons se répartissent en deux registres, haut et bas. Suivant leurs inflexions, ils se divisent en égaux et obliques. Ces derniers se divisent aussi en brefs et longs. Chaque ton est désigné par un signe orthographique suivi d'un nom en vietnamien (voir figure 5.2). L'absence de signe indique le ton zéro.

*ngang* : zéro

` *huyền* : descendant (grave)

' *hỏi* : retombant (interrogation)

~ *ngã* : remontant (tilde)

´ *sắc* : montant (aigu)

. *nặng* : intensif (point au dessous)

| inflexion / registre | égal | oblique | |
|---|---|---|---|
| | | bref | long |
| haut | zéro | montant | retombant |
| bas | descendant | intensif | remontant |

Figure 5.2 : Tableau des tons en vietnamien

La langue vietnamienne est monosyllabique. La structure d'une syllabe est semblable à celle de la transcription pinyin du chinois (voir chapitre 9), mais une syllabe vietnamienne peut porter plus de tons qu'une syllabe chinoise (pékinoise). La syllabe est composée de

deux parties : des voyelles finales, comme *a, em, ông, uông...* éventuellement précédées de consonnes initiales, comme *b, c, ng, ngh, qu...* Pour une syllabe, les tons sont placés au dessus de la voyelle principale, sauf le ton intensif qui doit être en dessous.

**Exemple** : Un mot vietnamien peut comporter une seule syllabe, comme

| | | | |
|---|---|---|---|
| *Đây* | Ici | *Đẫy* | Gros |
| *Đầy* | Plein | *Đấy* | Là-bas |
| *Đầy* | Pousser | *Đậy* | Couvre |

ou plusieurs syllabes, comme :

| | |
|---|---|
| *Ngôn ngữ* | Langue |
| *Ngôn ngữ học* | Linguistique |
| *Nhà ngôn ngữ học* | Linguiste |

Dans l'implémentation, on considère deux catégories en vue du traitement : les *caractères simples* qui n'ont pas de signes diacritiques et les *caractères diacrités* qui peuvent avoir un ou deux signes diacritiques. Parmi ces derniers, on distingue les *caractères accentués,* qui existent déjà dans l'alphabet latin (à, â, ê, é...), et les *caractères de base,* qui peuvent être accentués ou diacrités.

**Note 1** : L'ancienne écriture vietnamienne, qui est d'origine chinoise, utilise les caractères "Nôm" (écriture démotique sino-vietnamienne). La transcription a été entreprise vers le dix-septième siècle par les missionnaires portugais. En 1651, le jésuite Alexandre de Rhode a publié le premier dictionnaire vietnamien à Rome. Depuis lors, le vietnamien, comme les autres langues, a connu de nombreux changements, et sa position internationale a été rehaussée.

**Note 2** : Les locuteurs du Nord Vietnam parlent avec six tons, tandis que les locuteurs du Sud n'en ont que cinq : le retombant et le remontant sont confondus. L'existence de deux dialectes seulement reflète l'homogénéité de la langue.

## 1.2 Intérêt du traitement informatique sur le vietnamien

Le vietnamien, avec plus de 60 millions de locuteurs, est une langue importante parmi les langues tonales asiatiques.

Le vietnamien est la seule langue utilisée au Viêt-nam, dans tous les domaines : culture, littérature, politique, économie et administration. Aujourd'hui, la langue dispose de plus en plus de termes convenables pour exprimer précisément les notions scientifiques.

Ainsi, le traitement informatique du vietnamien est tout à fait nécessaire. Il permettra d'ailleurs de promouvoir l'enrichissement terminologique et l'informatisation en général.

## 1.3 Travaux en cours

Actuellement, l'informatique au Viêt-nam n'est pas encore très développée.

En ce qui concerne le traitement de texte vietnamien, notamment dans le domaine de la PAO, il existe à l'heure actuelle des outils disponibles sur les PC utilisés au Viêt-nam et à l'étranger. Ce sont des systèmes étendus, comme Ventura, Word..., ou développés, comme VNew (qui fonctionne avec Ventura), VietStar...

On espère que dans un avenir proche, des études plus élaborées seront entreprises dans le domaine linguistique : dictionnaires électroniques, bases de données lexicales, TAO, etc.

Malgré certaines difficultés concernant les matériels disponibles, la limitation des contacts internationaux après les guerres successives, etc., la situation est en train de changer.

# 2 Principes de la solution

Dans cette première approche, nous prenons contact avec le code source de Grif et faisons une réalisation significative. Il s'agit d'un travail simple et rapide.

Le principe qui nous a guidé est d'intervenir localement et de se limiter à compléter les codes. Il est nécessaire de fournir une réalisation complète sur les aspects retenus (codage, saisie, restitution).

La limite acceptée de cette approche est de ne rien faire au niveau du dialogue. Notre extension au vietnamien sera appelée `Vgrif`.

## 2.1 Codes ASCII étendus

On a vu que, dans la version prototype de Grif, on utilise les codes ASCII standard pour les caractères latins, grecs (y compris les éléments graphiques et les symboles mathématiques dont chacun utilise un octet) et certains codes ASCII étendus pour représenter les commandes et les signes diacritiques. C'est à l'aide de cette représentation que Grif reconnaît exactement une touche frappée au clavier pour la traiter différemment. Il existe des codes étendus non utilisés et ceux-ci sont entièrement "vides".

Pour associer à chaque caractère vietnamien un code, nous prenons les codes étendus vides de Grif, d'où le nom de "méthode par remplissage des vides". Heureusement, il y a assez de vides pour coder entièrement tous les caractères. Nous arrivons donc à un système de codage utilisant les codes ASCII étendus avec les critères de codification suivants :

1. Codage de tous les caractères possibles, simples ou de base, accentués ou diacrités, en majuscules ou en majuscules ; on a donc 134 caractères et 9 signes simples à coder ;
2. Maintien des codes ASCII standard et des codes de commande de Grif ;
3. Utilisation du codage des polices standard en PostScript pour l'impression.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | À | Ã | Á | Â | È | É | Ê | Ì | Í | | Ò | Õ | Ó | Ô | Ù |
| 1 | Ú | à | ã | á | â | è | é | ê | ì | í | ò | õ | ó | ô | ù | ú |
| 2 | | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ‘ | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | |
| 8 | Ă | Ả | Ạ | Ă | Ẳ | Ẵ | Ắ | Ặ | À | Ẩ | Ẫ | Ấ | Ậ | Đ | Ẻ | Ẽ |
| 9 | | | | | | | | | | | | | ˆ | ´ | ` | ¨ |
| A | ˘ | ’ | ’ | ˜ | . | - | Ẹ | È | Ẻ | Ẽ | É | Ệ | Ỉ | Ĩ | Ị | Ỏ |
| B | Ọ | Ò | Ổ | Ỗ | Ố | Ộ | Ơ | Ờ | Ở | Ỡ | Ớ | Ợ | Ủ | Ũ | Ụ | Ư |
| C | Ừ | Ử | Ữ | Ứ | Ự | Ỳ | Ỷ | Ỹ | Ý | Ỵ | ă | ả | ạ | ă | ẳ | ẵ |
| D | ắ | ặ | à | ẩ | ẫ | ấ | ậ | đ | ẻ | ẽ | ẹ | è | ể | ễ | é | ệ |
| E | ỉ | ĩ | ị | ỏ | ọ | ò | ổ | õ | ó | ộ | ơ | ờ | ở | ỡ | ớ | ợ |
| F | | ủ | ũ | ụ | ư | ừ | ử | ữ | ứ | ự | ỳ | ỷ | ỹ | ý | y | |

codes ASCII standard

codes réservés strictement aux commandes de Grif

codes non utilisés

Figure 5.3 : Tableau des codes ASCII étendus pour le vietnamien

Dans l'organisation du codage, on dispose les codes de 1 à 31 en décimal (ou de 01 à 1F en hexadécimal) pour coder tous les caractères latins accentués qui existent dans la police standard en PostScript. Cette disposition permet de simplifier la création d'une nouvelle police qui sera utilisée pour l'impression. De 128 à 253 décimal (80 à FD hexadécimal), on code tous les caractères restants à raison d'un par code.

Le codage place les majuscules avant les minuscules, les caractères accentuées avant les caractères diacrités, les caractères diacrités dans l'ordre grave, interrogation, tilde, aigu et point au dessus, pour respecter l'ordre du dictionnaire [Il Lê & Nguyễn 89]

On a vu que, dans le code ASCII, la considération du 6-ième bit (ou une différence de 32 en décimal) permet la reconnaissance d'un caractère en majuscule ou en minuscule. Mais cela n'est pas généralisable aux caractères ajoutés par "remplissage des vides". Cela complique l'algorithme de recherche et de remplacement d'une chaîne de caractères avec ou non distinction MAJ/min utilisé dans Grif. On s'est borné à une répartition séparée des

codes majuscules et des codes minuscules. Par exemple, les codes majuscules se trouvent dans les intervalles (01,10), (41, 5A), (80, 8F) et (A6, C9) en hexadécimal.

## 2.2 Saisie par composition des touches

Nous avons développé la méthode de saisie par composition des touches déjà utilisée dans Grif. Un caractère accentué ou diacrité est décomposé en deux parties : le caractère de base et les signes diacritiques. Par exemple, le caractère accentué â est formé par le caractère de base a et le signe circonflexe ^ ; le caractère è est formé par le caractère de base e et les deux signes circonflexe ^ et grave `, etc.

À partir de cette décomposition, on construit une représentation interne sous forme de vecteurs et de tables.

D'une façon générique, on établit un vecteur de signes $s_1$, $s_2$..., $s_M$, appelé V1 et un vecteur de lettres $l_1$, $l_2$..., $l_K$, appelé V2. Chaque lettre de V2 peut prendre un ou deux signes diacritiques dans V1. Outre les M signes simples, on peut avoir N signes composés de forme ($s_i$, $s_j$). Les deux vecteurs sont triés alphabétiquement. Comme les signes simples et composés sont des caractères affichables, on ajoute le caractère espace dans V2 pour entrer ces signes en tant que tels.

On construit ensuite deux tables, dites table de vérification et table de consultation. La table de vérification, appelée T1, est une matrice carrée MxM dont chaque ligne et chaque colonne correspond à un signe dans V1. T1 donne l'autorisation de combinaison de deux signes, comme $s_i$ et $s_j$ dans la figure 5.4. Si la combinaison est possible, T1 donne une valeur différente de zéro. Cette valeur est celle de la ligne (i, j) de la table de consultation. Si le caractère entré n'a qu'un seul signe diacritique, T1 n'est pas prise en compte.

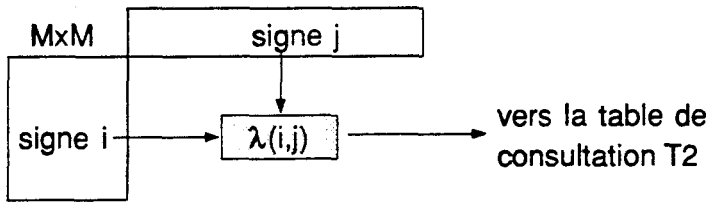



Figure 5.4 : Table de vérification T1

La table de consultation, appelée T2, donne le code du caractère à afficher quand l'entrée est bien terminée. T2 est une matrice (M+N)xK dont chaque colonne correspond à une lettre $l_k$ de V2 et chaque ligne correspond soit à un signe simple, ligne i, soit à un signe composé, ligne λ(i,j). La valeur repérée est ainsi une valeur c(i,k), ou c(λ(i,j),k), qui doit être différente de zéro. Elle se trouve à la ligne λ(i,j) déterminée par T1 décrite plus haut, et dans la colonne k déterminée par l'insertion de la lettre $l_k$. Si le caractère entré n'a qu'un seul signe diacritique $s_i$, seule T2 sera explorée pour avoir la valeur c(i,k).

Si V2 contient encore un blanc, la table de consultation dispose d'une autre colonne correspondant à ce blanc. Donc on a la valeur c(i) pour le signe simple ou c(i, j) pour le signe composé à afficher. La figure 5.5 ci-dessous présente l'organisation de T2.

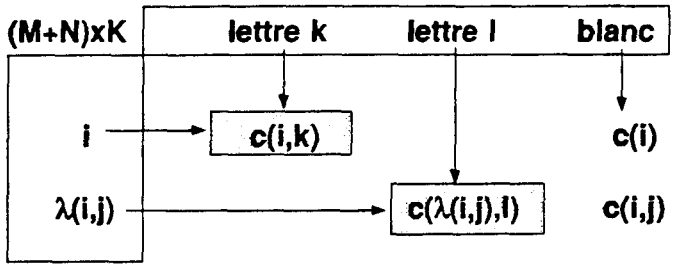



Figure 5.5 : Table de consultation T2

À l'aide de ces vecteurs et tables de données, on peut effectuer la saisie en composition à gauche ou en composition à droite. Pour les caractères qui portent deux signes diacritiques, l'ordre d'insertion de ces deux signes peut être quelconque. Afin de simplifier à l'implémentation, les deux types de composition ne peuvent pas être appliqués en même temps. Pour changer de mode pendant la saisie, le type de composition doit être désigé par une commande de bascule.

## 2.3 Création des polices

### Pour l'affichage

On peut utiliser deux outils interactifs : `Fontedit` sous le logiciel `SunView` [IV Sun 86] et `Xfedor`, un éditeur de polices [III.3 Dardailler 89b] dans l'environnement X-Window.

`Fontedit` est particulièrement adapté aux stations de travail Sun, pour créer des polices en format Vfont. Dans sa version prototype, Grif peut utiliser les polices de ce format pour afficher son texte (adapté au serveur X-Window version 10). Mais avec l'utilisation du format SNF (adapté au serveur X-Window version 11), il faut les traduire, d'abord en format BDF (pour "Bitmap Distribution Format"), puis en format SNF, par les commandes `vtobdf` et `bdftosnf` (voir figure 5.6).

L'éditeur `Xfedor` crée des polices en format BDF dans des fenêtres X. Comme `Fontedit`, l'utilisateur peut manipuler directement l'image par point de chaque caractère. Quand les ressources correspondantes sont établies correctement, on peut utiliser avec cohérence les algorithmes de mise en ligne et de justification de Grif.

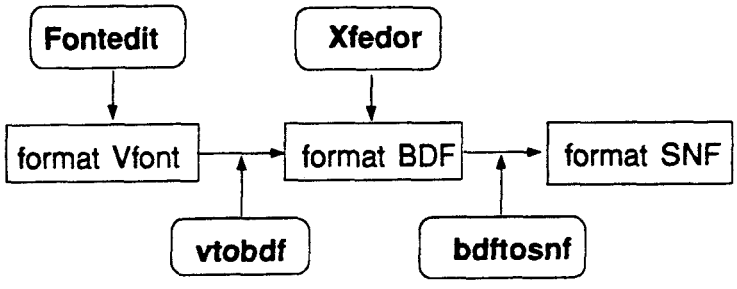



Figure 5.6 : Création des polices du vietnamien

Dans sa version prototype, les alphabets traitables par Grif (latin et grec) sont introduits à l'avance dans un fichier de données des menus. Il reste un problème pour insérer l'alphabet vietnamien dans cet ensemble. Une solution est d'ajouter "à la main" ce nom, et de corriger certains paramètres dans les logiciels de gestion des fenêtres.

Note : Dans sa version originale, `vtobdf` ne travaille qu'avec des codes sur 7 bits. Cela empêche de traduire les polices du vietnamien créées par `Fontedit` en format BDF. Nous y avons fait une extension pour l'adapter aux codes étendus.

### Pour l'impression directe en PostScript

La création des polices d'affichage ne permet pas encore de les utiliser dans l'impression directe d'un texte vietnamien. Le problème est la création dynamique de nouvelles polices de caractères diacrités en PostScript et la justification d'un texte complexe pouvant contenir plusieurs alphabets différents.

La première méthode proposée est la redéfinition des polices en latin existantes. En voici le principe : pour restituer les caractères diacrités qui n'existent pas dans l'ensemble des caractères imprimables, on peut restituer d'abord les caractères de base, puis les signes diacritiques, choisis parmi l'ensemble des signes spéciaux imprimables.

On peut également utiliser deux autres méthodes pour construire l'image d'un caractère : la *création d'une police en mode point* ou la *création d'une police vectorielle* ("outline") [III.3 Adobe 85, 87].

En mode point, les caractères dans la police sont crées par la description "bit-map" des images comme des données, puis ils sont dessinés par les sous-programmes.

Dans la méthode vectorielle, une nouvelle police est créée par la description géométrique du contour de ses caractères. Chaque caractère est décrit au moyen des opérateurs graphiques de PostScript.

Cependant, ces méthodes sont plus difficiles à réaliser en pratique que la première.

La création de polices en PostScript ne change pas beaucoup les algorithmes de justification d'un texte complexe. L'idée de la méthode est de calculer l'espace réel entre deux mots d'un texte donné séparés par un blanc. Sur la ligne à imprimer, cette valeur est déterminée par le nombre de blancs et la différence des longueurs : longueur effective du texte envoyé (composé des caractères différents du blanc) et longueur réservée à placer.

### Transcodage des caractères accentués

On a aussi la possibilité de traduire les caractères diacrités vietnamiens dans un formateur TEX. Il s'agit de récrire les schémas de traduction en langage T pour chaque classe de documents. Ainsi, il faut adapter au compilateur T le traitement des changements d'alphabet et le transcodage des caractères diacrités. Le problème est de trouver une méthode de transcodage.

Selon une solution proposée par B. E. Vogel [III.4 Vogel 89] qui est développée sur PC TEX, les caractères diacrités sont définis à l'aide des marques de signes attachées aux caractères de base. Par exemple, le caractère *â* est défini par `\ha`, le caractère *ă* est

défini par \ga, le caractère *à* est défini par \gha, où \h, \g et \gh sont des marques attachées au caractère *a*, etc. Si dans TEX, on définit :

```
\def\g{\'}
\def\h{\^}
\def\gh{\leavevmode\raise1.161ex\rlap{\kern.15em\accent"12}
        \raise1.465ex\rlap{\kern.1em\accent"5E }}
```

on peut écrire le transcodage au moyen d'un schéma de traduction en langage T :

```
TEXTTRANSLATE Vietnamien
   '\001' -> '\'{a}'; { pour à }
   '\004' -> '\^{a}'; { pour â }
   '\322' -> { pour à }
        '\leavevmode\raise1.161ex\rlap
          {\kern.15em\accent"12 }
        \raise1.465ex\rlap{\kern.1em\accent"5E }{a}'
   END
```

# 3 Implémentation

## 3.1 Contexte du travail

La réalisation a été divisée en deux parties : manipulation des caractères diacrités et restitution à l'écran et sur papier (en langage PostScript).

Pour la saisie, on n'a implémenté que la méthode de composition à gauche. Ce choix permet de simplifier le travail par l'utilisation du même algorithme de saisie de Grif. Enfin, on a voulu que, au cours de la saisie d'un caractère diacrité, l'utilisateur soit guidé par des messages indiquant ce qui a déjà été entré.

Pour la création des polices d'affichages, on a utilisé `Fontedit` pour ajouter et compléter les caractères diacrités vietnamiens à partir des polices existantes de Grif. On rappelle que c'est un gros travail de créer au total 3x3x6 = 54 fichiers de police de trois familles (Times, Helvetica, Courier) en trois styles (romain, gras, italique) et de six tailles différentes (9, 13, 17, 22, 33, 48 en pixel).

Ici, on rencontre une autre difficulté dans la mise en ligne des majuscules écrit en latin et en vietnamien. Puisqu'il n'existe pas de caractères majuscules latins accentués dans cette implémentation, il faut agrandir suffisamment la taille des polices correspondantes pour placer les caractères diacrités.

L'étape suivante est la création des polices en PostScript. On utilise la méthode de redéfinition des polices.

L'implémentation travaille principalement avec le médiateur et on fait fonctionner Grif sur une station Sun-3 sous Unix et sous X-Window V11R3.

## 3.2 Tables de données de saisie

La syntaxe de combinaison de gauche pour la saisie des caractères avec signes simples est la suivante :

```
Meta Signe Lettre
```

`Méta` représente la touche `Left` ou `Right` sur le clavier Sun ; `Signe` représente le type de signe à placer ; `Lettre` est le caractère de base sur laquelle on veut placer le signe.

Il s'agit de choisir les touches alphanumériques. On considère que le choix de celle-ci est libre, mais il vaut mieux choisir de façon à rappeler le signe à utiliser. Par exemple, la touche ? peut rappeler le signe *interrogation*, la touche ) rappelle le signe *crochet*, etc.

La table ci-dessous nous présente les touches utilisées pour représenter les signes simples. Par convention, on utilise les noms abrégés pour les vecteurs et tableaux de données.

| La touche : | nom abrégé : | représente le signe : |
|---|---|---|
| u | Br | *brève* |
| ^ | Ci | *circonflexe* |
| ) | Cr | *crochet* |
| ` | Gr | *grave* |
| ? | In | *interrogation* |
| ~ | Ti | *tilde* |
| ' | Ai | *aigu* |
| . | Po | *point au dessous* |
| - | Ba | *barre* |

Dans la définition, on a introduit le signe Ba pour traiter les caractères Đ et đ. Par exemple, pour entrer le caractère đ, on doit taper

```
Meta - d
```

De même, pour entrer le caractère Đ, on doit taper

```
Meta - D
```

La combinaison de deux signes simples donne un signe composé. La syntaxe est la suivante :

```
Signe1 Signe2 Lettre
```

`Signe1` et `Signe2` représentent les signes simples à placer ; `Lettre` est le caractère de base qu'on veut placer le signe composé.

Voici 15 compositions possibles dans la représentation de leur nom abrégé :

| Nom abrégé : | représente ce signe : | Nom abrégé : | représente ce signe : |
|---|---|---|---|
| BrGr | *brève-grave* | CiAi | *circonflexe-aigu* |
| BrIn | *brève-interrog.* | CiPo | *circonflexe-point* |
| BrTi | *brève-tilde* | CrGr | *crochet-grave* |
| BrAi | *brève-aigu* | CrIn | *crochet-interrog.* |
| BrPo | *brève-point* | CrTi | *crochet-tilde* |
| CiGr | *circonflexe-grave* | CrAi | *crochet-aigu* |
| CiIn | *circonflex-interrog.* | CrPo | *crochet-point* |
| CiTi | *circonflexe-tilde* | | |

Par exemple, pour entrer le caractère *ũ* qui contient un signe composé, on peut taper

```
Meta ) Meta ~ u
```

Mais on peut aussi taper

```
Meta ~ Meta ) u
```

Maintenant, on construit les vecteurs V1 de taille M = 8 et V2 de taille K = 15. Avec 15 signes composés plus le signe simple Ba, on a donc N = 16.

```
V1 = { Br,Ci,Cr,Gr,In,Ti,Ai,Po }
V2 = { A,D,E,I,O,U,Y,a,d,e,i,o,u,y,Blanc }
```

V2 contient les caractères D et d comme on l'a expliqué plus haut. On ajoute encore dans V2 un blanc pour entrer les signes diacritiques en tant que tels. En outre, il n'y a pas de signe Ba dans V1, car ce vecteur est construit pour vérifier seulement la possibilité de combinaison de deux signes diacritiques appartenant à une lettre. Les éléments de ces deux vecteurs sont indexés à partir de zéro.

Ensuite, on construit les tableaux T1 et T2. Le tableau de vérification T1 est constitué d'une matrice carrée MxM = 8x8 dont chaque ligne ou chaque colonne correspond à un élément dans le vecteur V1.

| | *Br* | *Ci* | *Cr* | *Gr* | *In* | *Ti* | *Ai* | *Po* |
|---|---|---|---|---|---|---|---|---|
| *Br* | 0 | 0 | 0 | 9 | 10 | 11 | 12 | 13 |
| *Ci* | 0 | 0 | 0 | 14 | 15 | 16 | 17 | 81 |
| *Cr* | 0 | 0 | 0 | 19 | 20 | 21 | 22 | 23 |
| *Gr* | 9 | 14 | 19 | 0 | 0 | 0 | 0 | 0 |
| *In* | 10 | 15 | 20 | 0 | 0 | 0 | 0 | 0 |
| *Ti* | 11 | 16 | 21 | 0 | 0 | 0 | 0 | 0 |
| *Ai* | 12 | 17 | 22 | 0 | 0 | 0 | 0 | 0 |
| *Po* | 13 | 18 | 23 | 0 | 0 | 0 | 0 | 0 |

Figure 5.7 : Tableau T1[8,8]

Le tableau de consultation T2 est une matrice (M+N)xK = 24x15 dont chaque colonne correspond à un élément dans le vecteur V2. La dernière colonne correspond au blanc et fournit seulement les codes des signes simples. Les huit premières lignes du T2 correspondent aux signes simples de V1. La neuvième ligne correspond au signe Ba. Les quinze dernières lignes correspondent aux signes composés.

| | *A* | *D* | *E* | *I* | *O* | *U* | *Y* | *a* | *d* | *e* | *i* | *o* | *u* | *y* | *bl* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Br* | 80 | 0 | 0 | 0 | 0 | 0 | 0 | CA | 0 | 0 | 0 | 0 | 0 | 0 | A0 |
| *Ci* | 4 | 0 | 7 | 0 | E | 0 | 0 | 14 | 0 | 17 | 0 | 1D | 0 | 0 | 9C |
| *Cr* | 0 | 0 | 0 | 0 | B6 | BF | 0 | 0 | 0 | 0 | 0 | EA | F3 | 0 | A1 |
| *Gr* | 1 | 0 | 5 | 8 | B | 15 | C5 | 11 | 0 | 15 | 18 | 1A | 1E | F9 | 9E |
| *In* | 81 | 0 | 8E | AC | AF | BC | C6 | CB | 0 | D8 | E0 | E3 | F0 | FA | A2 |
| *Ti* | 2 | 0 | 8F | AD | C | BD | C7 | 12 | 0 | D9 | E1 | 1B | F1 | FB | A3 |
| *Ai* | 3 | 0 | 6 | 9 | D | 10 | C8 | 13 | 0 | 16 | 19 | 1C | 1F | FC | 9D |
| *Po* | 82 | 0 | A6 | AE | B0 | BE | C9 | CC | 0 | DA | E2 | E4 | F2 | FD | A4 |
| *Ba* | 0 | 8D | 0 | 0 | 0 | 0 | 0 | 0 | D7 | 0 | 0 | 0 | 0 | 0 | A5 |
| *BrGr* | 83 | 0 | 0 | 0 | 0 | 0 | 0 | CD | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *BrIn* | 84 | 0 | 0 | 0 | 0 | 0 | 0 | CE | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *BrTi* | 85 | 0 | 0 | 0 | 0 | 0 | 0 | CF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *BrAi* | 86 | 0 | 0 | 0 | 0 | 0 | 0 | D0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *BrPo* | 87 | 0 | 0 | 0 | 0 | 0 | 0 | D1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *CiGr* | 88 | 0 | A7 | 0 | B1 | 0 | 0 | D2 | 0 | DB | 0 | E5 | 0 | 0 | 0 |
| *CiIn* | 89 | 0 | A8 | 0 | B2 | 0 | 0 | D3 | 0 | DC | 0 | E6 | 0 | 0 | 0 |
| *CiTi* | 8A | 0 | A9 | 0 | B3 | 0 | 0 | D4 | 0 | DD | 0 | E7 | 0 | 0 | 0 |
| *CiAi* | 8B | 0 | AA | 0 | B4 | 0 | 0 | D5 | 0 | DE | 0 | E8 | 0 | 0 | 0 |
| *CiPo* | 8C | 0 | AB | 0 | B5 | 0 | 0 | D6 | 0 | DF | 0 | E9 | 0 | 0 | 0 |
| *CrGr* | 0 | 0 | 0 | 0 | B7 | C0 | 0 | 0 | 0 | 0 | 0 | EB | F4 | 0 | 0 |
| *CrIn* | 0 | 0 | 0 | 0 | B8 | C1 | 0 | 0 | 0 | 0 | 0 | EC | F5 | 0 | 0 |
| *CrTi* | 0 | 0 | 0 | 0 | B9 | C2 | 0 | 0 | 0 | 0 | 0 | ED | F6 | 0 | 0 |
| *CrAi* | 0 | 0 | 0 | 0 | BA | C3 | 0 | 0 | 0 | 0 | 0 | EE | F7 | 0 | 0 |
| *CrPo* | 0 | 0 | 0 | 0 | BB | C4 | 0 | 0 | 0 | 0 | 0 | EF | F8 | 0 | 0 |

Figure 5.8 : Tableau T2[24,15]

**Exemple** : L'insertion de deux signes Ti et Cr est autorisée et T1 donne la valeur 21. La 21-ième ligne CrTi de T2 est donc consultée. Grâce au vecteur V2, l'entrée suivante de la lettre u détermine la 12-ième colonne de T2. À partir des coordonnées [21,12], on obtient le code du caractère *ü* F6 en hexadécimal. Ces consultations sont aussi indiquées dans les figures 5.7 et 5.8.

Dans la réalisation, si l'entrée d'un signe simple, ou composé, est correcte, l'insertion suivante de n'importe quel caractère ne donne pas d'erreur. Par exemple, si l'utilisateur

entre le signe composé BrGr suivi d'un caractère D, seul D est pris en compte et affiché, car il n'existe pas le caractère D avec le signe composé brève-grave.

Note : Sur le clavier Sun, les codes de contrôle ASCII de 1 à 31 en décimal peuvent être entrés directement par la combinaison de la touche `Control` et d'une touche alphabétique. Cette possibilité permet d'entrer plus rapidement les caractères accentués de codes inférieurs à 32.

## 3.3 Dialogue par la méthode "à la main"

Notre implémentation crée une version en vietnamien de Grif (`Vgrif`) qui permet à l'utilisateur de dialoguer dans sa langue avec Grif.

L'utilisateur de Grif est guidé en permanence pour effectuer des actions structurales (sélection, création, destruction...) ou des commandes au cours de l'édition de son document [III.2 Vatton 87].

Selon la nature des intitulés textuels affichés sur les sous-fenêtres de dialogue (cases de menus, de messages, ou de commandes), nous avons deux types d'items :

1. *intitulés structuraux :* Ils sont créés dynamiquement à partir des schémas de structure en langage S. Ce sont des noms de types d'éléments (texte, image, références...), ou de types d'objets structurés (section, sous-section, paragraphe...) affichés dans les cases de menus et de commandes.
2. *intitulés fixes :* Ce sont les messages, les textes associés aux boutons de la souris, les intitulés de commandes d'édition (`INSERER`, `COUPER`, `COLLER` et `COPIER`) et les intitulés en tête des sous-fenêtres (Documents et vues, Sélection, Recherche...) qui sont groupés dans des modules.

   Par exemple, la version française de Grif utilise deux fichiers `francais.p` et `txtF.c` pour les intitulés fixes. Ainsi, les noms d'alphabet (Latin et Grec) et les attributs typographiques fournis par les polices de caractères disponobles (Familles de polices, Style et taille) se trouvent dans ces intitulés fixes.

L'écriture des chaînes de caractères vietnamiens utilise la même solution que celle de Grif : les caractères non standard sont écrits sous la forme de leur code octal (de 1 à 37 et de 200 à 377) précédé d'un signe '\'.

Par exemple, le mot "Tìm kiếm" ("Recherche") s'écrit `T\030m ki\336m`.

Nous devons donc réécrire des schémas de structure en langage S dont les intitulés structuraux sont écrits en vietnamien. Pour les intitulés fixes, nous créons deux nouveaux fichiers, `vietnamien.p` et `txtV.c`, avec l'écriture déterminée.

Afin d'afficher le nom d'alphabet vietnamien dans le menu de changement d'alphabets, nous avons dû ajouter directement cet intitulé dans le fichier `vietnamien.p`. C'est pourquois, nous parlons de la méthode "à la main".

D'autre part, nous voulons une image de document entièrement en vietnamien. Le document disposera des titres (introduits dans un schéma de présentation en langage P) en

vietnamien, comme les titres Résumé, Mots-Clés, Table des matières, Index... dans un document en français. Ces titres sont décrits dans un schéma de présentation sous forme des constantes.

**Exemple** : Dans le schéma de présentation d'un article en français, la constante "Résumé" a été déclarée comme suit :

```
CONST
        CstResume = TEXT 'R\37sum\37 : ';
```

Pour un article en vietnamien, ce schéma doit être réécrit avec les redéclarations des constantes textuelles, comme la constante "Tóm tắt" suivant :

```
CONST
        CstResume = TEXT 'T\034m t\320t :';
```

La méthode "à la main" que nous avons utilisée dans cette implémentation permet d'effectuer un dialogue en vietnamien pour produire un document qui peut contenir une langue quelconque parmi les langues disponibles ou combiner ces langues.

## 3.4 Impression en langage PostScript

La dernière étape de l'implémentation est l'impression directe d'un texte vietnamien en langage PostScript. Comme nous l'avons dit plus haut, il y a deux problèmes à considérer : la redéfinition de polices et la justification du texte en utilisant les nouvelles polices (avec les anciennes).

Dans la redéfinition, d'abord, on reprend toutes les polices en latin qui ont été construites par Grif, après, on les renomme selon l'alphabet vietnamien. Ensuite, on utilise la méthode de "remplissage des vides" pour insérer tous les caractères accentués possibles existant en PostScript et les caractères de base. L'ordre de remplissage est déjà défini dans le tableau de codes ASCII étendus (figure 5.3). Enfin, on définit les signes diacritiques (simples et composés) par l'écriture des sous-programmes adéquats en PostScript.

À titre d'exemple, le signe composé CrPo est composé des deux signes simples Cr et Po. Le signe Cr est défini par l'apostrophe droite et le calcul des coordonnées de son dessin sur les lettres O o U u. De même avec le signe Po, défini par le point et le calcul des coordonnées de son dessin sur les lettres le concernant. Voici l'utilisation des signes simples en PostScript :

```
Br = /breve                 Ti = /tilde
Ci = /circumflexe           Ai = /acute
Cr = /quoteright            Po = /period
Gr = /grave                 Ba = /endash
In = /quoteright
```

À l'aide des polices construites, la justification des lignes d'un texte vietnamien est ainsi effectuée en deux étapes. À la première, après avoir déterminé la police concernée, le texte vietnamien est considéré comme un texte latin composé des caractères dans cette police et est justifié normalement. À la seconde, on lance l'impression des signes diacritiques nécessaires sur chaque caractère suivant leur apparition dans le texte. Il est clair que cette méthode est aussi utilisable pour un texte écrit sur plusieurs alphabets.

**Exemple** : Pour imprimer avec justification le titre de cette thèse ("Contribution à l'informatique multilingue : extension d'un éditeur de documents structurés")

*Đóng góp cho ứng dụng tin học trên nhiều ngôn ngữ : mở rộng một trình xử lý văn bản có cấu trúc.*

l'éditeur réalise d'abord la justification du texte

*Dóng góp cho ung dung tin hoc trên nhiêu ngôn ngu : mo rông môt trình xu ly van ban có câu trúc.*

dont les caractères accentués *ó, ê, ô, ì* existent en latin. Après, tous les signes supplémentaires sont imprimés par les sous-programmes dans leurs positions correctes.

# Chapitre 6 : Évaluation de la méthode



Imprimé par Grif le 20/3/91

# 1 Intérêt

## 1.1 Combinaison des alphabets

Avec le premier résultat obtenu (version vietnamienne `Vgrif`), l'utilisateur de Grif peut maintenant travailler sur un document écrit entièrement en vietnamien. La manipulation des signes diacritiques est facilitée par la compatibilité entre la saisie des caractères vietnamiens et celle des caractères latins. La réalisation permet, d'une part, une homogénéité des fonctions disponibles : pour un texte quelconque, les commandes ont le même effet.

D'autre part, chaque alphabet dans l'ensemble des alphabets traitables correspond à une version de Grif, et est utilisé spécifiquement comme une langue de dialogue : l'utilisateur peut la choisir pour produire son document sur d'autres alphabets.

La possibilité de combinaison des alphabets (vietnamien, latin et grec) dans un même document se fait à l'aide d'une commande de changement d'alphabet au cours de l'édition du document. Dans ce cas, le texte sélectionné est transformé en un texte de l'autre alphabet choisi par l'utilisateur. Grif considère les noms d'alphabets comme des attributs à présenter au niveau de chaque caractère sur la ligne.

Voici un exemple de combinaison de langues : *Comment dire merci ? En grec, on dit* ευχαριστῶ, *en hongrois köszönöm, en vietnamien cảm ơn. Comment dire au revoir ? En turc, on dit güle-güle, en hongrois a viszontlátásra, en vietnamien tạm biệt, etc.*

## 1.2 Extension de la méthode

La méthode peut être étendue aux autres langues écrites avec l'alphabet latin (slovaque, suédois, turc...) ou plus généralement, aux systèmes d'écriture qui utilisent un ensemble de moins de 256 signes typographiques : cyrillique, grec..., à condition que la méthode de saisie par composition des caractères diacrités soit utilisable.

La difficulté est de maintenir la compatibilité entre l'affichage sur l'écran et l'impression sur papier. Dans tous les cas, on peut créer des polices pour l'affichage ainsi que pour l'impression en PostScript ; mais cette dernière tâche demande beaucoup de travail pour arriver à un résultat esthétiquement satisfaisant.

## 1.3 Travail simple

La méthode que nous avons développée pour le vietnamien ne demande qu'un travail simple et donne rapidement des résultats dans de nombreux cas pratiques. Cette première étude nous a permis de nous familiariser avec Grif, qui est un gros programme.

Dans ce travail, l'`Éditeur` et le `Médiateur` sont les deux composants principaux que nous avons modifiés. À titre indicatif, voici le volume de ces modifications (essentiellement des ajouts) :

Les modules d'`Éditeur` (en Pascal) :

| *Description du traitement :* | *Nombre de lignes ajoutées :* |
|---|---|
| Commande de recherche de texte | 24 |
| Commande de sortie directe en PostScript | 3 |
| Création des images abstraites | 5 |
| Création des paragraphes | 8 |
| Gestion de la sélection structurelle | 27 |
| Lecture d'un fichier de la forme pivot | 15 |
| Messages et menus en français | 15 |
| Traitement de changement d'alphabet | 15 |
| Traitement de mise en ligne | 70 |
| *Total:* | 182 |

Les modules de `Médiateur` (en Pascal et en C) :

| *Description du traitement :* | *Nombre de lignes ajoutées :* |
|---|---|
| Affichage des boîtes de texte | 10 |
| Défilement dans les fenêtres | 9 |
| Insertion des caractères et gestion des commandes | 36 |
| Gestion des catalogues et polices | 12 |
| Saisie par composition des touches | 205 |
| Gestion des fenêtres X-Window | 24 |
| Gestion des sorties en PostScript | 238 |
| Gestion des fichiers PostScript | 480 |
| *Total:* | 1014 |

Voici le pourcentage de modification :

| *Composant :* | *Pourcentage de modification :* |
|---|---|
| `Éditeur` | 0.5% |
| `Médiateur` | 4.1% |

Le caractère très structuré et modulaire de Grif nous a donc permis d'arriver à ce premier résultat au moyen de modifications très ponctuelles, qui seraient très faciles à reprendre dans le produit en C actuellement commercialisé, si besoin était.

