Etude et réalisation d'un système permettant la construction de réseaux d'automates d'état [sic] finis : application à la production de documents en braille abrégé

Blaise Mathieu

THESE

présentée à

et à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR INGENIEUR

par

Blaise MATHIEU

ETUDE ET REALISATION D'UN SYSTEME PERMETTANT LA CONSTRUCTION DE RESEAUX D'AUTOMATES D'ETAT FINIS.

Application à la production de documents en braille abrégé.

Thèse soutenue le 26 septembre 1980 devant la commission d'examen

G. VEILLON	Président
J. COURTIN V. JOLOBOFF	Examinateurs

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1979-1980

Président : M. Philippe TRAYNARD

Vice-Présidents : M. Georges LESPINARD
M. René PAUTHENET

PROFESSEURS DES UNIVERSITES

MM.	ANCEAU François	Informatique fondamentale et appliquée
	BENOIT Jean	Radioéléctricité
	BESSON Jean	Chimie Minérale
	BLIMAN Samuel	Electronique
	BLOCH Daniel	Physique du Solide - Cristallographie
	BOIS Philippe	Mécanique
	BONNETAIN Lucien	Génie Chimique
	BONNIER Etienne	Métallurgie
	BOUVARD Maurice	Génie Mécanique
	BRISSONNEAU Pierre	Physique des Matériaux
	BUYLE-BODIN Maurice	Electronique
	CHARTIER Germain	Electronique
	CHERADAME Hervé	Chimie Physique Macromoléculaires
Mme	CHERUY Arlette	Automatique
MM.	CHIAVERINA Jean	Biologie, Biochimie, Agronomie
	COHEN Joseph	Electronique
	COUMES André	Electronique
	DURAND Francis	Métallurgie
	DURAND Jean-Louis	Physique Nucléaire et Corpusculaire
	FELICI Noël	Electrotechnique
	FOULARD Claude	Automatique
	GUYOT Pierre	Métallurgie Physique
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du Solide - Cristallographie
	LACOUME Jean-Louis	Géographie - Traitement du Signal
	LANCIA Roland	Electronique - Automatique
	LESIEUR Marcel	Mécanique
	LESPINARD Georges	Mécanique
	LONGEQUEUE Jean-Pierre	Physique Nucléaire Corpusculaire
	MOREAU René	Mécanique
	MORET Roger	Physique Nucléaire Corpusculaire
	PARIAUD Jean-Charles	Chimie - Physique
	PAUTHENET René	Physique du Solide - Cristallographie
	PERRET René	Automatique

.../...

MM.	PERRET Robert	Electrotechnique
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	POLOUJADOFF Michel	Electrotechnique
	POUPOT Christian	Electronique - Automatique
	RAMEAU Jean-Jacques	Chimie
	ROBERT André	Chimie Appliquée et des matériaux
	ROBERT François	Analyse numérique
	SABONNADIERE Jean-Claude	Electrotechnique
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée
M.	SOHM Jean-Claude	Chimie - Physique
Mme	SCHLENKER Claire	Physique du Solide - Cristallographie
MM.	TRAYNARD Philippe	Chimie - Physique
	VEILLON Gérard	Informatique fondamentale et appliquée
	ZADWORNY François	Electronique

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

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahïm	Maître de Recherche
	BRONOEL Guy	Maître de Recherche
	CARRE René	Maître de Recherche
	DAVID René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	KAMARINOS Georges	Maître de Recherche
	KLEITZ Michel	Maître de Recherche
	LANDAU Ioan-Doré	Maître de Recherche
	MERMET Jean	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique)

E.N.S.E.E.G.

MM.	ALLIBERT Michel	
	BERNARD Claude	
	CAILLET Marcel	
Mme	CHATILLON Catherine	
MM.	COULON Michel	
	HAMMOU Abdelkader	
	JOUD Jean-Charles	
	RAVAINE Denis	
	SAINFORT	C.E.N.G.

.../...

MM. SARRAZIN Pierre
SOUQUET Jean-Louis
TOUZAIN Philippe
URBAIN Georges — Laboratoire des Ultra-Réfractaires ODEILLO

E.N.S.M.E.E.