Enfin, nous avons pu reprendre dans le travail suivant une partie des ressources (polices, tableaux) et des modules de traitements spécifiques réalisés dans cette première extension.

# 2 Limites de cette approche

## 2.1 Niveau du traitement

L'ajout du vietnamien dans Grif ne s'est traduit que par une intervention au niveau du traitement de chaînes.

Cette limitation volontaire permet certes le traitement de documents contenant du vietnamien, mais la saisie et surtout la correction du texte restent lourdes, et la qualité d'impression est moyenne.

Par exemple, pour corriger un ou deux signes diacritiques sur un caractère déjà entré, il faut tout effacer (caractère de base et signes), et retaper. Évidemment, une meilleure solution serait de corriger seulement le signe incorrect.

Du point de vue esthétique, la construction des caractères diacrités par combinaison de signes préexistant en PostScript ne donne pas une très bonne qualité. En effet, il est difficile de placer correctement les signes diacritiques dans leur position pour chaque changement de taille des caractères.

## 2.2 Limites de la réalisation

Cette première réalisation montre quatre désavantages de la méthode employée. Le premier est la limite de la méthode du "remplissage des vides". Le codage obtenu ne permet pas d'effectuer simplement les commandes de recherche et de remplacement en prenant en compte la distinction MAJ/min, notamment pour les chaînes contenant les codes étendus. De plus, puisqu'il n'y a pas encore de traitement indépendant de la touche frappée au clavier dans la méthode de saisie directe, certains codes étendus restent strictement réservés aux commandes de Grif (ceux de rang 144 à 155 en décimal) et ne donc peuvent pas être utilisés pour le codage des caractères.

Le deuxième désavantage concerne le changement d'attributs : de même que pour les attributs typographiques, le texte sélectionné peut être soit modifié par une nouvelle présentation, soit écrit sur un autre alphabet donné. Il est nécessaire de différencier ce traitement pour la cohérence. Par exemple, un caractère latin ne peut pas se transformer en un caractère chinois, ou l'inverse, à l'aide d'un changement d'alphabets. En effet, il n'existe souvent pas de correspondance sur le plan de la représentation interne des caractères entre les alphabets.

Le troisième désavantage est l'invariabilité des attributs typographiques des caractères d'un alphabet par rapport à ceux de l'autre. Par exemple, si l'éditeur travaille avec l'alphabet cyrillique, il ne peut pas utiliser le même menu typographique qu'avec l'alphabet latin. Il faut donc avoir une gestion dynamique des ressources disponibles.

Le dernier concerne le dialogue. D'ailleurs, comme ce point n'a pas été traité dans `Vgrif`, l'utilisateur ne peut pas choisir librement une langue de dialogue différente des langues du document pendant l'édition.

## 2.3 Difficultés d'adaptation aux nouvelles versions de Grif

Bien que les modifications apportées au code de Grif soient faibles, elles ne se présentent pas encore comme des modules indépendants. La plupart des paramètres ont été introduits "à la main" dans les programmes sources. On pourrait bien sûr reporter ces modifications dans les nouvelles versions de Grif, mais il serait difficile de les améliorer en parallèle aux autres évolutions de Grif, et d'intégrer ces améliorations au fur et à mesure.

# 3 Perspectives

## 3.1 Perfectionnement de cette approche

Il serait tout-à-fait possible d'améliorer ce que nous avons fait, pour obtenir une version de Grif permettant de traiter des documents contenant du texte écrit dans les langues utilisant de "petits" systèmes d'écriture, et s'écrivant comme le français, de gauche à droite et de haut en bas.

On arriverait à un résultat analogue à ce qu'on peut faire sous une version "européenne" du Mac OS avec les texteurs (traitements de texte) usuels.

Le problème de la saisie et de la correction des erreurs de frappe pourrait être traité de façon un peu plus générale, en considérant les polices comme des ressources munies de méthodes de saisie et de correction.

En travaillant assez sur les polices, on arriverait sans doute à bien améliorer la qualité de la restitution à l'imprimante.

Enfin, on arriverait certainement à remanier le code de Grif pour isoler beaucoup mieux ce qui est spécifique des traitements de chaînes de caractères.

## 3.2 Des limites intrinsèques

Cependant, nous buterions sur l'impossibilité de généraliser à des systèmes d'écriture utilisant de grands jeux de caractères, comme le chinois et/ou s'écrivant de façon différente du français, comme l'arabe.

De plus, l'utilisation du niveau de présentation (les polices) pour traiter des phénomènes ressortissant au niveau linguistique (les langues et leurs systèmes d'écriture) ne permettrait pas de réaliser un certain nombre de fonctions naturellement attendues par les utilisateurs, comme la recherche en fonction de la langue, avec ou non respect des diacritiques et autres distinctions (comme MAJ/min).

Enfin, nous aurions trouvé dommage de poursuivre dans une voie limitée alors que des réalisations commerciales, comme le Star de Xerox montrent qu'on peut faire mieux, et que nous avions déjà commencé, dans un travail antérieur, à étudier et à implémenter, dans le mini-éditeur PÉPÉ, une voie plus générique.

## 3.3 Vers une approche plus radicale

Nous avons donc cherché à relever ce défi, en réalisant une version de Grif permettant de traiter le chinois, et de le faire non pas au moyen d'une extension spécifique, mais en nous inspirant de l'esprit de généricité de ses concepteurs, et en utilisant le plus possible les techniques de génie logiciel qu'ils ont mis en œuvre, de façon à ce que le multilinguisme soit effectué au niveau adéquat, et réalisé dans des modules clairement identifiables.

# Partie C
# Une approche générique : introduction du langage E dans Grif. Application au chinois



Imprimé par Grif le 20/3/91

# Introduction

Dans la partie précédente, nous avons réalisé une première extension de Grif, qui consiste à utiliser les codes ASCII étendus pour représenter les caractères avec signes diacritiques multiples et à développer sous Grif une méthode de saisie par composition des touches.

Bien que ce codage permette de représenter tous les caractères de tous les systèmes d'écriture utilisant moins de 256 signes typographiques, et que la méthode proposée soit utilisable dans des situations de réel multilinguisme et pas seulement de "localisation", nous avons conclu qu'il fallait rechercher d'autres stratégies pour étendre le nombre de langues traitables par Grif, dans un contexte totalement multilingue, d'une façon qui facilite des applications linguistiques ultérieures.

Le premier problème considéré est l'utilisation d'un système de codage adéquat. Suivant le cas, un code peut être représenté dynamiquement sur plus d'un octet (comme une transcription, par exemple). Ensuite, il est nécessaire de rechercher un autre moyen plus générique pour pouvoir attacher à chaque langue une méthode de saisie convenable. De plus, le nombre des contraintes de la restitution multilingue demande d'utiliser des méthodes paramétrables. Le dernier problème considéré est la possibilitité de rendre le dialogue de l'éditeur lui aussi totalement multilingue.

Comme nous l'avons indiqué, aucune approche de la conception d'un STTM utilisée aujourd'hui ne permet de résoudre d'une façon générique et globale les problèmes posés par le multilinguisme. Il y a trois exigences que nous devons satisfaire : la flexibilité, la portabilité et la généricité. La flexibilité permettra d'offrir un large éventail de solutions possibles. La portabilité assurera une indépendance totale entre les ressources disponibles et le matériel utilisé. La généricité donnera la possibilité d'enrichir l'ensemble de langues naturelles et de systèmes d'écriture traitables par le système.

La comparaison des stratégies possibles nous amène à penser que l'approche la plus adéquate est une approche par langage. C'est ce qui nous a conduit à définir un langage E spécifique, analogue aux langages de Grif.

Le langage E permet d'écrire des *schémas* de transcription d'entrée. Pour une langue donnée, on peut construire plusieurs schémas différents activables simultanément sous Grif. Chaque schéma décrit les conventions d'entrée d'un caractère dans le jeu de caractères retenu, sa représentation interne adaptée à une application linguistique, et sa restitution sur un matériel donné, en fonction des ressources disponibles.

Le chapitre 7 présente l'étude des stratégies possibles et aboutit à une solution plus générique, le langage E. Le chapitre suivant décrit la sémantique, la syntaxe et la mise en œuvre du langage E, toutes trois homogènes à celles des autres langages de Grif. Enfin, le chapitre 9 présente l'utilisation pratique de E, à travers une extension multilingue de Grif permettant de prendre en compte dans un même document du chinois, du vietnamien, du russe, du grec, en sus du français et de l'anglais.

# Chapitre 7 : Étude des stratégies possibles et choix d'une solution par langage

# 1 Stratégies possibles

Nous avons envisagé trois stratégies pour atteindre les objectifs exposés dans l'introduction précédente :

1. Intégration des méthodes existantes.
2. Extension des langages de Grif.
3. Introduction d'un nouveau langage.

## 1.1 Intégration des méthodes existantes

Le principe est de compléter des logiciels, pas à pas, par l'introduction de méthodes supplémentaires, puis par leur combinaison dans le code source de Grif, pour lui donner effectivement des possibilités de traitement prévues. Cette solution ne concerne que la programmation, mais ne concerne pas les langages de Grif. Les quatre procédés envisageables sont : l'utilisation de codages mixtes, le mélange des méthodes de saisie, l'extension de la restitution et la gestion "à la main" des menus.

### Utilisation de codages mixtes

Actuellement, vu l'absence d'un système de codage commun et universel pour multi-alphabet, l'utilisation unique d'une méthode ne permet pas d'étendre le nombre d'applications. Il est possible d'utiliser des codages mixtes. Suivant le cas, le codage utilisé peut être sur deux octets, en taille fixe (comme la méthode d'Anderson), ou variable (codage "souple" utilisé dans Star de Xerox), ou même sur plusieurs octets comme certaines transcriptions (des grands caractères CARIX, par exemple).

Cette méthode permet d'étendre Grif à nouvelles langues dont chacune appartient à un type de codage. Par exemple, pour le traitement des caractères arabes, hébreu..., l'utilisation du même code que Grif est possible ; mais pour les caractères chinois, ou japonais, il est obligatoire d'utiliser des codes à deux octets, etc.

On constate qu'avec des codages mixtes, la notion de texte représenté par des chaînes de caractères simples n'est plus suivie. En effet, le texte sera une suite quelconque de représentations internes et par conséquent les traitements de chaînes (comparaison, tri alphabétique, coupure de sous-chaînes...) changeront.

Nous pourrions utiliser trois méthodes de codage :

1. méthode d'Anderson (voir chapitre 2) : codage en taille fixe sur deux octets, commun pour tous les alphabets ;
2. méthode "souple" : codage mixte d'un et de deux octets seulement ;

3. méthode "transcription" : le codage de chaque caractère est de taille variable, même à l'intérieur d'un jeu donné ; chaque application transforme la transcription vers sa représentation interne propre ; l'avantage est qu'on peut choisir de prendre comme alphabet de base un ensemble universel et portable de caractères.

Le travail suivant est de résoudre les problèmes de saisie et de restitution sur le codage intégré. En effet, l'ajout d'une nouvelle langue demande de construire les ressources correspondantes.

### Mélange des méthodes de saisie

Une méthode de saisie ne s'adapte qu'à un ensemble limité de jeux de caractères (appelés alphabets par brièveté dans notre implémentation ainsi que dans certaines parties de la discussion) ; en revanche, la saisie d'un jeu peut être effectuée par l'utilisation de méthodes différentes.

Afin d'utiliser simultanément plusieurs méthodes dans Grif, comme saisie par transcription et saisie par choix, présentées plus haut, il faut introduire des modules de saisie indépendants. En fait, ce sont des sous-programmes dont chacun correspond à une méthode et dont les paramètres sont le jeu de caractères en cours et le mode d'entrée (clavier, souris, ou peut-être fichier).

Par exemple, si on a un ensemble d'alphabets $A_1, A_2, \ldots, A_N$ et un ensemble de méthodes disponibles $M_1, M_2, \ldots, M_M$, le choix d'une méthode ne dépend que de l'alphabet courant :

**si** Alphabet = $A_i$ **alors** Methode = $M_j$ **fsi**

Mais, si l'alphabet $A_i$ dispose de certaines méthodes $M_{j1}, M_{j2}, \ldots, M_{jK}$, et si l'utilisateur ne veut pas la méthode par défaut, dans ce cas, il peut en choisir une :

**si** Choix_Methode **alors**
  **choix** Methode **dans** ( $M_{j1}, M_{j2}, \ldots, M_{jK}$ ) **fchoix**
**fsi**

La construction des modules de saisie demande donc de disposer d'un module de gestion d'entrée. Dans l'extension de Grif pour le vietnamien, nous avons utilisé le même traitement, car il n'y avait qu'une seule méthode de saisie des caractères diacrités. Ainsi, le module introduit interprète une touche c du clavier (ou un bouton de la souris) comme soit une commande, soit l'entrée selon une méthode de saisie déjà déterminée par l'utilisateur. Et voici ce traitement :

**si** c = $Cmd_k$ **alors** Exec_$Cmd_k$
**sinon** Exec_methode(c)
**fsi**

La saisie est toujours un problème délicat en fonction de la performance de la méthode utilisée. Les utilisateurs, qui sont très divers, souhaitent la facilité d'apprentissage, la commodité de manipulation, ou la vitesse de saisie (nombre minimum de frappes pour un caractère). L'expérience d'autres systèmes et la disponibilité des jeux de caractères sont aussi des facteurs considérables.

Nous proposons donc d'utiliser en même temps les deux méthodes : saisie par composition des touches et saisie par choix.

### Extension de la restitution

Le principe est que la résolution se fait indépendamment des schémas de présentation. Parmi les trois problèmes présentés au chapitre 2 (création des ressources de restitution, composition des caractères, et mise en lignes et en paragraphes), voici une liste de propriétés souhaitables au niveau de la restitution :

1. introduction de contrôles de pagination destinés à éviter les veuves, les orphelins, et le changement de page entre certains paragraphes : ces contrôles sont réalisés par l'ajout de deux nouveaux items dans le menu de la commande `Présentation` ;
2. création de nouvelles polices de caractères pour enrichir les propriétés typographiques : soulignement, mise en relief... ;
3. traitement du crénage de deux lettres majuscules successives dans un mot en vérifiant leur position d'une façon compatible en affichage et en impression ;
4. affichage en priorité de tous les caractères disponibles en PostScript ;
5. coupure des mots en fin de ligne (césure) automatique ou contrôlée par la construction de dictionnaires et d'algorithmes adaptés aux critères linguistiques et typographiques spécifiques de chaque langue traitable par Grif ; cette tâche demande des travaux importants ;
6. correction automatique des fautes d'orthographe.

Il y a donc deux possibilités d'extension :

* modification de modules existants, notamment dans le `Médiateur` ;
* ajout direct de nouveaux modules de traitement.

### Dialogue par l'extension de la méthode "à la main"

Nous pouvons étendre la méthode "à la main" qui a déjà servi pour le vietnamien (voir chapitre 5).

Il s'agit de modifier directement les structures de données et les modules de gestion d'interface utilisateur de Grif. La modification des structures de données donne un indicateur sur la langue de dialogue, ainsi que sur la police de caractères utilisées (pour afficher les menus, les commandes et les messages sur les sous-fenêtres de dialogue). Nous ajoutons encore dans le menu principal de Grif la commande `DIALOGUER` pour changer la langue de dialogue. Cette commande doit être au niveau des commandes d'édition (`COUPER`, `COPIER`...).

La méthode "à la main" a utilisé deux types d'intitulés de dialogue (structural et fixe) et le nom d'alphabet vietnamien a été inséré directement dans le fichier de messages (intitulés fixes). Le désavantage manifeste de cette méthode est qu'il n'y avait pas de gestion dynamique des noms des alphabets traitables et des attributs typographiques des polices de caractères correspondantes (qui sont différents d'une langue à l'autre).

Afin d'obtenir une description indépendante, nous pouvons grouper les noms des jeux de caractères introduits, les noms des polices et les attributs typographiques (fournis par les ressources correspondantes) dans un module indépendant. Ce type d'items nous donne des *intitulés complémentaires*. Le fichier contenant ces intitulés sera appelé *fichier de compléments*.

Ainsi, nous avons trois types d'intitulés pour le dialogue. Selon le codage utilisé, ces intitulés textuels sont écrits comme des chaînes de caractères (méthode utilisée pour les caractères diacrités vietnamiens). Les caractères codés sur deux octets (chinois, japonais...) sont alors juxtaposés successivement (comme la représentation des textes mixtes dans Star de Xerox, voir chapitre 1). Il en résulte qu'il est difficile d'utiliser la méthode de codage "transcription" dans cette stratégie, car pour un caractère de l'alphabet donné, il n'y aucune de liaison entre sa représentation interne et son code dans la police utilisée.

L'extension de la méthode "à la main" nous donne un gros ensemble de schémas et de fichiers, de trois types :

1. les schémas de structure en langage S et les schémas de de présentation en langage P d'une classe de documents : pour chaque type de schéma, nous devons réécrire un schéma identique dans la langue disponible ;
2. les fichier de messages : pour chaque langue, nous devons écrire des fichier de messages, comme `Vietnamien.p` et `txtV.c` ;
3. le fichier de compléments.

Avec cette méthode, nous pouvons créer une version multilingue de Grif qui permet à l'utilisateur de changer la langue de dialogue au cours de manipulation de son document.

## 1.2 Extension des langages de Grif

Le principe est d'étendre en même temps les trois langages (S, P et T) de Grif. Il y a deux niveaux d'extension :

1. ajouter de nouveaux mots-clés et les extensions sémantiques correspondantes, en conservant la syntaxe du langage ;
2. modifier totalement la sémantique et la syntaxe du langage.

La modification utilise en tout cas le même méta-langage et mène à une recompilation compatible de tous les schémas. Comme le niveau 2 demande un travail difficile, dans notre présentation, nous choisissons le niveau 1 pour faciliter l'étude des solutions, en respectant la nature de chaque langage de Grif.

### Extension du langage S

Nous proposons deux possibilités d'extension, sur les attributs et sur la structure d'une classe d'objets.

Le langage S permet de définir globalement un attribut, appelé *langue*. Celui-ci ne sert qu'à contrôler les attributs typographiques affectés à un élément textuel, mais ce n'est pas le cas pour la gestion des alphabets. Dans la version prototype de Grif, les alphabets (latin et grec) existent par défaut et leur choix est géré indépendamment des schémas.

> **Exemple** : On peut utiliser l'attribut langue pour décrire un document bilingue de Grif (en français et en anglais). Dans son schéma de structure en S, on le définit globalement comme suit :
>
> ```
> ATTR
>       Langue = Francais, Anglais ;
> ```
>
> Grâce à cette définition, on peut faire afficher le texte français dans un style de caractères autre que celui du texte anglais. Dans le schéma de présentation en P, on utilise le même attribut langue pour décrire les règles de présentation suivantes :
>
> ```
> ATTRIBUTES
>       Langue = Francais :
>                Style : Roman ;
>                { texte français en caractères romains }
>       Langue = Anglais :
>                Style : Italics ;
>                { texte anglais en caractères italiques }
> ```
>
> Avec ces règles, changer la valeur de l'attribut langue (français, ou anglais) provoque un changement du style.

L'extension sur les attributs se fait en affectant à l'attribut langue de nouvelles valeurs. Mais il faut concilier deux niveaux : celui des attributs typographiques et celui des alphabets.

Ainsi, tous les alphabets traitables, qui sont introduits à l'avance, sont déclarés par leur nom dans le schéma de structure comme attributs. Lorsqu'un élément textuel, qui utilise l'aphabet courant, est créé, l'utilisateur peut lui attribuer un autre alphabet. C'est le compilateur S qui produit les procédures nécessaires.

La deuxième possibilité concerne la redéfinition de la structure d'une classe d'objets pour enrichir ses propriétés dans la présentation.

Prenons par exemple le problème de la mise en paragraphes. Nous désirons définir trois types de paragraphe : horizontal, vertical et diagonal. Nous laissons de côté les types rond ou courbe à cause de leur complexité. Ils pourraient faire l'objet d'une extension ultérieure. Pour chaque type de paragraphe, l'ordre de restitution des caractères dépend de l'alphabet courant. Par défaut, c'est l'ordre de gauche à droite et de haut en bas. Mais s'il

agit d'un texte arabe, ou d'un texte chinois (pour lequel cette valeur par défaut a été modifiée par l'utilisateur), l'ordre s'inverse.

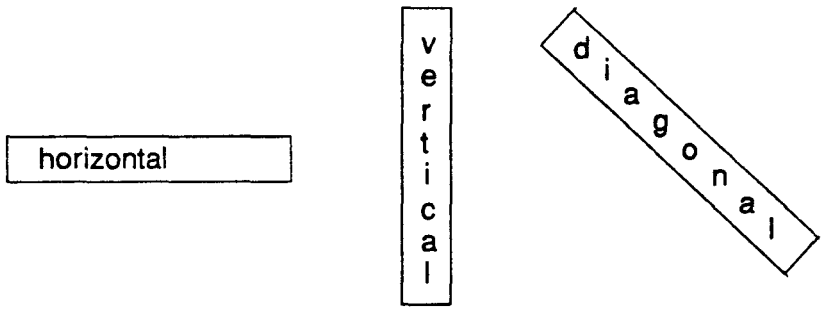



Figure 7.1 : Trois types de paragraphe introduits dans le langage S

On peut considérer que les types horizontal et vertical sont des variations du type diagonal. Dans le schéma de structure de la classe de paragraphes, on peut décrire :

```
Type_Paragraphe = CASE OF
                  Horizontal;
                  Vertical;
                  Diagonal;
                  END;
```

Les deux possibilités d'extension présentées n'entraînent pas de gros changement dans la sémantique du langage S. Dans les schémas de structure, il faut introduire aussi des descriptions concernant la saisie et la restitution pour les alphabets traitables par Grif. L'introduction peut agir sur la notion de texte (élément terminal) qui peut combiner différents systèmes d'écriture.

### Extension du langage P

Notre extension sert uniquement à la restitution, en déclarant les aspects physiques de la présentation des objets d'un document. L'extension du langage P doit se faire en même temps que celle du langage S. Nous avons deux possibilités : redéfinir les boîtes de présentation des paragraphes, et déclarer les attributs typographiques possibles.

La redéfinition des boîtes résoud les problèmes de mise en lignes et en paragraphes imbriqués par l'extension du langage S.

Dans la version prototype de Grif, il existe un seul type de paragraphe (boîte englobante) qui est construit par la concaténation, de gauche à droite, des boîtes d'éléments (boîtes englobées). Ainsi, chaque fois qu'un paragraphe est créé, il est complété progressivement, ligne par ligne. Sur chaque ligne, les caractères d'un texte sont aussi entrés et affichés dans le même ordre.

L'extension consiste à adapter aux schémas de présentation la description des nouvelles propriétés d'objets. Ce sont les nouveaux types de paragraphe (horizontal, vertical ou

diagonal). Un paragraphe peut comporter un mélange de sens d'écriture (de gauche à droite ou l'inverse, de haut en bas ou l'inverse).

La déclaration des attributs est introduite dans les règles de présentation pour chaque alphabet présent. Ces attributs ne sont pas fixés : ils varient suivant les ressources et les alphabets disponibles. L'extension doit permettre une gestion dynamique des paramètres pour le choix des caractères à afficher.

### Extension du langage T

À partir de la structure logique spécifique d'une classe d'objets du document décrite par le langage S, l'extension du langage T vise à redéfinir des règles de traduction qui peuvent être associées :

* aux trois types de paragraphes définis,
* aux attributs globaux concernant les alphabets disponibles,
* aux caractères diacrités complexes, idéographiques, ou autres.

Il est important de voir que la traduction dépend beaucoup de la capacité du formateur pour l'alphabet concerné. Par exemple, l'introduction d'un texte chinois, arabe, ou hébreu... dans TEX, ou Troff est un travail délicat, à cause des capacités nécessaires à la gestion des objets générés.

## 1.3 Introduction d'un nouveau langage

Le principe est d'introduire un nouveau langage de conception homogène à celle des autres langages de Grif.

Ainsi, en s'appuyant sur le modèle de haut niveau déjà acquis dans Grif, nous introduisons les *descriptions linguistiques* pour compléter la structure abstraite qui est spécifiée par la structure logique et la présentation physique du document. Ces descriptions sont réalisées par la définition du nouveau langage. Son implémentation donne l'édition des documents structurés multilingues.

Il y a quatre contraintes dans les descriptions linguistiques :

1. la prise en compte d'application existantes, fondées sur tel ou tel codage d'un système d'écriture donné ;
2. les méthodes de saisie et les aspects de restitution ;
3. l'exploitation dynamique des ressources disponibles ;
4. l'ouverture pour plusieurs applications.

Nous nous sommes inspiré du langage LT (Langage de Transduction) utilisé au GETA pour écrire des transcripteurs, dans le cadre de recherches en traduction automatique multilingue [III.1 Boitet 83] [III.1 Lepage 86]. Le langage LT a surtout été utilisé pour sortir les textes traduits sur les matériels disponibles, attendant chacun une transcription différente.

Par exemple, si l'on dispose d'une imprimante qui peut sortir des textes en alphabet thaï où chaque caractère correspond à un code hexadécimal, on peut sortir un texte traduit en thaï où chaque caractère est transcrit par un caractère latin suivi d'un chiffre.

# 2 Discussion

## 2.1 Adéquation et inadéquation des solutions

À partir des trois stratégies proposées, nous faisons une évaluation en détail sur chaque méthode.

### Pour le codage

Les trois méthodes proposées sont estimées comme suit :

| *Méthode* | *Adéquation* | *Inadéquation* |
|---|---|---|
| "Anderson" | homogénéité du traitement sur les chaînes de texte | mauvaise adaptation aux systèmes d'écriture à grands jeux de caractères (chinois, japonais...), pas de réalisation |
| "Souple" | gain en place de mémoire et en temps d'exécution | difficulté du traitement linguistique |
| "Transcription" | bonne adaptation à diverses applications | place mémoire (grand nombre d'octets utilisés pour représenter un caractère) |

Nous considérons que la troisième stratégie (introduction d'un nouveau langage) peut utiliser effectivement l'une de ces trois méthodes de codage, parce qu'on peut décrire indépendamment les formes de représentation interne d'un système d'écriture hors de l'édition du document par rapport aux ressources d'affichage.

Par contre, les deux premières stratégies (intégration des méthodes et extension des langages S, P, T), où le codage doit être intégré à l'avance dans l'éditeur, ne permettent pas d'utiliser la méthode de codage "transcription", parce qu'il est difficile de faire une correspondance entre le code interne d'un caractère (transcription sur plusieurs octets) et son adresse dans la police utilisée pour l'afficher.

### Pour la saisie

Les deux méthodes (composition des touches et saisie par choix) sont autant utilisables dans les trois stratégies présentées. Cependant, avec la troisième stratégie, les données de saisie nécessaires (les tables des caractères de base et des signes diacritiques pour la saisie par composition des touches, les entrées pour la saisie par choix) sont décrites et gérées dynamiquement ; elles ne dépendent pas du matériel utilisé. En revanche, ce n'est pas le cas pour les deux premières stratégies.

### Pour la restitution

Analysons les trois stratégies présentées selon les quatre aspects principaux de la restitution :

| *Aspects de restitution* | *Intégration des méthodes* | *Extension des langages S, P et T* | *Introduction d'un nouveau langage* |
|---|---|---|---|
| Variation morphologique | possible | possible mais délicate | bien résolue |
| Mise en paragraphes | possible | bien résolue | possible mais délicate |
| Typographie (gestion dynamique) | possible | possible | bien résolue |
| Composition des caractères | possible | possible mais délicate | possible mais délicate |

### Pour le dialogue

La méthode "à la main" dans la stratégie "intégration des méthodes existantes" donne la possibilité de changer la langue de dialogue dans une version de Grif multilingue. Mais nous ne pouvons pas utiliser la méthode de codage "transcription" pour le dialogue.

D'autre part, il n'y a pas moyen de réaliser un dialogue multilingue par extension des langages S, P et T (deuxième stratégie). Cependant, on pourrait essayer de combiner ce type d'extension avec la méthode "à la main".

La troisième stratégie est donc la meilleure, car le nouveau langage introduit permet d'appliquer effectivement un changement des langues de dialogue.

## 2.2 Comparaison des stratégies

L'évaluation des méthodes pour les problèmes multilingues retenus nous donne la comparaison suivante :

| *Stratégie* | *Adéquation* | *Inadéquation* |
|---|---|---|
| 1. Intégration des méthodes | introduction arbitraire et non limitée des méthodes | . solution partielle,<br>. non paramétrable,<br>. pas de moyen de description des aspects retenus,<br>. nombre d'alphabets traitables fixé à l'avance. |
| 2. Extension des langages S, P, T | approche générique et globale | . insuffisance de la description des aspects multilingues,<br>. difficulté de concilier la nature des langages de Grif et les aspects multilingues retenus,<br>. nombre d'alphabets traitables fixé à l'avance. |
| 3. Introduction d'un nouveau langage | . approche générique et globale,<br>. possibilité d'introduire de nouveaux alphabets | . insuffisance pour certains aspects de la restitution,<br>. problèmes de place mémoire et de temps d'exécution dus à la richesse des alphabets traitables et des ressources. |

Nous arrivons finalement à deux solutions possibles :

1. combinaison des deux stratégies 1 et 2 (intégration des méthodes existantes et extension des langages S, P, T en même temps) ;
2. stratégie 3 (introduction d'un nouveau langage).

# 3 Choix d'une solution

## 3.1 Langage de transcription d'entrée

Guidés par le souci d'obtenir une solution générique, et des modifications très localisées et modulaires du source de Grif, nous choisissons la deuxième solution (stratégie 3, par langage), et définissons un *langage de transcription d'entrée*, ou langage E.

Un programme écrit dans ce langage sera appelé un *schéma de transcription*. Il est clair que, pour un système d'écriture donné, on peut disposer de divers schémas de transcription.

Le langage E doit être simple à réaliser. Il y a trois étapes à effectuer :

1. construction du modèle sémantique ;
2. description de la syntaxe du langage ;
3. implémentation.

## 3.2 Problèmes à résoudre en priorité

Du fait de la diversité des problèmes posés, le langage E ne vise qu'à résoudre en priorité les quatre problèmes suivants :

1. description du codage utilisé dans la représentation interne ; il faut pouvoir décrire les divers types de codage présentés plus haut ;
2. description des séquences d'entrée utilisées dans chaque méthode de saisie, et la possibilité de combinaison de trois méthodes : saisie directe, saisie par composition des touches et saisie par choix ;
3. description de certaines caractéristiques linguistiques nécessaires et des attributs typographiques concernant la restitution ;
4. pour chaque langue donnée, liaison étroite entre l'entrée, le codage et la restitution : à partir d'une forme de saisie d'un caractère dans l'alphabet, il faut pouvoir produire une ou plusieurs représentations internes correspondant à ce caractère, et son ou ses adresses dans une police déterminée.

## 3.3 Intérêt du langage E

L'introduction du langage E présente un intérêt théorique dans la conception d'un système de traitement de documents structurés multilingues. C'est le complément du modèle de document de Grif par les descriptions linguistiques présenté dans la figure 7.2.

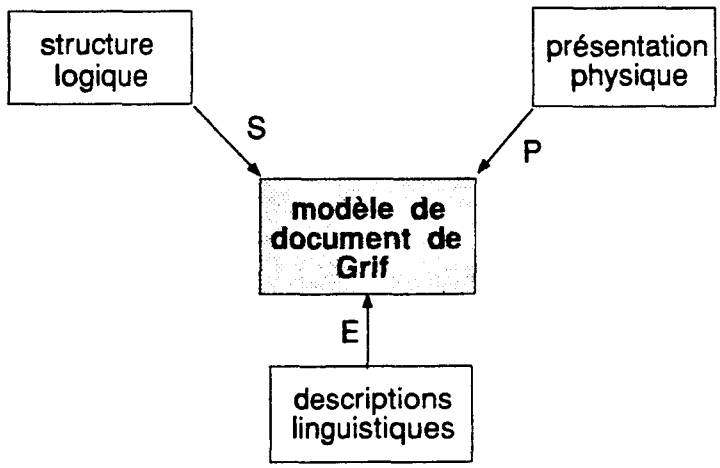



Figure 7.2 : Complément du modèle de document de Grif

En pratique, nous avons les trois avantages suivants :

1. portabilité et lisibilité de la représentation des caractères traités dans les schémas de transcription ;
2. large possibilité de création et de modification de l'ensemble des langues naturelles traitables par Grif : toutes les formes de saisie et de transcription, ainsi que les codes servant à les restituer dans les schémas de transcription, sont décrits indépendamment de l'édition du document ;
3. simplicité du travail de programmation : on peut utiliser et appliquer dans cette implémentation les algorithmes existants de Grif.

Dans le chapitre suivant, on présente en détail la définition du langage E et son implémentation. Une discussion sur les axes de développement ultérieurs sera donnée à la fin du chapitre.

# Chapitre 8 : Le langage E, une solution générique

# 1 Spécification

## 1.1 Caractéristiques générales

La caractéristique essentielle du langage E est son homogénéité avec les autres langages de Grif. Ainsi, sa mise en œuvre est effectuée en trois étapes distinctes : *définition, compilation* et *application.*

Dans l'étape de définition, on construit la sémantique du langage à l'aide des propriétés introduites. Pour le langage E, il s'agit d'un complément du modèle de document de Grif grâce aux descriptions linguistique retenues. La syntaxe de E utilise la même méta-grammaire que celle des langages S, P et T de Grif. D'une part, la syntaxe décrite doit être claire et pertinente ; d'autre part, elle doit correspondre naturellement à la sémantique visée.

L'étape suivante est l'écriture des schémas de transcription dans le langage E. Les schémas sont compilés à l'aide du compilateur E, qui produit des *tables de transcription.* Comme on l'a vu, la création et la modification des schémas sont effectuées par les superutilisateurs.

La dernière étape est l'application des schémas de transcription compilés. C'est aussi l'étape d'édition de Grif : les utilisateurs finals ne peuvent que créer et modifier leurs documents, en s'appuyant sur les schémas prédéfinis et sur les langues disponibles.

## 1.2 Sémantique

Il y a deux composants principaux des descriptions linguistiques pour compléter le modèle de document de Grif : les *paramètres d'édition* et les *règles de transcription*.

Les *paramètres d'édition* servent notamment à entrer et à restituer les parties textuelles du document dans les langues disponibles. Pour une langue donnée, il y a deux catégories de paramètres d'édition : les *paramètres d'entrée* et les *paramètres de sortie.* Les paramètres d'entrée fournissent les informations nécessaires au cours de la saisie des signes du ou des jeux de caractères associés à la langue.

On propose donc les paramètres d'entrée suivants : langue, jeu de caractères et méthode de saisie. Ces trois paramètres sont décrits par leur nom dans le schéma. Pour le paramètre méthode de saisie, trois méthodes sont actuellement disponibles : saisie directe, saisie par composition des touches et saisie par choix.

Les paramètres de sortie servent à la restitution (affichage sur l'écran, ou impression sur papier). L'impression est faite soit directement en langage PostScript, soit par traduction de la forme pivot d'un document Grif, sauvé dans un fichier, vers un formateur, comme TEX, ou LaTEX.

La présentation physique de ces paramètres dépend strictement des ressources du matériel utilisé. Sur l'ensemble des polices de caractères disponibles pour le traitement, on doit fournir en paramètres les propriétés nécessaires. La première propriété est l'indicateur sur les codes d'affichage (ou numéros d'image) qui donne le nombre d'octets utilisés, un octet ou deux octets, pour représenter un code.

La deuxième propriété est l'indicateur MAJ/min qui indique que l'écriture de la langue concernée a la taille majuscule et minuscule. Certaines langues n'ont pas de taille, comme le chinois, le japonais, le thaï, etc. L'introduction de cet indicateur permet une description cohérente dans certaines applications : indexation, recherche et/ou remplacement avec (ou non) distinction MAJ/min.

La troisième propriété concerne les polices dont chacune est décrite par les aspects : nommage, classification selon la famille, le style, la taille..., et propriétés typographiques. L'ensemble des polices de caractères disponibles constitue les ressources de restitution.

On choisit une des ressources disponibles pour l'affichage des caractères entrés au clavier par l'utilisateur. Cette police, appelée police de dialogue, est aussi déclarée comme un paramètre de sortie. Par exemple, quand Grif travaille avec l'alphabet latin, on peut choisir la police de caractères Times, de style roman et de taille de 13 pixels en hauteur.

Après avoir défini les paramètres d'édition, on construit les règles de transcription qui permettent la transformation d'une entrée en une représentation interne et le passage de ce codage au code d'affichage. La représentation interne est une transcription lisible, portable, et adaptée aux traitements linguistiques. Les codes d'affichage sont pris dans la police de dialogue déclarée dans les paramètres de sortie.

Une entrée est définie en prenant en compte l'utilisation d'un clavier QWERTY standard. Dans l'application multilingue, c'est une suite de frappes pour former une expression d'entrée. Pour des raisons de performances, le choix de la forme de saisie doit satisfaire des contraintes : temps de manipulation, facilité d'apprentissage...

Il en résulte que la définition des règles de transcription offre une grande généricité pour utiliser plusieurs langues en même temps dans l'éditeur. Selon les besoins d'application, on peut choisir un type de transcription et une forme de saisie convenables.

Dans une règle, une entrée, parmi plusieurs possibles, peut donner une ou plusieurs transcriptions, chacune correspondant à un caractère. Un caractère peut avoir également un ou plusieurs codes d'affichage (numéros d'image). Nous proposons actuellement trois types de règle de transcription, *normal*, *homophone* et *morphologique* correspondant aux relations : (1 -> 1 -> 1), (1 -> N -> N) et (1 -> 1 -> N). La relation des composants est donnée dans la figure 8.1.

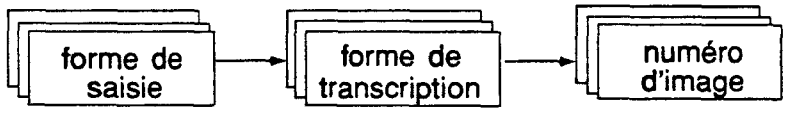



Figure 8.1 : Relation dans les règles de transcripion

La figure 8.2 présente une règle de transcription de type homophone (1 -> N -> N) pouvant décrire N caractères différents.

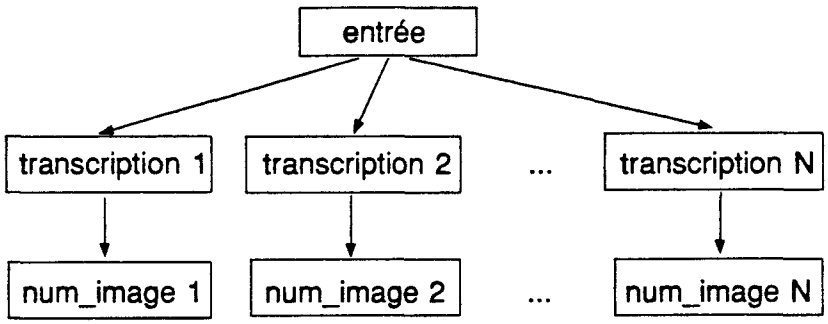



Figure 8.2 : Une règle de transcription de type homophone

## 1.3 Syntaxe

Tout schéma de transcription commence par le mot-clé `TRANSCRIPTION` et se termine par le mot-clé `END`. Le mot-clé `TRANSCRIPTION` est suivi du nom du schéma de transcription. Ce nom est suivi d'un point-virgule.