MM. BISCONDI Michel
BOOS Jean-Yves
GUILHOT Bernard
KOBILANSKI André
LALAUZE René
LANCELOT Françis
LE COZE Jean
LESBATS Pierre
SOUSTELLE Michel
THEVENOT François
THOMAS Gérard
TRAN MINH Canh
DRIVER Julian
RIEU Jean

E.N.S.E.R.G.

MM. BOREL Joseph
CHEHIKIAN Alain
VIKTOROVITCH Pierre

E.N.S.I.E.G.

MM. BORNARD Guy
DESCHIZEAUX Pierre
GLANGEAUD François
JAUSSAUD Pierre
Mme JOURDAIN Geneviève
MM. LEJEUNE Gérard
PERARD Jacques

E.N.S.H.G.

M. DELHAYE Jean-Marc

E.N.S.I.M.A.G.

MM. COURTIN Jacques
LATOMBE Jean-Claude
LUCAS Michel
VERDILLON André

Je tiens à remercier,

Monsieur Gérard VEILLON, Professeur à l'Institut National Polytechnique de Grenoble, qui m'a permis d'obtenir les moyens nécessaires pour travailler dans des conditions satisfaisantes, et qui m'a fait l'honneur de présider le jury de cette thèse ;

Monsieur Jacques COURTIN, Professeur à l'Université des Sciences Sociales de Grenoble, qui a bien voulu se charger de la direction de ces recherches ; ses critiques et ses conseils judicieux ont été pour une grande part dans la réussite de ce travail ;

Monsieur Vania JOLOBOFF, Ingénieur de la Société Européenne de Miniordinateurs et de Systèmes, qui a accepté de juger cette thèse et de participer au jury ;

et tous ceux qui m'ont aidé dans mes recherches, notamment Monsieur Ernest GRANDJEAN qui m'a fait profiter de son expérience et de sa grande connaissance du système PIAF.

Je veux exprimer ma gratitude au Laboratoire Brigitte Frybourg du CNAM à Paris et à son directeur, Monsieur le Professeur M. AVAN ; leur collaboration a été très bénéfique, particulièrement en ce qui concerne la mise au point du traducteur en braille abrégé.

Je dois dire ici que mon travail et ma vie au laboratoire ont été très facilités grâce à la gentillesse de tous ceux qui m'ont spontanément aidé à résoudre mes problèmes particuliers.

Enfin, je remercie vivement Marie-José DOREL qui a assuré la dactylographie de cette thèse et le Service de Reproduction qui en fait l'impression.

SOMMAIRE

INTRODUCTION

Cet ouvrage comprend deux parties : la première partie est consacrée aux rappels concernant le Transducteur Général d'Etats Finis du système PIAF (Programmes Interactifs de l'Analyse du Français) et à la description de certaines applications réalisées à partir du Transducteur Général d'Etats Finis. La deuxième partie est consacrée à une extension du Transducteur Général d'Etats Finis à un réseau d'automates (appelé MULTPIAF) permettant d'envisager de nouvelles applications.

I - LE TRANSDUCTEUR GENERAL D'ETATS FINIS ET SES APPLICATIONS

L'équipe Intelligence Artificielle du Laboratoire IMAG a conçu et réalisé des outils de traitement de la langue naturelle. Ces travaux ont donné naissance au système PIAF qui comporte plusieurs modules [Cour1, Cour2, CtDj, Grj2, Chia, ChCGV, Jol] :

1) Un Transducteur Général d'Etats Finis ou TGEF qui est un automate d'états finis permettant l'analyse et la transduction de données séquentielles à l'aide d'une grammaire et d'un dictionnaire donnés en paramètres du programme.
Ce module a primitivement été conçu pour réaliser l'analyse morphologique du Français. Il permet de découper la phrase en mots et pour chacun de ces mots de fournir des renseignements morphologiques tels que catégories et variables grammaticales.
Nous verrons que le TGEF peut avoir beaucoup d'autres applications que celle mentionnée ci-dessus.

2) Un analyseur syntaxique qui permet, à partir de la phrase découpée en mots, des renseignements fournis par l'analyseur morphologique et d'une liste de relations de dépendances entre catégories, de construire les différentes structures possibles d'une phrase (structure arborescente).

Sur chaque arc des structures arborescentes produites, il est possible d'appliquer une règle hors-contexte permettant de contrôler l'accord entre les variables grammaticales (accord en genre et en nombre).

Notre étude a porté uniquement sur le TGEF.

A la suite d'une demande du Laboratoire Brigitte Frybourg (Laboratoire pour la réinsertion professionnelle des handicapés) du CNAM à Paris, nous nous sommes intéressés aux problèmes concernant l'édition de livres en braille.

Notre étude a donc porté sur la mise au point d'un traducteur en braille abrégé à l'aide du TGEF qui permet de traduire un texte français en braille abrégé et de le produire sur le terminal braille papier SAGEM TEM 8 br avec toutes les conventions spécifiques à l'édition du braille.

Nous nous sommes aussi intéressés à la traduction du braille abrégé vers l'intégral car un tel traducteur pourrait être un élément intéressant d'un outil de rédaction, pour les non-voyants, utilisant l'ordinateur.

Enfin, nous exposerons les travaux réalisés par l'équipe Intelligence Artificielle concernant le contrôle de saisie, détection de fautes de frappe et d'orthographe d'usage, travaux qui ont été repris par le Laboratoire Brigitte Frybourg.

Un tel outil nous semble important pour l'édition de livres en braille car le principal problème concernant cette édition n'est pas tant la traduction en braille abrégé que la saisie correcte, rapide et bon marché des informations textuelles.

La première partie se composera donc :

- de rappels concernant le TGEF,
- d'une description du traducteur en braille abrégé et du traducteur inverse,
- d'une description du contrôle de saisie à l'aide du TGEF.

II - EXTENSION DU TGEF

L'utilisation du TGEF dans différentes applications nous a montré qu'il semblait intéressant de développer un réseau permettant d'implanter plusieurs TGEF pour une même application, chaque TGEFétant affecté à une tâche particulière.

Par exemple, en ce qui concerne l'édition de livres en braille, il serait nécessaire de créer une chaîne complète de traitement, c'est-à-dire un système permettant la saisie automatique (bandes photo-composeuses, lecteur optique) ou manuelle des informations, de contrôler cette saisie et de traduire en braille abrégé en vue de l'édition.

Chacune de ces tâches peut être réalisée par un ou plusieurs TGEF. Notamment, deux TGEF seront nécessaires pour faire l'analyse et le contrôle des données car ces dernières contiennent des informations de deux types : informations textuelles contrôlées par un analyseur morphologique et des informations typographiques utilisées pour l'édition et qu'il faut modifier pour permettre la traduction en braille abrégé.

En cours de travail, le système gèrera l'appel des différents TGEF.

Nous avons donc développé un nouveau système appelé MULTPIAF qui permet de créer un réseau de TGEF en spécifiant les noms des différents TGEF composant le réseau et l'organisation interne de ce réseau.

Le système MULTPIAF conserve toutes les propriétés du TGEF :

- interactivité pendant la création ou la modification du réseau et en cours de fonctionnement,
- généralité d'applications : les informations concernant le réseau sont des paramètres du système,
- langage simple : le langage utilisé dans l'éditeur qui permet de créer ou de modifier le réseau est un langage simple accessible notamment aux non-informaticiens.

La deuxième partie se compose donc :

- d'une présentation générale du système MULTPIAF et d'une description des différents types d'appels possibles des automates ;

- d'une analyse du langage accepté par un tel système ;
- de rappels sur l'éditeur du TGEF et d'une description de celui de MULTPIAF et du langage utilisé pour créer ou modifier le réseau ;
- d'une présentation des principaux algorithmes de fonctionnement du système et d'applications expérimentales réalisées à l'aide de ce système.

PARTIE I

APPLICATIONS DU SYSTEME PIAF

CHAPITRE 1

RAPPELS SUR LE TRANSDUCTEUR GENERAL D'ETATS FINIS

Le Transducteur Général d'Etats Finis ou TGEF est un système permettant de faire l'analyse et la transduction de langages d'états finis de façon interactive en fonction de paramètres fournis par l'utilisateur, ces paramètres étant une grammaire à validations-saturations (équivalente à une grammaire d'états finis) et un dictionnaire.

Le système propose des outils de Définition et des outils de Manipulation des différentes données textuelles. Un éditeur permet le contrôle interactif des différentes transactions :

1) Les outils de Définition permettent la création et la mise à jour du dictionnaire et de la grammaire.

2) Les outils de Manipulation permettent l'analyse et la transduction de données textuelles en fonction du dictionnaire et de la grammaire définis précédemment.

I.1.1. LA GRAMMAIRE [Courl]

Elle se compose principalement d'une liste de règles et de modèles.