Après la déclaration du nom du schéma, on déclare les constantes, les paramètres d'édition, et les règles de transcription de l'alphabet concerné dans l'ordre donné. La déclaration de constantes peut être absente, mais les autres déclarations sont obligatoirement présentes.

```
Transcription = 'TRANSCRIPTION' NomSchema ';'
                [ 'CONST' SuiteConst ]
                  'PARAM_IN' SuiteParamIn
                  'PARAM_OUT' SuiteParamOut
                  'CODES' SuiteDefTrans
                  'END' .
NomSchema = NAME .
```

La déclaration des constantes est introduite par le mot-clé `CONST` et constituée d'une liste de constantes. Chaque constante dans la liste est déclarée par son nom suivi d'un signe `'='` et d'une valeur. La valeur d'une constante est soit une chaîne de caractères délimitée par des apostrophes, soit un nombre entier sans signe. Cette valeur est suivie d'un signe `';'`. L'utilisation des constantes permet de simplifier l'écriture de certaines déclarations et donc d'éviter des erreurs.

```
SuiteConst = DefConst < DefConst > .
DefConst   = NomConst '=' ValConst ';' .
NomConst   = NAME .
ValConst   = Chaine / Entier .
```

```
Chaine     = STRING .
Entier     = NUMBER .
```

**Exemple** : On déclare les constantes suivantes :

```
CONST
      a = 'a'; e = 'e'; i = 'i'; { lettres a, e, i, etc. }
      Circ = 158; { signe circonflexe ^, code donné en valeur décimale }
```

La déclaration de tous les paramètres d'édition du schéma doit être terminée par un point-virgule. D'abord, on déclare les paramètres d'entrée au moyen du mot-clé `PARAM_IN` suivi d'une suite de paramètres. Dans l'ordre de déclaration, on trouve le nom de la langue et du jeu de caractères, la méthode et le type de la fenêtre de saisie, la déclaration des lettres qui peuvent avoir des signes diacritiques, les signes diacritiques eux-mêmes, et enfin le nombre maximum de signes diacritiques qu'une lettre peut porter.

```
SuiteParamIn = Langue
               Jeu
               Entree
             [ Fenetre ]
             [ LettreAcc ]
             [ Signes ]
             [ MaxSignSLettre ]
```

Pour chacun de ces paramètres, on donne le mot-clé qui indique le paramètre lui-même et la valeur qui est un nom, séparés par un signe `'='`.

```
Langue = 'Language' '=' NomLangue ';' .
NomLangue = NAME .
Jeu = 'Set' '=' NomJeu ';' .
NomJeu = NAME .
```

Après la déclaration de la langue et du jeu correspondant, le schéma doit désigner une méthode de saisie des caractères. Dans un schéma, il n'est pas nécessaire de déclarer la saisie directe. On considère que la méthode de composition des touches peut être utilisée pour entrer les caractères avec signes diacritiques et que les caractères sans signes diacritiques sont entrés par interprétation des touches. Le mot-clé `LEFTCOMPO` indique la méthode de composition à gauche, `RIGHTCOMPO` indique la composition à droite, et `SELECT` déclare la méthode de saisie par choix.

```
Entree = 'Entry' '=' Methode .
Methode = 'LEFTCOMPO' / 'RIGHTCOMPO' / 'SELECT' .
```

Après cette déclaration, on déclare tous les signes diacritiques et toutes les lettres de l'alphabet qui peuvent les recevoir. Ces caractères sont énumérés sous forme symbolique dans des listes. Une liste est souvent délimitée par des parenthèses et chaque élément

introduit dans la liste est déclaré soit par un nom de constante prédéfinie, soit sous forme de chaîne, soit par un nombre, et séparé du suivant par une virgule.

```
LettreAcc = 'Letters' '=' ListeSymbole ';' .
Signes = 'Signs' '=' ListeSymbole ';' .

ListeSymbole = '(' SuiteSymb ')' .
SuiteSymb = Symbole < ',' Symbole > .
Symbole = NomConst / Chaine .
```

Le nombre maximum de signes diacritiques possibles sur une lettre doit être indiqué dans le schéma. Cette information est utilisée par le module traitant la saisie par composition des touches. Le traitement actuel permet qu'une lettre porte au maximum deux signes diacritiques.

```
MaxSignSLettre = 'Max_Signs' '=' NbSignes .
NbSignes = NUMBER .
```

**Exemple** : L'ensemble de règles suivantes illustre la déclaration de la méthode de saisie par composition des touches dans un schéma de transcription du français :

```
TRANSCRIPTION Francais;
PARAM_IN
   Language = francaise; { langue française }
   Set = latin;
   Entry = RIGHTCOMPO; { composition à droite }
   Letters =  { lettres diacritées écrites sous forme de chaînes }
         ('A','C','E','I','O','U','a','c','e','i','o','u');
   Signs = ( { représentés par les codes ASCII étendus }
            157,  { accent aigu }
            156,  { accent grave }
            158,  { accent circonflexe }
            159,  { tréma }
            160); {cédille}
   Max_Signs = 1; { un seul signe diacritique pour une lettre }
```

Après la présence du mot-clé `SELECT`, on trouve obligatoirement la déclaration du type de la fenêtre associée à la méthode de saisie par choix. On propose trois types de fenêtre selon les propriétés des expressions d'entrée spécifiques aux méthodes de saisie par choix : fenêtre `WPHON` correspond à la saisie phonétique, fenêtre `WGRAPH` correspond à la saisie par composition graphique d'un caractère et fenêtre `WMORPHO` utilisée en particulier pour traiter des variations morphologiques.

```
Fenetre = 'Window' '=' NomFen ';' .
NomFen = 'WPHON' / 'WGRAPH' / 'WMORPHO' .
```

La déclaration des paramètres de sortie dans un schéma de transcription se fait dans l'ordre d'apparition des indicateurs sur les codes d'affichage et sur la taille MAJ/min, puis des déclarations de polices de caractères. Pour chacun des paramètres, on donne le mot-clé, qui indique le paramètre lui-même, suivi d'une valeur.

```
SuiteParamOut = Octets
                MajMin
              [ Famille ]
              [ Style ]
                Relief
                FonteDialogue

Octets = 'Bytes' '=' NbOctets ';' .
NbOctets = NUMBER .

MajMin = 'UppLow' '=' Taille ';' .
Taille = 'Yes' / 'No' .
```

Les valeurs des paramètres `Famille`, `Style` et `Relief` sont ainsi des listes.

```
Famille = 'Family' '=' ListePolice ';' .
ListePolice = '(' SuiteNomPolice ')' .
SuiteNomPolice = NomPolice < ',' NomPolice > .
NomPolice = STRING .

Style = 'Style' '=' ListeEvidence ';' .
ListeEvidence = '(' SuiteNomEvid ')' .
SuiteNomEvid = NomEvid < ',' NomEvid > .
NomEvid = STRING .

Relief = 'Size' '=' ListeRelief ';' .
ListeRelief = '(' SuiteCorps ')' .
SuiteCorps = Corps < ',' Corps > .
Corps = NUMBER .
```

Le dernier paramètre de sortie déclare le nom de la police servant au dialogue pendant la saisie d'un texte. Pour pouvoir utiliser certains caractères spéciaux autres que le signe `'-'` selon les conventions du méta-langage, y compris le signe `'_'`, ce nom est une chaîne. Par exemple, la chaîne `'cyr-s25'` désigne le nom de la police de caractères cyrilliques de hauteur 25 pixels. Rappelons que les noms des polices utilisées dans l'environnement X-WINDOW utilisent les formats standard BDF.

```
PoliceDialogue = 'DialFont' '=' NomPolice ';' .
NomPolice = STRING .
```

Exemple : Les paramètres de sortie du schéma de transcription du français sont donnés par l'ensemble des règles suivantes :

```
PARAM_OUT
   Bytes = 1;      { code d'affichage sur un octet }
   UppLow = Yes;  { écriture MAJ/min }
   Family=('Times','Helvetica','Courier');{familles de caractères }
   Style = ('Roman', 'Bold', 'Italic');   {style}
   Size = ( 9, 13, 17, 22, 33, 48); {différents corps}
   DialogFont = 'ltr13'; {nom de police de dialogue}
```

La déclaration des règles de transcription dans l'alphabet, qui est introduite par le mot clé CODES, est la partie importante du schéma. On y trouve, après ce mot-clé, toutes les définitions de transformation des caractères de l'alphabet donné. Une transformation est déclarée par les signes composés '->' ('-' et '>' successifs). Le signe ';' termine une définition. L'expression d'entrée dans une règle est une concaténation de symboles.

```
SuiteDefTrans = DefTrans < DefTrans > .
DefTrans = ExpEntree '->' RprsCodage ';' .
ExpEntree = Symbole <Symbole > .
```

Le passage d'une transcription (représentation interne du caractère) au code d'affichage est aussi déclaré par les signes composés '->'. Pour simplifier la syntaxe, la forme de transcription, selon les types prédéfinis de l'utilisateur, est une chaîne de caractères. Une présentation plus précise sera donnée dans la section suivante.

La sémantique du langage E permet de faire correspondre à une entrée du clavier trois types de règle différents. En fait, c'est une représentation de code RprsCodage qui est normale, homophone, ou morphologique. Dans l'application, on peut utiliser les règles de transcription de type homophone pour les caractères chinois, de type morphologique pour l'arabe, etc.

```
RprsCodage = TransNormal / TransHomo / TransMorpho .
TransNormal = Transcript '->' CodeInfo .
Transcript = STRING .

CodeInfo = [ BaseNum ] Entier .
BaseNum = '\' / '$' .
```

La déclaration du type homophone et morphologique utilise des listes.

```
TransHomo = '(' SuiteTrNormal ')' .
SuiteTrNormal = TransNormal < ',' TransNormal > .

TransMorpho = Transcript '->' ListeCodeInfo .
ListeCodeInfo = '(' SuiteCodeInfo ')' .
SuiteCodeInfo = CodeInfo < ',' CodeInfo > .
```

**Exemple** : Voici quelques règles de transcription de type normal des caractères accentués français, relatives à une transcription intermédiaire utilisée au GETA, Grenoble (voir chapitre 2) :

```
CODES { constantes et paramètres déclarés plus haut }
   Circ a -> 'a!3' -> 3; {'a!3' représentant le caractère â avec le code
                          d'affichage 3 dans la fonte 'ltr13'}
   Circ e -> 'e!3' -> 11; { caractère ê }
   Circ i -> 'i!3' -> 15; { caractère î }
```

# 2 Implémentation

## 2.1 Construction du compilateur E

Le principe de construction du compilateur de schémas de transcription écrits en langage E est analogue à celui des compilateurs S, P et T de Grif. Il y a deux phases : phase de génération des tables et phase de génération du code.

Dans la première phase, à partir de la description (en méta-langage M) du langage E présentée plus haut, nous utilisons le compilateur de M. La compilation, qui utilise la table des règles syntaxiques de M, produit deux tables différentes. La première, qui est construite de la même façon avec celle de M, contient les règles syntaxiques de E pour l'analyse syntaxique. La seconde contient les symboles pour la génération du code du compilateur.

Dans la deuxième phase, en utilisant les deux tables produites, le compilateur E engendre les tables de transcription.

La figure 8.3 présente le principe de fonctionnement du compilateur E :

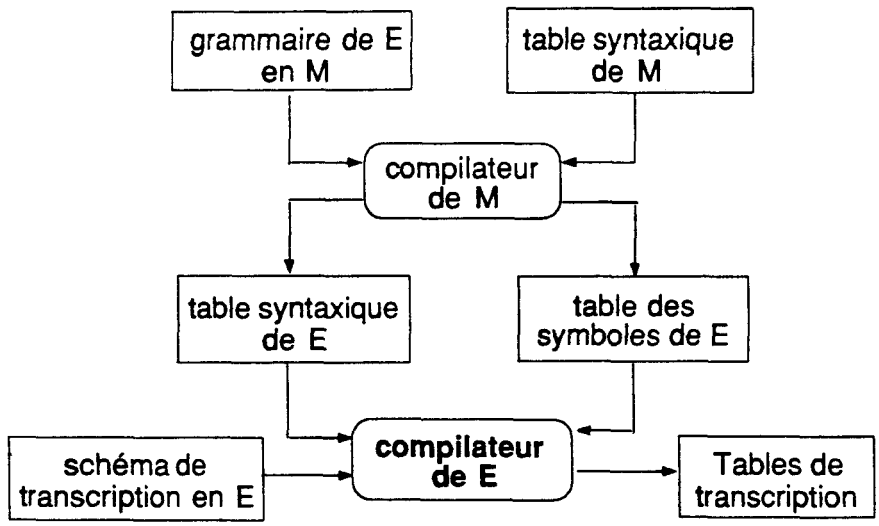



Figure 8.3 : Fonctionnement du compilateur E

## 2.2 Construction des tables de transcription

Les schémas de transcription écrits en langage E sont compilés et donnent lieu à des tables de transcription. La compilation des schémas est faite avant toute exécution de l'éditeur Grif. Quand Grif fonctionne, il charge préalablement ces tables en mémoire.

Les tables de transcription sont organisées pour réaliser le schéma suivant :

Expression d'entrée --> Transcription --> Code d'affichage

Suivant la méthode de saisie déclarée, un schéma peut donner différents types de tables de transcription. La figure 8.4 ci-dessous indique les liaisons entre les tables.

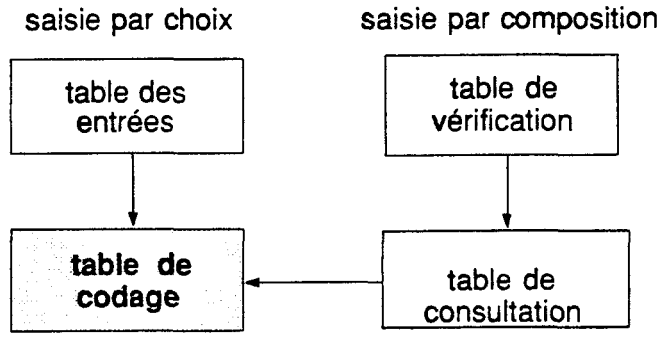



Figure 8.4 : Tables de transcription

La *table de codage* est constituée d'éléments à trois champs. Le premier est réservé à la représentation interne de la transcription du caractère. Le deuxième contient son code d'affichage dans la police déclarée. Nous indiquons plus loin comment on peut construire les noms de ces polices. Le troisième contient l'adresse de l'élément suivant. S'il s'agit du dernier élément, ce champ vaut zéro. Tous les éléments de la table sont triés alphabétiquement selon les champs de la clé de transcription afin de diminuer les temps de recherche.

Les *tables de consultation* et les *tables de vérification* ne sont créées que quand, dans le schéma de transcription, on a déclaré la méthode de saisie par composition des touches (à gauche ou à droite). La construction de ces deux tables, T1 et T2, est la même que celle que nous avons présentée dans le chapitre 5 pour le vietnamien. En effet, on dispose du vecteur V1 de M signes simples et du vecteur V2 de K caractères de base. Donc T1 est une matrice carrée MxM et T2 est une matrice (M+N)xK, où N est le nombre de signes composés. Cependant, T1 n'est créée que si, dans le schéma de transcription, on déclare qu'une lettre peut porter plusieurs signes diacritiques (deux au maximum actuellement). Dans ce cas, T2 donne l'adresse à consulter dans la table de codage.

Si, dans le schéma de transcription, on déclare la méthode de saisie par choix SELECT suivie d'un type de fenêtre, la *table des entrées* avec une fenêtre de saisie correspondante sera créée. Cette table contient des éléments à trois champs. Le premier donne l'entrée, qui est une chaîne de caractères prévue par le schéma. Le deuxième est un pointeur qui

indique l'adresse du premier élément de la table de codage correspondant à l'expression contenue dans le premier champ. Le dernier contient le nombre des éléments ayant la même expression de saisie. Comme la table de codage, la table des entrées est aussi triée alphabétiquement.

Dans la figure 8.5 ci-dessous, on présente en détail les structures de données pour la saisie par choix SELECT.

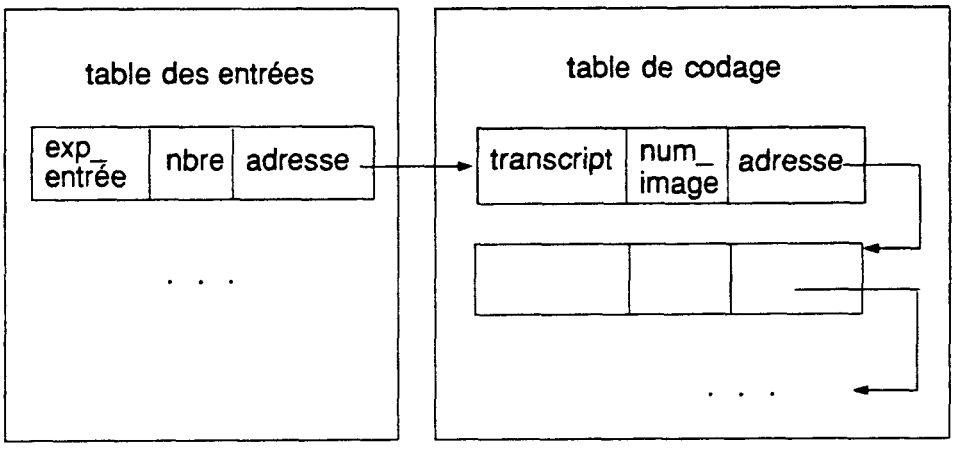



Figure 8.5 : Structures de données pour la saisie par choix

### Représentation du texte

Après avoir construit les tables de transcription, il s'agit de stocker des données dans les champs par le compilateur E, notamment les représentations internes dans les tables de codages. Pour cela, il faut chercher des règles d'écriture des transcriptions pour préciser la définition du symbole `Transcript` introduit dans la syntaxe du langage E.

On rappelle que, dans Grif, un élément textuel est représenté par une chaîne de caractères ASCII. Dans l'arbre abstrait, un nœud correspondant à un morceau de texte porte ou hérite un indicateur d'alphabet. Cette structure est conforme à la représentation pivot qui est définie par :

```
Texte = [ NomAlphabet ] ChaineCar OctetNul .
NomAlphabet = CARACTERE .
ChaineCar = STRING .
OctetNul = NIL .
```

{ Par convention, CARACTERE représente un seul caractère ASCII imprimable ; l'absence de l'indicateur indique l'alphabet latin par défaut ; la chaîne est terminée par un octet nul. }

Dans le sens du langage E, ChaineCar est remplacé par une suite de transcriptions ou une suite d'unités ; en revanche, l'indicateur d'alphabet est remplacé par le nom du jeu déclaré dans les paramètres d'entrée. En fait, un caractère dans le jeu donné correspond à une unité qui peut être l'un de trois type différents : simple, composé ou symbole d'échappement.

```
Texte = [ NomJeu ] ChaineCar OctetNul .
ChaineCar = < Unite > .
Unite = CarSimple | CarCompose | SymbEchapp .
```

L'unité CarSimple représente un caractère quelconque codé sur un octet : on revient au codage usuel de Grif.

```
CarSimple = CARACTERE .
```

Actuellement, on ne donne pas une description définitive pour le codage. Par l'ouverture du langage E, suivant les applications, on peut introduire de nouvelles représentations pour CarCompose.

En s'appuyant sur une transcription intermédiaire utilisée au GETA, on peut par exemple définir le codage des caractères avec signes diacritiques par deux parties : nom de caractère NomCar et diacritique Diacr. Cette transcription peut être adaptée aux alphabets non latin, comme arabe, hébreu, thaï...

La partie NomCar représente l'identificateur du caractère dans le jeu de caractères donné.

La partie Diacr peut être vide, s'il s'agit d'un caractère simple, mais elle peut contenir plusieurs signes diacritiques ou des sous parties diacritiques. Par exemple, un caractère arabe peut avoir deux signes diacritiques, tandis qu'un signe diacritique thaï peut se décomposer en voyelle et ton.

Dans ce genre de transcription, l'écriture d'un signe diacritique est constituée d'un indicateur de signe suivi du code numérique du diacritique.

```
CarCompose = NomCar [ Diacr ] | Diacr  .
NomCar = LETTRE .
Diacr = SousDiacr [ SousDiacr ] .
SousDiacr = IndicSigne CodeSigne .
IndicSigns = '!' .
CodeSigne = NUMBER .
```

La dernière unité SymbEchapp permet d'éviter l'ambiguïté dans la combinaison des caractères simples et composés dans le texte.

```
SymbEchapp = '\' .
```

**Exemple** : Voici quelques représentations internes utilisant la transcription intermédiaire :

| *Alphabet :* | *Forme de transciption :* | *Représente :* |
|---|---|---|
| Latin | Caracte!2re | Caractère |
| Latin | a\\b | a\b |
| Latin | Co!3te!1\0 | Côté0 |
| Vietnamien | tie!3!1n | tiến ("avancer") |
| Grec | Euxaistw | Ευχαιστῶ("Merci") |

On trouvera en annexe l'utilisation de la transcription intermédiaire dans l'écriture des schémas de transcription de l'alphabet latin et du vietnamien.

### Structures de données de Grif

Avec l'ajout des tables de transcription, les principales structures de données de Grif sont données dans la figure 8.6 (phase d'édition) :

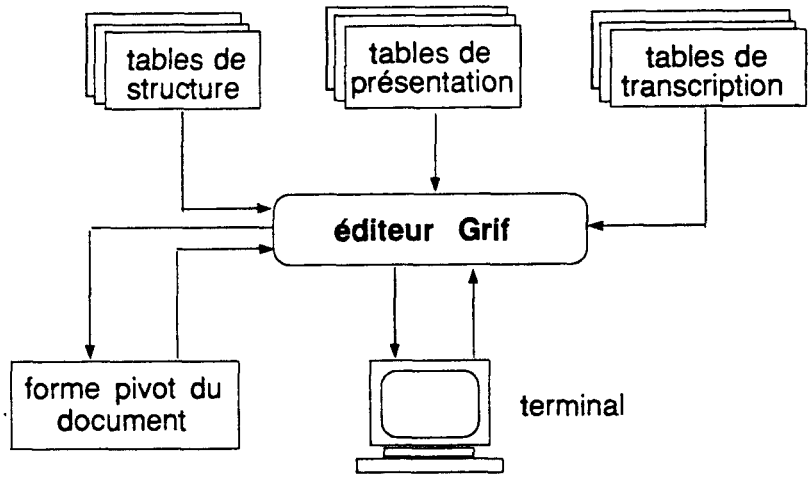



Figure 8.6 : Structures de données de Grif

### Fenêtre de saisie

La fenêtre de saisie, dont les caractéristiques sont identiques aux fenêtres de dialogue de Grif, est créée en même temps que son ensemble de sous-fenêtres. Les deux premières sous-fenêtres sont réservées à l'affichage du type de fenêtre de saisie et de l'entrée au clavier (écho). Il y a deux sous-fenêtres particulières, '<' et '>', qui sont deux cases de commande. Les sous-fenêtres restantes sont réservées à l'affichage des caractères correspondant à une entrée, ordonnés par fréquences croissantes. Le rang est indiqué, et le défilement est possible.

Sur la fenêtre de saisie, on peut exécuter les commandes suivantes :

* `AGauche` : fait défiler à gauche la liste des caractères ayant la même expression d'entrée.
* `ADroite` : fait défiler à droite.
* `ESC` : annule une entrée.
* `Choix` : désigne le caractère voulu et valide le choix.
* Caractère arbitraire ? ou * (joker).

Sur la figure 8.7, il y a 9 sous-fenêtres. Cependant, il est difficile d'en prévoir le nombre, comme on l'a vu sur les définitions du codage dans un schéma. Suivant les besoins, on peut fixer approximativement ce nombre de sous-fenêtres.

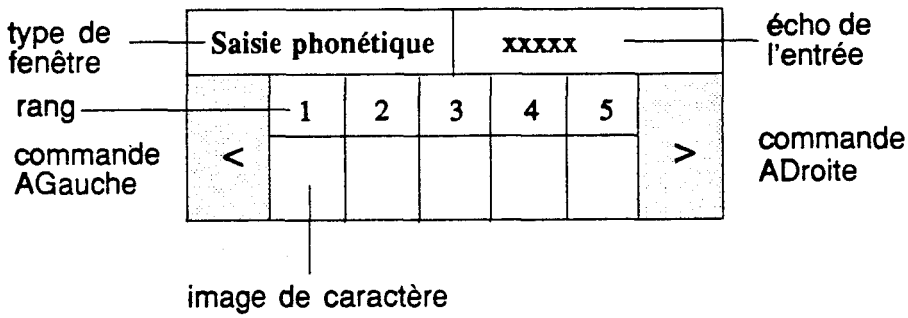



Figure 8.7 : Fenêtre de saisie phonétique

## 2.3 Gestion des ressources

Les paramètres de sortie dans un schéma de transcription permettent la construction des noms de polices de caractères. Par convention, un nom est créé par la concaténation, dans l'ordre, du nom d'alphabet (ou non), de la famille de police, du style (s'il y en a) et enfin de la taille des caractères. Ainsi :

```
NomPolice = [ NomAlphabet ] NomFamille [ Style ] Corps .
```

L'absence du nom d'alphabet ou du style dans le nom de police construit dépend du nom de polices disponibles dans les ressources de Grif. On rappelle que ce sont des noms au format BDF sous X-Window.

**Exemple** : Dans le schéma de transcription du chinois, on déclare :

```
Family = ( 'Beijing' );
Size = ( 16, 24 );
```

On peut alors construire les deux noms de polices `Beijing16` et `Beijing24`, qui correspondront aux deux fichiers `beijing16.snf` et `beijing24.snf` du format SNF.

Avec les outils de création de nouvelles polices, notamment `xfedor` (voir chap 5), l'enrichissement des ressources est possible. Par exemple, on peut introduire des polices de caractères de nouveau style, comme souligné, relief..., puis les déclarer dans les paramètres de sortie d'un schéma de transcription. Il s'agit d'une compatibilité avec l'impression, soit directe en langage PostScript, soit par traduction vers un formateur.

## 2.4 Écriture des intitulés de dialogue

Au cours de l'édition du document, la langue de dialogue est déterminée par une commande, appelée `DIALOGUER` (voir chapitre 7). Le menu de cette commande, identique à celui de la commande de changement d'alphabets `Alphabets` de Grif, constitue une liste des noms des alphabets disponibles. Or, la commande `Aphabets` agit sur les éléments textuels du document. Par contre, la commande `DIALOGUER` agit uniquement sur les intitulés de dialogue, après la détermination de la langue de dialogue.

Cette langue permet alors d'utiliser la police de dialogue et les paramètres de sortie nécessaires à partir des tables de transcription correspondantes. Il s'agit d'effectuer une traduction entre les représentations des intitulés textuels de dialogue avant de les afficher sur les sous-fenêtres.

En fait, il y a deux formes de représentation de ces intitulés, interne et externe. Les intitulés internes, servant à l'affichage, sont créés temporellement par la commande `DIALOGUER` et résident en mémoire. Ce sont des chaînes de caractères formées par la concaténation des numéros d'image, ou codes d'affichage dont chacun est représenté sur un ou deux octets, suivant la police de dialogue courante. Une modification des structures de données dans l'interface utilisateur de Grif est alors indispensable.

Les intitulés externes sont créés par le superutilisateur et sauvegardés dans l'ensemble des schémas (en S, en P et en E) et des fichiers de messages. Dans l'implémentation, cet ensemble est appelé *ensemble des fichiers de dialogue*. Les intitulés externes sont écrits en s'appuyant sur le type de transcription défini et utilisé dans les schémas de transcription (présenté plus haut).

Dans la stratégie "intégration des méthodes" (voir chapitre 7), nous avons déjà proposé trois types d'intitulés externes : structural, fixe et complémentaire. Cette méthode ajoute directement les intitulés complémentaires dans les modules concernés. Pour les différencier d'avec la méthode "à la main", les intitulés complémentaires seront appelés les *intitulés linguistiques* (dus des schémas de transcription).

La séparation des intitulés de dialogue suivant leur nature montre son intérêt dans la gestion dynamique des langues traitées. Parce que les messages sont toujours fixés à l'avance, tandis que les intitulés linguistiques, fournis par les tables de transcription, sont toujours variables.

La langue de dialogue, qui est déjà déclarée dans un schéma de transcription, peut être donnée en paramètre lors l'appel de Grif sous Unix. Dans le but de faciliter l'écriture des

intitulés externes, nous utilisons l'anglais comme langue de dialogue par défaut. Ainsi, les fichiers de dialogue écrit en anglais sont considérés comme les *fichiers de messages par défaut.*

La figure 8.8 présente la traduction de forme externe en forme interne des intitulés de dialogue par la commande `DIALOGUER`.

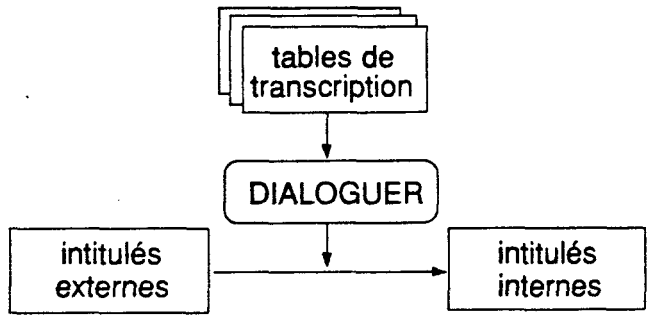



Figure 8.8 : Transduction des intitulés de dialogue

# 3 Perspectives

L'implémentation du langage E est terminée. Bien sûr, il faudra étendre l'ensemble des langues disponibles, mettre au point l'ensemble de la réalisation, et chercher à ce qu'elle soit réutilisable dans d'autres versions de Grif.

Comme on l'a vu, il faut aussi étendre S, P et T pour traiter un certain nombre de points non traitables au niveau de E, et/ou qui concernent clairement la structure abstraite, la présentation ou la traduction (vers PostScript ou autres).

## 3.1 Travail en cours

### Augmentation du nombre de systèmes d'écriture traités

Le problème est de traiter des signes diacritiques plus complexes (thaï...), des variations morphologiques, ou des ligatures (arabe, hébreu, grec...) pour enrichir l'ensemble de langues traitables. La solution est d'introduire des nouveaux modules supplémentaires indépendants, notamment des méthodes de saisie et de restitution, en résolvant la combinaison de plusieurs signes diacritiques, l'analyse de contexte, etc.

### Adaptation aux nouvelles versions de Grif

Il y a deux possibilités d'adaptation : le groupement des déclarations (types, constantes, variables, procédures externes), des données (images des caractères chinois à imprimer, par exemple), et des programmes de traitement multilingue dans de nouveaux modules (ou répertoires) indépendants, et la localisation des modifications.

Le groupement n'engendre pas beaucoup de difficultés, car l'adaptation se fait au niveau de la liaison des modules compilés et de la définition des procédures externes (en Pascal et en C).

En revanche, la localisation permet d'adapter parfaitement à la nouvelle version de Grif les parties étendues localisées, notamment le traitement des structures de données textuelles, l'appel des procédures et la modification des versions de X-Window (en X11R4, par exemple). L'adaptation demande d'abord de comparer deux versions, puis d'insérer ou de modifier suivant le cas.

### Utilisation de plusieurs schémas de transcription

Afin de simplifier le travail, nous avons supposé dans l'implémentation actuelle du langage E que chaque jeu de caractères introduit n'a qu'un seul schéma de transcription activable dans Grif. S'il existe une langue qui est déclarée sur deux, ou plusieurs schémas de transcription différents, cela peut entraîner des ambiguïtés. Le problème posé est la possibilité d'utiliser en même temps plusieurs schémas de transcription (pour chaque jeu de caractères disponible).

La solution est de gérer simultanément plusieurs méthodes de saisie des éléments du jeu de caractères donné. En fait, en s'appuyant sur le codage déterminé et sur les ressousces d'affichage correspondantes de ce jeu, les schémas de transcription se distinguent uniquement par la déclaration de la méthode de saisie et des entrées dans les règles de transcription. Dans ce cas, les noms de schéma peuvent être indexés. C'est le compilateur E qui fournira un nouveau menu pour basculer entre les méthodes de saisie disponibles.

Par exemple, on décrirait dans le schéma de transcription `Vietnamien1.SCH` la méthode de composition à gauche, et dans le schéma `Vietnamien2.SCH` la méthode de composition à droite. La compilateur E reconnaitrait le nom d'alphabet utilisé en regroupant ces deux schémas.

### Transduction de deux formes de représentation pivot

L'utilisation des transcriptions pour le codage, ou particulièrement la modification des codes existants dans certains cas donne une différence inévitable entre la forme pivot de la version prototype de Grif et celle de l'extension. La conséquence directe de cette différence est que la forme pivot d'un document déjà édité et stockée dans un fichier (forme pivot prototype) est devenue incompréhensible pour la nouvelle version multilingue de Grif.

Pour traduire la forme pivot prototype vers la forme pivot étendue, il faut construire un transducteur qui utilise aussi des tables de transcription. En principe, ce programme fait correspondre un caractère qui a un code d'affichage à une représentation en langage E.

La figure 8.9 ci-dessous présente la transduction de deux formes pivots.

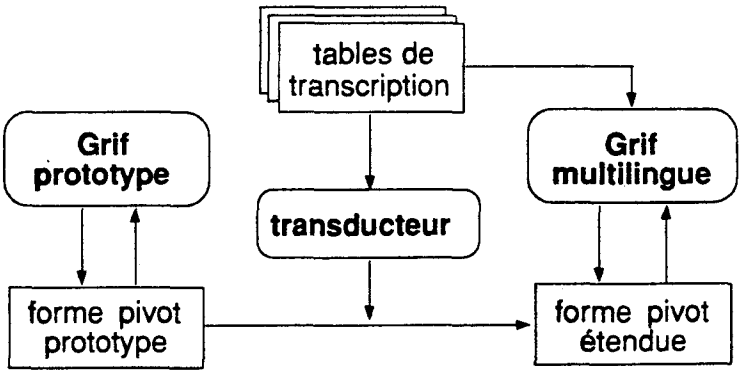



Figure 8.9 : Traduction de l'ancienne forme pivot vers la nouvelle

## 3.2 Extension des autres langages de Grif

### Extension de S et de P pour compléter le dialogue

Dans la réalisation actuelle, le traitement des intitulés de dialogue n'est pas encore adéquat. Nous avons conservé la syntaxe de S et de P, et créé parallèlement pour chaque langue disponible un schéma correspondant (avec l'indexation des noms de schéma).

Par exemple, pour les intitulés structuraux, il existe des schémas de structure et de présentation parallèles pour le vietnamien, le grec, le chinois, etc. Dans les schémas pour le chinois actuellement, un caractère chinois (jeu de caractères hanzi) est représenté par deux caractères ASCII imprimables (en code GB). Ainsi, pour la classe des articles scientifiques, nous avons écrit des schémas de structure `ArticleVIE.SCH` et de présentation `ArticleVIEP.SCH` pour le vietnamien, de même `ArticleHAN.SCH` et `ArticleHANP.SCH` pour le chinois, etc.

Nous avons également un ensemble de schémas de transcription dans les différentes langues pour les intitulés linguistiques et un ensemble de fichiers de messages.

La solution proposée est d'étendre les langages S et P en même temps. En utilisant le type de transcription, chaque intitulé textuel (nom d'élément et d'objet structurés, titres de constante, etc.) est écrit complètement dans toutes les langues disponibles.

**Exemple** : On peut déclarer la constante 'Résumé :' en anglais, en français et en vietnamien dans un schéma de présentation comme suit :

```
CONST
     CsteResume = TEXT BEGIN
                  Anglais : 'Abstract :';
                  Français : 'Re!lsume!l :';
                  Vietnamien : 'To!lmta!12!lt :';
                  END ;
```

Cette solution est bien plus cohérente et facile à maintenir, bien qu'il faille modifier et recompiler tous les schémas de structure et de présentation chaque fois qu'une nouvelle

langue est ajoutée. Comme il s'agit d'une modification assez lourde, nous n'avons pas cru pouvoir la mener à bien dans le cadre de ce travail.

Une solution plus simple est d'étendre seulement les compilateurs de S, de P et de E. Le principe est de ramasser tous les intitulés textuels de dialogue dans tous les schémas disponibles, puis de les traduire (à la main) en forme externe en utilisant le type de transcription défini, et de les sauvegarder dans des fichiers de dialogue.

Les quatre étapes de la réalisation sont alors :

1. *écriture des schémas* : tous les schémas (S, P et E) ne sont qu'écrits en anglais (avec la syntaxe déjà définie pour chaque langage) ;
2. *ramassage des intitulés de dialogue* : la compilation des schémas disponibles ramasse les intitulés textuels en anglais, puis les rassemble dans les fichiers particuliers, appelés *fichiers d'intitulés* ; les intitulés peuvent être séparés simplement par un retour de ligne ;
3. *transcription (ou traduction) à la main* : à partir des fichiers d'intitulés, le superutilisateur (ou l'utilisateur, s'il peut le faire) transcrit chaque ligne dans la langue donnée selon le type de transcription correspondant ;
4. *application* : Grif utilise les fichiers d'intitulés en passant de l'un à l'autre si l'utilisateur change de langue de dialogue ;

Dans la deuxième étape, il est possible de séparer les intitulés en anglais en trois fichiers différents suivant les catégories (S, P et E). Cette méthode donne un classement cohérent, mais, produit un nombre de fichiers appréciable.

Du point de vue pratique, la troisième étape demande un travail proportionnel au nombre de langues introduites et de schémas disponibles, et certainement assez lourd.

### Extension de S et de P pour la restitution

Nous pouvons appliquer l'extension des langages S et P pour définir les nouveaux types d'objets, comme ceux de paragraphe, servant à la mise en paragraphes. Cette solution a déjà été proposée dans le chapitre 7.

### Extension de T pour la traduction

Nous pouvons aussi appliquer l'extension du langage T pour traduire un document multilingue vers un formateur, comme nous l'avons proposé dans le chapitre 7.

## 3.3 Introduction d'un nouveau langage

Pour résoudre effectivement les problèmes de césure ou de typographie fine, il est vraisemblable qu'il faudra introduire un autre langage. Ce langage devrait permettre d'écrire des règles grammaticales, ainsi que des dictionnaires. En tout cas, il faudra aussi assurer la cohérence avec les langages déjà définis.

# Chapitre 9 : Application au chinois



Imprimé par Grif le 20/3/91

# 1 Informatisation du chinois

## 1.1 Introduction à l'écriture chinoise

L'écriture chinoise [II Alleton 76] [II Cousquer 88, 89] [II Malherbe 83] est l'une des plus anciennes écritures utilisées aujourd'hui. Elle comprend dizaines de milliers de caractères idéographiques *hanzi* ( 汉 字 ) dont chacun correspond à une syllabe parlée (parfois à deux, rarement à trois). Il n'y a pas de rapport systématique entre la prononciation et la graphie.

La graphie, ou le dessin d'un caractère, est une composition de traits élémentaires inscrits fictivement dans un carré de taille constante. Certains assemblages de traits apparaissant dans plusieurs caractères sont appelés clés ou radicaux et ont la signification. Le nombre de clés, varie suivant les analyses calligraphiques des dictionnaires. Les caractères sont tracés selon un ordre très précis.