1) Une règle se compose d'une étiquette qui permet de la désigner, d'affectations ou de calculs de variables et de catégories, d'un ensemble de validations et de saturations sous forme de listes de règles et d'un indicateur d'état final.

- Catégories et variables : Elles sont utilisées pour réaliser la transduction et sont affectées ou calculées au cours de l'application de la règle. Dans le cas de l'analyse morphologique du français, les variables seront les variables grammaticales (masculin, féminin, singulier, pluriel, ...) et les catégories seront les catégories grammaticales (substantif commun, verbe, ...).

  Ces catégories et ces variables sont facultatives ; elles sont déclarées dans la grammaire selon l'application. Les variables sont calculées, c'est-à-dire que l'on peut affecter, additionner ou soustraire une valeur à une variable. Une catégorie peut être affectée ou peut prendre la valeur qu'elle avait dans l'état précédent.

Nous indiquons, ici, deux éléments importants qui sont le code morphologique (CM) et le code dérivation (CD). Le code morphologique est une liste de règles et nous verrons dans le fonctionnement de l'automate son utilisation.
Le CD permet de conserver les différentes valeurs prises par une catégorie pendant l'analyse d'un mot. Il est possible au niveau de chaque règle d'indiquer comment le CD doit conserver les valeurs, conservation soit des valeurs antérieures, soit de la nouvelle valeur, soit de la concaténation des valeurs antérieures et de la nouvelle valeur.

- Un ensemble de validations et de saturations[1] qui entre dans le calcul des états suivants possibles de l'automate. Ce calcul sera exposé dans le paragraphe concernant le fonctionnement de l'automate.
- Un indicateur (FIM) permettant de signaler à l'automate qu'il se trouve dans un état final.

2) Les modèles :

Ils servent à faire le lien entre les règles de la grammaire et les éléments du dictionnaire ; chaque élément du dictionnaire fait référence à un modèle qui lui-même fait référence à un ensemble de règles applicables, ce qui permet de regrouper tous les éléments du dictionnaire qui provoquent le même comportement de l'automate dans un état donné.

Les modèles contiennent :

- une liste de règles à appliquer ;
- un ensemble de validations et de saturations qui entre dans le calcul des états suivants possibles de l'automate ;
- une liste d'affectations de catégories et de variables.

- Une liste de règles à appliquer : c'est cette liste qui permet au modèle de faire référence à des règles à appliquer.

---

(1) Algorithmes pour le Traitement Interactif des Langues Naturelles - J. COURTIN.

- L'introduction d'une liste de validations et de saturations dans les modèles est une facilité offerte à l'utilisateur pour écrire la grammaire. Le fait d'avoir une liste dans les règles et une liste dans les modèles peut permettre de diminuer le nombre de règles.
- Une liste de valeurs de catégories et de variables. Ces valeurs peuvent être utilisées par les règles pour affecter ou calculer les variables ou les catégories.

I.1.2. LE DICTIONNAIRE [Cour1, Grj2]

Il peut être considéré comme une interface entre la chaîne d'entrée et l'automate en faisant correspondre à chaque élément une référence à un modèle.

De plus, il contient des informations qui peuvent être utilisées pour la transduction. Nous verrons pour chacune des applications envisagées l'utilisation de ces informations.

Un élément du dictionnaire contient :

1) une chaîne de caractères
2) des indicateurs
3) une référence à un modèle
4) un chaînage à un autre élément du dictionnaire.

La chaîne de caractères et la référence au modèle sont obligatoires, le reste des informations est facultatif.

1) La chaîne de caractères servira à faire l'identification avec la chaîne d'entrée.

2) Les indicateurs : il en existe six qui ont des fonctions différentes selon les applications. Nous donnons ici seulement l'indicateur (*) qui a la même fonction dans toutes les applications ; il signifie que lorsque l'automate a trouvé une solution, il n'a pas le droit d'en rechercher une autre en redécoupant la chaîne de caractères, par exemple.

3) Le nom du modèle sert directement de référence.

4) Le chaînage à un autre élément : il est appelé chaînage synonyme

(indicateur S) et est spécifiquement indiqué par l'utilisateur. Il est notamment utilisé, couplé avec un indicateur (indicateur S), en traduction braille abrégé pour trouver la traduction d'une chaîne de caractères validée par l'automate.

I.1.3. FONCTIONNEMENT DE L'AUTOMATE

Nous donnons ci-dessous les principes fondamentaux du fonctionnement des automates ; pour plus de détails, il est possible de se reporter à la thèse de J. COURTIN [Cour1].

L'automate identifie la chaîne d'entrée aux éléments du dictionnaire et contrôle l'application des règles.

1) Identification dans le dictionnaire : il recherche, à partir de la gauche de la chaîne d'entrée, la première coïncidence avec les éléments du dictionnaire.
Du fait de son organisation, la première coïncidence sera la plus longue. S'il y a échec, l'automate choisira un élément plus court en prenant le premier fils de l'élément sélectionné, s'il existe.

2) Contrôle de l'application des règles : l'automate sélectionne un modèle référencé par l'élément choisi dans le dictionnaire, il applique une des règles valides.

Si aucune solution n'est possible, l'automate revient à l'état précédent.

I.1.4. APPLICATION DU TGEF

Du fait de sa grande souplesse et de son caractère général, le TGEF a servi dans beaucoup d'applications.

Nous en mentionnons, ci-dessous, quelques-unes.

Analyse morphologique et contrôle de saisie [Cour1, Cour2, CtDj]

Historiquement, le TGEF a été construit pour cette application. Le système réalise l'analyse morphologique de textes et détecte les fautes d'orthographe d'usage et de frappe en fonction d'un modèle de la langue au moment de la saisie des données. Dans ce cas, la transduction consistera en la production de catégories et de variables grammaticales qui seront les données de l'analyseur syntaxique.

Contrôle de saisie et documentation automatique [Cour1, Grj1]

Dans cette application, le système réalise deux opérations en parallèle :

1) le contrôle de saisie des informations documentaires ;
2) la réduction sous forme canonique de mots-clés simples ou composés qui seront utilisés dans un système documentaire.

Traduction de programmes d'un langage dans un autre langage [Cour1, Cha]

Le TGEF a servi de traducteur de programmes d'un langage dans un autre langage de même niveau, plus particulièrement du PL360 IBM en LP80 Iris 80 CII.

Cette méthode a été utilisée pour implanter les programmes du TGEF sur IRIS 80 à partir de la version existante sur IBM 360.

Applications au braille abrégé [Cour1, Mat1, Mat2, GiHu]

Nous avons utilisé le TGEF pour réaliser un traducteur en braille abrégé. Ce qui a principalement consisté à construire une grammaire et un dictionnaire (au sens de PIAF) formalisant les règles de la grammaire du braille abrégé et à réaliser des procédures de sorties permettant d'éditer les résultats d'une manière satisfaisante.
Nous nous sommes intéressés à la traduction inverse, c'est-à-dire la traduction du braille abrégé vers le français, en utilisant le même dictionnaire et la même grammaire.

Il en existe bien d'autres telles que la génération phonétique, la correction automatique des fautes d'orthographe, etc.

Les applications concernant le contrôle de saisie, la traduction en braille abrégé et la traduction inverse seront le sujet des chapitres suivants.

CHAPITRE 2

APPLICATION DU TRANSDUCTEUR GENERAL D'ETATS FINIS POUR LA PRODUCTION DE DOCUMENTS EN BRAILLE ABREGE

INTRODUCTION

Les textes en braille se présentent sous la forme de chaînes de caractères codés sur six positions. La réalisation physique se fait par l'embossage de points sur un papier fort.

En général, les lignes sont de 31 ou 40 caractères.

Il existe deux modes de codification braille :

1) Un mode dit intégral ou un caractère braille correspond à un caractère graphique.
2) Un mode dit abrégé où l'on contracte certaines chaînes de caractères en un, deux,trois ou quatre caractères braille, ceci pour augmenter la rapidité d'écriture et de lecture et pour diminuer le volume très important des livres en braille. Contrairement à la sténographie, ces abréviations conservent l'orthographe et ont été définies en essayant d'éviter le plus possible les ambiguïtés à la lecture.

Le système d'abréviation est spécifique à chaque langue et son utilisation est plus ou moins grande selon les pays. D'autre part, certaines langues ne possèdent pas encore de système d'abréviation braille bien défini.

En ce qui concerne le français, les règles de production du braille abrégé ont été définitivement précisées dans "L'Abrégé Orthographique Etendu" publié par l'Association Valentin Hauy [AVH].