Par exemple, la clé 广 ("petite maison") se prononce *yán* et se dessine par trois traits dans l'ordre 丶 — et 丿. Si l'on continue à dessiner en ajoutant à ce caractère les trois traits dans l'ordre — 丨 et —, on obtient le caractère 庄 ("village") qui se prononce *zhuāng*, (du premier ton) etc.

C'est pour simplifier et normaliser l'écriture chinoise qu'on a mené en RPC des réformes. Pour la graphie, on a réduit le nombre de traits de certains caractères complexes et supprimé des caractères qui en étaient des variantes. Avec cette simplification, un caractère peut être en forme traditionnelle, ou en forme simplifiée. Dans l'écriture chinoise moderne actuelle, environ 8 000 caractères différents sont utilisés.

Pour la prononciation, on utilise la transcription phonétique *pinyin*( 拼 音 ) de la langue de Pékin (han). L'unité phonétique est formée d'une syllabe et d'un ton. La transcription pinyin ne code que les syllabes. Une syllabe est composée de deux parties : l'une des 36 voyelles, comme a, en, iao, iang... éventuellement précédées d'une des 23 consonnes, comme b, ch, zh... Le pékinois n'utilise effectivement que 420 syllabes. En principe, chaque syllabe peut se prononcer selon 5 tons différents : continu -, montant ˊ, rentrant ˇ, descendant ˋ et faible (ou neutre). En pratique, le pékinois n'utilise qu'environ 1 400 prononciations parmi les 2 100 possibles.

Il y a aussi le problème de correspondance graphie-prononciation. Une prononciation peut correspondre à plusieurs caractères homophones ; inversement, un caractère peut avoir plusieurs prononciations différentes, souvent par un changement de ton. Par exemple, la prononciation *guǎng* donne deux caractères 犷 ("sauvagerie") et 广 et, le caractère 了 peut avoir les trois prononciations dont chacune a une signification : *le* ("particule"), *liǎo* ("finir"), *liào* ("regarder loin).

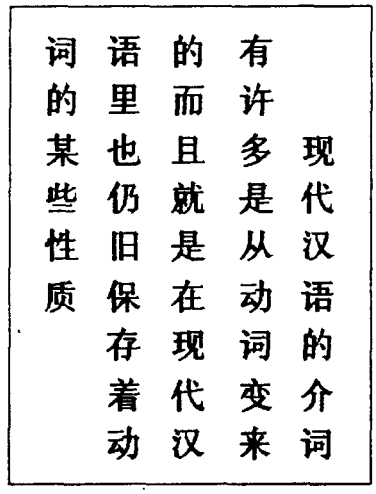
Dans l'écriture traditionnelle, les caractères chinois sont rangés en colonnes verticales, de haut en bas et de droite à gauche. Il existe aussi une disposition en lignes horizontales, écrites de droite à gauche, qu'on trouve dans certains ouvrages anciens. Actuellement, la disposition horizontale de gauche à droite est adoptée en RPC et devient publique. Toutefois, les trois possibilités peuvent coexister dans un même document (journal, par exemple), surtout à Taiwan.

Le chinois ancien était monosyllabique. Par contre, en chinois moderne, un mot est souvent polysyllabique, et s'écrit avec autant de caractères que de syllabes. Dans ce cas, on peut relier leur pinyin pour former une unité grammaticale.

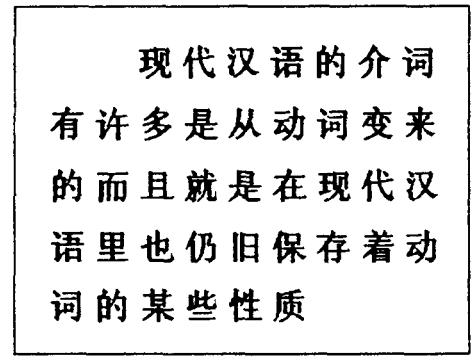
La figure 9.1 présente deux façons d'écrire, verticalement et horizontalement (avec le renfoncement) d'un texte chinois dont le pinyin est écrit en reliant les transcriptions des caractères :

*xiàndài hànyǔ de jiècí jǒu xǔduō shì cóng dòngcí biànlái de érqiě jiùshì zài xiàndài hànyǔ lǐ, yě réngjiù bǎocún zhe dongcí de mǒuxiè xìngzhì*

Traduction : "En ce qui concerne les prépositions *(jiècí)* du *(de)* chinois *(hànyǔ)* moderne *(xiàndài)*, il y en a *(yǒu)* beaucoup *(xǔduō)* qui sont *(shì...de)* dérivées *(biànlái)* de *(cóng)* verbes *(dòngcí)* et *(érqiě)* même *(jiùshì)* dans *(zài...lǐ)* la langue chinoise *(hànyǔ)* moderne *(xiàndài)*, elles conservent *(bǎocún zhe)* aussi *(yě)* encore *(réngjiù)* certaines *(mǒuxiè)* caractéristiques *(xìngzhì)* des verbes *(dòngcí)*.




écriture en colonne de haut en bas, de droite à gauche




écriture en ligne de haut en bas, de gauche à droite

Figure 9.1 : Deux dispositions d'un texte chinois

## 1.2 Codification

Les systèmes de codage sur deux octets des caractères chinois (GB 2312-80 en RPC, JIS C 6226 au Japon et BIG-5 en Taiwan) sont largement utilisés à l'heure actuelle. Cependant, pour des applications diverses, on a proposé plusieurs méthodes qui utilisent plusieurs octets par caractère, de taille fixe ou variable. Parmi eux, on peut citer : le codage utilisant six chiffres [III.1 Qiao 90] ou huit lettres fixes [III.1 Tao 66], le codage étendu [III.1 Cousquer 89] et le codage phonétique-structural PS [Boitet & Tchéou 90].

### Code GB 2312-80

Le code GB 2312-80 (désormais, on l'appellera simplement code GB, ou norme GB), qui a été publié en 1981 à Pékin (RPC) [III.1 GB 81], contient 6763 caractères chinois simplifiés usuels. À chacun est associé un code de deux octets. Ce codage contient également les alphabets latin, grec, cyrillique, les syllabaires chinois et japonais (katakana et hiragana), et divers symboles graphiques et typographiques.

On trouve dans le code GB deux caractéristiques : chaque code est formé de deux codes ASCII standard (imprimables), et les caractères sont répartis dans deux matrices qui correspondent à l'ordre pinyin (et ton) et à la nature de la graphie. Il y a le problème des caractères homophones (plusieurs caractères pour un pinyin), et un caractère peut avoir plusieurs pinyin.

La première matrice, à 40 lignes (de 16 à 55) et 94 colonnes (de 1 à 94), contient 3755 caractères (et 5 trous), classés en colonnes selon l'ordre pinyin. La deuxième matrice, à 32 lignes (de 56 à 87) et 94 colonnes, contient 3008 caractères, classés selon un système de clés (186). Donc les coordonnées d'un caractère chinois dans le code GB sont déterminées par le numéro de ligne (de 16 à 87) suivi d'un numéro de colonne (de 1 à 94). Les lignes de 1 à 9 contiennent les autres caractères. On n'utilise pas les lignes de 10 à 15.

Selon A. et E. Cousquer (Université de Lille 1 & CNRS) [II Cousquer 88b], d'une part, l'ordre des caractères dans ce code n'est pas homogène et on ne peut pas effectuer les tris lexicographiques. À titre d'exemple, les caractères de même clé dans ce code sont classés suivant le nombre de traits, puis pour une partie suivant la nature du premier trait. Pour une autre partie, ils sont classés suivant le pinyin en ordre alphabétique. D'autre part, le code GB ne correspond pas à une prononciation unique. Enfin, pour le travail en lexicographie et en linguistique chinoises, il est nécessaire de compléter ce code.

### Code utilisant six chiffres

Proposé par J. Qiao, Y. Qiao et S. Qiao [III.1 Qiao 90], le principe de codage est simple. Selon la décomposition graphique, chaque caractère chinois est divisé en six sections structurées dont chacune est représentée par un chiffre décimal de 1 à 6 (figure 9.2).

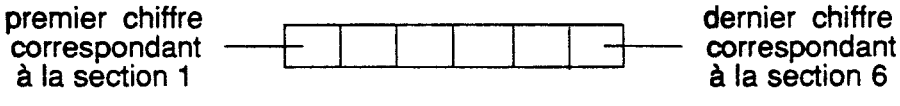



Figure 9.2 : Division d'un caractère chinois et son code

Il y a quatre types de structure :

1. Simple et double (figure 9.3a) : quatre structures simples correspondent aux sections 1 (coin haut-gauche), 2 (coin bas-gauche), 3 (coin haut-droite) et 4 (coin bas-droite) ; deux structures doubles correspondent aux sections 5 (combinaison de 1 et 3) et 6 (2 et 4).
2. Deux triplets en haut et en bas (figure 9.3b) : celui du haut contient trois sections 1 (gauche), 5 (au milieu) et 3 (droite) ; le groupe en bas contient trois sections 2, 6 et 4.
3. Simple-double en haut et triplet en bas (figure 9.3c) : le groupe du haut contient trois sections 1 (gauche), 3 (droite) et 5 (combinaison de 1 et 3) ; celui du bas contient trois sections 2 (gauche), 6 (au milieu) et 4 (droite).
4. Triple en haut et simple-double en bas (figure 9.3d) : le groupe du haut contient trois sections 1 (gauche), 5 (au milieu) et 3 (droite) ; le groupe bas contient trois sections 2 (gauche), 4 (droite) et 6 (combinaison de 2 et 4).

La figure 9.3 présente la division et le codage d'un caractère du premier type (structure simple et double) :

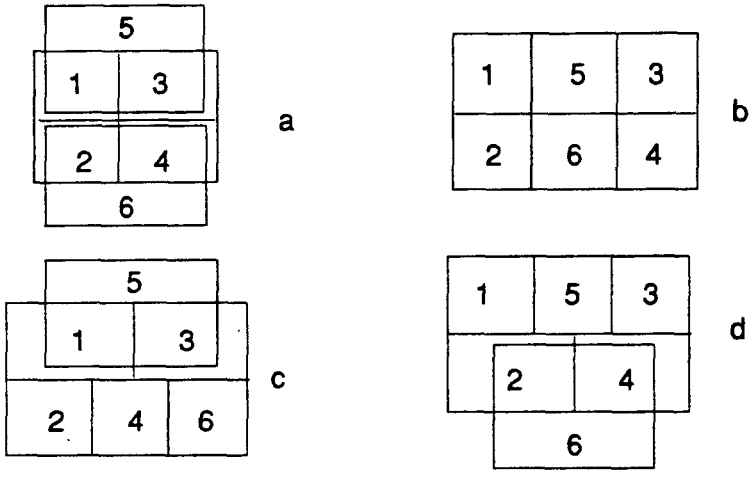



Figure 9.3 : Division d'un caractère chinois et son code

Le codage d'un caractère est effectué en quatre phases :

1. choisir un type de structure (de 1 à 4) ;
2. décomposer en 6 sections correspondant au type choisi ;
3. à l'aide d'une table donnée, déterminer le code de chaque section occupée par un groupe de traits (dans la table, chaque groupe de traits correspond à un code de 1 à 9) ;
4. former le code complet en reliant six chiffres (les sections inoccupées prennent valeur zéro).

Par exemple, le caractère 占 ("occuper") est classé dans le type 1 avec la décomposition suivante : la section 2 contient 口 qui appartient au groupe 9 ; la section 3 contient — du groupe 1 et la section 5 contient | du groupe 3 ; enfin, les sections 1, 4 et 6 ne sont pas occupées et prennent la valeur zéro. On obtient donc le code 091030. La décomposition est indiquée dans la figure 9.4 suivante :

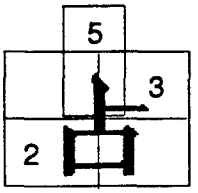



Figure 9.4 : Structure double haut-bas

Il y a aussi une autre méthode de codage présentée dans [III.4 Tao 66] qui utilise huit caractères alphanumériques latins en format fixe. Son principe est aussi la décomposition graphique d'un caractère en traits élémentaires selon les règles d'écriture (position et direction). Par exemple, le caractère 面 ("visage") est codé par la suite `'_/OI--IO'` correspondant à la décomposition des traits — 丿 口 | — — | respectivement. Si le nombre de traits est inférieur à huit, la suite est complétée par des zéros.

### Code étendu

D'après A. Cousquer [III.1 Cousquer 89], le principe de codage est de remplacer chaque caractère chinois par un mot de telle sorte que les algorithmes de tri alphabétique classiques pour l'anglais et pour le français permettent d'obtenir un ordre lexicographique cohérent. Cela conduit à un code comportant nettement plus de deux octets par caractère chinois. On considère que ce n'est plus un obstacle, étant donnés les progrès rapides dans les tailles de mémoire des ordinateurs.

La proposition est donc d'utiliser un système de codage où, en utilisant des conventions du méta-langage utilisé dans Grif, le code d'un caractère chinois serait de la forme :

```
Ideogramme = Pinyin Discriminant Police Code_Info .
```

Le `Pinyin` dans le code se compose d'une `Syllabe` qui est une chaîne de caractères alphabétiques suivie d'un `Ton` qui est un chiffre.

```
Pinyin = Syllabe Ton .
Syllabe = STRING .
Ton = '0' / { ton faible }
      '1' / { ton continu - }
      '2' / { ton montant ´ }
      '3' / { ton rentrant ˇ }
      '4' . { ton descendant ` }
```

Le `Discriminant` contient le nombre de traits du caractère suivi du numéro de sa clé. Pour réduire le nombre d'octets utilisés dans la codification, on peut représenter les deux composants `NbTraits` et `NumCle` sur deux octets.

```
Discriminant = NbTraits NumCle .
NbTraits = NUMBER .
NumCle = NUMBER .
```

La `Police` indique le jeu choisi qui est code GB. C'est aussi un code sur un octet.

```
Police = NUMBER .
```

Enfin, le `Code_Info` est un code sur deux octets repérant la graphie dans ce jeu : par exemple, le code GB dans le cas de cette norme.

```
Code_Info = Octet1 Octet2 .
Octet1 = CHARACTER .
Octet2 = CHARACTER .
```

Cette codification offre l'avantage de l'ouverture : tout caractère introduit y trouve sa place sous forme simplifiée ou non ; elle n'est pas limitée en nombre comme peut l'être la norme GB.

### Code PS

Le codage phonético-structural PS a été proposé par Ch. Boitet et F. Tchéou (GETA, IMAG, UJF & CNRS) [III.1 Boitet & Tchéou 90]. En utilisant un ensemble réduit de caractères contenu dans l'alphabet du langage PL/I, et donc dans l'ISO 646, un code d'un caractère chinois est de la forme répétive :

```
Ideogramme = Syllabe Ton '-' NbTraits [ '/' Cle ] .
```

La `Syllabe` est écrite avec des lettres majuscules de A à Z et correspond au pinyin du caractère. Le nombre de lettres dans une syllabe va de 1 (comme A, M...) à 6 (comme CHUANG, SHUANG, ZHUANG...). Le caractère Ü (U tréma) est remplacé par "U:" car le tréma n'est pas universel dans les codages actuels.

```
Syllabe = STRING .
```

Le `Ton` prend une valeur entière de 1 à 5 correspondant aux tons continu, montant, rentrant, descendant et faible.

```
Ton = '1' / '2' / '3' / '4' /'5' .
```

Le signe `'-'` permet de séparer le `Ton` et le `NbTraits` qui est le nombre de traits utilisés pour dessiner le caractère.

```
NbTraits = NUMBER .
```

Les éléments `Syllabe`, `Ton` et `NbTraits` ne suffisent pas à la représentation d'un caractère dans certains cas, parce que plusieurs caractères chinois peuvent avoir la même syllabe, le même ton et le même nombre de traits. La décomposition du caractère avec sa clé sémantique permet le plus souvent d'éviter l'ambiguïté. La clé d'un idéogramme est aussi représentée de la même façon et peut être ambiguë, car plusieurs caractères peuvent avoir la même syllabe, le même ton, le même nombre de traits et la même clé, et ainsi de suite. On itère alors cette décomposition, après par un signe `'/'` jusqu'à éliminer l'ambiguïté.

```
Cle = Ideogramme .
```

**Exemple** : La représentation complète `HAO3-6/NU:3-3` représente le caractère chinois 好 "bon" en français. Ce caractère est transcrit par le pinyin `HAO` au troisième ton (rentrant) et comporte 6 traits avec la clé sémantique `NU:3`( 女 , "femme"), elle-même au troisième ton et à 3 traits.

## 1.3 Manipulation des caractères chinois

À cause de la complexité du problème, on a développé plusieurs méthodes différentes. Pour un système de traitement informatique de texte, on peut utiliser une méthode unique, mais il existe des systèmes qui permettent de manipuler en même temps plusieurs méthodes. En principe, on peut les diviser en trois classes : saisie directe, saisie phonétique et saisie par décomposition graphique.

### Saisie directe

À l'aide de la disposition de tous les caractères possibles d'une façon particulière (en tableau bidimentionnel, ou en pages tournantes), l'utilisateur peut entrer directement le caractère voulu sur un "grand clavier" à touches par les étapes de choix. Cependant, le temps d'apprentissage de cette méthode est long et le nombre de caractères disponibles est relativement limité.

### Saisie phonétique

Très largement utilisée [III.4 Archer & alii 88], [III.4 IEEE 85], [III.4 Mohr 82], la saisie phonétique est adoptée dans plusieurs systèmes commercialisés, comme *TianMaII* ( 天马 ) [III.4 Asia 89] sur les PC, *WinText* de WinSoft (voir chapitre 1) sur les Macintosh, etc.

Le principe de la méthode (le même que la saisie par choix que nous l'avons présentée au chapitre 2) est simple. En utilisant un clavier latin, l'utilisateur entre le pinyin (6 lettres au maximum) suivi éventuellement d'un ton (de 1 à 4) du caractère, et puis, il choisit le caractère voulu parmi la liste des caractères homophones correspondants. La longueur d'une telle liste atteint quelquefois une centaine selon les dictionnaires choisis. Par exemple, l'entrée *yi4* correspond à 60 caractères simplifiés différents selon la norme GB,

à 71 caractères selon le dictionnaire *Xinhua* ( 新 华 ), et à 154 selon le dictionnaire *Cihai*( 辞 海 ).

Pour augmenter la performance de la méthode, on a effectué des améliorations, notamment sur l'entrée pinyin, ou sur le choix du caractère voulu dans la liste homophone. Par exemple, sur l'entrée pinyin, dans le système TienMa, on n'entre que le pinyin pour le choix d'un caractère ; mais on peut obtenir directement le résultat par la combinaison de deux caractères successivement en une entrée (comme *hanyu* pour 汉 语 "langue chinoise"), ou par l'utilisation des règles simplifiées (comme l'entrée majuscule *Li* au troisième ton pour 李 qui est un nom de famille), etc.

En ce qui concerne la sélection du caractère cherché, on compte sur la fréquence des caractères utilisés dans un document. En fait, la fréquence n'est pas unique. Dans le projet ALEXIS réalisé au GEDIS, A. Cousquer a proposé un système de gestion dynamique des listes homophones pour la méthode pinyin dérivé : le dernier caractère utilisé est toujours automatiquement placé en tête de la liste en cours. L'intérêt considérable de la méthode réside dans la vitesse de saisie et le maintien de l'ordre identique des caractères homophones pour chaque utilisation propre.

Actuellement, il y a environ 500 méthodes brevetées marchant avec les claviers usuels PC (sur environ 1 000 méthodes étudiées en RPC), et les meilleures, comme celle du Pr. Li à la SJTU, permettent d'entrer en moyenne 1,5 caractères par touche frappée, grâce à l'utilisation de collocations. Donc il faut pouvoir intégrer facilement et simultanément plusieurs méthodes.

### Saisie par décomposition graphique

Il y a plusieurs méthodes : décomposition en traits élémentaires de Hanze Talk [III.4 WinSoft 90], méthode d'analyse "des quatre coins" [III.4 Tao 66], division en six sections présentée plus haut, etc.

On peut aussi citer la méthode *wubizixing* ( 五 笔 字 型 "composition par cinq traits"), inventée par Wang Y. M. [III.4 Zhang & Cousquer 88] en RPC. Le principe de la décomposition est basé sur la fréquence d'apparition de ces éléments graphiques qui peuvent être des clés. Il y a au total 130 motifs d'éléments sélectionnés et distribués sur les touches d'un clavier QWERTY. L'organisation de ces touches permet de faciliter l'apprentissage. Le nombre de frappes en moyenne pour obtenir un caractère de la méthode ne dépasse pas quatre touches.

C'est plus que les meilleurs méthodes destinées aux chinois, mais c'est utilisable par des gens qui ne savent pas le chinois et doivent occasionnement entrer des caractères sans en connaître la prononciation.

À titre d'exemple, le caractère 好 ("bon") est décomposé en trois éléments 女 了 et — correspondant aux trois touches successives V, B et G. Cependant, les deux dernières peuvent être composés pour une seule frappe B correspondant à 子 .

Dans les méthodes dérivées (du projet ALEXIS au GEDIS, par exemple), une entrée est terminée par un blanc. Mais pour chaque touche frappée au clavier, si l'entrée courante

est correcte, on fait afficher immédiatement tous les caractères correspondants pour augmenter la vitesse de saisie. Par exemple, les entrées V, VB et VBG sont correctes. L'utilisateur peut entrer un blanc quand le caractère voulu est affiché.

## 1.4 Problème de création des polices

La difficulté principale de l'affichage des caractères chinois est de créer les polices. nécessaires. Les recherches portent sur la réduction de la place de stockage et du travail manuel à effectuer.

La taille de la matrice carrée contenant l'image d'un caractère chinois dépend du degré de résolution du matériel utilisé. Avec des milliers de caractères usuels, le stockage demande une espace de mémoire énorme.

Par exemple, on dispose actuellement au GEDIS de deux polices au format SNF avec des matrices 16x16 et 24x24, et d'un jeu utilisant des matrices 28x28 (fournies avec un terminal multilingue conçu au laboratoire CATAB de l'université J. Moulin à Lyon). Ce sont des polices de caractères de la norme GB utilisées dans notre travail. Voici quelques chiffres (pour 6763 caractères) :

| *Taille de matrice d'image :* | *Nombre d'octets utilisés :* |
|---|---|
| 16x16 | 625 112 |
| 24x24 | 936 364 |
| 28x28 | 801 444 |

Pour pouvoir réduire la place de stockage, on utilise deux méthodes, la compression d'image, et la recomposition de caractères à partir des composants graphiques (clés, traits). Dans l'impression, on a besoin des polices dans différents styles et corps avec des images fines. Il est nécessaire de réduire le travail à effectuer.

L'une des méthodes proposées est la définition des caractères suivant des clés et des traits de base. Un caractère est déterminé par les clés, et une clé est déterminée par les traits de base. La description repose sur la disposition des composants et sur les règles de leur composition. Cette méthode permet de produire différents styles à partir d'une description unique, par exemple, à l'aide du logiciel METAFONT .

# 2 Principes de la solution

En utilisant le langage E, la réalisation du chinois sur Grif est faite en même temps que les autres alphabets : cyrillique, grec, latin et vietnamien. Avec cette extension, nous créerons une version française `mgrif` multilingue.

La résolution se compose de trois phases : utilisation du codage mixte, organisation des données et adaptation des traitements.

## 2.1 Utilisation du codage mixte

Actuellement, pour simplifier la réalisation, nous utilisons les codes GB de deux octets (pour le chinois) en même temps que les codes ASCII (pour les alphabet latin, vietnamien, grec...). Ainsi, les transcriptions ne sont qu'écrits formellement dans les schémas de transcription. Le compilateur E ne prend pas en compte la partie de transcription dans les règles de transcription.

Les codes GB sont fournis par un fichier de données , appelé `TABLE`, et par les polices de caractères chinois existant au GEDIS.

La structure du fichier `TABLE` est simple. Ce sont des enregistrements dont chacun se compose de deux parties : l'entrée précédée de la liste de codes GB des caractères homophones. L'entrée est une chaîne de caractères minuscules de a à z représentant la syllabe ou pinyin, terminée par un chiffre de 1 à 4 représentant un ton ; l'absence de ton représente le ton faible. Le code GB est représenté par deux codes ASCII de 7 bits.

Dans ce fichier, un caractère peut avoir plusieurs entrées. Les enregistrements sont triés dans l'odre alphabétique des entrées.

Par exemple, la syllabe *ba* et tons correspondent aux 5 enregistrements *ba, ba1, ba2, ba3, ba4*.

Il y a deux fichiers de polices disponibles `beijing16` et `beijing24` dont chaque caractère chinois est dessiné sur un carré 16x16 ou 24x24 respectivement.

## 2.2 Organisation des données

Les données (tables de transcription, tables de structure et tables de présentation), créées par l'étape de compilation, sont fournies à l'éditeur comme des paramètres.

Les schémas de transcription doivent être compilés préalablement. Leur écriture demande des polices de caractères disponibles et des conventions d'entrée-codage. En principe, il n'y a pas de limitation du nombre de schémas de transcription présents, mais il existe des contraintes concernant le traitement d'entrée-sortie adapté et la capacité réelle du matériel utilisé. Par exemple, le traitement des caractères arabes n'est pas encore disponible.

En considérant que le fonctionnement de Grif se divise en deux moments distincts : initiation et édition de document, une table de structure, ou une table de présentation, n'est appelée qu'au moment d'édition à l'aide d'une demande de l'utilisateur. Les tables de transcription, en revanche, sont chargées une seule fois au moment de démarrage, avant que l'éditeur soit en service. Par contre, un fichier de dialogue peut être appelé, soit au moment de démarrage, soit au moment d'édition.

La figure 9.5 ci-dessous présente l'organisation des données :

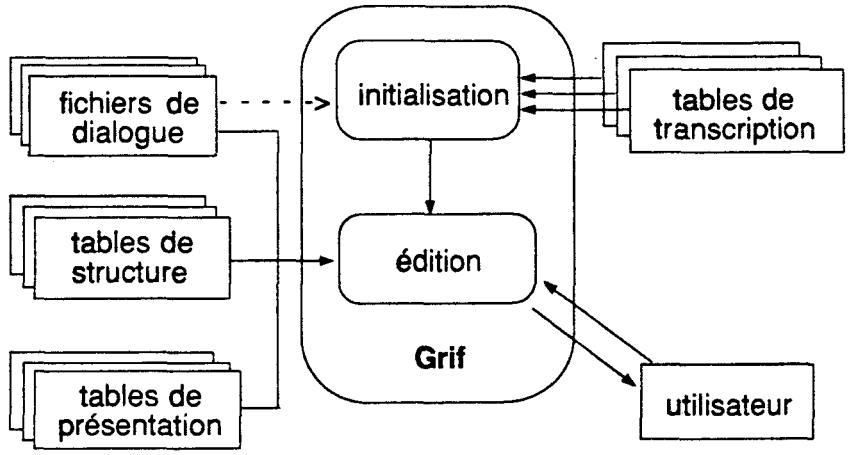



Figure 9.5 : Organisation des données

## 2.3 Adaptation des traitements

La réalisation vise à construire des modules indépendants de ceux de Grif. L'extension de Grif donne trois classes de modules : non modifiés, modifiés et nouveaux.

Les modules non modifiés concernent la gestion des arbres abstraits dans `Editeur` (création, modification) et la gestion de l'environnement.

Les modules modifiés, qui se trouvent notamment dans `Médiateur`, concernent la manipulation des images abstraites pour le processus d'entrée-sortie, le chargement des schémas, des polices de caractères et la recherche de texte. Il s'agit d'une adaptation des structures de données et d'un traitement cohérent sur l'ensemble des fonctions interactives.

Les modules nouveaux traitent l'entrée et la sortie, notamment pour les caractères diacritiques (latins et vietnamiens) et idéographiques (chinois).

Il y a sept fonctions suivantes à réaliser :

1. chargement des tables de transcription et des fichiers de dialogue ;
2. gestion d'entrée au clavier et des fenêtres de saisie ;
3. saisie par composition des touches et saisie par choix ;
4. affichage des caractères chinois sur les fenêtre de dialogue (saisie, recherche et remplacement, menu, messages) et sur les vues ;
5. échange des alphabets ;
6. adaptation des commandes de Grif ;
7. gestion des tables de transcription et des données pour la restitution en PostScript.

# 3 Implémentation

## 3.1 Écriture des schémas de transcription

Les noms des fichiers concernant la transcription sont formés par un préfixe, qui est le nom de l'alphabet concerné suivi d'un suffixe déterminé comme suit :

| *Suffixe :* | *Correspondant à :* |
|---|---|
| `.SCH` | schéma de transcription |
| `.TRS` | schéma de transcription compilé |
| `.DIA` | fichier de dialogue |

Les préfixes fournissent des intitulés qui sont énumérés dans le menu de la commande d'échange des alphabets.

La compilation utilise la commande `transc` suivi éventuellement du nom du schéma de transcription à compiler. La fonction `RdSchTransc` permet de charger en mémoire les schémas de transcription compilés.

Voici les fichiers utilisés dans notre implémentation :

| *Schéma de transcription :* | *Schéma compilé :* | *Fichier de dialogue :* |
|---|---|---|
| `Chinois.SCH` | `Chinois.TRS` | `Chinois.DIA` |
| `Grec.SCH` | `Grec.TRS` | `Grec.DIA` |
| `Latin.SCH` | `Latin.TRS` | `Latin.DIA` |
| `Russe.SCH` | `Russe.TRS` | `Russe.DIA` |
| `Vietnamien.SCH` | `Vietnamien.TRS` | `Vietnamien.DIA` |

Actuellement, seuls les deux schémas de transcription pour les alphabets latin et vietnamien sont tout à fait complets. Les deux schémas de transcription pour les alphabets grec et russe ne contiennent pas les signes diacritiques. Le schéma de transcription du chinois est plus particulier et présenté ci-dessous :

### Schéma de transcription du chinois

On a vu l'intérêt du code PS : portabilité, lisibilité et possibilité d'effectuer les travaux linguistique d'une façon cohérente. Nous proposons l'utilisation de ce codage pour écrire un schéma de transcription du chinois. Cependant, l'écriture complète en code PS de tous les caractères chinois usuels demande un gros travail et un grand espace mémoire.

Voici quatre conventions d'écriture :

1. jeu de caractères est *hanzi* simplifié contenant les 6763 caractères usuels de la norme GB ;
2. saisie phonétique avec les entrées pinyin du fichier `TABLE` ;
3. utilisation du codage PS sous forme d'une chaîne de caractères pour les règles de transcription ;

4. utilisation des codes GB pour les codes d'affichage : dans le schéma, ils sont écrits en décimal sous la forme simplifiée ij, où i et j sont des nombres de deux chiffres représentant les coordonnées dans la matrice de codes GB. La valeur correcte d'un code d'affichage est donc 256(i+32) + (j+32)..

L'écriture complète d'un schéma de transcription du chinois demande un temps appréciable. À titre d'illustration, on ne donne ci-dessous que les définitions des premiers caractères du code GB.

```
TRANSCRIPTION Chinois;
PARAM_IN
  Language = chinoise;        { langue chinoise }
  Set = Hanzi;              { jeu hanzi simplifié }
  Entry  = SELECT;            { saisie par choix }
  Window = WPHON;             { fenêtre de saisie phonétique }
PARAM_OUT
  Bytes = 2;                  { code sur deux octets }
  UppLow = No;                { pas de taille MAJ/min }
  Family = ( 'Beijing' );     { pour créer les noms de police }
  Size = ( 16, 24 );          { correspondant à deux polices }
  DialogFont = 'Beijing16';{ police de dialogue }
CODES
  'a'  -> (
          'A5-8/KOU3-3'  -> 2639, { 呵 }
          'A5-10/KOU3-3' -> 1601  { 啊 }
          );
  'a1' -> (
          'A1-7/FU4-2'   -> 1602, { 阿 }
          'A1-6/KOU3-3'  -> 6325, { 吖 }
          'A1-12/JIN1-5' -> 7925, { 锕 }
          'A1-12/ROU4-4' -> 7571  { 腌 }
          );
  'a2' -> 'A2-13/KOU3-3' -> 6436; { 嗄 }
  {...}
END
```

En principe, chaque prononciation peut correspondre à une règle de transcription, c'est à dire qu'il faut écrire au maximum 1 400 règles. Mais en réalité, au total, il y a environ 500 entrées possibles pour 6763 caractères usuels de la norme GB. Dans le schéma, on peut réduire le nombre d'entrées, ou simplifier les entrées dans les règles, si cela ne donne pas d'ambiguïté.

## 3.2 Saisie phonétique des caractères chinois

La fenêtre de saisie de type phonétique est décrite par le schéma de transcription. Lorsque `mgrif` est en mode de saisie d'un texte chinois, la fonction `CrFnSaisie` la crée dès la première frappe au clavier d'une entrée du caractère. Elle se trouve au dessous et à droite de la fenêtre principale de `mgrif`. Cependant, l'utilisateur peut la déplacer.

Dans la fenêtre de saisie (voir figure 9.6), il y a 24 sous-fenêtres dont la première est réservée à l'affichage, en inversion vidéo, du type de la fenêtre. La deuxième est utilisée pour afficher l'entrée pinyin (écho) au clavier. Les deux sous-fenêtres < et > sont des cases de commande. Les vingt restantes sont réservées à l'affichage des caractères homophones correspondant à une entrée pinyin+ton, avec leur rang. Le défilement est possible.

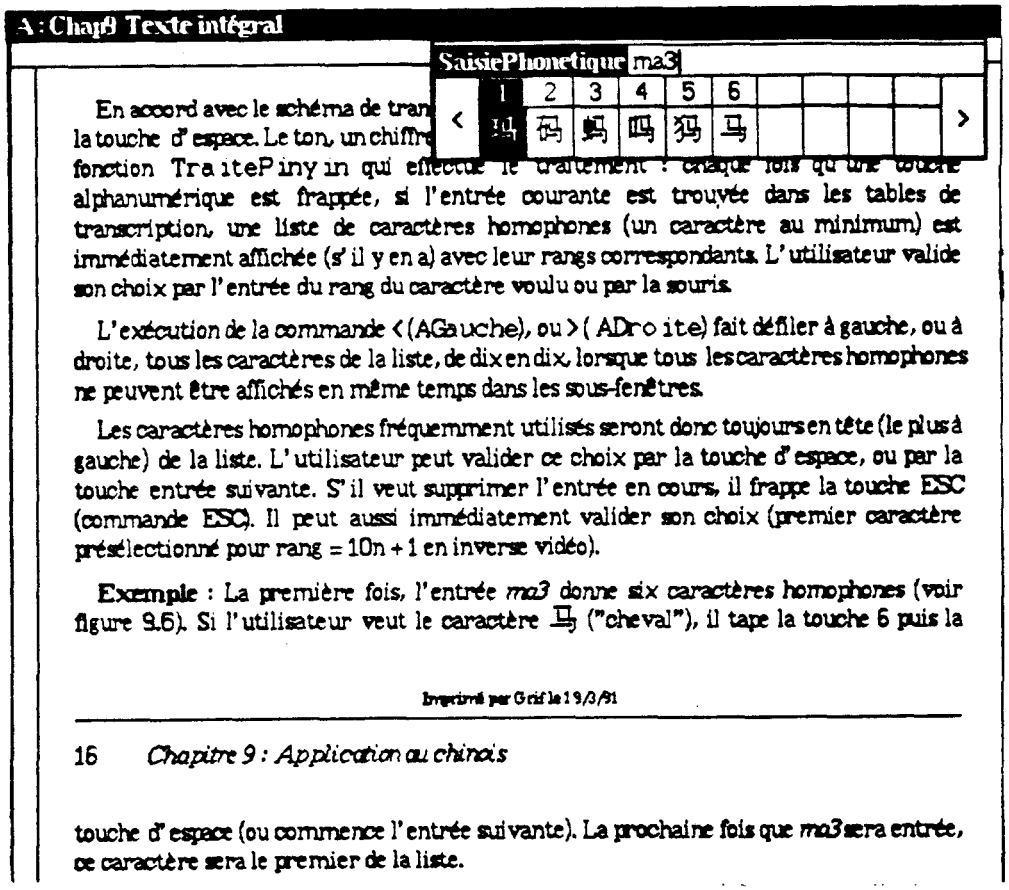



Figure 9.6 : Fenêtre de saisie phonétique pinyin

En accord avec le schéma de transcription, l'utilisateur tape l'entrée et la termine par la touche d'espace. Le ton, un chiffre de 1 à 4, a aussi l'effet de terminer une entrée. C'est la fonction `TraitePinyin` qui effectue le traitement : chaque fois qu'une touche alphanumérique est frappée, si l'entrée courante est trouvée dans les tables de transcription, une liste de caractères homophones (un caractère au minimum) est immédiatement affichée (s'il y en a) avec leur rangs correspondants. L'utilisateur valide son choix par l'entrée du rang du caractère voulu ou par la souris.

L'exécution de la commande < (`AGauche`), ou > (`ADroite`) fait défiler à gauche, ou à droite, tous les caractères de la liste, de dix en dix, lorsque tous les caractères homophones ne peuvent être affichés en même temps dans les sous-fenêtres.

Les caractères homophones fréquemment utilisés seront donc toujours en tête (le plus à gauche) de la liste. L'utilisateur peut valider ce choix par la touche d'espace, ou par la touche entrée suivante. S'il veut supprimer l'entrée en cours, il frappe la touche `ESC` (commande `ESC`). Il peut aussi immédiatement valider son choix (premier caractère présélectionné pour rang = 10n + 1 en inverse vidéo).

**Exemple** : La première fois, l'entrée *ma3* donne six caractères homophones (voir figure 9.6). Si l'utilisateur veut le caractère 马 ("cheval"), il tape la touche 6 puis la touche d'espace (ou commence l'entrée suivante). La prochaine fois que *ma3* sera entrée, ce caractère sera le premier de la liste.

Une entrée peut contenir un caractère arbitraire * ou ? (joker). Dans ce cas, à l'aide des commandes < et >, l'utilisateur fait défiler toutes les entrées possibles trouvées par l'éditeur, puis choisit le caractère voulu.

Si la touche frappée n'est pas une commande, la fonction `SaisieTexte` décide quelle méthode de saisie sera activée en vérifiant l'alphabet courant. La commande `TraiteCompose` permet de manipuler les caractères avec signes diacritiques latins, ou vietnamiens.

## 3.3 Impression des caractères chinois en langage PostScript

En langage PostScript, un caractère chinois est imprimé grâce à la description graphique d'un carré de taille déterminée par les polices d'affichage. En principe, ce carré peut devenir rectangle pour des effets de style ou de mise en relief. Les données fournies à cette description sont prises dans un fichier existant au GEDIS, qui contient le jeu des caractères de la norme GB sous forme de matrice 28x28 (bitmap).

On utilise la méthode de "remplissage progressif" pour créer des sous-polices en mode point, puisque le langage PostScript actuel ne permet pas de construire des polices dont le nombre de caractères dépasse 255. On a donc environ 27 sous-polices différentes à construire pour les 6763 caractères de la norme GB.

Le principe est décrit comme suit : au cours de chargement du fichier contenant la forme pivot du document de `mgrif`, les caractères chinois sont sélectionnés séparément pour remplir un vecteur de taille 255 de telle sorte que leurs valeurs de code d'affichage en décimal soient toujours en ordre croissant. La création permet de disposer, au maximum, de 254 caractères (sauf le blanc qui existe par défaut) nommés suivant l'ordre de leur présence dans une sous-police effective.

Quand le vecteur est rempli, ou quand le chargement du document est bien terminé, la sous-police correspondante est construite à partir de la description des images en mode

point, et nommée suivant son ordre d'apparition. En fait, l'image d'un caractère chinois (suite de chiffres hexadécimaux) demande environ 300 octets de mémoire.

La restitution d'un texte chinois sur une page pose deux problèmes reliés : le changement des sous-polices créées et la justification. En effet, le changement doit être effectué pendant la justification, par exemple si la première moitié des caractères d'une ligne appartient à la sous-police numérotée p et la dernière moitié à la sous-police numérotée p+1.

La justification d'un texte mixte utilise la même méthode que pour les textes latins. On compte d'abord le nombre de blancs effectifs sur la ligne justifiée pour fixer la distance entre deux caractères imprimés. On détermine la police ou la sous-police correspondant à chaque caractère, selon qu'il s'agit de caractères non-chinois ou chinois. Les caractères diacrités de codes ASCII étendus et les caractères chinois sont appelés sous forme du code en octal `'\xxx'`.

Pour réduire encore la taille du fichier PostScript produit, seulement des éléments apparus (images et symboles graphiques) pendant le chargement du fichier du document pivot de `mgrif` sont pris en compte, au lieu d'en disposer complètement par défaut comme des sous-programmes de PostScript. On introduit donc un compteur `MarqueImage` pour marquer les images et un compteur `MarqueSymbole` pour marquer les symboles graphiques.

## 3.4 Commandes sur les alphabets

Grâce aux schémas de transcription prédéfinis, `mgrif` dispose maintenant d'un ensemble d'alphabets qui est géré dynamiquement pendant la production des documents. Ainsi, le dialogue est effectué par l'exécution des commandes sur les alphabets : changement lors de l'entrée d'un texte multi-alphabet et fonction de recherche et de remplacement.

Pour changer d'alphabet, l'utilisateur utilise la commande `Alphabets`.

L'exécution de cette commande n'est plus identique à celle de la commande de changement des attributs de la version prototype (voir chapitre 4). Grâce à la gestion dynamique, la commande affiche la liste des alphabets disponibles à choisir dans le menu suivant la présence des schémas de transcription.

De plus, comme le code d'un caractère n'est plus identique d'un alphabet à l'autre, l'exécution de la commande `Alphabets` crée un nouvel élément textuel en mode "squelette". Selon la sélection en cours, l'élément créé sera positionné à l'endroit souhaité.

Par exemple, la figure 9.7 présente les différentes étapes d'exécution de la commande `Alphabets` pour insérer un caractère chinois dans un texte en français. L'utilisateur entre ce texte dans son document et désigne l'endroit à insérer par la souris. Dans ce cas, c'est un morceau de texte sélectionné en inversion vidéo désigné par la boîte grise (a). Ensuite, il utilise la commande `Alphabets` et choisit l'alphabet chinois. Un élément

textuel vide est créé (une boîte) et affiché sous forme de "squelette" (b). Enfin, il entre le caractère chinois voulu dans la boîte (c). Le curseur, désigné par le signe circonflexe, se trouve immédiatement après ce caractère.

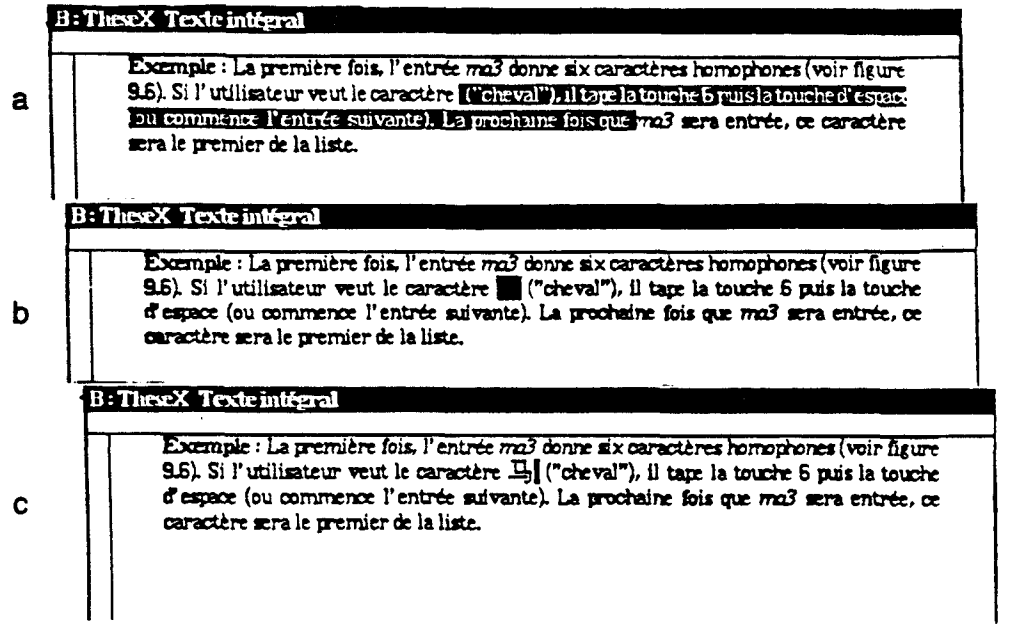



Figure 9.7 : Insertion d'un caractère chinois dans un texte en français

Le traitement de recherche et de remplacement d'un texte multilingue améliore la cohérence : l'alphabet d'un texte à rechercher ou à remplacer doit être désigné et trouvé avant la recherche des caractères. Dans l'exécution, un caractère est trouvé par son code de transcription qui est une chaîne de caractères ASCII. Dans la sous-fenêtre du menu `Recherche`, le texte est entré et affiché correctement selon l'alphabet.

Les étapes d'exécution sont :

1. sélection de la commande `Recherche` sur le menu principal de `mgrif` ;
2. désignation du type (texte) et de la direction (en avant ou en arrière) de la recherche ;
3. sélection de l'alphabet à chercher et à remplacer ;
4. entrée du texte à chercher et du texte à remplacer, dans cette étape, les textes entrés précédemment sont conservés avec leur alphabet ;
5. exécution de la recherche et du remplacement.

Pour changer la langue de dialogue, l'utilisateur utilise la commande `DIALOGUER`.

La figure 9.8 présente un exemple de la combinaison de langues traitables dans Grif multilingue.

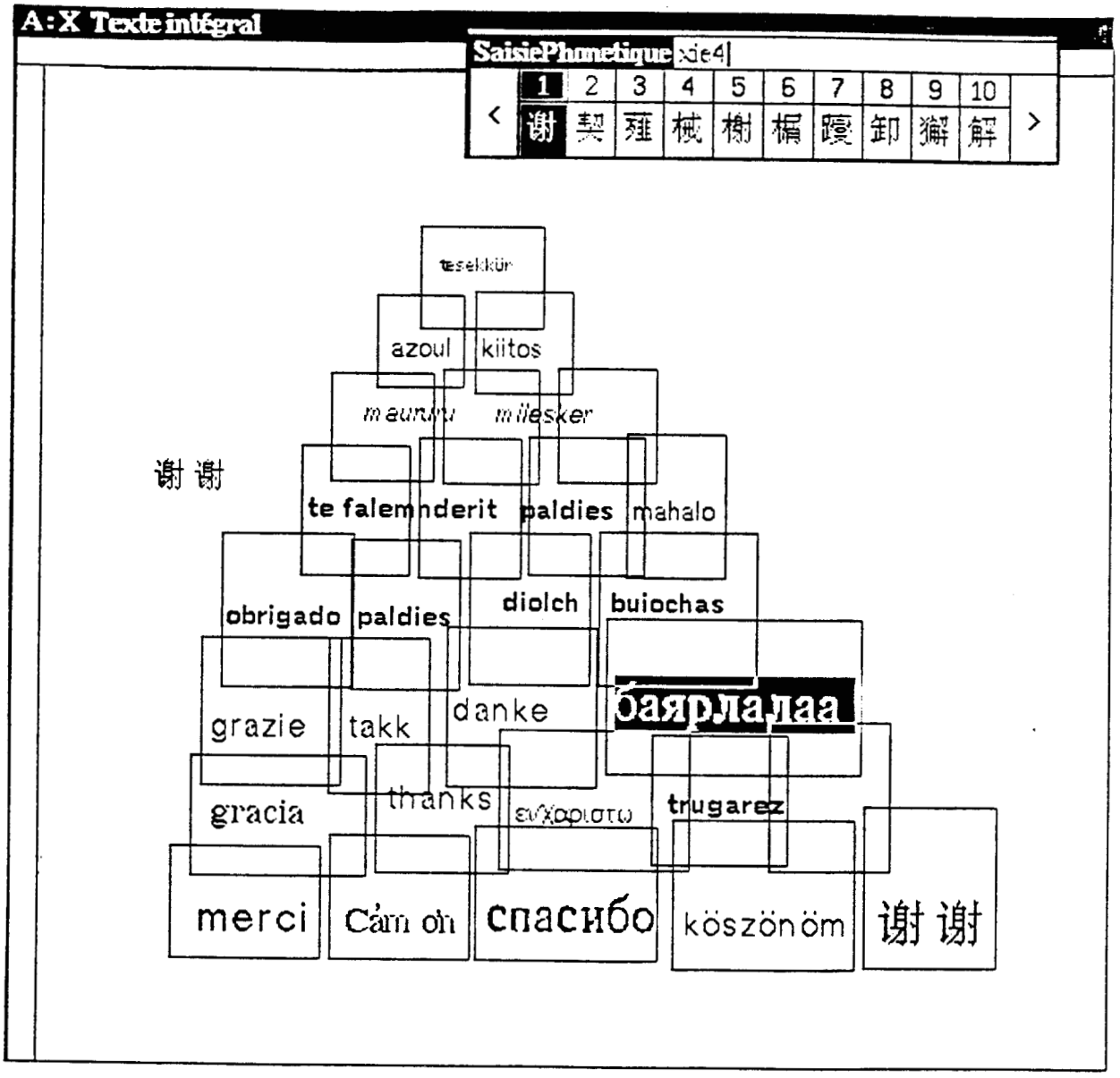



Figure 9.8 : Combinaison de langues traitables dans `mgrif`

# Conclusion

## 1 Contribution de la thèse

Ce travail est une étude des solutions possibles pour la conception d'un STTM en général et la réalisation d'une extension de l'éditeur Grif pour rendre ce produit multilingue. Nous avons abordé plusieurs aspects des problèmes du multilinguisme, de la documentation structurée et de l'interactivité. Nous en proposons une solution pour Grif.

À partir d'une adaptation de Grif au vietnamien où nous utilisons la méthode de codage appelée "remplissage des vides" pour représenter les caractères avec signes diacritiques vietnamiens, nous proposons trois stratégies possibles en résolvant les problèmes suscités par le multilinguisme (codage, saisie, restitution et dialogue). Nous sommes arrivé à une solution générique qui est l'introduction du langage E dans Grif. C'est un langage de description des conventions d'entrée, de codage et d'affichage sur un système d'écriture. L'application de E permet de compléter la notion de modèle de document de Grif (spécifié par la structure logique et la présentation physique) par un nouveau niveau, celui de la description linguistique.

L'implémentation réalisée sur le langage E nous a donné une version de Grif multilingue `mgrif` qui peut traiter efficacement les caractères latins (y compris les caractères vietnamiens), grecs, cyrilliques et chinois (norme GB 2312-80).

Nous avons ainsi obtenu le premier éditeur de documents structuré réellement multilingue.

## 2 Problèmes existants

L'aspect multilingue pose encore des problèmes que nous n'avons pas résolus. Voici ceux qui nous paraissent les plus importants.

Certains caractères de ponctuation, comme deux points, point-virgule, point d'interrogation... et la façon d'écrire l'initiale en majuscule dans certains mots à l'intérieur d'une phrase changent d'une langue à l'autre. Les différences entre l'anglais et le français sur ce point sont de bons exemples. Le premier problème concernant la typographie fine est le suivant : peut-on éliminer les erreurs d'orthographe dans un texte multilingue ?

Le deuxième est la coupure des mots en fin de ligne. Ce problème est bien résolu dans certains systèmes de traitement de texte comme TEX. Mais, pour l'éditeur Grif qui travaille en mode interatif, à l'heure actuelle, on n'a pas encore réalisé le traitement de la coupure des mots. Dans l'étude des solutions possibles, nous rencontrons des difficultés inévitables, dues à deux facteurs : la possibilité de conciliation des aspects divers du multilinguisme et la capacité des matériels disponibles.

Pour les deux problèmes cités, une solution proposée consiste à définir un autre langage particulier intervenant comme le langage E. Ce langage permet d'une part de décrire des règles d'écriture des caractères spéciaux avec l'orthographe cohérente d'une langue donnée. D'autre part, pour les langues dont l'écriture contient la coupure des mots, ce langage permet décrire des informations nécessaires au traitement. L'étape suivante est de construire les dictionnaires de suffixes, de préfixes et les algorithmes d'analyse. Ces dictionnaires permettent également de vérifier la correction orthographe des mots dans une chaîne.

Un problème connu dans la restitution d'un texte multilingue est celui de l'ordre d'écriture dans la mise en lignes et en paragraphes. Comment restituer un texte de Grif dans l'ordre arbitraire (vertical, horizontal, diagonal ou quelconque) ? L'extension à la fois des langages S et P dans Grif, comme nous l'avons proposée, est possible. Cela permet de décrire plusieurs images différentes à présenter d'un même document. Par exemple, pour la restitution verticale, un paragraphe peut être présenté comme une colonne et une page se compose d'une suite de ces colonnes.

Un autre problème est l'exportation d'un document de Grif vers un formateur ou une norme. Nous avons abordé ce problème avec le vietnamien où la résolution n'est pas très difficile. En utilisant les systèmes de codage sur un octet, l'extension du langage T pour un nombre de systèmes d'écriture correspondants est résoluble (comme il y a des extensions de TEX pour l'arabe par exemple). Le problème devient très difficile à résoudre avec les caractères chinois, japonais, coréen..., bien qu'il existe des solutions d'insertion de ces caractères dans TEX (comme les méthode proposée par A. Cousquer au LIFL, Université de Lille 1). Jusqu'à maintenant, nous n'avons pas encore abordé effectivement ces problèmes, qui restent ouverts.

Enfin, dans la première version de notre implémentation, nous n'avons pas encore résolu la recherche et le remplacement d'une chaîne écrite sur plusieurs alphabets, comme le traitement des expressions du $\lambda$-calcul.

# 3 Perspectives

Les descriptions linguistiques fournies par le langage E nous permettent de compléter le modèle de document de Grif. Nous avons proposé un procédé de conception d'un STTM générique par les trois approches, structurelle, interactive et multilingue. Ainsi, la continuation de l'étude pour résoudre complètement les problèmes existants est nécessaire.

Dans ce but, notre implémentation de Grif doit être améliorée pour obtenir un maximum de performances et pour s'adapter aux nouvelles versions de Grif. Parallèlement avec les possibilités de traitement des problèmes de restitution (césure, typographie fine, ordre d'écriture...), Grif doit pouvoir traiter d'autres systèmes d'écriture complexe, comme ceux de l'arabe, de l'hébreu, du thaï, etc.

D'autre part, étant donné que Grif ne fonctionne que sous X-Window, et en tenant compte des évolutions en cours de ce système, l'aspect de dialogue devrait être résolu, comme la gestion des fenêtres, au niveau d'application "client" de X-Window.

Comme le traitement automatisé des langues naturelles connaît un grand développement, l'existence d'un Grif multilingue ouvre de nombreuses perspectives. Par exemple, il devrait être assez aisé de définir une classe de documents "bitextes", pour obtenir à partir de Grif un éditeur bilingue, avec synchronisation automatique des paragraphes. De même, on pourrait construire une interface conviviale avec des bases lexicales multilingues.

Enfin, la possibilité de raffiner le modèle de document permettrait d'associer des "types d'énoncés" aux différents niveaux, et d'obtenir ainsi un premier niveau d'aide à la rédaction, et/ou de critique textuelle, moyennant bien sûr l'interface avec des linguiciels appropriés.

# ANNEXES



Imprimé par Grif le 20/3/91

# ANNEXE 1 Tableau des langues naturelles prises en compte par les CARIX (en 1986)

| Langue | Nom abgéré de l'alphabet | Jeu de caractères | Million de personnes[1] |
|---|---|---|---|
| albanais | ALB | romain | 2 |
| allemand | ALL | romain | 100 |
| amharique (éthiopien) | AMH | amharique | 30 |
| anglais-américain | ANG | romain | 350 |
| arabe | ARB | arabe | 200 |
| arménien | ARM | arménien | 4 |
| basque | BAS | romain | 4 |
| bengali | BEN | bengali | 100 |
| biélorusse | BIE | cyrillique | 8 |
| birman | BIR | birman | 50 |
| bulgare | BUL | cyrillique | 8 |
| cambodgien | CAM | khmer | 6 |
| catalan | CAT | romain | 6 |
| chinois | HAN | hanzi | 1 000 |
| cinghalais | CIN | cinghalais | 10 |
| coréen | COR | hangul/hanzi | 55 |
| danois | DAN | romain | 5 |
| espagnol | ESP | romain | 250 |
| estonien | EST | romain | 1 |
| finnois | FIN | romain | 4 |
| français | FRA | romain | 66 |
| gaélique | GAE | romain | 0.5 |
| géorgien | GEO | géorgien | 4 |
| grec | GRC | grec | 9 |
| hébreu | HEB | hébreu | 3 |
| hindi/ourdou | IND | devanagari/arabe (étendu) | 600 |
| hongrois | HON | romain | 12 |

Imprimé par Grif le 20/3/91

| Langue | Nom abgéré de l'alphabet | Jeu de caractères | Million de personnes[1] |
|---|---|---|---|
| islandais | ISL | romain | 0.2 |
| italien | ITA | romain | 60 |
| japonais | JAP | kanjis/kanas | 120 |
| laotien | LAO | thaï | 3 |
| latin (langue morte) | LAT | romain | 0 |
| letton | LET | romain | 2 |
| lithuanien | LIT | romain | 3 |
| macédonien | MAC | cyrillique | 3 |
| malais-indonésien | MAL | romain/arabe (étendu) | 120 |
| mongol | MON | mongol/cyrillique | 6 |
| néerlandais-afrikaans | NED | romain | 25 |
| norvégien | NOR | romain | 4 |
| ouïgour | OUI | arabe/ouïgour | 5 |
| ouzbek | OUZ | cyrillique | 10 |
| persan | PER | arabe (étendu) | 40 |
| polonais | POL | romain | 35 |
| portugais-brésilien | POR | romain | 120 |
| roumain-moldave | ROU | romain | 25 |
| russe | RUS | cyrillique | 120 |
| sanskrit (langue morte) | SAN | devanagari | 0 |
| serbo-croate | SRB | romain/cyrillique | 13 |
| slovaque | SVA | romain | 4 |
| slovène | SVN | romain | 2 |
| suédois | SUE | romain | 8 |
| swahili (souahéli) | SWA | romain/arabe | 30 |
| tamoul (malabar) | TAM | tamoul | 80 |
| télougou | TEL | télougou | 40 |
| tchèque | TCH | romain | 11 |
| thaï | TAI | thaï | 65 |
| tibétain | TIB | tibétain | 4.5 |

Imprimé par Grif le 20/3/91

| Langue | Nom abgéré de l'alphabet | Jeu de caractères | Million de personnes(1) |
|---|---|---|---|
| turc | TUR | romain | 50 |
| ukrainien | UKR | cyrillique | 45 |
| vietnamien | VIE | romain | 60 |

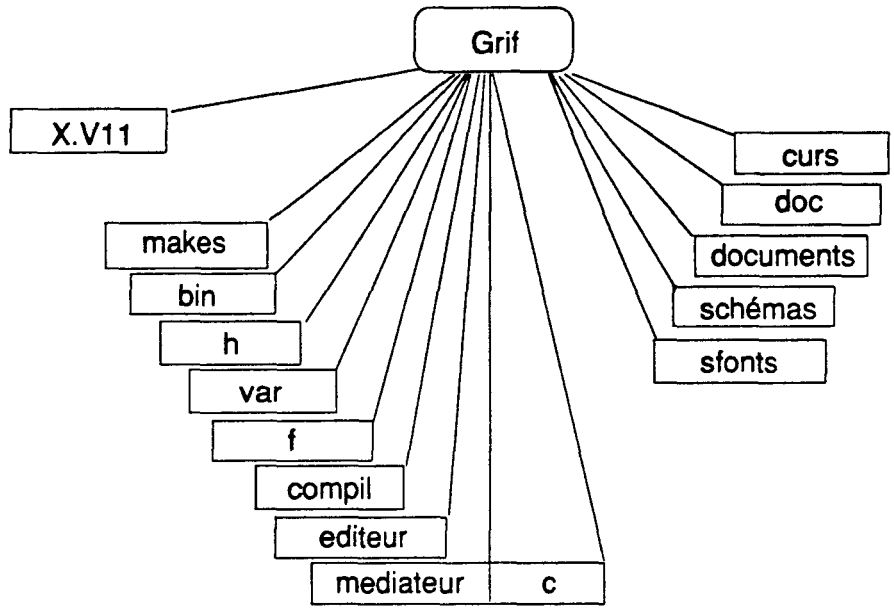
**Remarque :** Ce tableau, qui est basé sur une statistique lors de travaux du PN-TAO (Projet National de Traduction Assistée par Ordinateur) [III.1 Boitet & al. 86b], ne donne pas toutes les langues naturelles du monde. Le document "Code pour la représentation des noms de langue" [III.1 ISO 85] donne une liste alphabétique de 136 langues avec les indicatifs de langue à deux lettres.

La dernière colonne (1) indique le nombre, en millions, de personnes dont la langue était la langue maternelle en 1986.

Imprimé par Grif le 20/3/91

# ANNEXE 2 Organisation des programmes de Grif

Les programmes sources (en Pascal et en C) de Grif sont organisés en modules groupés dans 15 répertoires. La figure suivante présente la hiérarchie de répertoires :




Organisation des programmes de Grif

Le système Grif contient trois groupes de répertoires dont les logiciels occupent un volume d'environ 14 Méga-octets sur bande. Le premier est le répertoire X.V11 qui contient l'environnement X-Window nécessaire au fonctionnement de l'application Grif.

Le deuxième groupe contient 5 répertoires, rangés par ordre alphanumérique, qui sont utilisés par l'application :

| *Nom :* | *Contenu :* |
|---|---|
| `curs` | curseurs et trames de gris. |
| `doc` | documentation disponible. |
| `documents` | documents au format pivot produits par Grif. |
| `schemas` | tous les schémas en langages S, P et T nécessaires à l'éditeur. |
| `sfonts` | polices de caractères en format SNF. |

Le troisième groupe contient 9 répertoires qui permettent de construire l'application :

| *Nom :* | *Fonction :* |
|---|---|
| `makes` | pour construire tous les programmes du répertoire `bin`. |
| `bin` | tous les programmes exécutables concernant l'application Grif. |
| `h` | structures de données. |
| `var` | variables globales. |
| `f` | définition des procédures externes des modules de Grif. |
| `compil` | pour construire les compilateurs (S, P et T) de Grif (écrits en Pascal). |
| `editeur` | modules d'édition écrits en Pascal. |
| `mediateur` | gestion d'interface utilisateur et de sortie (modules écrits en Pascal). |
| `c` | modules d'interface utilisateur écrits en C. |

Le tableau suivant présente le volume des quatre composants logiciels :

| *Composant logiciel:* | *Nombre de modules :* | *Nombre de lignes :* |
|---|---|---|
| les compilateurs | 14 en Pascal | 14 532 |
| les traducteurs | 2 en Pascal | 2 214 |
| `Éditeur` | 31 en Pascal | 36 726 |
| `Médiateur` | 18 en Pascal | 14 159 |
| | 13 en C | 10 522 |
| *Total:* | 78 | 78 153 |

Le nombre d'octets des programmes exécutables dans le répertoire `bin` est présenté dans le tableau suivant :

| *Programme:* | *Description :* | *Nombre d'octets :* |
|---|---|---|
| `grif` | éditeur Grif | 1 327 104 |
| `print` | sortie directe en PostScript | 679 936 |
| `export` | exportation vers un formateur | 655 360 |
| `import` | traduction des fichiers ASCII | 131 072 |
| `str` | compilateur S | 114 688 |
| `prs` | compilateur P | 147 456 |
| `tra` | compilateur T | 122 880 |
| `printstr` | visualisation d'un schéma S | 81 920 |
| `printprs` | visualisation d'un schéma P | 114 688 |
| `chstr` | contrôle des schémas S | 122 880 |
| `grm` | traduction des grammaires | 73 728 |
| *Total:* | 11 programmes | 3 571 712 |

# ANNEXE 3 Grammaire du langage E

```
{*************************************************************************
      Cette grammmaire definit le langage de description des schemas de
    transcription qui sont utilises dans l'editeur Grif.

      Une grammaire est specifiee par une suite de regles.  Chaque regle
    est constituee d'un symbole de la grammaire suivi du signe = et de la
    partie droite.  Dans la partie droite, le formalisme utilise les
    conventions suivantes :

      ' '    delimite les mots-cles du langage.
      [ ]    delimite une partie optionelle.
      < >    delimite une repetition (de zero a n fois).
      /      indique un choix.
      { }    delimite un commentaire.

      NAME   suite de lettres majuscules/minuscules, de chiffres et de
    caractere de soulignement _, commen çant par une lettre.

      STRING suite de caracteres quelconques delimites par des apostrophes.

      NUMBER entier positif ou nul, sans signe, c-a-d une suite de chiffres
    decimaux.

      La justaposition des symboles indique la concatenation de ces symboles.

      La fin d'une regle est marquee par un point.

      Le fichier META.LAN donne une description formelle de ce formalisme.

      H. Kh. PHAN   Juillet  1990
*************************************************************************}


Transcription = 'TRANSCRIPTION' NomSchema ';'{ Nom du schema de transcription }
              [ 'CONSTS' ListConst ]          { Liste de constantes }
                'PARAMS_IN' ListParamIn       { Liste de parametres des entrees}
                'PARAMS_OUT' ListParamOut     { Liste de parametres des sorties}
                'CODES' ListDefTrans          { Definition de l'alphabet }
                'END' .                       { Fin du schema }

NomSchema = NAME .
{***
      Definition des constantes
***}
ListConst = DefConst < DefConst > .
DefConst  = NomConst '=' ValConst ';' .
NomConst  = NAME .
ValConst  = Chaine / Entier .
Chaine    = STRING .
```

```
Entier    = NUMBER .
{***
      Definition des parametres des entrees
***}
ListParamIn = Langue         { Declaration de la langue }
              Jeu            { Declaration du jeu de caracteres }
              Entree         { Declaration de la methode de saisie }
            [ Fenetre   ]    { Declaration de la fenetre de saisie }
            [ LettreAcc ]    { Declaration des lettres portant les signes }
            [ Signes    ]    { Declaration des noms de signe diacritique }
            [ MaxSignes ] .  { Le nombre maximum de signes par lettre }

Langue   = 'Language' '=' NomLangue ';' .
NomLangue = NAME .

Jeu = 'Set' '=' NomJeu ';' .
NomJeu = NAME . { Seul premier caractere significatif actuellement }

Entree    = 'Entry' '=' Method ';' .

Method = 'LEFTCOMPO'  /
         'RIGHTCOMPO' /  { Composition des touches dans l'ordre :
                           'LEFTCOMPO'  = signe [signe] lettre
                           'RIGHTCOMPO' = lettre signe [signe] }
         'SELECT' .      { Saisie par choix }

Fenetre  = 'Wind' '=' NomFen ';' .

NomFen    = 'WPHON'  /    { Entree phonetique }
            'WGRAPH' /    { Entree graphique }
            'WMORPHO' .   { Entree morphologique }

LettreAcc = 'Letters' '=' '(' SuiteSymb ')' ';' .
SuiteSymb = Symbol < ',' Symbol > .
Symbol    = NomConst / Chaine .

Signes    = 'Signs' '=' '(' SuiteSymb ')' ';' .

MaxSignes = 'Max_Signs' '=' NbSignes ';' .
NbSignes  =  NUMBER .
{***
      Definition des parametres des sorties
***}
ListParamOut =   Octets      { Nombre d'octets par code d'affichage }
                 MajMin      { Ecriture en majuscule/minuscule }
               [ Famille ]   { Famille de caracteres du jeu }
               [ Style   ]   { Style de caractere }
                 Relief      { Relief de caractere }
                 PoliceDial. { Police de dialogue en saisie }

Octets    = 'Bytes' '=' NbOctets ';' .
NbOctets  = NUMBER .

MajMin    = 'UppLow' '=' Taille ';' .
Taille    = 'Yes' /  { Taille majuscule/minuscule }
            'No' .   { Pas de taille }

Famille   = 'Family' '=' '(' SuiteFami ')' ';' .
SuiteFami = NomFami < ',' NomFami > .
NomFami   = STRING .
```

```
Style     = 'Style' '=' '(' SuiteEvid ')' ';' .
SuiteEvid = Evidence < ',' Evidence > .
Evidence  = STRING .

Relief    = 'Size' '=' '(' SuiteCorps ')' ';' .
SuiteCorps= Corps < ',' Corps > .
Corps     = NUMBER .        { Un entier de deux chiffres }

PoliceDial = 'DialogFont' '=' NomPolice ';' .
NomPolice  = NAME .   { Nom sans suffixe, comme .snf, par exemple }
{***
      Definition des transcriptions dans l'alphabet
***}
ListDefTrans   = DefTrans < DefTrans > .

DefTrans      = ExpEntree '-' '>' RprsCodage ';' .
ExpEntree     = Symbol < Symbol > .
RprsCodage    = TransCode  / ListTransCode .
ListTransCode = '(' SuiteTrsCode ')' .
SuiteTrsCode  = TransCode < ',' TransCode > .

TransCode     = Transcript '-' '>' CodeInfo .
Transcript    = STRING .

CodeInfo      = [ BaseNum ] NUMBER .
BaseNum       = '$' /   { Code en hexadecimal }
                '\' .   { Code en octal }

END
```

# ANNEXE 4 Description des tables de transcription

```
{* ================================================================== *
 * typetransc.h :
 * Declaration des constantes et types pour les schemas de transcription
 * Les types sont utilises dans les modules entrees-sorties du Mediateur
 * en Pascal

 *      PHAN Huy Khanh - Juillet 1990
 * ------------------------------------------------------------------ *)

const

        MaxLgExp =  7;    { longueur d'une expression d'entree }
        MaxLgRprs = 10;   { longueur d'une representation interne }
        LgExtNom = 15;    { longueur d'un Nom extensif }
        LgChaine = 255;   { longueur de Chaine }
        NbMaxSigne = 20;  { nombre maximum de signes diacr. de la langue }
        NbMaxLettre = 30;     { nombre maximum de lettres accentues }
        NbMaxSignCompo= 30;   { nombre maximum de signes composes }
        NbMaxSchTransc= 10;   { nombre maximum de shemas de transcription }
        NbMaxDefTrans = 600;  { nombre maximum de DefTrans }
        NbMaxRprsCode = 7800;{ nombre maximum de RprsCode }

type

   Express = packed array [1..MaxLgExp] of char;
   BuffTCode = packed array [1..MAX_POS] of char;
   PtrExtNom = ^ExtNom;
   ExtNom  = packed array [1..LgExtNom] of char;{ Nom extensif }
   PtrChaine = ^Chaine;
   Chaine = packed array [1..LgChaine] of char;{ Chaine de caracteres }

   Code2Oct = record
     Oct1: char; { Premier octet }
     Oct2: char  { Deuxieme octet }
     end;

   PtrRprsCode = ^RprsCode;
   RprsCode = record
     RCBuff:    BuffTCode;      { Forme de transcription d'un RprsCode }
     RCCodInfo: Code2Oct;       { code d'affichage de deux octets }
     RCSuiv:    integer         { adresse du RprsCode suivant }
     end;

   PtrDefTrans = ^DefTrans;
   DefTrans = record
     DTLgExp:   shortint;       { longueur d'une expression d'entree }
     DTEntree:  Express;        { expression d'entree }
     DTOffset:  shortint;       { adresse du premier RprsCode }
     DTNbe:     shortint        { nombre de RprsCodes ayant la memme entree }
     end;
```

```
     {*Description d'un schema de transcription pour le traitement
informatique de textes multilingue.
     Pour verifier l'autorisation de combinaison des signesdiacritiques,
on utilise dans la table 1 le type char pour gagner la place.
     Si la valeur d'un element de la table 1 est un code inferieure a 255,
elle sera le rang a considerer dans la table 2. En effet, un code
d'affichage (code ASCII etendu dans cette application) a une valeur
conventionne inferieure a 255.                                          *)

 PtrSchTrs = ^SchTrs;
 SchTrs = record
      TrsNom:                Nom;      { nom de la structure generique }
      TrsCode:               integer;  { code identifiant la version }
      { ======================= les parametres d'entrees }
      TrsJeu:          Nom;      { nom du jeu de caractere }
      TrsLang:         Nom;      { nom de la langue }
      TrsAlphabet:char;          { nom abrege de l'alphabet }
      TrsEntree:       Nom;      { nom de la methode de saisie }
      TrsFenetre:      Nom;      { nom de la fenetre de saisie }
      TrsNbLettres:    shortint;{ nombre de lettres accentuees }
      TrsNbSignes:     shortint;{ nombre de signes diacritiques }
      TrsMaxSignes:    shortint;{ nombre max. de signes par lettre}
      TrsLettres:      Chaine;   { chaine des codes de lettre accentuee }
      TrsSignes:       Chaine;   { chaine des codes de signe diacritique }
      { ======================= les parametres de sorties }
      TrsCode2byte:    boolean; { representation d'un code d'affichage }
      TrsMajMin:       boolean; { ecriture majuscule/minuscule}
      TrsNbPolice:     shortint; { nombre de police utilisees }
      TrsPolice:       array [1..MAX_POLICE] of ExtNom;
                               { Tableau de familles de caracteres }
      TrsNbEvidence:   shortint; { nombre de niveaux de mise en evidence }
      TrsEvidence:     array [1..MAX_EVIDENCE] of ExtNom;
                               { Tableau de styles }
      TrsNbTaille:     shortint; { nombre de tailles utilisees }
      TrsTaille:       array [1..MAX_LTAILLE] of integer;
                               { Tableau de tailles }
      TrsDPolice:      ExtNom;   { nom de la police de dialogue }
      TrsNbRprsCode:   shortint;{ nombre de RprsCode }
      TrsRprsCode:     PtrRprsCode;{ pointeur sur le premier RprsCode }
      { ======================= les tables de transcription }
      case TrsSCompose: boolean of
      { saisie par l'entree d'une expression }
      false:(TrsNbDefTrans:   shortint;{ nombre de DefTrans }
             { pointeur vers la premiere definition de transcription }
             TrsPtrDefTrs: PtrDefTrans );
      { saisie par la composition des touches }
      true: ({ nombre de lignes (signes composes) de la Table 2 }
             TrsNbSignCompo:shortint;
             { table de verification d'autorisation de combinaison des
               signes diacritiques)
             TrsTable1: array [1..NbMaxSigne] of Chaine;
             { table de codes d'affichage (ASCII)}
             TrsTable2: array [1..NbMaxSignCompo] of Chaine )
      end;
```