L'Abrégé utilise les 64 possibilités du caractère braille, certains codes sont donc utilisés plusieurs fois : soit avec la valeur attribuée à ce caractère, soit comme composant d'une abréviation.

Exemple : ô peut signifier (ô) ou (dr) ou (ant).
En général, les ambiguïtés à la lecture sont levées par la position du caractère dans le mot ou le groupe de mots.
Dans l'exemple précédent, ô représente (ô) devant une consonne, (dr) devant une voyelle et (ant) en fin de mot.

I.2.1. APPLICATION DU TGEF A LA TRADUCTION EN BRAILLE ABREGE [Mat1, Mat2]

I.2.1.1. Les règles du braille abrégé

Les abréviations en braille peuvent se séparer en trois groupes :

1) Les abréviations du premier groupe concernent les abréviations de mots très courants tels que les articles, les pronoms ou les conjonctions .... Ces abréviations utilisent un ou deux codes braille et n'obéissent à aucune contrainte particulière.

Exemple : (le) s'abrège par (l), (mais) s'abrège par (x), etc.

2) Les abréviations du second groupe concernent les abréviations de groupes de lettres pouvant être utilisées dans un mot.

Pour lever les ambiguïtés qu'il peut y avoir entre les différentes significations d'un même code, ces abréviations sont soumises à certaines contraintes.

Ces contraintes peuvent être générales, valables pour toutes les abréviations de ce groupe, ou particulières à un symbole d'abréviation.

Exemple de contrainte générale : interdiction d'utiliser consécutivement le même code braille :
drôle ne peut s'abréger (ôôle).

Exemple de contrainte particulière : (re) peut s'abréger (') mais uniquement en début de mot et devant une consonne.

Les règles d'abréviation sont faites de manière à conserver le découpage syllabique (interdiction d'abréger un groupe de lettres se trouvant à cheval sur deux syllabes, sauf pour les redoublements de lettres comme (tt) ou (ll), et les règles de prononciation (interdiction d'abréger (ien) dans "ils chatient" par exemple, mais autorisé dans "il revient").

3) Les abréviations du troisième groupe concernent des abréviations spéciales de mots ou groupes de mots.

Des mots ou groupes de mots considérés comme très fréquents se sont vus attribuer des abréviations spéciales. Il existe pour chacune de ces abréviations des dérivations permettant d'abréger les dérivations du mot correspondant. Par exemple : lumière s'abrège lm, lumineux lmx, lumineuse lmse, luminosité lmst, etc.

Actuellement, la réduction est de l'ordre d'un tiers ; ce faible résultat tient peut-être au fait que les mots ou groupes de mots choisis dans le troisième groupe d'abréviations ne correspondent plus exactement aux mots les plus fréquents du français,produisant une certaine inefficacité de ce dernier groupe.

I.2.1.2. Application du TGEF à la traduction en braille abrégé

L'étude de la grammaire du braille abrégé montre qu'elle est équivalente à une grammaire d'états finis donc à une grammaire à validations-saturations. Il est par conséquent possible d'utiliser le TGEF pour produire du braille abrégé.

Dans cette application, le TGEF segmente la chaîne d'entrée à l'aide de l'identification d'éléments dans le dictionnaire et contrôle la concaténation de ces éléments à l'aide de la grammaire (30 règles et 30 modèles pour la grammaire du braille abrégé français).

La transduction sera, dans ce cas, la mémorisation d'un appel d'une procédure ayant en paramètre l'élément identifié. Lorsque l'automate arrivera dans un état final, ces appels seront effectués dans l'ordre où ils auront été mémorisés.

Ces procédures permettent d'associer à chacun des éléments identifiés leur traduction d'une manière ad hoc. Le résultat du travail du TGEF sera donc la concaténation de la traduction de ces différents éléments ayant servi à l'analyse.

Le dictionnaire est composé de mots ou de groupes de mots, de lettres ou de groupes de lettres servant à la segmentation et auxquels on associe leur traduction en utilisant le chaînage synonyme mentionné dans le chapitre précédent. Les éléments servant à l'analyse possèdent l'indicateur (S) alors que leur traduction ne le possède pas.

Nous avons prévu quatre procédures différentes de traduction :

- DICT : cette fonction recopie l'élément identifié sans rechercher de traduction associée. Elle est utilisée pour toutes les chaînes de caractères qui ne possèdent pas d'abréviations.

- SYNO : cette fonction permet de parcourir le chaînage synonyme pour retrouver l'élément ne possédant pas l'indicateur (s) et de le prendre comme traduction. Si cet élément n'existe pas, alors un message d'erreur est envoyé à l'utilisateur.

- COMMENT : cette fonction a un rôle très particulier ; elle permet de recopier une chaîne comprise entre deux séparateurs sans l'abréger. A la rencontre du séparateur "début de chaîne" identifié dans le dictionnaire, elle va tout d'abord rechercher en parcourant les chaînages synonymes le séparateur "fin de chaîne" correspondant, et recopier tous les caractères de la chaîne d'entrée jusqu'à la rencontre du séparateur "fin de chaîne".

Cette fonction est très importante car dans l'édition d'un texte en braille abrégé, il est souvent nécessaire de ne pas abréger certaines chaînes : titres, noms propres, mots étrangers, etc.

- CADRAGE : cette fonction est utilisée pour l'édition, elle permet de réaliser physiquement les fins de paragraphes ou d'alinéas. Au caractère identifié dans le dictionnaire qui provoque l'application de cette fonction, est associée la chaîne de caractères à mettre sur la ligne de sortie suivante. Cette chaîne de caractères sera pratiquement toujours constituée d'un certain nombre de blancs.

Il est à remarquer que les propriétés du TGEF ont grandement facilité la mise au point de ce traducteur notamment :

- l'interactivité qui a été très utile pendant la phase de mise au point du dictionnaire et de la grammaire ;
- l'identification prioritaire de la plus grande coïncidence entre la chaîne d'entrée et les éléments du dictionnaire, ce qui permet de reconnaître sans séparateurs particuliers des locutions à abréger, comme "pour ainsi dire", "à présent", etc, et à interdire ou à forcer l'abréviation d'exceptions aux règles générales d'abréviations en introduisant directement le mot litigieux dans le dictionnaire

Si aucune règle ne conduit à un résultat par un système de retour-arrière, l'automate peut rechercher des éléments plus courts et trouver une solution.

I.2.1.3. Exemples de traduction

Nous donnons ici quelques exemples de la méthode utilisée par le TGEF pour effectuer la traduction de mots en braille abrégé. La représentation de la traduction en braille abrégé sera donnée, ci-dessous, par la valeur graphique des caractères braille correspondants.

Traduction de : autorisation_

Le caractère (_) représente le caractère blanc.

Données	Elément identifié	Procédure	Traduction correspondante
autorisation_	au	SYNO	k
torisation_	t	DICT	t
orisation_	or	SYNO	;
l'abréviation de or est interdite devant une voyelle			
orisation_	o	DICT	o
risation_	r	DICT	r
isation_	i	DICT	i
sation_	s	DICT	s
ation_	ation	SYNO	â
-	-	DICT	-

Ce dernier élément a provoqué l'état final de l'automate.

Résultat de la traduction : ktorisâ_

I.2.2. TRADUCTION INVERSE DU BRAILLE ABREGE

Nous nous sommes intéressés à la traduction inverse du braille abrégé, traduction de l'abrégé vers l'intégral, pour plusieurs raisons.

- L'apparition sur le marché de terminaux en braille, utilisant notamment le braille éphémère[1], et les progrès constants de la micro-informatique vont permettre, dans un futur proche, de fournir des outils de rédaction très efficaces aux non-voyants, ce qui facilitera la communication entre personnes voyantes et personnes non-voyantes.

---

(1) Braille éphémère : système électronique permettant l'affichage du braille à l'aide de petites pointes pouvant apparaître ou disparaître sur une plage tactile.

Or, il est souvent plus facile à un aveugle d'écrire en braille abrégé qu'en braille intégral.

La mise au point d'un tel outil peut donc avoir une certaine utilité.

- Les méthodes envisagées pour résoudre ce problème peuvent être généralisées à d'autres systèmes d'abréviations utilisées dans la saisie de textes.

Un système d'abréviation est parfaitement réversible s'il existe une bijection entre les éléments à abréger et les abréviations. Dans ce cas, le système ne présente aucune ambiguïté, et il est très facile de passer de l'intégral vers l'abrégé ou inversement de l'abrégé vers l'intégral.