# ANNEXE 5 Schémas de transcription utilisés

```
{*************************************************************************
     Ce schema decrit la transcription des caracteres accentues Latin

     H. Kh. PHAN   Juillet 1990
*************************************************************************}

TRANSCRIPT Latin;

CONSTS { utilises dans l'ecriture des chaines de caracteres }

{ Les lettres avec
 un blanc pour entrer les signes diacritiques simples }
        bl = ' ';
        A = 'A';        a = 'a';
        C = 'C';        c = 'c';
        E = 'E';        e = 'e';
        I = 'I';        i = 'i';
        N = 'N';        n = 'n';
        O = 'O';        o = 'o';
        U = 'U';        u = 'u';
        Y = 'Y';        y = 'y';

{ Les signes diacritiques }
        GR = 156; { ` accent grave }
        AC = 157; { ' accent aigu }
        CI = 158; { ^ accent circonflexe }
        DI = 159; { : trema }
        CE = 160; { , cedille }
        TI = 161; { ~ tilde }
CA = 162; { V circonflexe inverse }
OG = 163; { cedille inverse }
        EN = 164; { / o oblique ou - barre }
        RI = 165; { o ronde }
HU = 166; { " umlaut }
        BR = 167; { v breve }
PE = 168; { . point }
MA = 169; { - macron }
        EL = 170; { e ligature gauche du e }

PARAMS_IN
        Set = Romain;
        Language = default;
        Entry  = LEFTCOMPO;
        Letters= (bl, 'A', 'C', 'E', I, N, O, U, Y, a, c, e, i, n, o, u, y);
        Signes = (AC, GR, CI, DI, CE, TI, CA, OG, EN, PE, MA, EL, HU, RI, BR);
        Max_Signe = 1; { un seul signe diacritique par lettre }

PARAMS_OUT
        Bytes  = 1; { un octet par code d'affichage }
```

```
        UppLow = Yes;
        Family = ('Times', 'Helvetica', 'Courier');
        Style  = ('Roman', 'Bold', 'Italic');
        Size   = (9, 13, 17, 22, 33, 48);
        DialogFont = 'ltr13';

CODES
        AC bl -> '!1' -> 194; { aigu }
        AC A -> 'A!1' -> 133;   AC a -> 'a!1' ->  2; { A a aigu }
        AC E -> 'E!1' -> 141;   AC e -> 'e!1' -> 31; { E e aigu }
        AC I -> 'I!1' -> 145;   AC i -> 'i!1' -> 14; { I i aigu }
        AC O -> 'O!1' -> 151;   AC o -> 'o!1' -> 20; { O o aigu }
        AC U -> 'U!1' -> 159;   AC u -> 'u!1' -> 27; { U u aigu }
        GR bl -> '!2' -> 193; { grave }
        GR A -> 'A!2' -> 132;   GR a -> 'a!2' ->  1; { A a grave }
        GR E -> 'E!2' -> 140;   GR e -> 'e!2' ->  9; { E e grave }
        GR I -> 'I!2' -> 144;   GR i -> 'i!2' -> 13; { I i grave }
        GR O -> 'O!2' -> 150;   GR o -> 'o!2' -> 19; { O o grave }
        GR U -> 'U!2' -> 158;   GR u -> 'u!2' -> 26; { U u grave }
        CI bl -> '!3' -> 195; { circonflexe }
        CI A -> 'A!3' -> 134;   CI a -> 'a!3' ->  3; { A a circonflexe }
        CI E -> 'E!3' -> 142;   CI e -> 'e!3' -> 11; { E e circonflexe }
        CI I -> 'I!3' -> 146;   CI i -> 'i!3' -> 15; { I i circonflexe }
        CI O -> 'O!3' -> 152;   CI o -> 'o!3' -> 21; { O o circonflexe }
        CI U -> 'U!3' -> 160;   CI u -> 'u!3' -> 28; { U u circonflexe }
        DI bl -> '!4' -> 200; { trema }
        DI A -> 'A!4' -> 136;   DI a -> 'a!4' ->  5; { A a trema }
        DI E -> 'E!4' -> 143;   DI e -> 'e!4' -> 12; { E e trema }
        DI I -> 'I!4' -> 147;   DI i -> 'i!4' -> 16; { I i trema }
        DI O -> 'O!4' -> 154;   DI o -> 'o!4' -> 23; { O o trema }
        DI U -> 'U!4' -> 176;   DI u -> 'u!4' -> 29; { U u trema }
        DI Y -> 'Y!4' -> 181;   DI y -> 'y!4' -> 30; { Y y trema }
        CE bl -> '!5' -> 203; { cedille }
        CE C -> 'C5!' -> 139;   CE c -> 'c5!' ->  8; { C c cedille }
        EL A -> 'A!e' -> 138;   EL a -> 'a!e' ->  7; { AE ae ligature }
        EL O -> 'O!e' -> 155;   EL o -> 'o!e' -> 24; { OE oe ligature }
        TI bl -> '!6' -> 196; { tilde }
        TI A -> 'A!6' -> 135;   TI a -> 'a!6' -> 97; { A a tilde }
        TI O -> 'O!6' -> 153;   TI o -> 'o!6' -> 22; { O o tilde }
        TI N -> 'N!6' -> 149;   TI n -> 'n!6' -> 18; { N n tilde }
        EN O -> 'O!9' -> 156;   EN o -> 'o!9' -> 25; { O o oblique }
        RI bl -> '!10' -> 202; { ronde }
        RI A -> 'A!10'-> 137;   RI a -> 'a!10'->  6; { A a ronde }
        BR bl -> '!12' -> 198; { breve }
END
```

```
{*****************************************************************************
     Ce schema decrit la transcription des caracteres accentues vietnamiens

     H. Kh. PHAN   Juillet 1990.
*****************************************************************************}

TRANSCRIPT Vietnamien;

CONSTS { utilises dans l'ecriture des chaines de caracteres }
   { Les lettres }
   A = 'A';   a = 'a';
   D = 'D';   d = 'd';
   E = 'E';   e = 'e';
   I = 'I';   i = 'i';
   O = 'O';   o = 'o';
   U = 'U';   u = 'u';
   Y = 'Y';   y = 'y';
   { Les signes }
   CI = 158; { ^ circonflexe }
   BR = 167; { u breve }
   VQ = 170; { ) crochet }
   GR = 156; { ` grave }
   QU = 171; { ? question }
   TI = 161; { ~ tilde }
   AC = 157; { ' aigu }
   PE = 168; { . point }
   EN = 164; { - barre (pour D et d) }

PARAMS_IN
   Set = Romain;
   Language = vietnamien;
   Entry = LEFTCOMPO;
   Letters = (A, D, E, I, O, U, Y, a, d, e, i, o, u, y);
   Signes  = (CI, BR, VQ, GR, QU, TI, AC, PE, EN);
   Max_Signe = 2; { eventuellement deux signes par lettre }

PARAMS_OUT
   Bytes  = 1; { un octet par code d'affichage }
   UppLow = Yes;
   Family = ('Times', 'Helvetica', 'Courier');
   Style  = ('Romain', 'Bold', 'Italic');
   Size   = (9, 13, 17, 22, 33, 48);
   DialogFont = 'vtr13';

CODES
{***
     signe simple
***}
   CI A -> 'A!3'  ->  20;   CI a -> 'a!3'  ->   4; { A a circonflexe }
   CI E -> 'A!3'  ->  23;   CI e -> 'a!3'  ->   7; { E e circonflexe }
   CI O -> 'O!3'  ->  29;   CI o -> 'a!3'  ->  14; { O o circonflexe }
   BR A -> 'A!12' -> 202;   BR a -> 'a!12' -> 130; { A a breve }
   VQ O -> 'O!15' -> 231;   VQ o -> 'o!15' -> 180; { O o crochet }
   VQ U -> 'U!15' -> 240;   VQ u -> 'u!15' -> 189; { U u crochet }
   GR A -> 'A!2'  ->  17;   GR a -> 'a!2'  ->   1; { A a grave }
   GR E -> 'E!2'  ->  21;   GR e -> 'e!2'  ->   5; { E e grave }
   GR I -> 'I!2'  ->  24;   GR i -> 'i!2'  ->   8; { I i grave }
```

```
   GR O -> 'O!2'  ->  26;   GR o -> 'o!2'  ->  11; { O o grave }
   GR U -> 'U!2'  ->  30;   GR u -> 'u!2'  ->  15; { U u grave }
   GR Y -> 'Y!2'  -> 246;   GR y -> 'y!2'  -> 195; { Y y grave }
   QU A -> 'A!16' -> 200;   QU a -> 'a!16' -> 128; { A a interrogation }
   QU E -> 'E!16' -> 213;   QU e -> 'e!16' -> 141; { E e interrogation }
   QU I -> 'I!16' -> 221;   QU i -> 'i!16' -> 170; { I i interrogation }
   QU O -> 'O!16' -> 224;   QU o -> 'o!16' -> 173; { O o interrogation }
   QU U -> 'U!16' -> 237;   QU u -> 'u!16' -> 186; { U u interrogation }
   QU Y -> 'Y!16' -> 247;   QU y -> 'y!16' -> 196; { Y i interrogation }
   TI A -> 'A!6'  ->  18;   TI a -> 'a!6'  ->   2; { A a tilde }
   TI E -> 'E!6'  -> 214;   TI e -> 'e!6'  -> 142; { E e tilde }
   TI I -> 'I!6'  -> 222;   TI i -> 'i!6'  -> 171; { I i tilde }
   TI O -> 'O!6'  ->  27;   TI o -> 'o!6'  ->  12; { O o tilde }
   TI U -> 'U!6'  -> 238;   TI u -> 'u!6'  -> 187; { U u tilde }
   TI Y -> 'Y!6'  -> 248;   TI y -> 'y!6'  -> 197; { Y y tilde }
   AC A -> 'A!1'  ->  19;   AC a -> 'a!1'  ->   3; { A a aigu }
   AC E -> 'E!1'  ->  22;   AC e -> 'e!1'  ->   6; { E e aigu }
   AC I -> 'I!1'  ->  25;   AC i -> 'i!1'  ->   9; { I i aigu }
   AC O -> 'O!1'  ->  28;   AC o -> 'o!1'  ->  13; { O o aigu }
   AC U -> 'U!1'  ->  31;   AC u -> 'u!1'  ->  16; { U u aigu }
   AC Y -> 'Y!1'  -> 249;   AC y -> 'y!1'  -> 198; { Y y aigu }
   PE A -> 'A!13' -> 201;   PE a -> 'a!13' -> 129; { A a point }
   PE E -> 'E!13' -> 215;   PE e -> 'e!13' -> 143; { E e point }
   PE I -> 'I!13' -> 223;   PE i -> 'i!13' -> 172; { I i point }
   PE O -> 'O!13' -> 225;   PE o -> 'o!13' -> 174; { O o point }
   PE U -> 'U!13' -> 239;   PE u -> 'u!13' -> 188; { U u point }
   PE Y -> 'Y!13' -> 250;   PE y -> 'y!13' -> 199; { Y y point }
   EN D -> 'D!9'  -> 252;   EN d -> 'd!9'  -> 251; { D d barre }
{***
     signe compose
***}
   { A a breve grave }
   BR GR A -> 'A!12!2'  -> 203;   BR GR a -> 'a!12!2'  -> 131;
   { A a breve interrog }
   BR QU A -> 'A!12!16' -> 204;   BR QU a -> 'a!12!16' -> 132;
   { A a breve tilde }
   BR TI A -> 'A!12!6'  -> 205;   BR TI a -> 'a!12!6'  -> 133;
   { A a breve aigu }
   BR AC A -> 'A!12!1'  -> 206;   BR AC a -> 'a!12!1'  -> 134;
   { A a breve point }
   BR PE A -> 'A!12!13' -> 207;   BR PE a -> 'a!12!13' -> 135;
   { A a circonflexe grave }
   CI GR A -> 'A!3!2'   -> 208;   CI GR a -> 'a!3!2'   -> 136;
   { E e circonflexe grave }
   CI GR E -> 'E!3!2'   -> 216;     CI GR e -> 'e!3!2' -> 165;
   { O o circonflexe grave }
   CI GR O -> 'O!3!2'   -> 226;   CI GR o -> 'o!3!2'   -> 175;
   { A a circonflexe interrog }
   CI QU A -> 'A!3!16'  -> 209;   CI QU a -> 'a!3!16'  -> 137;
   { E e circonflexe interrog }
   CI QU E -> 'E!3!16'  -> 217;   CI QU e -> 'e!3!16'  -> 166;
   { O o circonflexe interrog }
   CI QU O -> 'O!3!16'  -> 227;   CI QU o -> 'o!3!16'  -> 176;
   { A a circonflexe tilde }
   CI TI A -> 'A!3!6'   -> 210;   CI TI a -> 'a!3!6'   -> 138;
   { E e circonflexe tilde }
   CI TI E -> 'E!3!6'   -> 218;   CI TI e -> 'e!3!6'   -> 167;
   { O o circonflexe tilde }
   CI TI O -> 'O!3!6'   -> 228;   CI TI o -> 'o!3!6'   -> 177;
   { A a circonflexe aigu }
```

```
CI AC A -> 'A!3!1'   -> 211;   CI AC a -> 'a!3!1'   -> 139;
{ E e circonflexe aigu }
CI AC E -> 'E!3!1'   -> 219;   CI AC e -> 'e!3!1'   -> 168;
{ O o circonflexe aigu }
CI AC O -> 'O!3!1'   -> 229;   CI AC o -> 'o!3!1'   -> 178;
{ A a circonflexe point }
CI PE A -> 'A!3!13'  -> 212;   CI PE a -> 'a!3!13'  -> 140;
{ E e circonflexe point }
CI PE E -> 'E!3!13'  -> 220;   CI PE e -> 'e!3!13'  -> 169;
{ O o circonflexe point }
CI PE O -> 'O!3!13'  -> 230;   CI PE o -> 'o!3!13'  -> 179;
{ O o crochet grave }
VQ GR O -> 'O!15!2'  -> 232;   VQ GR o -> 'o!15!2'  -> 181;
{ U u crochet grave }
VQ GR U -> 'U!15!2'  -> 241;   VQ GR u -> 'u!15!2'  -> 190;
{ O o crochet interrog }
VQ QU O -> 'O!15!16' -> 233;   VQ QU o -> 'o!15!16' -> 182;
{ U u crochet interrog }
VQ QU U -> 'U!15!16' -> 242;   VQ QU u -> 'u!15!16' -> 191;
{ O o crochet tilde }
VQ TI O -> 'O!15!6'  -> 234;   VQ TI o -> 'o!15!6'  -> 183;
{ U u crochet tilde }
VQ TI U -> 'U!15!6'  -> 243;   VQ TI u -> 'u!15!6'  -> 192;
{ O o crochet aigu }
VQ AC O -> 'O!15!1'  -> 235;   VQ AC o -> 'o!15!1'  -> 184;
{ U u crochet aigu }
VQ AC U -> 'U!15!1'  -> 244;   VQ AC u -> 'u!15!1'  -> 193;
{ O o crochet point }
VQ PE O -> 'O!15!13' -> 236;   VQ PE o -> 'o!15!13' -> 185;
{ U u crochet point }
VQ PE U -> 'U!15!13' -> 245;   VQ PE u -> 'u!15!13' -> 194;

END
```

```
{*************************************************************************
 Ce schema decrit la transcription de l'alphabet Grec

            H. Kh. PHAN     Juillet 1990
*************************************************************************}

TRANSCRIPT Grec;

CONSTS { Pour simplifier l'ecriture des chaines de caracteres }

{ Les lettres }
        a = 'a';          b = 'b';
        c = 'c';          d = 'd';

{ Les signes }
        S = 161; { pour entrer un symbol }

PARAMS_IN
        Set = Greque;
        Language = grec;
        Entry = LEFTCOMPO; { composition a gauche }
        Letters = (a, b, c, d);
        Signes  = (S);
        Max_Signe = 1; { un seul signe par lettre }

PARAMS_OUT
        Bytes  = 1; { un octet par code d'affichage }
        UppLow = Yes;
        Family = ('Times', 'Helvetica', 'Courier');
        Style  = ('Romain', 'Bold', 'Italic');
        Size   = (9, 13, 17, 22, 33, 48);
        DialogFont = 'gtr13';

CODES
{ Pour la saisie de 4 symboles mathematiques existant dans Grif }
        S a -> 'E!1' -> 1; { union }
        S b -> 'e!1' -> 3; { insertion }
        S c -> 'A!2' -> 2; { egal et inferieur a }
        S d -> 'a!2' -> 4; { egal et superieur a }

END
```

```
{*************************************************************************
  Ce schema decrit la transcription des caracteres accentues Russes

            H. Kh. PHAN   Juillet 1990
*************************************************************************}

TRANSCRIPT Russe;
CONSTS { Pour simplifier l'ecriture des chaines de caracteres }
{ Les lettres }
        A =  65;        a =  97;
        B = 'B';        b = 'b';
        C = 'C';        c = 'c';
        E = 'E';        e = 'e';
        I = 'I';        i = 'i';
        O = 'O';        o = 'o';
        R = 'R';        r = 'r';
        S = 'S';        s = 's';
        T = 'T';        t = 't';
        U = 'U';        u = 'u';
        X = 'X';        x = 'x';
        V = 'V';        v = 'v';
{ Les signes }
        SS = 161; { accent symbolise }
PARAMS_IN
        Set = Ciryllique;
        Language = russe;
        Entry = LEFTCOMPO;
        Letters = ( A, B, C, E, I, O, R, S, T, U, V, X,
                                a, b, c, e, i, o, r, s, t, u, v, x );
        Signes  = ( SS );
        Max_Signe = 1; { un seul signe par lettre }
PARAMS_OUT
        Bytes  = 1; { un octet par code d'affichage }
        UppLow = Yes;
        Family = ('cyr-s');
        { Style  = (); }
        Size   = ( 25 );
        DialogFont = 'cyr-s25';
CODES
{ la transcription n'est pas utilisee, representee par * }
        SS A -> '*' -> 17;   SS a -> '*' -> 62;
        SS B -> '*' -> 39;   SS b -> '*' -> 96;
        SS C -> '*' -> 95;   SS c -> '*' -> 25;
        SS E -> '*' -> 19;   SS e -> '*' -> 31;
        SS I -> '*' -> 16;   SS i -> '*' -> 60;
        SS O -> '*' -> 37;   SS o -> '*' -> 36;
        SS R -> '*' -> 94;   SS r -> '*' ->  1;
        SS S -> '*' -> 64;   SS s -> '*' -> 22;
        SS T -> '*' -> 35;   SS t -> '*' -> 34;
        SS U -> '*' -> 30;   SS u -> '*' -> 61;
        SS V -> '*' -> 18;   SS v -> '*' ->  4;
        SS X -> '*' -> 38;   SS x -> '*' -> 43;
END
```

```
{*************************************************************************
 Ce schema decrit la transcription des caracteres Chinois

            H. Kh. PHAN     Juillet 1990
*************************************************************************}

TRANSCRIPT Chinois;

PARAMS_IN
        Set = Hanzi; { jeu de caracteres simplifies }
        Language = chinois;
        Entry = SELECT; { saisie par choix }
        Wind = WPHON;    { fenetre de saisie phonetique }

PARAMS_OUT
        Bytes  = 2; { code GB : deux octets par code d'affichage }
        UppLow = No;
        Family = ( 'beijing' );
        { Style  = ();}
        Size   = (16, 24);
        DialogFont = 'beijing16';

CODES
{ ces regles ne sont pas encore utilisees dans la realisation }
        'a'     -> ('a1' -> 2639, 'a2' -> 1601, 'a3' -> 1602);
        'a1'-> ('a4' -> 2639, 'a5' -> 6325, 'a6' -> 1601, 'a7' -> 7925,
                        'a8' -> 7571, 'a9' -> 1602);
        'a2'-> ('a10' -> 2639, 'a11' -> 6436, 'a12' -> 1601);
        'a3'-> ('a13' -> 2639, 'a14' -> 1601);
        'a4'-> ('a15' -> 2639, 'a16' -> 1601);
        'ai1'->('a17' -> 1603, 'a18' -> 6263, 'a19' -> 1604, 'a20' -> 1605,
                                'a21' -> 1606, 'a22' -> 7945, 'a23' -> 1607);
        'ai2'->('a24' -> 6263, 'a25' -> 1604, 'a26' -> 2084, 'a27' -> 1608, 'a28
' -> 1609);
        'ai3'->('a29' -> 1610, 'a30' -> 8616, 'a31' -> 1605, 'a32' -> 6440, 'a33
' -> 1611);

END
```

# ANNEXE 6 Données techniques de programmation

## Les nouveaux modules introduits

Les tableaux suivants donnent le nombre d'octets utilisés dans les modules introduits ordonnés suivant les répertoires :

Le répertoire `chinois` qui a été ajouté dans l'ensemble de 15 répertoires de Grif (voir annexe 2) contient des données entrée-sortie pour les caractères chinois.

| *Module :* | *Description :* | *Nombre d'octets :* |
|---|---|---|
| `CODE0` | Code GB pour l'affichage | 15 642 |
| `Pinyin.data` | Données en pinyin | 46 063 |
| `TABLE` | Entrée en pinyin | 14 991 |
| `CC` | Image des caractères chinois | 801 444 |
| *Total* | 4 modules | 878 140 |

Le répertoire `h` pour les constantes et les types de données :

| *Module :* | *Description :* | *Nombre d'octets :* |
|---|---|---|
| `consttrans` | Constantes pour les schémas E | 744 |
| `constsaisie` | Constantes pour la saisie | 1 382 |
| `typetrans` | Types pour les schémas E | 5 022 |
| `typesaisie` | Types pour la saisie | 755 |
| *Total* | 4 modules | 7 903 |

Le répertoire `var` pour les variables globales :

| *Module :* | *Description :* | *Nombre d'octets :* |
|---|---|---|
| `saisie` | Pour `Éditeur` | 677 |
| `saisiec` | Pour `Médiateur` | 1 889 |
| `gb` | Pour imprimer des car. chinois | 851 |
| *Total* | 3 modules | 3 147 |

Le répertoire `f` pour la définition des procédures externes :

| *Module :* | *Description :* | *Nombre d'octets :* |
|---|---|---|
| `alphabet` | Gestion des alphabets | 643 |
| `fnsaisie` | Gestion des fenêtres de saisie | 280 |
| `gbcc` | Traitement des codes GB | 336 |
| `rdschtrans` | Lecture d'un schéma E | 589 |
| `saisie` | Gestion des méthodes de saisie | 319 |
| `tab` | Chargement des schémas E | 541 |
| `wrschtrans` | Écriture d'un schéma E | 263 |
| `media` | Service d'affichage | 732 |
| *Total* | 8 modules | 3 703 |

Le répertoire `makes` pour la construction des programmes du répertoire `bin`:

| *Module :* | *Description :* | *Nombre d'octets :* |
|---|---|---|
| `mk_mgrif` | Construction de `mgrif` | 17 123 |
| `mk_mprint` | Construction de `mprint` | 9 291 |
| `mk_transc` | Construction de `transc` | 1 869 |
| *Total* | 3 modules | 28 283 |

Le répertoire `compil` pour la construction du compilateur E :

| *Module :* | *Description :* | *Nombre d'octets :* |
|---|---|---|
| `TRANSC.GRM` | Codes syntaxiques de E | 2 150 |
| `transc.p` | Compilation de E | 30 818 |
| `wrschtrans.p` | Sauvegarde d'un schéma E | 10 606 |
| *Total* | 3 modules | 43 574 |

Le répertoire `Éditeur` :

| *Module :* | *Description :* | *Nombre d'octets :* |
|---|---|---|
| `alphabet` | Changement d'alphabet | 11 932 |
| `rdschtrans` | Chargement des schémas E | 13 326 |
| *Total* | 2 modules | 25 258 |

Le répertoire `Médiateur` :

| *Module :* | *Description :* | *Nombre d'octets :* |
|---|---|---|
| `média` | Conversion des données | 5 374 |
| `fnsaisie` | Gestion des fenêtres de saisie | 8 591 |
| `saisie` | Traitement de saisie | 22 235 |
| `tab` | Initialisation des tables de E | 15 327 |
| `gbcc` | Dessin des caractères chinois | 12 578 |
| *Total* | 5 modules | 64 105 |

## Extention sur les deux composants Éditeur et Médiateur

Les modules d'`Éditeur` en Pascal :

| *Description du traitement :* | *Nombre de lignes ajoutées :* |
|---|---|
| Traitement de changement d'alphabet | 399 |
| Chargement des schémas de transcription | 478 |
| Chargement des schémas de structure | 3 |
| Manipulatiuon des arbres abstraits | 21 |
| Commande de recherche de texte | 159 |
| Traitement des commandes d'édition | 182 |
| Création des images abstraites | 13 |
| Trace et mise au point des arbres abstraits | 43 |
| Démarrage de l'éditeur | 29 |
| Messages et menus en français | 54 |
| Manipulation des images | 4 |
| Lecture d'un fichier de la forme pivot | 122 |
| Transforme dans la forme pivot | 66 |
| Application des règles de présentation | 14 |
| Commande de sortie directe en PostScript | 11 |
| *Total* | 1598 |

Les modules de `Médiateur` en Pascal et en C :

| *Description du traitement :* | *Nombre de lignes ajoutées :* |
|---|---|
| Conversion des données | 194 |
| Affichage des boîtes de texte | 44 |
| Gestion des commandes locales | 131 |
| Défilement dans les fenêtres | 15 |
| Programme de l'éditeur Grif | 24 |
| Gestion des images abstraites | 25 |
| Mise en lignes des boîtes | 2 |
| Trace des états du `Médiateur` | 15 |
| Gestion des sélection | 1 |
| Gestion des fenêtres de saisie | 299 |
| Traitement des images de caractères chinois | 455 |
| Traitement de saisie | 738 |
| Initialisation des tables de transcription | 503 |
| Gestion des catalogues | 24 |
| Gestion des entrées/sorties pour X | 5 |
| Gestion des polices de caractères | 77 |
| Gestion des menus | 155 |
| Gestion des sorties en PostScript | 124 |
| Gestion des fichiers PostScript | 1035 |
| Gestion des fenêtres X-Window | 78 |
| *Total* | 6100 |

Voici le pourcentage de modification :

| *Composant :* | *Pourcentage de modification :* |
|---|---|
| `Éditeur` | 4.35 % |
| `Médiateur` | 24.72% |

# ANNEXE 7 Code ASCII étendu pour l'alphabet latin

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | à | á | â | a | ä | å | æ | ç | | è | ê | ë | ì | í | î |
| 1 | ï | bl | ñ | ò | ó | ô | õ | ö | œ | ø | ù | ú | û | ü | ÿ | é |
| 2 | | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ‘ | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | |
| 8 | ß | ł | š | ž | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë |
| 9 | Ì | Í | Î | Ï | Ł | Ñ | Ò | Ó | Ô | Õ | Ö | Œ | Ø | Š | Ù | Ú |
| A | Û | ¡ | ¢ | £ | / | ¥ | *f* | § | ¤ | ' | “ | « | ‹ | › | fi | fl |
| B | Ü | – | † | ‡ | · | Ÿ | ¶ | • | ‚ | „ | ” | » | … | ‰ | Ž | ¿ |
| C | ¯ | ` | ´ | ^ | ~ | - | ˘ | ˙ | ¨ | ˚ | ˛ | ˝ | ˇ | | | |
| D | | | | | | | | | | | | | | | | |
| E | | | | | | | | | | | | | | | | |
| F | | | | | | | | | | | | | | | | |

code ASCII standard

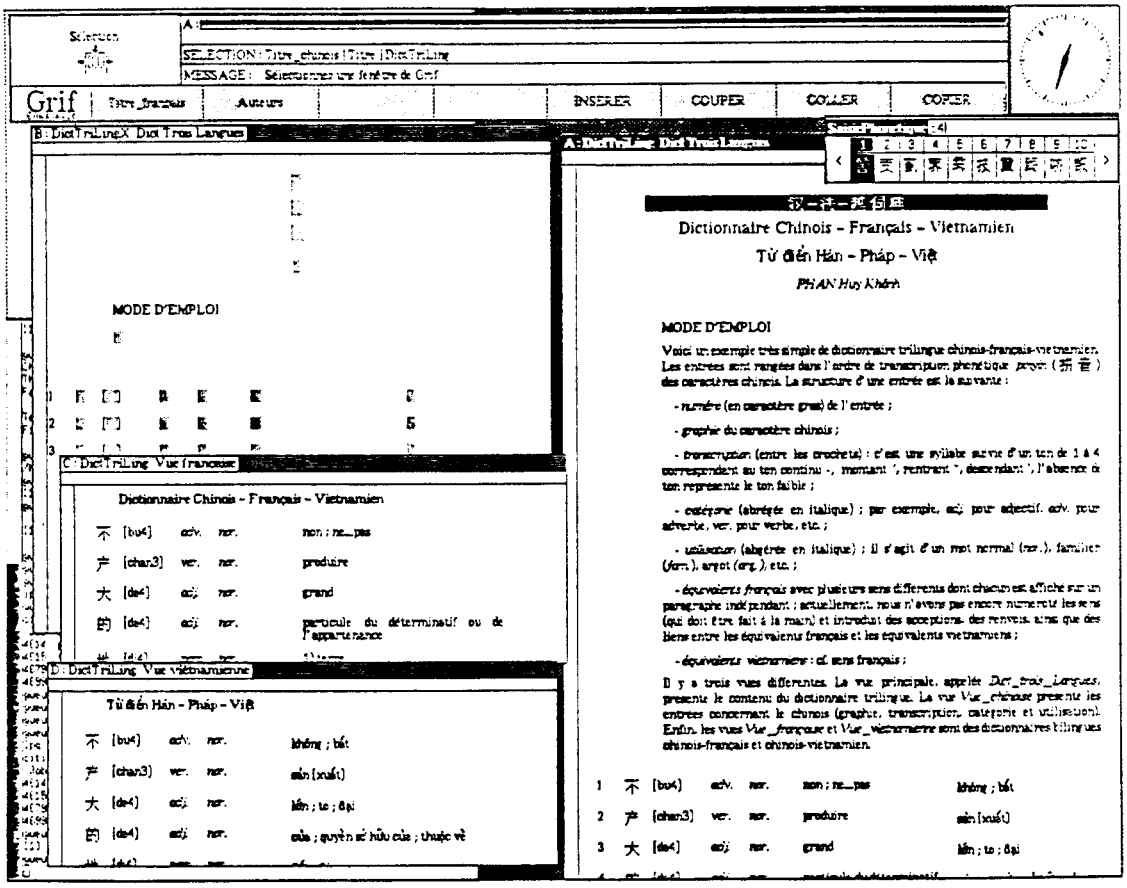
code non utilisé

bl un blanc non sécable

Imprimé par Grif le 20/3/91

# ANNEXE 8 Dictionnaire trilingue Chinois – Français – Vietnamien

Cette annexe présente un exemple d'une classe de documents : "DictTriLing" utilisée dans Grif pour produire un dictionnaire trilingue Chinois - Français - Vietnamien. L'image suivante présente l'édition de ce dictionnaire à partir de sa forme de "squelette" et les différentes vues sous Grif multilingue :




Ensuite, nous donnons quelques pages imprimées de ce dictionnaire. Nous donnons aussi le schéma de structure en langage S, le schéma de présentation en langage P de la classe "DictTriLing", ainsi que l'arbre abstrait et l'image abstraite représentant cette image concrète.

# 汉 - 法 - 越 伺 典

# Dictionnaire Chinois - Français - Vietnamien

# Từ điển Hán - Pháp - Việt

*PHAN Huy Khánh*

## MODE D'EMPLOI

Voici un exemple très simple de dictionnaire trilingue chinois-français-vietnamien. Les entrées sont rangées dans l'ordre de transcription phonétique *pinyin* ( 拼 音) des caractères chinois. La structure d'une entrée est la suivante :

- *numéro* (en caractère gras) de l'entrée ;

- *graphie* du caractère chinois ;

- *transcription* (entre les crochets) : c'est une syllabe suivie d'un ton de 1 à 4 correspondant au ton continu ¯, montant ´, rentrant ˇ, descendant `, l'absence de ton représente le ton faible ;

- *catégorie* (abrégée en italique) ; par exemple, *adj.* pour adjectif, *adv.* pour adverbe, *ver.* pour verbe, etc. ;

- *utilisation* (abgérée en italique) ; il s'agit d'un mot normal (*nor.*), familier (*fam.*), argot (*arg.*), etc. ;

- *équivalents français* avec plusieurs sens différents dont chacun est affiché sur un paragraphe indépendant ; actuellement, nous n'avons pas encore numéroté les sens (qui doit être fait à la main) et introduit des acceptions, des renvois, ainsi que des liens entre les équivalents français et les équivalents vietnamiens ;

- *équivalents vietnamiens* : cf. sens français ;

Il y a trois vues différentes. La vue principale, appelée *Dict_trois_Langues*, présente le contenu du dictionnaire trilingue. La vue *Vue_chinoise* présente les entrées concernant le chinois (graphie, transcription, catégorie et utilisation). Enfin, les vues *Vue_française* et *Vue_vietnamienne* sont des dictionnaires bilingues chinois-français et chinois-vietnamien.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 不 | [bu4] | *adv.* | *nor.* | non ; ne...pas | không ; bất |
| 2 | 产 | [chan3] | *ver.* | *nor.* | produire | sản [xuất] |
| 3 | 大 | [da4] | *adj.* | *nor.* | grand | lớn ; to ; đại |
| 4 | 的 | [da4] | *adj.* | *nor.* | particule du déterminatif ou de l'appartenance | của ; quyền sở hữu của ; thuộc về |

| | | | | | | |
|---|---|---|---|---|---|---|
| 5 | 地 | [di4] | *nom* | *nor.* | 1) terre<br>2) suffixe d'adverbe | đất ; địa |
| 6 | 动 | [dong4] | *ver.* | *nor.* | bouger ; toucher | động ; chạm |
| 7 | 个 | [ge4] | *adj.* | *nor.* | spécificatif | để chỉ rõ ; để định rõ |
| 8 | 工 | [gong1] | *nom* | *nor.* | 1) ouvrier<br>2) travaux | công ; công việc ; công nhân |
| 9 | 国 | [guo2] | *nom* | *nor.* | pays ; État | nước ; quốc gia |
| 10 | 和 | [he1] | *var.* | *nor.* | 1) paix<br>2) et ; avec | bình ; hòa bình |
| 11 | 门 | [men1] | *adj.* | *nor.* | suffixe de pluriel | các ; những |
| 12 | 人 | [ren1] | *nom* | *nor.* | être humain | người ; nhân |
| 13 | 时 | [shi1] | *nom* | *nor.* | temps ; heure | giờ ; thì giờ ; thời giờ |
| 14 | 为 | [wei4] | *prép.* | *nor.* | pour | thay ; thay cho ; đổi lấy ; làm |
| 15 | 要 | [yao4] | *ver.* | *nor.* | vouloir | muốn ; háo ; thèm |
| 16 | 有 | [you3] | *ver.* | *nor.* | avoir | có ; cảm thấy ; được ; mua được |
| 17 | 以 | [yi3] | *prép.* | *nor.* | par ; avec | bởi ; với, cùng với |

# 汉-法-越伺典

| | | | | |
|---|---|---|---|---|
| 1 | 不 | [bu4] | *adv.* | *nor.* |
| 2 | 产 | [chan3] | *ver.* | *nor.* |
| 3 | 大 | [da4] | *adj.* | *nor.* |
| 4 | 的 | [da4] | *adj.* | *nor.* |
| 5 | 地 | [di4] | *nom* | *nor.* |
| 6 | 动 | [dong4] | *ver.* | *nor.* |
| 7 | 个 | [ge4] | *adj.* | *nor.* |
| 8 | 工 | [gong1] | *nom* | *nor.* |
| 9 | 国 | [guo2] | *nom* | *nor.* |
| 10 | 和 | [he1] | *var.* | *nor.* |
| 11 | 门 | [men1] | *adj.* | *nor.* |
| 12 | 人 | [ren1] | *nom* | *nor.* |
| 13 | 时 | [shi1] | *nom* | *nor.* |
| 14 | 为 | [wei4] | *prép.* | *nor.* |
| 15 | 要 | [yao4] | *ver.* | *nor.* |
| 16 | 有 | [you3] | *ver.* | *nor.* |
| 17 | 以 | [yi3] | *prép.* | *nor.* |

Dictionnaire Chinois - Français - Vietnamien

| | | | | | |
|---|---|---|---|---|---|
| 1 | 不 | [bu4] | *adv.* | *nor.* | non ; ne...pas |
| 2 | 产 | [chan3] | *ver.* | *nor.* | produire |
| 3 | 大 | [da4] | *adj.* | *nor.* | grand |
| 4 | 的 | [da4] | *adj.* | *nor.* | particule du déterminatif ou de l'appartenance |
| 5 | 地 | [di4] | *nom* | *nor.* | 1) terre<br>2) suffixe d'adverbe |
| 6 | 动 | [dong4] | *ver.* | *nor.* | bouger ; toucher |
| 7 | 个 | [ge4] | *adj.* | *nor.* | spécificatif |
| 8 | 工 | [gong1] | *nom* | *nor.* | 1) ouvrier<br>2) travaux |
| 9 | 国 | [guo2] | *nom* | *nor.* | pays ; État |
| 10 | 和 | [he1] | *var.* | *nor.* | 1) paix<br>2) et ; avec |
| 11 | 门 | [men1] | *adj.* | *nor.* | suffixe de pluriel |
| 12 | 人 | [ren1] | *nom* | *nor.* | être humain |
| 13 | 时 | [shi1] | *nom* | *nor.* | temps ; heure |
| 14 | 为 | [wei4] | *prép.* | *nor.* | pour |
| 15 | 要 | [yao4] | *ver.* | *nor.* | vouloir |
| 16 | 有 | [you3] | *ver.* | *nor.* | avoir |
| 17 | 以 | [yi3] | *prép.* | *nor.* | par ; avec |

Từ điển Hán - Pháp - Việt

| | | | | | |
|---|---|---|---|---|---|
| 1 | 不 | [bu4] | *adv.* | *nor.* | không ; bất |
| 2 | 产 | [chan3] | *ver.* | *nor.* | sản [xuất] |
| 3 | 大 | [da4] | *adj.* | *nor.* | lớn ; to ; đại |
| 4 | 的 | [da4] | *adj.* | *nor.* | của ; quyền sở hữu của ; thuộc về |
| 5 | 地 | [di4] | *nom* | *nor.* | đất ; địa |
| 6 | 动 | [dong4] | *ver.* | *nor.* | động ; chạm |
| 7 | 个 | [ge4] | *adj.* | *nor.* | để chỉ rõ ; để định rõ |
| 8 | 工 | [gong1] | *nom* | *nor.* | công ; công việc ; công nhân |
| 9 | 国 | [guo2] | *nom* | *nor.* | nước ; quốc gia |
| 10 | 和 | [he1] | *var.* | *nor.* | bình ; hòa bình |
| 11 | 门 | [men1] | *adj.* | *nor.* | các ; những |
| 12 | 人 | [ren1] | *nom* | *nor.* | người ; nhân |
| 13 | 时 | [shi1] | *nom* | *nor.* | giờ ; thì giờ ; thời giờ |
| 14 | 为 | [wei4] | *prép.* | *nor.* | thay ; thay cho ; đổi lấy ; làm |
| 15 | 要 | [yao4] | *ver.* | *nor.* | muốn ; háo ; thèm |
| 16 | 有 | [you3] | *ver.* | *nor.* | có ; cảm thấy ; được ; mua được |
| 17 | 以 | [yi3] | *prép.* | *nor.* | bởi ; với, cùng với |

```
{*****************************************************************************
          Ce schema decrit la structure d'un dictionnaire trilingue.

          H. Kh. PHAN          Mars 1991
 *****************************************************************************}

STRUCTURE DictTriLing;

DEFPRES DictTriLingP;

ATTR

   Langue     = Chinois, Fran\10ais, Vi\13tnamien;
   Importance= D\37finition,  Mot_important;

STRUCT

   DictTriLing = BEGIN
   Titre = BEGIN
               Titre_chinois      = Texte_chinois;
               Titre_fran\10ais   = Texte_fran\10ais;
               Titre_vi\13tnamien= Texte_vi\13tnamien;
               END;
   Auteurs = LIST OF (Auteur = Texte_fran\10ais);
   Mode_Emploi = LIST OF (Paragr_Mode_Emploi = Texte_fran\10ais);
   CorpsDict = LIST [2..*] OF (Entr\37e =
               BEGIN
               Graphie       = Texte_chinois;
               Transcription= Texte_fran\10ais;
               Cat\37gorie   = Texte_fran\10ais;
               Utilisation   = Texte_fran\10ais;
               Equival_fran\10ais    = LIST OF
                           (Sens_fran\10ais=Texte_fran\10ais);
               Equival_vi\13tnamien = LIST OF
                           (Sens_vi\13tnamien=Texte_vi\13tnamien);
               END);
   END;

   Texte_chinois      = Text WITH Langue=Chinois;
   Texte_fran\10ais   = Text WITH Langue=Fran\10ais;
   Texte_vi\13tnamien= Text WITH Langue=Vi\13tnamien;
END
```

```
{*************************************************************************
          Ce schema decrit la presentation utilisee pour l'edition
          d'un dictionnaire trilingue avec des pages.

          B. Kh. PHAN          Mars 1991
 *************************************************************************}

PRESENTATION DictTriLing;

VIEWS

   Dict_Trois_Langues, Vue_chinoise, Vue_fran\10aise, Vue_vi\13tnamienne;

PRINT

   Dict_Trois_Langues, Vue_chinoise, Vue_fran\10aise, Vue_vi\13tnamienne;

COUNTERS

   CptPage : Rank of Page;
   CptEntr\37e: Rank of Entr\37e;

DEFAULT

   BEGIN
   HorizRef : Enclosed . HRef;
   VertRef : * . Left;
   Width : Enclosing . Width;
   Height : Enclosed . Height;
   VertPos : Top = Previous . Bottom;
   HorizPos : Left = Previous . Left;
   Justify : Enclosing =;
   LineSpacing : Enclosing =;
   Break: Yes;
   Visibility: Enclosing =;
   Font : Enclosing =;
   Style : Enclosing =;
   Size : Enclosing =;
{   Adjust : Enclosing =; }
   Indent : Enclosing =;
   Depth : 0;
   END;

BOXES

   BoiteNumPage :
          BEGIN
          Content : (Text '- ' Value (CptPage, Arabic) Text ' -');
          VertPos : Top = Previous SAUT_PAGE . Bottom + 0.3 cm;
          HorizPos: VMiddle = Previous SAUT_PAGE . VMiddle;
          Height:  0.8 cm;
          Size: 2;
          Font: Times;
          Style: Roman;
          END;

   BoiteBasPage:
          BEGIN
          Height: 0.6 cm;
          Width: 1 pt;
          VertPos: Bottom = Next SAUT_PAGE . Top;
          HorizPos: Left = Next SAUT_PAGE . Left;
          Content: (Graphics ' ');
          END;

   LaPage :
```

```
            BEGIN
            Width: 15 cm;
            Height: 26 cm;
            VertPos: Top = Enclosing . Top + 2 cm;
            HorizPos: Left = Enclosing . Left + 3 cm;
            IF Not One(CptPage) CreateAfter (BoiteNumPage);
            IF Not One(CptPage) CreateBefore (BoiteBasPage);
            END;

   BoiteInstrFran:
            BEGIN
Width : Enclosing . Width * 25%;
            VertPos: Top = Previous . Bottom + 1 cm;
            HorizPos: Left = Next . Left;
            Content: Text 'MODE D''EMPLOI';
            Style: Bold;
            END;

   NumeroEntr\37e:
            BEGIN
            Size: Enclosing=;
Style: Bold;
            VertPos: Top = Enclosing . Top;
            Content: (VALUE (CptEntr\37e, Arabic));
IN Vue_chinoise
               BEGIN
               VertPos: Top = Enclosing . Top + 0.3;
               Size: Enclosing =;
               Width: 2;
               END;
            END;

   CrochetOuvr:
            Content: Text '[';

   CrochetFerm:
            Content: Text ']';

RULES

    DictTriLing:
            BEGIN
            Page(LaPage);
            Font: Times;
            Style: Roman;
            Size : 2;
            Width : Enclosing . Width;
            HorizPos : VMiddle = Enclosing . VMiddle;
            Visibility: 5;
            Indent : 0;
            { Adjust : Left; }
            Justify : Yes;
            LineSpacing : 1;
            END;

   Titre :
            BEGIN
            Width : Enclosing . Width * 80%;
            Size : Enclosing + 1;
            VertPos : Top = Enclosing . Top + 1 cm;
            HorizPos : VMiddle = Enclosing . VMiddle;
Justify : No;
Indent : 0;
            END;

   Titre_chinois :
            BEGIN
Width: Enclosing . Width;
```

```
            VertPos : Top = Enclosing . Top;
                      Line(VMiddle);
            {  Adjust : VMiddle; }
                      IN Vue_chinoise
               BEGIN
               Visibility: 2;
               Size: Enclosing - 1;
      VertPos: Top = Enclosing . Top + 1 cm;
                           Line(Left);
               END;
   IN Vue_fran\10aise
      Visibility: 0;
   IN Vue_vi\13tnamienne
      Visibility: 0;
            END;

      Titre_fran\10ais :
            BEGIN
   Width: Enclosing . Width;
            VertPos : Top = Previous . Bottom + 0.5;
                      Line(VMiddle);
            {  Adjust : VMiddle; }
   IN Vue_chinoise
      Visibility: 0;
            IN Vue_fran\10aise
               BEGIN
               Visibility: 2;
               Size: Enclosing - 1;
      Style: Bold;
      VertPos: Top = Enclosing . Top + 1 cm;
                           Line(Left);
               END;
   IN Vue_vi\13tnamienne
      Visibility: 0;
            END;

      Titre_vi\13tnamien :
            BEGIN
   Width: Enclosing . Width;
            VertPos : Top = Previous . Bottom + 0.5;
                      Line(VMiddle);
            {  Adjust : VMiddle; }
   IN Vue_chinoise
      Visibility: 0;
   IN Vue_fran\10aise
      Visibility: 0;
            IN Vue_vi\13tnamienne
               BEGIN
               Visibility: 2;
               Size: Enclosing - 1;
      Style: Bold;
               VertPos: Top = Enclosing . Top + 1 cm;
               Line(Left);
               END;
            END;

      Auteurs :
            BEGIN
            Width: Enclosing . Width * 80 %;
                      Style: Italics;
            VertPos : Top = Previous . Bottom + 1.5;
   BorizPos : VMiddle = Enclosing . VMiddle;
            Indent: 0;
            Justify : No;
   IN Vue_chinoise
      Visibility: 0;
   IN Vue_fran\10aise
```

```
   Visibility: 0;
IN Vue_vi\13tnamienne
   Visibility: 0;
          END;

   Auteur :
          BEGIN
          Width: Enclosing . Width;
          VertPos : Top = Previous . Bottom + 0.2 cm;
                    Line(VMiddle);
Justify: No;
          {  Adjust : VMiddle; }
          END;

   Mode_Emploi :
          BEGIN
          Width : Enclosing . Width * 87%;
CreateBefore(BoiteInstrFran);
          VertPos : Top = Previous . Bottom + 0.5 cm;
          HorizPos : Right = Enclosing . Right;
Justify: No;
Indent: 0;
IN Vue_chinoise
   Visibility: 0;
IN Vue_fran\10aise
   Visibility: 0;
IN Vue_vi\13tnamienne
   Visibility: 0;
          END;

Paragr_Mode_Emploi :
   BEGIN
   Width: Enclosing . Width;
   VertPos: Top = Previous . Bottom + 0.4 cm;
   Line(Left);
   Justify: Yes;
   END;

   CorpsDict :
          BEGIN
          Indent : 1.5;
          VertPos : Top = Previous . Bottom + 1.5 cm;
          HorizPos : VMiddle = Enclosing . VMiddle;
          IN Vue_chinoise
             BEGIN
             VertPos: Top = Previous . Bottom + 1 cm;
             HorizPos: Left = Enclosing . Left;
             END;
          IN Vue_fran\10aise
             BEGIN
             VertPos: Top = Previous . Bottom + 1 cm;
             HorizPos: Left = Enclosing . Left;
             END;
          IN Vue_vi\13tnamienne
             BEGIN
             VertPos: Top = Previous . Bottom + 1 cm;
             HorizPos: Left = Enclosing . Left;
             END;
          END;

{************************** Une Entree Chinois **************************}
   Entr\37e :
          BEGIN
          Create(NumeroEntr\37e);
          Width: Enclosing . Width;
          VertPos : Top = Previous . Bottom + 0.5 cm;
          HorizPos : Left = Enclosing . Left;
```

```
            NoBreak1 : 2;
            NoBreak2 : 2;
                     END;

{--------------- Graphie   ---------------------}
   Graphie :
            BEGIN
            Width : 0.75 cm;
            VertPos : Top = Enclosing . Top;
            HorizPos: Left = Enclosing . Left + 0.8 cm;
Line(Left);
Indent: 0;
            IN Vue_chinoise
               BEGIN
               VertPos: Top = Enclosing . Top;
               HorizPos: Left = Enclosing . Left + 1 cm;
               END;
            END;

{---------------- Transcription ----------------}
   Transcription :
            BEGIN
            Create (CrochetOuvr);
            CreateLast (CrochetFerm);
            Width : 1.5 cm;
            VertPos : Top = Enclosing . Top;
            HorizPos : Left = Enclosing . Left + 1.65 cm;
Line(Left);
Indent: 0;
            END;

{---------------- Categorie -------------------}
   Cat\37gorie :
            BEGIN
            Style: Italics;
            Width : 1 cm;
            VertPos : Top = Enclosing . Top;
            HorizPos : Left = Enclosing . Left + 3.4 cm;
Line(Left);
Indent: 0;
            END;

{----------------- Utilisation ----------------}
   Utilisation :
            BEGIN
            Style: Italics;
            Width : 1 cm;
            VertPos : Top = Enclosing . Top;
            HorizPos : Left = Enclosing . Left + 4.5 cm;
Line(Left);
Indent: 0;
            END;

{------------------ Sens Francais -------------}
   Equival_fran\10ais :
            BEGIN
            Width : 4 cm;
            VertPos : Top = Enclosing . Top;
            HorizPos : Left = Enclosing . Left + 6.1 cm;
IN Vue_chinoise
   Visibility: 0;
            IN Vue_fran\10aise
   BEGIN
               Width : 6 cm;
   VertPos: Top = Enclosing . Top;
               HorizPos: Left = Enclosing . Left + 7 cm;
   END;
```

```
IN Vue_vi\13tnamienne
   Visibility: 0;
          END;

   Sens_fran\10ais :
          BEGIN
          Width : Enclosing . Width;
          VertPos : Top = Previous . Bottom + 0.3 ;
                    Line(Left);
Indent: 0;
          END;

{--------------- Sens Vietnamien --------------}
   Equival_vi\13tnamien :
          BEGIN
          Width : 4 cm;
          VertPos : Top = Enclosing . Top;
          HorizPos : Left = Enclosing . Left + 10.7 cm;
IN Vue_chinoise
   Visibility: 0;
IN Vue_fran\10aise
   Visibility: 0;
IN Vue_vi\13tnamienne
   BEGIN
   Width : 6 cm;
             VertPos: Top = Enclosing . Top;
             HorizPos: Left = Enclosing . Left + 7 cm;
             END;
          END;

   Sens_vi\13tnamien :
          BEGIN
          Width : Enclosing . Width;
          VertPos : Top = Previous . Bottom + 0.3;
          HorizPos : Right = Enclosing . Right;
          Line(Left);
Indent: 0;
          END;

   TEXTE :
          Width : Enclosed . Width;

ATTRIBUTES

   Langue = Chinois: Font: Times;

   Langue = Fran\10ais: Font: Times;

   Langue = Vi\13tnamien: Font: Times;

   Importance = D\37finition: Style: Italics;

   Importance = Mot_important: Style: Bold;

END { fin schema DictTriLing }
```

```
{*************************************************************************
        Arbre abstrait d'un dictionnaire trilingue      H. Kh. PHAN Mars 1991

        Les alphabets utilises : L = Latin
                                 V = Vietnamien
                                 G = Grec
                                 R = Russe
                                 C = Chinois
        Dans cette version, les caracteres ASCII non imprimables sont
  illisibles dans les elements textuels.
 *************************************************************************}

DictTriLing(DictTriLing) Label50002
  SAUT_PAGE(DictTriLing) Number=  1 View= 1 Begin of element
  Titre(DictTriLing)
    Titre_chinois(DictTriLing)
      TEXTE(DictTriLing) Pres(Size vue1) Lg=13 Alphabet=C
      '::# #-# 7(# #-# T=# KE# 5c|'
    Titre_frabis(DictTriLing)
      TEXTE(DictTriLing) Lg=44 Alphabet=L
      'Dictionnaire Chinois - Frabis - Vietnamien|'
    Titre_vitnamien(DictTriLing)
      TEXTE(DictTriLing) Lg=25 Alphabet=V
      'T in Hn - Php - Vit|'
  Auteurs(DictTriLing)
    Auteur(DictTriLing)
      TEXTE(DictTriLing) Lg=14 Alphabet=L
      'PHAN Huy Khnh|'
  Mode_Emploi(DictTriLing)
    Paragr_Mode_Emploi(DictTriLing) Pres(InterL vue1)
      TEXTE(DictTriLing) Lg=150 Alphabet=L
      'Voici un exemple tr      s simple de dictionnaire trilingue chinois-fra..
      TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=6 Alphabet=L
      'pinyin|'
      TEXTE(DictTriLing) Lg=3 Alphabet=L
      ' ( |'
      TEXTE(DictTriLing) Lg=3 Alphabet=C
      'F4# Rt|'
      TEXTE(DictTriLing) Lg=70 Alphabet=L
      ' ) d'un caract   re chinois. La structure d'une entre est la suivan...'
    Paragr_Mode_Emploi(DictTriLing) Pres(Indent vue1, InterL vue1)
      TEXTE(DictTriLing) Lg=2 Alphabet=L
      '- |'
      TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=6 Alphabet=L
      'numro|'
      TEXTE(DictTriLing) Lg=34 Alphabet=L
      ' (en caract      re gras) de l'entre ;|'
    Paragr_Mode_Emploi(DictTriLing) Pres(Indent vue1, InterL vue1)
      TEXTE(DictTriLing) Lg=2 Alphabet=L
      '- |'
      TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=7 Alphabet=L
      'graphie|'
      TEXTE(DictTriLing) Lg=23 Alphabet=L
      ' du caract       re chinois ;|'
    Paragr_Mode_Emploi(DictTriLing) Pres(Indent vue1, InterL vue1)
      TEXTE(DictTriLing) Lg=2 Alphabet=L
      '- |'
      TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=13 Alphabet=L
      'transcription|'
      TEXTE(DictTriLing) Lg=179 Alphabet=L
```

```
  ' (entre les crochets) : c'est une syllabe suivie d'un ton de 1  4...'
Paragr_Mode_Emploi(DictTriLing) Pres(Indent vuel, InterL vuel)
  TEXTE(DictTriLing) Lg=2 Alphabet=L
  '- |'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=9 Alphabet=L
  'catgorie|'
  TEXTE(DictTriLing) Lg=38 Alphabet=L
  ' (abgre en italique) ; par exemple, |'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=4 Alphabet=L
  'adj.|'
  TEXTE(DictTriLing) Lg=16 Alphabet=L
  ' pour adjectif, |'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=4 Alphabet=L
  'adv.|'
  TEXTE(DictTriLing) Lg=15 Alphabet=L
  ' pour adverbe, |'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=4 Alphabet=L
  'ver.|'
  TEXTE(DictTriLing) Lg=19 Alphabet=L
  ' pour verbe, etc. ;|'
Paragr_Mode_Emploi(DictTriLing) Pres(Indent vuel, InterL vuel)
  TEXTE(DictTriLing) Lg=2 Alphabet=L
  '- |'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=11 Alphabet=L
  'utilisation|'
  TEXTE(DictTriLing) Lg=52 Alphabet=L
  ' (abgre en italique) ; il s'agit d'un mot normal (|'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=4 Alphabet=L
  'nor.|'
  TEXTE(DictTriLing) Lg=13 Alphabet=L
  '), familial (|'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=4 Alphabet=L
  'fam.|'
  TEXTE(DictTriLing) Lg=14 Alphabet=L
  '), argotique (|'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=4 Alphabet=L
  'arg.|'
  TEXTE(DictTriLing) Lg=9 Alphabet=L
  '), etc. ;|'
Paragr_Mode_Emploi(DictTriLing) Pres(Indent vuel, InterL vuel)
  TEXTE(DictTriLing) Lg=2 Alphabet=L
  '- |'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=20 Alphabet=L
  'quivalents franais|'
  TEXTE(DictTriLing) Lg=300 Alphabet=L
  ' avec plusieurs sens diffrents dont chacun est affich sur un par...'
Paragr_Mode_Emploi(DictTriLing) Pres(Indent vuel, InterL vuel)
  TEXTE(DictTriLing) Lg=2 Alphabet=L
  '- |'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=23 Alphabet=L
  'quivalents vietnamiens|'
  TEXTE(DictTriLing) Lg=22 Alphabet=L
  ' : cf. sens franais ;|'
Paragr_Mode_Emploi(DictTriLing) Pres(InterL vuel)
  TEXTE(DictTriLing) Lg=57 Alphabet=L
  'Il y a trois vue diffrentes. La vue principale, appele |'
  TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=18 Alphabet=L
  'Dict_trois_Langues|'
  TEXTE(DictTriLing) Lg=56 Alphabet=L
  ', prsente le contenu du dictionnaire trilingue. La vue |'
```

```
    TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=12 Alphabet=L
    'Vue_chinoise|'
    TEXTE(DictTriLing) Lg=112 Alphabet=L
    ' prsente les entres concernant le chinois (graphie, transcriptio...'
    TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=13 Alphabet=L
    'Vue_fraaise|'
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    ' et |'
    TEXTE(DictTriLing)(ATTR Importance=Dfinition)  Lg=16 Alphabet=L
    'Vue_vietnamienne|'
    TEXTE(DictTriLing) Lg=73 Alphabet=L
    ' sont des dictionnaires bilingues chinois-fraais et chinois-viet...'
CorpsDict(DictTriLing)
  Entre(DictTriLing)
    Graphie(DictTriLing)
      TEXTE(DictTriLing) Lg=1 Alphabet=C
      '2;|'
    Transcription(DictTriLing)
      TEXTE(DictTriLing) Lg=3 Alphabet=L
      'bu4|'
    Catgorie(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'adv.|'
    Utilisation(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'nor.|'
    Equival_fraais(DictTriLing)
      Sens_fraais(DictTriLing)
        TEXTE(DictTriLing) Lg=14 Alphabet=L
        'non ; ne...pas|'
    Equival_vitnamien(DictTriLing)
      Sens_vitnamien(DictTriLing)
        TEXTE(DictTriLing) Lg=11 Alphabet=V
        'khng ; bt|'
  Entre(DictTriLing)
    Graphie(DictTriLing)
      TEXTE(DictTriLing) Lg=1 Alphabet=C
      '2z|'
    Transcription(DictTriLing)
      TEXTE(DictTriLing) Lg=5 Alphabet=L
      'chan3|'
    Catgorie(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'ver.|'
    Utilisation(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'nor.|'
    Equival_fraais(DictTriLing)
      Sens_fraais(DictTriLing)
        TEXTE(DictTriLing) Lg=8 Alphabet=L
        'produire|'
    Equival_vitnamien(DictTriLing)
      Sens_vitnamien(DictTriLing)
        TEXTE(DictTriLing) Lg=10 Alphabet=V
        'sn [xut]|'
  Entre(DictTriLing)
    Graphie(DictTriLing)
      TEXTE(DictTriLing) Lg=1 Alphabet=C
      '4s|'
    Transcription(DictTriLing)
```

```
    TEXTE(DictTriLing) Lg=3 Alphabet=L
    'da4|'
  Catgorie(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'adj.|'
  Utilisation(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'nor.|'
  Equival_frabis(DictTriLing)
    Sens_frabis(DictTriLing)
      TEXTE(DictTriLing) Lg=5 Alphabet=L
      'grand|'
  Equival_vitnamien(DictTriLing)
    Sens_vitnamien(DictTriLing)
      TEXTE(DictTriLing) Lg=14 Alphabet=V
      'ln ; to ; i|'
Entre(DictTriLing)
  Graphie(DictTriLing)
    TEXTE(DictTriLing) Lg=1 Alphabet=C
    '5D|'
  Transcription(DictTriLing)
    TEXTE(DictTriLing) Lg=3 Alphabet=L
    'da4|'
  Catgorie(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'adj.|'
  Utilisation(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'nor.|'
  Equival_frabis(DictTriLing)
    Sens_frabis(DictTriLing)
      TEXTE(DictTriLing) Lg=48 Alphabet=L
      'particule de l'apparte- nance ou du dterminatif|'
  Equival_vitnamien(DictTriLing)
    Sens_vitnamien(DictTriLing)
      TEXTE(DictTriLing) Lg=33 Alphabet=V
      'ca ; quyn s hu ca ; thuc v|'
SAUT_PAGE(DictTriLing) Number=  2 View= 1 Computed
Entre(DictTriLing)
  Graphie(DictTriLing)
    TEXTE(DictTriLing) Lg=1 Alphabet=C
    '5X|'
  Transcription(DictTriLing)
    TEXTE(DictTriLing) Lg=3 Alphabet=L
    'di4|'
  Catgorie(DictTriLing)
    TEXTE(DictTriLing) Lg=3 Alphabet=L
    'nom|'
  Utilisation(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'nor.|'
  Equival_frabis(DictTriLing)
    Sens_frabis(DictTriLing)
      TEXTE(DictTriLing) Lg=8 Alphabet=L
      '1) terre|'
    Sens_frabis(DictTriLing)
      TEXTE(DictTriLing) Lg=20 Alphabet=L
      '2) suffixe d'adverbe|'
  Equival_vitnamien(DictTriLing)
    Sens_vitnamien(DictTriLing)
```

```
      TEXTE(DictTriLing) Lg=9 Alphabet=V
      't ; a|'
Entre(DictTriLing)
  Graphie(DictTriLing)
    TEXTE(DictTriLing) Lg=1 Alphabet=C
    '6/|'
  Transcription(DictTriLing)
    TEXTE(DictTriLing) Lg=5 Alphabet=L
    'dong4|'
  Catgorie(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'ver.|'
  Utilisation(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'nor.|'
  Equival_fraais(DictTriLing)
    Sens_fraais(DictTriLing)
      TEXTE(DictTriLing) Lg=16 Alphabet=L
      'bouger ; toucher|'
  Equival_vitnamien(DictTriLing)
    Sens_vitnamien(DictTriLing)
      TEXTE(DictTriLing) Lg=11 Alphabet=V
      'ng ; chm|'
Entre(DictTriLing)
  Graphie(DictTriLing)
    TEXTE(DictTriLing) Lg=1 Alphabet=C
    '8v|'
  Transcription(DictTriLing)
    TEXTE(DictTriLing) Lg=3 Alphabet=L
    'ge4|'
  Catgorie(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'adj.|'
  Utilisation(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'nor.|'
  Equival_fraais(DictTriLing)
    Sens_fraais(DictTriLing)
      TEXTE(DictTriLing) Lg=12 Alphabet=L
      'spcificatif|'
  Equival_vitnamien(DictTriLing)
    Sens_vitnamien(DictTriLing)
      TEXTE(DictTriLing) Lg=22 Alphabet=V
      ' ch r
```

```
|'
    Entre(DictTriLing)
      Graphie(DictTriLing)
        TEXTE(DictTriLing) Lg=1 Alphabet=C
        '9$|'
      Transcription(DictTriLing)
        TEXTE(DictTriLing) Lg=5 Alphabet=L
        'gong1|'
      Catgorie(DictTriLing)
        TEXTE(DictTriLing) Lg=3 Alphabet=L
        'nom|'
      Utilisation(DictTriLing)
        TEXTE(DictTriLing) Lg=4 Alphabet=L
        'nor.|'
      Equival_fraais(DictTriLing)
        Sens_fraais(DictTriLing)
          TEXTE(DictTriLing) Lg=10 Alphabet=L
          '1) ouvrier|'
        Sens_fraais(DictTriLing)
          TEXTE(DictTriLing) Lg=10 Alphabet=L
          '2) travaux|'
      Equival_vitnamien(DictTriLing)
        Sens_vitnamien(DictTriLing)
          TEXTE(DictTriLing) Lg=28 Alphabet=V
          'cng ; cng vic ; cng nhn|'
    Entre(DictTriLing)
      Graphie(DictTriLing)
        TEXTE(DictTriLing) Lg=1 Alphabet=C
        '9z|'
      Transcription(DictTriLing)
        TEXTE(DictTriLing) Lg=4 Alphabet=L
        'guo2|'
      Catgorie(DictTriLing)
        TEXTE(DictTriLing) Lg=3 Alphabet=L
        'nom|'
      Utilisation(DictTriLing)
        TEXTE(DictTriLing) Lg=4 Alphabet=L
        'nor.|'
      Equival_fraais(DictTriLing)
        Sens_fraais(DictTriLing)
          TEXTE(DictTriLing) Lg=11 Alphabet=L
          'pays ; tat|'
      Equival_vitnamien(DictTriLing)
        Sens_vitnamien(DictTriLing)
          TEXTE(DictTriLing) Lg=15 Alphabet=V
          'nc ; quc gia|'
    Entre(DictTriLing)
      Graphie(DictTriLing)
        TEXTE(DictTriLing) Lg=1 Alphabet=C
        ':M|'
      Transcription(DictTriLing)
        TEXTE(DictTriLing) Lg=3 Alphabet=L
        'he1|'
      Catgorie(DictTriLing)
        TEXTE(DictTriLing) Lg=4 Alphabet=L
        'var.|'
      Utilisation(DictTriLing)
        TEXTE(DictTriLing) Lg=4 Alphabet=L
        'nor.|'
      Equival_fraais(DictTriLing)
```

```
      Sens_fraais(DictTriLing)
        TEXTE(DictTriLing) Lg=7 Alphabet=L
        '1) paix|'
      Sens_fraais(DictTriLing)
        TEXTE(DictTriLing) Lg=12 Alphabet=L
        '2) et ; avec|'
    Equival_vitnamien(DictTriLing)
      Sens_vitnamien(DictTriLing)
        TEXTE(DictTriLing) Lg=15 Alphabet=V
nh|'    'bh ; ha b
  Entre(DictTriLing)
    Graphie(DictTriLing)
      TEXTE(DictTriLing) Lg=1 Alphabet=C
      'CE|'
    Transcription(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'men1|'
    Catgorie(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'adj.|'
    Utilisation(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'nor.|'
    Equival_fraais(DictTriLing)
      Sens_fraais(DictTriLing)
        TEXTE(DictTriLing) Lg=18 Alphabet=L
        'suffixe de pluriel|'
    Equival_vitnamien(DictTriLing)
      Sens_vitnamien(DictTriLing)
        TEXTE(DictTriLing) Lg=11 Alphabet=V
        'cc ; nhng|'
  Entre(DictTriLing)
    Graphie(DictTriLing)
      TEXTE(DictTriLing) Lg=1 Alphabet=C
      'HK|'
    Transcription(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'ren1|'
    Catgorie(DictTriLing)
      TEXTE(DictTriLing) Lg=3 Alphabet=L
      'nom|'
    Utilisation(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'nor.|'
    Equival_fraais(DictTriLing)
      Sens_fraais(DictTriLing)
        TEXTE(DictTriLing) Lg=11 Alphabet=L
        'tre humain|'
    Equival_vitnamien(DictTriLing)
      Sens_vitnamien(DictTriLing)
        TEXTE(DictTriLing) Lg=12 Alphabet=V
        'ngi ; nhn|'
  Entre(DictTriLing)
    Graphie(DictTriLing)
      TEXTE(DictTriLing) Lg=1 Alphabet=C
      'J1|'
    Transcription(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'shi1|'
    Catgorie(DictTriLing)
```

```
    TEXTE(DictTriLing) Lg=3 Alphabet=L
    'nom|'
  Utilisation(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'nor.|'
  Equival_fraais(DictTriLing)
    Sens_fraais(DictTriLing)
      TEXTE(DictTriLing) Lg=13 Alphabet=L
      'temps ; heure|'
  Equival_vitnamien(DictTriLing)
    Sens_vitnamien(DictTriLing)
      TEXTE(DictTriLing) Lg=24 Alphabet=V
      'gi ; thgi ; thi gi|'
Entre(DictTriLing)
  Graphie(DictTriLing)
    TEXTE(DictTriLing) Lg=1 Alphabet=C
    'N*|'
  Transcription(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'wei4|'
  Catgorie(DictTriLing)
    TEXTE(DictTriLing) Lg=5 Alphabet=L
    'prp.|'
  Utilisation(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'nor.|'
  Equival_fraais(DictTriLing)
    Sens_fraais(DictTriLing)
      TEXTE(DictTriLing) Lg=4 Alphabet=L
      'pour|'
  Equival_vitnamien(DictTriLing)
    Sens_vitnamien(DictTriLing)
      TEXTE(DictTriLing) Lg=31 Alphabet=V
      'thay ; thay cho ; i ly ; lm|'
Entre(DictTriLing)
  Graphie(DictTriLing)
    TEXTE(DictTriLing) Lg=1 Alphabet=C
    'R*|'
  Transcription(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'yao4|'
  Catgorie(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'ver.|'
  Utilisation(DictTriLing)
    TEXTE(DictTriLing) Lg=4 Alphabet=L
    'nor.|'
  Equival_fraais(DictTriLing)
    Sens_fraais(DictTriLing)
      TEXTE(DictTriLing) Lg=7 Alphabet=L
      'vouloir|'
  Equival_vitnamien(DictTriLing)
    Sens_vitnamien(DictTriLing)
      TEXTE(DictTriLing) Lg=17 Alphabet=V
      'mun ; ho ; thm|'
Entre(DictTriLing)
  Graphie(DictTriLing)
    TEXTE(DictTriLing) Lg=1 Alphabet=C
    'SP|'
  Transcription(DictTriLing)
```

```
        TEXTE(DictTriLing) Lg=4 Alphabet=L
        'you3|'
      Catgorie(DictTriLing)
        TEXTE(DictTriLing) Lg=4 Alphabet=L
        'ver.|'
      Utilisation(DictTriLing)
        TEXTE(DictTriLing) Lg=4 Alphabet=L
        'nor.|'
      Equival_fraais(DictTriLing)
        Sens_fraais(DictTriLing)
          TEXTE(DictTriLing) Lg=5 Alphabet=L
          'avoir|'
      Equival_vitnamien(DictTriLing)
        Sens_vitnamien(DictTriLing)
          TEXTE(DictTriLing) Lg=31 Alphabet=V
; cm thy }cc ; mua c|'
   Entre(DictTriLing)
     Graphie(DictTriLing)
       TEXTE(DictTriLing) Lg=1 Alphabet=C
       'RT|'
     Transcription(DictTriLing)
       TEXTE(DictTriLing) Lg=3 Alphabet=L
       'yi3|'
     Catgorie(DictTriLing)
       TEXTE(DictTriLing) Lg=5 Alphabet=L
       'prp.|'
     Utilisation(DictTriLing)
       TEXTE(DictTriLing) Lg=4 Alphabet=L
       'nor.|'
     Equival_fraais(DictTriLing)
       Sens_fraais(DictTriLing)
         TEXTE(DictTriLing) Lg=10 Alphabet=L
         'par ; avec|'
     Equival_vitnamien(DictTriLing)
       Sens_vitnamien(DictTriLing)
         TEXTE(DictTriLing) Lg=19 Alphabet=V
         'bi ; vi, cng vi|'
SAUT_PAGE(DictTriLing) Number=  3 View= 1 Computed

**** Scrapbook ****
Empty
```

```
{*************************************************************************
        Image abstraite d'un dictionnaire trilingueH. Kh. PHAN Mars 1991

        Les alphabets utilises : L = Latin
                                 V = Vietnamien
                                 G = Grec
                                 R = Russe
                                 C = Chinois
        Dans cette version, les caracteres ASCII non imprimables sont
  illisibles dans les elements textuels.
 *************************************************************************

1 DictTriLing Vol:1603 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
      VRef: VertRef = Pave1.Left
      HRef: HorizRef = Pave2.HorizRef
      VertPos: Top = Window.Top
      HorizPos: VMiddle = Window.VMiddle
      Width: Window.Width+425pt
      Height: default
      line:N CT:N CQ:Y

2  SAUT_PAGE Vol:1 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:N Pave Pres:N Nature:comp
        VRef: VertRef = Pave2.Left
        HRef: HorizRef = Pave3.HorizRef
        VertPos: Top = Pave1.Top
        HorizPos: Left = Pave1.Left
        Width: default
        Height: default
        line:N CT:N CQ:N

3    SAUT_PAGE Vol:1 View1 Visib:5 Font:t HighL:0 Size:1pt Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:N Secable:N Pave Pres:N Nature:graphics
          alphabet=L'h'
          VRef: PosRef = nil
          HRef: PosRef = nil
          VertPos: PosRef = nil
          HorizPos: PosRef = nil
          Width: 425pt
          Height: 1pt

4  Titre Vol:82 View1 Visib:5 Font:t HighL:0 Size:2 Indent:0 Plane:0 Justif:N
 Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
        VRef: VertRef = Pave4.Left
        HRef: HorizRef = Pave5.HorizRef
        VertPos: Top = Pave1.Top+28 pt
        HorizPos: VMiddle = Pave1.VMiddle
        Width: Pave1.Width*80%
        Height: default
        line:N CT:N CQ:N

5    Titre_chinois Vol:13 View1 Visib:5 Font:t HighL:0 Size:2 Indent:0
 Plane:0 Justif:N Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
          VRef: VertRef = Pave5.Left
          HRef: HorizRef = Pave6.HorizRef
          VertPos: Top = Pave4.Top
          HorizPos: PosRef = nil
          Width: Pave4.Width
          Height: default
```

```
          line:Y align:center

6      TEXTE Vol:13 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
Justif:N Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=C '::# #-# 7(# #-# T=# KE# 5c|'
            VRef: VertRef = Pave6.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
            Height: default

7    Titre_fraais Vol:44 Viewl Visib:5 Font:t HighL:0 Size:2 Indent:0
 Plane:0 Justif:N Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
          VRef: VertRef = Pave7.Left
          HRef: HorizRef = Pave8.HorizRef
          VertPos: Top = Pave5.Bottom+5
          HorizPos: Left = Pave5.Left
          Width: Pave4.Width
          Height: default
          line:Y align:center

8      TEXTE Vol:44 Viewl Visib:5 Font:t HighL:0 Size:2 Indent:0 Plane:0
 Justif:N Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'Dictionnaire Chinois - Fraais - Vietnamien|'
            VRef: VertRef = Pave8.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
            Height: default

9    Titre_vitnamien Vol:25 Viewl Visib:5 Font:t HighL:0 Size:2 Indent:0
 Plane:0 Justif:N Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
          VRef: VertRef = Pave9.Left
          HRef: HorizRef = Pavel0.HorizRef
          VertPos: Top = Pave7.Bottom+5
          HorizPos: Left = Pave7.Left
          Width: Pave4.Width
          Height: default
          line:Y align:center

10      TEXTE Vol:25 Viewl Visib:5 Font:t HighL:0 Size:2 Indent:0 Plane:0
 Justif:N Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=V 'T in Hn - Php - Vit|'
            VRef: VertRef = Pavel0.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
            Height: default

11  Auteurs Vol:14 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:N Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
        VRef: VertRef = Pavell.Left
        HRef: HorizRef = Pavel2.HorizRef
        VertPos: Top = Pave4.Bottom+15
        HorizPos: VMiddle = Pavel.VMiddle
        Width: Pavel.Width*80%
        Height: default
```

```
        line:N CT:N CQ:N

12    Auteur Vol:14 View1 Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:N Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
          VRef: VertRef = Pavel2.Left
          HRef: HorizRef = Pavel3.HorizRef
          VertPos: PosRef = nil
          HorizPos: PosRef = nil
          Width: Pavell.Width
          Height: default
          line:Y align:center

13      TEXTE Vol:14 View1 Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:N Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'PHAN Huy Khnh|'
            VRef: VertRef = Pavel3.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
            Height: default

14  Mode_Emploi.BoiteInstrFran Vol:13 View1 Visib:5 Font:t HighL:1 Size:1
 Indent:0 Plane:0 Justif:Y Linespace:10 Modif:N Secable:Y Pave Pres:Y Nature:tex
        alphabet=L 'MODE D'EMPLOI|'
        VRef: VertRef = Pavel4.Left
        HRef: PosRef = nil
        VertPos: Top = Pavell.Bottom+28 pt
        HorizPos: Right = Pavel5.Right
        Width: Pavel.Width*87%
        Height: default

15  Mode_Emploi Vol:1454 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
       VRef: VertRef = Pavel5.Left
        HRef: HorizRef = Pavel6.HorizRef
        VertPos: Top = Pavel4.Bottom+14 pt
        HorizPos: Right = Pavel.Right
        Width: Pavel.Width*87%
        Height: default
        line:N CT:N CQ:N

16    Paragr_Mode_Emploi Vol:232 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0
 Plane:0 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:comp
          VRef: VertRef = Pavel6.Left
          HRef: HorizRef = Pavel7.HorizRef
          VertPos: PosRef = nil
          HorizPos: PosRef = nil
          Width: Pavel5.Width
          Height: default
          line:Y align:left

17      TEXTE Vol:150 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
 alphabet=L 'Voici un exemple trs simple de dictionnaire trilingue chino...'
            VRef: VertRef = Pavel7.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
```

```
            Height: default

18      TEXTE Vol:6 View1 Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'pinyin|'
            VRef: VertRef = Pave18.Left
            HRef: PosRef = nil
            VertPos: Top = Pave17.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

19      TEXTE Vol:3 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L ' ( |'
            VRef: VertRef = Pave19.Left
            HRef: PosRef = nil
            VertPos: Top = Pave18.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

20      TEXTE Vol:3 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=C 'F4# Rt|'
            VRef: VertRef = Pave20.Left
            HRef: PosRef = nil
            VertPos: Top = Pave19.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

21      TEXTE Vol:70 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
 alphabet=L ' ) d'un caractre chinois. La structure d'une entre est la ...'
            VRef: VertRef = Pave21.Left
            HRef: PosRef = nil
            VertPos: Top = Pave20.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

22    Paragr_Mode_Emploi Vol:42 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10
 Plane:0 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:comp
          VRef: VertRef = Pave22.Left
          HRef: HorizRef = Pave23.HorizRef
          VertPos: Top = Pave16.Bottom+11 pt
          HorizPos: Left = Pave16.Left
          Width: Pave15.Width
          Height: default
          line:Y align:left

23      TEXTE Vol:2 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L '- |'
            VRef: VertRef = Pave23.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
```

```
            Height: default

24      TEXTE Vol:6 View1 Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'numro|'
            VRef: VertRef = Pave24.Left
            HRef: PosRef = nil
            VertPos: Top = Pave23.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

25      TEXTE Vol:34 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L ' (en caractre gras) de l'entre ;|'
            VRef: VertRef = Pave25.Left
            HRef: PosRef = nil
            VertPos: Top = Pave24.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

26    Paragr_Mode_Emploi Vol:32 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10
 Plane:0 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:comp
          VRef: VertRef = Pave26.Left
          HRef: HorizRef = Pave27.HorizRef
          VertPos: Top = Pave22.Bottom+11 pt
          HorizPos: Left = Pave22.Left
          Width: Pave15.Width
          Height: default
          line:Y align:left

27      TEXTE Vol:2 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L '- |'
            VRef: VertRef = Pave27.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
            Height: default

28      TEXTE Vol:7 View1 Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'graphie|'
            VRef: VertRef = Pave28.Left
            HRef: PosRef = nil
            VertPos: Top = Pave27.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

29      TEXTE Vol:23 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L ' du caractre chinois ;|'
            VRef: VertRef = Pave29.Left
            HRef: PosRef = nil
            VertPos: Top = Pave28.Bottom
            HorizPos: PosRef = nil
            Width: default
```

```
            Height: default

30     Paragr_Mode_Emploi Vol:194 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10
 Plane:0 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:comp
            VRef: VertRef = Pave30.Left
            HRef: HorizRef = Pave31.HorizRef
            VertPos: Top = Pave26.Bottom+11 pt
            HorizPos: Left = Pave26.Left
            Width: Pave15.Width
            Height: default
  line:Y align:left

31       TEXTE Vol:2 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L '- |'
            VRef: VertRef = Pave31.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
            Height: default

32       TEXTE Vol:13 View1 Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'transcription|'
            VRef: VertRef = Pave32.Left
            HRef: PosRef = nil
            VertPos: Top = Pave31.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

33       TEXTE Vol:179 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
 alphabet=L ' (entre les crochets) : c'est une syllabe suivie d'un ton de...'
            VRef: VertRef = Pave33.Left
            HRef: PosRef = nil
            VertPos: Top = Pave32.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

34     Paragr_Mode_Emploi Vol:111 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10
 Plane:0 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:comp
            VRef: VertRef = Pave34.Left
            HRef: HorizRef = Pave35.HorizRef
            VertPos: Top = Pave30.Bottom+11 pt
            HorizPos: Left = Pave30.Left
            Width: Pave15.Width
            Height: default
            line:Y align:left

35       TEXTE Vol:2 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L '- |'
            VRef: VertRef = Pave35.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
```

```
            Height: default

36      TEXTE Vol:9 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'catgorie|'
            VRef: VertRef = Pave36.Left
            HRef: PosRef = nil
            VertPos: Top = Pave35.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

37      TEXTE Vol:38 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L ' (abgre en italique) ; par exemple, |'
            VRef: VertRef = Pave37.Left
            HRef: PosRef = nil
            VertPos: Top = Pave36.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

38      TEXTE Vol:4 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'adj.|'
            VRef: VertRef = Pave38.Left
            HRef: PosRef = nil
            VertPos: Top = Pave37.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

39      TEXTE Vol:16 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L ' pour adjectif, |'
            VRef: VertRef = Pave39.Left
            HRef: PosRef = nil
            VertPos: Top = Pave38.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

40      TEXTE Vol:4 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'adv.|'
            VRef: VertRef = Pave40.Left
            HRef: PosRef = nil
            VertPos: Top = Pave39.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

41      TEXTE Vol:15 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L ' pour adverbe, |'
            VRef: VertRef = Pave41.Left
            HRef: PosRef = nil
            VertPos: Top = Pave40.Bottom
            HorizPos: PosRef = nil
            Width: default
```

```
            Height: default

42      TEXTE Vol:4 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'ver.|'
            VRef: VertRef = Pave42.Left
            HRef: PosRef = nil
            VertPos: Top = Pave41.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

43      TEXTE Vol:19 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L ' pour verbe, etc. ;|'
            VRef: VertRef = Pave43.Left
            HRef: PosRef = nil
            VertPos: Top = Pave42.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

44    Paragr_Mode_Emploi Vol:113 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10
 Plane:0 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:comp
          VRef: VertRef = Pave44.Left
          HRef: HorizRef = Pave45.HorizRef
          VertPos: Top = Pave34.Bottom+11 pt
          HorizPos: Left = Pave34.Left
          Width: Pavel5.Width
          Height: default
          line:Y align:left

45      TEXTE Vol:2 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L '- |'
            VRef: VertRef = Pave45.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
            Height: default

46      TEXTE Vol:11 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'utilisation|'
            VRef: VertRef = Pave46.Left
            HRef: PosRef = nil
            VertPos: Top = Pave45.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

47      TEXTE Vol:52 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L ' (abgre en italique) ; il s'agit d'un mot normal (|'
            VRef: VertRef = Pave47.Left
            HRef: PosRef = nil
            VertPos: Top = Pave46.Bottom
            HorizPos: PosRef = nil
            Width: default
```

```
            Height: default

48      TEXTE Vol:4 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'nor.|'
            VRef: VertRef = Pave48.Left
            HRef: PosRef = nil
            VertPos: Top = Pave47.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

49      TEXTE Vol:13 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L '), familial (|'
            VRef: VertRef = Pave49.Left
            HRef: PosRef = nil
            VertPos: Top = Pave48.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

50      TEXTE Vol:4 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'fam.|'
            VRef: VertRef = Pave50.Left
            HRef: PosRef = nil
            VertPos: Top = Pave49.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

51      TEXTE Vol:14 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L '), argotique (|'
            VRef: VertRef = Pave51.Left
            HRef: PosRef = nil
            VertPos: Top = Pave50.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

52      TEXTE Vol:4 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'arg.|'
            VRef: VertRef = Pave52.Left
            HRef: PosRef = nil
            VertPos: Top = Pave51.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

53      TEXTE Vol:9 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L '), etc. ;|'
            VRef: VertRef = Pave53.Left
            HRef: PosRef = nil
            VertPos: Top = Pave52.Bottom
            HorizPos: PosRef = nil
            Width: default
```

```
            Height: default

54     Paragr_Mode_Emploi Vol:322 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10
 Plane:0 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:comp
           VRef: VertRef = Pave54.Left
           HRef: HorizRef = Pave55.HorizRef
           VertPos: Top = Pave44.Bottom+11 pt
           HorizPos: Left = Pave44.Left
           Width: Pave15.Width
           Height: default
           line:Y align:left

55       TEXTE Vol:2 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
             alphabet=L '- |'
             VRef: VertRef = Pave55.Left
             HRef: PosRef = nil
             VertPos: PosRef = nil
             HorizPos: PosRef = nil
             Width: default
             Height: default

56       TEXTE Vol:20 View1 Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
             alphabet=L 'quivalents fran|'
             VRef: VertRef = Pave56.Left
             HRef: PosRef = nil
             VertPos: Top = Pave55.Bottom
             HorizPos: PosRef = nil
             Width: default
             Height: default

57       TEXTE Vol:300 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
 alphabet=L ' avec plusieurs sens diffrents dont chacun est affich sur ...'
             VRef: VertRef = Pave57.Left
             HRef: PosRef = nil
             VertPos: Top = Pave56.Bottom
             HorizPos: PosRef = nil
             Width: default
             Height: default

58     Paragr_Mode_Emploi Vol:47 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10
 Plane:0 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:comp
           VRef: VertRef = Pave58.Left
           HRef: HorizRef = Pave59.HorizRef
           VertPos: Top = Pave54.Bottom+11 pt
           HorizPos: Left = Pave54.Left
           Width: Pave15.Width
           Height: default
          line:Y align:left

59       TEXTE Vol:2 View1 Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
             alphabet=L '- |'
             VRef: VertRef = Pave59.Left
             HRef: PosRef = nil
             VertPos: PosRef = nil
             HorizPos: PosRef = nil
             Width: default
```

```
            Height: default

60      TEXTE Vol:23 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'quivalents vietnamiens|'
            VRef: VertRef = Pave60.Left
            HRef: PosRef = nil
            VertPos: Top = Pave59.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

61      TEXTE Vol:22 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:10 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L ' : cf. sens fraais ;|'
            VRef: VertRef = Pave61.Left
            HRef: PosRef = nil
            VertPos: Top = Pave60.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

62    Paragr_Mode_Emploi Vol:361 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0
 Plane:0 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:comp
          VRef: VertRef = Pave62.Left
          HRef: HorizRef = Pave63.HorizRef
          VertPos: Top = Pave58.Bottom+11 pt
          HorizPos: Left = Pave58.Left
          Width: Pavel5.Width
          Height: default
          line:Y align:left

63      TEXTE Vol:57 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
 alphabet=L 'Il y a trois vue diffrentes. La vue principale, appele |'
            VRef: VertRef = Pave63.Left
            HRef: PosRef = nil
            VertPos: PosRef = nil
            HorizPos: PosRef = nil
            Width: default
            Height: default

64      TEXTE Vol:18 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'Dict_trois_Langues|'
            VRef: VertRef = Pave64.Left
            HRef: PosRef = nil
            VertPos: Top = Pave63.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

65      TEXTE Vol:56 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
 alphabet=L ', prsente le contenu du dictionnaire trilingue. La vue |'
            VRef: VertRef = Pave65.Left
            HRef: PosRef = nil
            VertPos: Top = Pave64.Bottom
            HorizPos: PosRef = nil
            Width: default
```

```
            Height: default

66      TEXTE Vol:12 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'Vue_chinoise|'
            VRef: VertRef = Pave66.Left
            HRef: PosRef = nil
            VertPos: Top = Pave65.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

67      TEXTE Vol:112 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
 alphabet=L ' prsente les entres concernant le chinois (graphie, transc...'
            VRef: VertRef = Pave67.Left
            HRef: PosRef = nil
            VertPos: Top = Pave66.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

68      TEXTE Vol:13 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'Vue_fraaise|'
            VRef: VertRef = Pave68.Left
            HRef: PosRef = nil
            VertPos: Top = Pave67.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

69      TEXTE Vol:4 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L ' et |'
            VRef: VertRef = Pave69.Left
            HRef: PosRef = nil
            VertPos: Top = Pave68.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

70      TEXTE Vol:16 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
            alphabet=L 'Vue_vietnamienne|'
            VRef: VertRef = Pave70.Left
            HRef: PosRef = nil
            VertPos: Top = Pave69.Bottom
            HorizPos: PosRef = nil
            Width: default
            Height: default

71      TEXTE Vol:73 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:12 Modif:Y Secable:Y Pave Pres:N Nature:text
 alphabet=L ' sont des dictionnaires bilingues chinois-fraais et chinoi...'
            VRef: VertRef = Pave71.Left
            HRef: PosRef = nil
            VertPos: Top = Pave70.Bottom
            HorizPos: PosRef = nil
            Width: default
```

```
            Height: default

72  CorpsDict Vol:40 View1 Visib:5 Font:t HighL:0 Size:1 Indent:15 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
        VRef: VertRef = Pave72.Left
        HRef: HorizRef = Pave73.HorizRef
        VertPos: Top = Pave15.Bottom+43 pt
        HorizPos: VMiddle = Pave1.VMiddle
        Width: Pave1.Width
        Height: default
        line:N CT:N CQ:Y

73    Entre Vol:40 View1 Visib:5 Font:t HighL:0 Size:1 Indent:15 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
          VRef: VertRef = Pave73.Left
          HRef: HorizRef = Pave74.HorizRef
          VertPos: PosRef = nil
          HorizPos: Left = Pave72.Left
          Width: Pave72.Width
          Height: default
          line:N CT:N CQ:N

74      Entre.NumeroEntre Vol:1 View1 Visib:5 Font:t HighL:1 Size:1
 Indent:15 Plane:0 Justif:Y Linespace:10 Modif:N Secable:Y Pave Pres:Y
  Nature:text
            alphabet=L '1|'
            VRef: VertRef = Pave74.Left
            HRef: PosRef = nil
            VertPos: Top = Pave73.Top
            HorizPos: PosRef = nil
            Width: Pave73.Width
            Height: default

75      Graphie Vol:1 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
            VRef: VertRef = Pave75.Left
            HRef: HorizRef = Pave76.HorizRef
            VertPos: Top = Pave73.Top
            HorizPos: Left = Pave73.Left+23 pt
            Width: 21pt
            Height: default
            line:Y align:left

76        TEXTE Vol:1 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:text
              alphabet=C '2;|'
              VRef: VertRef = Pave76.Left
              HRef: PosRef = nil
              VertPos: PosRef = nil
              HorizPos: PosRef = nil
              Width: default
              Height: default

77      Transcription Vol:5 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0
 Plane:0 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
            VRef: VertRef = Pave77.Left
            HRef: HorizRef = Pave78.HorizRef
            VertPos: Top = Pave73.Top
            HorizPos: Left = Pave73.Left+47 pt
            Width: 43pt
```

```
            Height: default
            line:Y align:left

78        Transcription.CrochetOuvr Vol:1 View1 Visib:5 Font:t HighL:0 Size:1
 Indent:15 Plane:0 Justif:Y Linespace:10 Modif:N Secable:Y Pave Pres:Y
  Nature:text
                alphabet=L '[|'
                VRef: VertRef = Pave78.Left
                HRef: PosRef = nil
                VertPos: PosRef = nil
                HorizPos: PosRef = nil
                Width: Pave77.Width
                Height: default

79        TEXTE Vol:3 View1 Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:text
                alphabet=L 'bu4|'
                VRef: VertRef = Pave79.Left
                HRef: PosRef = nil
                VertPos: Top = Pave78.Bottom
                HorizPos: PosRef = nil
                Width: default
                Height: default

80        Transcription.CrochetFerm Vol:1 View1 Visib:5 Font:t HighL:0 Size:1
 Indent:15 Plane:0 Justif:Y Linespace:10 Modif:N Secable:Y Pave Pres:Y
  Nature:text
                alphabet=L ']|'
                VRef: VertRef = Pave80.Left
                HRef: PosRef = nil
                VertPos: Top = Pave79.Bottom
                HorizPos: PosRef = nil
                Width: Pave77.Width
                Height: default

81      Catgorie Vol:4 View1 Visib:5 Font:t HighL:2 Size:1 Indent:0
 Plane:0 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
              VRef: VertRef = Pave81.Left
              HRef: HorizRef = Pave82.HorizRef
              VertPos: Top = Pave73.Top
              HorizPos: Left = Pave73.Left+96 pt
              Width: 28pt
              Height: default
              line:Y align:left

82        TEXTE Vol:4 View1 Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:text
                alphabet=L 'adv.|'
                VRef: VertRef = Pave82.Left
                HRef: PosRef = nil
                VertPos: PosRef = nil
                HorizPos: PosRef = nil
                Width: default
                Height: default

83      Utilisation Vol:4 View1 Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
              VRef: VertRef = Pave83.Left
              HRef: HorizRef = Pave84.HorizRef
              VertPos: Top = Pave73.Top
```

```
            HorizPos: Left = Pave73.Left+128 pt
            Width: 28pt
            Height: default
            line:Y align:left

84        TEXTE Vol:4 Viewl Visib:5 Font:t HighL:2 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:text
              alphabet=L 'nor.|'
              VRef: VertRef = Pave84.Left
              HRef: PosRef = nil
              VertPos: PosRef = nil
              HorizPos: PosRef = nil
              Width: default
              Height: default

85      Equival_franis Vol:14 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:15
 Plane:0 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
            VRef: VertRef = Pave85.Left
            HRef: HorizRef = Pave86.HorizRef
            VertPos: Top = Pave73.Top
            HorizPos: Left = Pave73.Left+173 pt
            Width: 113pt
            Height: default
            line:N CT:N CQ:N

86        Sens_franis Vol:14 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0
 Plane:0 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
              VRef: VertRef = Pave86.Left
              HRef: HorizRef = Pave87.HorizRef
              VertPos: PosRef = nil
              HorizPos: PosRef = nil
              Width: Pave85.Width
              Height: default
              line:Y align:left

87          TEXTE Vol:14 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:text
                alphabet=L 'non ; ne...pas|'
                VRef: VertRef = Pave87.Left
                HRef: PosRef = nil
                VertPos: PosRef = nil
                HorizPos: PosRef = nil
                Width: default
                Height: default

88      Equival_vitnamien Vol:11 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:15
 Plane:0 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
            VRef: VertRef = Pave88.Left
            HRef: HorizRef = Pave89.HorizRef
            VertPos: Top = Pave73.Top
            HorizPos: Left = Pave73.Left+303 pt
            Width: 113pt
            Height: default
            line:N CT:N CQ:N

89        Sens_vitnamien Vol:11 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0
 Plane:0 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:comp
              VRef: VertRef = Pave89.Left
              HRef: HorizRef = Pave90.HorizRef
              VertPos: PosRef = nil
```

```
              HorizPos: Right = Pave88.Right
              Width: Pave88.Width
              Height: default
              line:Y align:left

90          TEXTE Vol:11 Viewl Visib:5 Font:t HighL:0 Size:1 Indent:0 Plane:0
 Justif:Y Linespace:10 Modif:Y Secable:Y Pave Pres:N Nature:text
                alphabet=V 'khng ; bt|'
                VRef: VertRef = Pave90.Left
                HRef: PosRef = nil
                VertPos: PosRef = nil
                HorizPos: PosRef = nil
                Width: default
                Height: default
```

# Références bibliographiques

## I INFORMATIQUE ET INTELLIGENCE ARTIFICIELLE


[Boullier 84] P. Boullier. *Contribution à la construction automatique d'analyseurs lexicographiques et syntaxiques.* Thèse de Doctorat d'État ès- Sciences Mathématiques, Université d'Orléans, janvier 1984, 449 p.

[Coutaz 86] J. Coutaz. *La construction d'Interfaces Homme-Machine.* Rapport IMAG RR 635-I, novembre 1986, 48 p.

[Coutaz 88] J. Coutaz. *Interface Homme-Ordinateur : Conception et Réalisation.* Thèse de Doctorat d'État, Université Joseph Fourier (Grenoble 1), décembre 1987, 402 p.

[Lenat 84] D. Lenat. *Les logiciels et l'intelligence artificielle.* Pour la Science, novembre 1984, 132-143.


## II LINGUISTIQUE ET INFORMATIQUE MULTILINGUE


[Alleton 76] V. Alleton. *L'écriture chinoise.* "Que sais-je", Presses Universitaires de France, 1976, 128 p.

[Boitet 85] Ch. Boitet. *Traduction (Assistée) par Ordinateur : ingéniérie logicielle et linguicielle.* Conf. AFCET RF & IA, Grenoble, nov. 1985, 15-75.

[Boris 89a] M. Boris. *Recentrage du marché mondial de la traduction. Vers la bureautique multilingue.* DECISION n° 246 - 9, octobre 1989, 66-68.

[Boris 89b] M. Boris. *Du traitement de texte multilingue au système de traduction. Le PC polyglotte.* DECISION n° 246 - 9, octobre 1989, 70-74.

[Cihai 83] Cihai (辞 海). *Large Dictionary of Chinese Characters and Words.* 4th edition, Ci Shu, Shanghai, 1983.

[Coulon et Kayser 86] D. Coulon, D. Kayser. *Informatique et langue naturelle : Présentation générale des méthodes d'interprétation des textes écrits.* T.S.I., 5/2, 1986, 103-128.

[Cousquer 88a] A. et E. Cousquer. *Informatique et Étude du Vocabulaire Scientifique Chinois.* C.L.A.O., vol. XVII n° 1, juin 1988, 129-145.

[Cousquer 88b] A. et E. Cousquer. *Informatisation du chinois, l'écriture chinoise.* AFCET/INTER-FACES, n° 64, février 1988, 5-14.

[Cousquer 89a] A. et E. Cousquer et H. Zhang. *Langue et Mathématiques.* Rapport de recherche, GEDIS, LIFL, 1989, 143 p.

[Cousquer et Zhang 89b] E. Cousquer et H. Zhang. *Linguistique et langue chinoise.* Rapport de recherche, GEDIS, LIFL, 1989, 104 p.



Imprimé par Grif le 20/3/91

[Fafiotte 90] G. Fafiotte. *Apprentissage assisté par ordinateur des caractères chinois*. Rapport de recherche, GETA, IMAG, juin 1990, 10 p.

[Galinski 84] Ch. Galinski. *Japanese and Chinese terminologies in European terminological data banks*. Wien: Infoterm, 1984 (Infoterm 1-84), 26 p.

[Galinski 88] Ch. Galinski. *The Chinese script and its impact on the development of specialized languages and terminology in East Asia*. Wien: Infoterm, 1988 (Infoterm 1-88), 27 p.

[Kantor 81] P. Kantor. *Le chinois sans peine*. ASSIMIL, France 1981, 124 p.

[Lê & Nguyễn 89] Lê Kh. K. & Nguyễn L. *Từ điển Việt-Pháp (Dictionnaire Vietnamien-Français)*. Éditions des Sciences Sociales, Hànội, 1989, 1134 p.

[Malherbe 83] M. Malherbe. *Les langues de l'humanité (Une encyclopédie des 3000 langues dans le monde)*. Éditions Seghers, Paris, 1983, 443 p.

[Moreaux 86] M. A. Moreaux. *Reconnaissance des formes en linguistique. Exemple d'application en analyse et génération du français*. CERTAL, INALCO, Colloque Langues et Informatique, Paris, 1986, 11-21.

[Sabah 88] G. Sabah. *Le traitement automatique du langage écrit*. Séminaire "Langue et Informatique. Un jeu culturel, scientifique et industriel", Paris, 12-13 décembre 1988, 73-84.

[Streeter 72] M. L. Sreeter. *DOC, 1971: A Chinese Dialect Dictionary on Computer*. Computer and Humanities, vol. 6/5, May 1972, 259-270.

[Vauquois et Boitet 84] B. Vauquois et Ch. Boitet. *Études sur les travaux en TAO portant sur des langues asiatiques (japonais, chinois, malais, thaï) et étude des problèmes liés au traitement de textes utilisant de grands jeux de caractères (idéogrammes)*. R. R. nº 83/739, Ass. Champollion/ADI, juin 1984, 81 p.

[Wang & al. 78] Wang H. (王) & al. *Dictionnaire Français-Chinois* (法汉词典). Édition Shanghai, 1978, 1494 p.

[Winograd 84] T. Winograd. *Les logiciels de traitement des langues naturelles*. Pour la Science, novembre 1984, 90-103.

## III PRODUCTION DE DOCUMENTS

### III.1 Codage multilingue

[Anderson 84] L. B. Anderson. *Multilingual Text Processing in a Two-Byte Code*. Proceedings of COLING-84, 2-6 July/1984, Stanford University, California, 1-4.

[Becker 84] J. D. Becker. *Le traitement de Texte Multilingue*. Pour la Science, septembre 1984, 66-76.

Imprimé par Grif le 20/3/91

[Boitet 83] Ch. Boitet. *Conventions de transcription pour la saisie et pour la révision des textes sous ARIANE-78.* Rapport DRET n° 64, GETA, Grenoble, décembre 1983, 15 p.

[Boitet et al. 86a] Ch. Boitet, D. Bachut, R. Gerber. *Ariane portable : Dossier d'Analyse Grands Caractères.* Version 2.0, PN-TAO & GETA, Grenoble, mai 1986, 46 p.

[Boitet et al. 86b] Ch. Boitet, D. Bachut, N. Verastegui, R. Gerber. *Ariane portable : Dossier de Spécification Externe Niveau général.* Version 2.1, PN-TAO & GETA, Grenoble, mars 1986, 33 p.

[Boitet et Tchéou 90] Ch. Boitet et F. X. Tchéou. *On a phonetic and structural encoding of Chinese characters in Chinese texts.* Proceedings of ROCLING III. R.O. C. Computational Linguistics Conference III (1990), 71-80.

[Cousquer 89] A. Cousquer. *Un codage du chinois.* Rapport de recherche, GEDIS, LIFL, Université de Lille 1, avril 1989, 3 p.

[GB81] GB (GuóBiao 国 标). *Code of Chinese graphic Character set for Information Interchange. Primary set GB 2312-80.* Fuxing-menwai Sanlihe, Beijing, China, 1981, 175 p.

[ISO 85] ISO/DID 639.2. *Code for the representation of names of languages.* Draft International Standard ISO/DID 639.2, submitted on 1985-09-19, International Organization for Standardization, 1985, 19 p.

[Kim & al. 88] Kim K., G. G. Belford & B. W. Kong. *A new Proposal for a Standard Hangul (or Korean Script) Code : How to Accomodate Both Databases and Word Processing.* Report n° UIUCDCS-R-88-1447, Department of Computer Science, University of Illinois at Urbana-Champaign, August 1988, 49 p.

[Kleinberg 87] D. Kleinberg. *KanjiTalk1.1 Usage note.* Apple Technical Publications Mail Stop 22-K, January 1987, 32 p.

[Lepage 86] Y. Lepage. *A language for transcriptions.* 11 th International Conference on Computational Linguistics. Proceedings, COLING'86 Bonn, August 1986, 402-404.

[Phan 88] Phan H. Kh. *CARIX : Tous les caractères pour toutes les langues.* Rapport de DEA, GETA, IMAG, juin 1988, 60 p.

[Qiao 90] J. Qiao, Y. Qiao & S. Qiao. *Six-Digit Coding Method.* Communication of the ACM, 33/5, May 1990, 491-494.

[Tayli & Al-Salamath 90] M. Tayli & A. Al-Salamath. *Building Bilingual Microcomputer Systems.* Communication of the ACM, 33/5, May 1990, 495-504.

[Vauquois et Robert 87] Ph. Vauquois, S. Robert. *Base lexicale Français-Anglais-Japonais : Adaptation de Prolog-CRISS aux Grands Jeux de Caractères.* Association Champollion R. I. DGT n° 20, GETA, IMAG, février 1987, 1-23.

Imprimé par Grif le 20/3/91

## III.2 Edition structurale

[André & al. 89] J. André, R. Furuta & V. Quint. *Structured Documents*. Cambridge University Press, 1989, 220 p.

[Girard 89] B. Girard. *La production de documents techniques assistée par ordinateur*. Hermes, Paris, 1989, 64 p.

[Miyabe & al. 90] Y. Miyabe, H. Ohta & K. Tsuga. *Structured Document Preparation System*. TUGboat, 11/3, 1990, 353-358.

[Quint 83] V. Quint. *Un système interactif pour la production des documents mathématiques*. T.S.I., 2/3, 1983, 179-190.

[Quint et Vatton 86a] V. Quint. I. Vatton. *Grif : An Interactive System for Structured Document Manipulation*. Text Processing and Document Manipulation, Processing of the International Conference, J. C. van Vliet ed., Cambridge University Press, 1986, 200-213.

[Quint et al. 86b] V. Quint, I. Vatton et H. Bedor. *Le système Grif*. T.S.I., 5/4, 1986, 337-341.

[Quint 87a] V. Quint. *Les systèmes pour la manipulation de documents structurés*. Cours INRIA Structure de/for Documents, INRIA, janvier 1987, 39-80.

[Quint 87b] V. Quint. *Une approche de l'édition structurée des documents*. Thèse de Doctorat d'État ès-Science Mathématiques, Université Joseph Fourier (Grenoble 1), mai 1987, 283 p.

[Quint & al. 90] V. Quint, M. Nanard & J. André. *Towards document engineering*. Rapport de recherche, n° 1244, INRIA, juin 1990, 17 p.

[Sperberg 90] C. M. Sperberg-McQueen & Lou Burnard. *ACH - ACL - ALLC Guidelines For the Encoding and Interchange of Machine-Readable Texts*. TEI P1, Draft Version 1.0, Chicago and Oxford, July 1990, 279 p.

[Vatton 87] I. Vatton. *L'interface Utilisateur du Système de Manipulation de Documents de Grif*. RR 661-I-, LGI, IMAG, mai 1987, 21 p.

## III.3 Polices et caractères

[André et Mlouka 1987] J. André, M. Mlouka. *Workshop on Font Design Systems*. INRIA-Sophia Antipolis, 18-19 May, 1987, 126.

[André 89] J. André. *Métriques de fontes en typographie traditionnelle*. Cahiers GUTenberg, n° 4, décembre 1989, 9-22.

[Dardailler 89a] D. Dardailler. *Normes et fontes*. Cahiers GUTenberg, n° 4, décembre 1989, 2-8.

[Dardailler 89b] D. Dardailler. *Contribution au service des polices de caractères dans le système de fenêtres X.* Thèse de doctorat nouveau régime, Sophia-Antipolis, décembre 1989, 2-8.

[Knuth 86] D. E. Knuth. *The Metafont book.* Addison Wesley Publishing Company, Reading, Massachusetts, 1986, 361 p.

## III.4 Saisie multilingue

[Amara & al. 89] F. Amara, A. Boualem, M. Mlouka, J.C. Franjaud & C. Lin. *TheChinese and Japanese Characters Input.* ISCIS IV, Çesme, Izmir, Turquie, Octobre 1989, 5p.

[Archer & al. 88] N. P. Archer, M. W. Chan, S. J. Huang & R. T. Liu. *A Chinese-English MicroComputer System.* Communications of the ACM, 31/8, August 1988, 977-892.

[Asia 89] Asia Communications Inc. *TIANMA II* (天马第二代) *user's manual.* Asia Communications Inc., Montreal, Qc, Canada 1989, 39 p.

[Becker 84] J. D. Becker. *Arabic Word Processing.* Communications of the ACM, 30/7, July 1987, 600-610.

[Boualem et al. 89] A. Boualem, C. Lin et M. MLouka. *Une approche d'aide à la saisie des caractères chinois.* IASTED International Symposium, Grindelwald, Suisse, février 1989, 4p.

[Cousquer 90] A. Cousquer. *En chinois dans le TEXte.* Cahiers GUTenberg, n° 6, juillet 1990, 15-24.

[Ferguson 86] M. J. Ferguson. *MultilingualTEX* . TEX for Documentation, Second European Conference. Strasbourg, June 1986, 19-21.

[Haralambous 89] Y. Haralambous. *TEX and latin alphabet languages.* TUGboat, 10/3, 1989, 342-345.

[IEEE 85] Contributions by Y. Chu, J. K. Huang, J. D. Becker, R. Matsuda, H. Makino & W. Cui. *Special Issue on Chinese/Kanji Text and data Processing.* IEEE Trans. on Computers, January 1985.

[Mohr 82] R. Mohr. *A handwriting Input for Chinese Characters.* Rapport de recherche, n° 82-R-001, Centre de Recherche en Informatique de Nancy, 1982, 11 p.

[Phan 90] Phan H. Kh. *Édition structurale des documents multilingues et application au vietnamien dans Grif.* Rapport de recherche n° 186, GEDIS, LIFL, avril 1988, 18 p.

[Tao 66] Tao H. C. *A Chinese Computer Alphabet For Automatic Machine Processing.* University of North Carolina, Chapel Hill, 1966, 18 p.

[Phan 91] Phan H. Kh. *Langage de transcription et adaptation de Grif à la langue chinoise.* Rapport de recherche n° xxx, GEDIS, LIFL, avril 1991, 26 p.

[Vogel 89] B. E. Vogel. *Printing Vietnamese characters by adding diacritical marks via TEX .* TUGboat, 10/2, 1989, 217-223.

[WinSoft 90] WinSoft. *WinKit Chinois. Version 5.1/W1. Hanzi. Mode d'emploi.* WinSoft, Grenoble, mars 1990, 16 p.

[Yhap 75] E. F. Yhap. *Keybord Method for Composing Chinese characters.* IBM Journal of Research & Development, January 1975, 60-70.

[Zhang et Cousquer 88] Zhang H. (张) et E. Cousquer. *Wubizixing (五笔字型), une méthode de saisie des caractères chinois.* Rapport de recherche n° 130, GEDIS, LIFL, avril 1988, 53 p.

## III.5 Traitement de textes et de documents

[André 82] J. André. *Bibliographie analytique sur les manipulations de textes.* T.S.I., 1/5, 1982, 445-455.

[Knuth 86] D. E. Knuth. *The TEXbook.* Addison Wesley Publishing Company, reading, Massachusetts, 1986, 483 p.

[Roy 84] Y. Roy. *TEX/W EB et le Traitement de Textes Mathématiques.* Masson, Paris, 1984, 102.

[SM 90] Contributions de F. Mireaux, J. M. Pinon, N. Belkhiter, P. Amar, I. Filotti, J. Désarménien, M. Kretz, Y. Roy, R. Strosser. *SM-90 Une Architecture Modulaire et ses Application (Actes des Journées SM-90 - 4, 5, 6 déc. 85, Versailles).* Eyrolles, Paris, 1986, 815-844.

[Seroul 89] R. Seroul. *Le petit Livre de TEX .* InterÉditions, Paris, 1989, 317 p.

[Williams 84] G. Williams. *The Apple Macintosh Computer.* Byte, February 1984, 30-54.

[WinSoft 88] WinSoft. *WinText, le traitement de textes multilingue pour Macintosh.* Version 2.0, WinSoft, Grenoble, septembre 1988, 392 p.

## III.6 Typographie

[Becker & Berry 89] Z. Becker & D. Berry. *triroff, an adaptation of the device- independent troff for formating tri-directional text.* Electronic Publishing, vol. 2(3), October 1989, 119-142.

[Buckle 86] N. Buckle. *Word Hyphenation in French.* Département d'informatique, Université du Québec à Hull, 1986, 108-113.

[Désarménien 86] J. Désarménien. *La division par ordinateur des mots français : application à TEX*. T.S.I., 5/4, 1986, 251-265.

[Johnson & Beach 88] J. Johnson & R. J. Beach. *Styles in Document Editing Systems*. IEEE Trans. on Computers, January 198, 32-43.

## IV PROGRAMMATION

[Adobe 85] Adobe Systems Incorporated. *PostScript Language Reference Manual*. Addition-Wesley, Reading, Massachusetts, 1985, 319 p.

[Adobe 87] Adobe Systems Incorporated. *PostScript par exemple. Outils pour l'édition électronique*. Texte français d'Alain Kadé. InterÉditions, Paris, 1987, 259 p.

[Borghi 90] B. Borghi. *Les fichiers cuisine d'Onc'PostScript. Fichier n° 3 : écrire en français avec PostScript*. Cahiers GUTenberg n° 6, juillet 1990, 60-62.

[Bourne 85] S. Bourne. *Le système UNIX*. InterÉditions, Paris, 1985, 398 p.

[Jones 89] O. Jones. *Introduction to the X Window System*. Prentice Hall, Inc., Englewood Cliffs, NewJersey, 1989, 511 p.

[Kernighan & Pike 86] B. W. Kernighan & R. Pike. *L'environnement de programmation UNIX*. InterÉditions, Paris, 1986, 372 p.

[Kernighan & Ritchie 84] B. W. Kernighan & D. M. Ritchie. *Le langage C*. Masson, Paris 1984, 218 p.

[Meyrowitz & van Dam 82] N. Meyrowitz & A. van Dam. *Interactive Editing Systems: Part II*. Computing Surveys, 14/3, September 1982, 366-368.

[Nebut 80] J. L. Nebut. *théorie et pratique du langage PASCAL*. Éditions TECHNIP, Paris, 1980, 273 p.

[O'Reilly & al. 88] T. O'Reilly, V. Quercia, and L. Lamb. *X Window System. User's Guide. Volume 1 & 2 & 3*. O'Reilly & Associates, Inc., 1988.

[Rifflet 86] JM. Rifflet. *La programmation sous UNIX*. McGraw-Hill, Paris 1986, 342 p.

[Sun 86] Sun Microsystems, Inc. *SunView™ Programmer's Guide*. Sun Micro-systems Inc., 2250 Garcia Avenue, Mountain View, CA, September 1986.

[Young 90] Douglas A. Young. The X Window System Programming and Applications with Xt. OSF/Motif Edition. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1990, 531 p.

Imprimé par Grif le 20/3/91

## 题目：

对多种语言计算科学的贡献．结构文件编辑程序的扩展．

## 摘要：

多种语言文件计算机处理有四个主要方面：多种语言的编码，输入装置上文本的采集，输出装置上文本的重建，以及对话．

多语言文件学科领域还有很多有待解决的问题．目前已有的工具只是解决部分问题，不太令人满意．本论文利用Grif（一种对话式的结构文件的生成系统），对多种语言文件处理系统在概念和实现方面提供了一些可能的解决方案．

第一步，采用扩展的ASCII编码和"键组成"方法，制成了带有区别符号的特征采集，实现了Grif对越语的扩展，以处理256个活版印刷符号以内的书写系统．

第二步，提出了一些更广泛的解决方法，以解决输入转换语言E的研制，该语言与Grif的其他语言类似．E语言的"语言描述"这一扩充完善了Grif文件模型．

这一工作将真正多种语言文件的处理的Grif进行了具体化，该文件可容纳汉字，运用拉丁字母书写的语言（包括越语），以及运用希腊字母和希里尔字母书写的语言．

**Đề tài :**

Đóng góp cho ứng dụng tin học trên nhiều ngôn ngữ. Mở rộng một hệ thống xử lý văn bản có tính cấu trúc.

**Tóm tắt :**

Dùng tin học để xử lý các văn bản viết được đồng thời trên nhiều ngôn ngữ liên quan đến bốn phương diện chủ yếu : sự mã hóa các văn bản nhiều ngôn ngữ trong máy, sự tiếp nhận trên một thiết bị vào, sự khôi phục trên một thiết bị ra, và sự hội thoại.

Nhiều vấn đề đặt ra bởi ứng dụng tin học trên nhiều ngôn ngữ vẫn còn chưa được giải quyết. Cho đến nay, những hệ thống xử lý văn bản thuộc loại này chỉ đưa ra được những lời giải mang tính cục bộ và chưa đáp ứng được những nhu cầu đòi hỏi. Luận án này trình bày những giải pháp thích hợp cho phép thiết kế và thực thi một hệ thống xử lý các văn bản viết được đồng thời trên nhiều ngôn ngữ trên cơ sở nới rộng Grif, một hệ thống xử lý các văn bản có tính cấu trúc.

Trong giai đoạn đầu, chúng tôi đã sử dụng bộ mã ASCII mở rộng (8 bits) và phương pháp "tổ hợp phím" để tiếp nhận những chữ cái có dấu, và như vậy, chúng tôi đã nới rộng được Grif cho tiếng Việt. Sự nới rộng này cho phép xử lý các văn bản viết được trên một số ngôn ngữ có bộ chữ cái nhỏ hơn 256 ký tự.

Trong giai đoạn hai, chúng tôi đã đề nghị nhiều lời giải tổng quát hơn để đi đến một giải pháp thích ứng, đó là sự định nghĩa một ngôn ngữ chuyển đổi mã trên các ký tự đưa vào từ bàn phím, gọi là ngôn ngữ E, tương tự như các ngôn ngữ đã có của Grif. Như thế, chúng tôi đã làm đầy đủ mô hình văn bản của Grif nhờ một sự "mô tả ngôn ngữ" viết trên ngôn ngữ E.

Cụ thể, chúng tôi đã xây dựng được một hệ thống xử lý văn bản trên Grif thực sự xử lý được nhiều ngôn ngữ, cho phép trong cùng một văn bản chứa đồng thời chữ Hán, các ngôn ngữ dùng chữ Latin (trong đó có tiếng Việt), chữ Hy-lạp và cả chữ Nga.

**Title:**

Contribution to multilingual text processing. Extension of an editor for structured documents.

**Abstract:**

There are four basic aspects in the area of multilingual document processing: storage of multilingual text, data acquisition from the keyboard, restitution through an output device, and dialogue.

Many problems posed by multilingual document processing are still open. Presently, the available tools solve the problemes only partially and unsatisfactory. The present thesis proposes possible solutions for the design and realization of a Multilingual Text Processing System (MTPS), starting from Grif, an interactive system for the manipulation of structured documents.

In the first step of this work, we have used the extended ASCII codes and the "typesetting of keys" method for acquiring the diacritic characters from the keyboard. The resulting extension of Grif for Vietnamese, can be directly adapted to handle any writing systems which uses less than 256 typographic symbols.

In the second step, we proposes more generic solutions and develop a language of transcription, called E language, which is analogous to the other languages of Grif. This extension makes Grif document models more complete, by providing a way to write (in E) a " linguistic description" of a class of documents.

The resulting implementation is a truly multilingual version of Grif, in which the same document can contain, Chinese, Greek, Russian, Vietnamese and European languages at the same time.

**Keywords:**

Document modelling, structured documents manipulation, multilingual text processing, writing systems, transcription, multilingualism.

**Titre :**

Contribution à l'informatique multilingue. Extension d'un éditeur de documents structurés.

**Résumé :**

Le traitement informatique de documents multilingues concerne quatre aspects essentiels : le codage d'un texte multilingue en mémoire, la saisie sur un dispositif d'entrée, la restitution sur un dispositif de sortie, et le dialogue.

De nombreux problèmes posés par le multilinguisme sont encore ouverts. Les outils disponibles actuellement ne donnent que des solutions partielles et peu satisfaisantes. Cette thèse présente des solutions possibles pour la conception et la réalisation d'un Système de Traitement de Texte Multilingue (STTM) à partir de Grif, un système interactif de production de documents structurés.

Dans la première étape du travail, on a utilisé les codes ASCII étendus et la méthode de "composition des touches" pour la saisie des caractères portant des signes diacritiques, et on a réalisé une extension de Grif au vietnamien, qui permet de traiter des systèmes d'écriture utilisant un ensemble de moins de 256 signes typographiques.

Dans une seconde étape, des solutions beaucoup plus générales ont été proposées pour arriver à une solution plus générique par le développement d'un langage de transcription d'entrée, appelé langage E, analogue aux autres langages de Grif. Grâce à cette extension, on complète le modèle de document de Grif par une "description linguistique", écrite en langage E.

Ce travail s'est concrétisé par la réalisation d'une version véritablement multilingue de Grif, dans laquelle un même document peut contenir à la fois du chinois, des langues utilisant les caractères latins (y compris le vietnamien), et des langues utilisant les alphabets grec et cyrillique.

**Mots-clés :**

Modèle de document, édition de documents structurés, traitement de texte multilingue, systèmes d'écriture, transcription, multilinguisme.