Si cette bijection n'existe pas, il est parfois encore possible de lever les ambiguïtés en utilisant le voisinage immédiat de la chaîne de caractères ambiguë. C'est le cas du braille abrégé français.

Il existe, enfin, des systèmes d'abréviation extrêmement ambigus où il est impossible de lever les ambiguïtés sans utiliser un contrôle de saisie.

C'est le cas de la sténotypie où il est possible d'utiliser le modèle syntaxique de PIAF pour lever les ambiguïtés [Cour2, Chom, Woods].

Nous avons voulu que le dictionnaire et la grammaire utilisés par le TGEF pour la traduction de l'intégral vers l'abrégé puissent servir aussi pour la traduction de l'abrégé vers l'intégral. Pour ceci, nous avons dû modifier le TGEF pour le faire fonctionner d'une manière réversible. Nous parlons ici de réversibilité car l'automate fonctionne dans le même sens, il part de l'état initial et va vers un état final, seulement l'entrée et la sortie sont inversées.

I.2.2.1. Mécanisme de fonctionnement

Pour réaliser cette traduction inverse, nous avons introduit le mécanisme suivant :

Identification : le TGEF identifie un élément du dictionnaire avec la chaîne d'entrée de la même manière que dans le fonctionnement normal (élément le plus long identifié en premier).

- Application d'une règle : pour appliquer une règle, l'automate va parcourir le chaînage synonyme (chaînage circulaire) jusqu'à trouver un élément qui possède une référence à un modèle ; on essaiera d'appliquer les règles demandées par le modèle.

Si aucune règle ne conduit l'automate dans un état final alors il y a recherche d'un autre élément possédant une référence à un modèle.

Le résultat sera donc la concaténation des éléments qui auront conduit à l'application des règles (élément portant la référence d'un modèle).

Comme dans le fonctionnement normal, les résultats ne sont effectivement produits que lorsque l'automate est dans un état final. Le même système de retour arrière permet de chercher une autre solution particulièrement si l'automate ne peut atteindre un état final.

La différence de fonctionnement réside donc dans la manière dont le dictionnaire est utilisé comme interface entre la zone de données et l'automate.

Il faut noter que le TGEF a pu être rendu réversible parce qu'un élément est rangé dans le dictionnaire avec sa traduction et qu'il est possible d'identifier l'élément ou sa traduction.

Pour détecter les ambiguïtés, il est possible de prévenir l'utilisateur lorsque l'automate découvre plusieurs solutions.

I.2.2.2. Réversibilité du braille abrégé

La codification braille possède soixante-quatre caractères ; en conséquence, pour réaliser un système d'abréviation, on a été obligé d'attribuer plusieurs significations, jusqu'à quatre, pour le même code.

Les ambiguïtés sont donc très nombreuses si l'on veut passer de l'abrégé à l'intégral. Nous les avons classées en trois types :

- 1) Les ambiguïtés prévues par la grammaire du braille abrégé et qui ont des contraintes d'utilisation pour faciliter la lecture. Par exemple, la plupart des ponctuations peuvent avoir une valeur d'abréviation, elles sont donc, en général, interdites en fin de mots.

Du fait de ces contraintes, ces ambiguïtés sont levées et ne posent pas de problème pour la réversibilité.

- 2) Les ambiguïtés qui n'ont pas été prévues par la grammaire du braille et qui sont en général implicitement levées par le lecteur. Pour les résoudre, nous avons utilisé les particularités d'emplois, en français, des chaînes de caractères que représentent les abréviations ambiguës.

Exemple : (â) peut signifier â ou être l'abréviation de ation (en fin de mot) ou de fr. L'abréviation de fr n'obéit à aucune contrainte. Nous avons donc ajouté la contrainte suivante :

- devant une voyelle, â abrège (fr) ;
- en fin de mots, â abrège (ation).

(â) peut toujours signifier l'abréviation de (ation) en fin de mots ; il ne semble pas qu'il existe de mots en français se terminant par â, ce qui lève l'ambiguïté ;

- partout ailleurs â signifie (â).

(k) signifie "k" ou l'abréviation de "au" sans contraintes spéciales pour faire la distinction.
Pour lever l'ambiguïté, on ajoute :

- (k) abrège "au" devant une consonne et en fin de mots, ailleurs il signifie "k" ; excepté pour quelques mots comme "krypton" ou "kayak" que l'on entre dans le dictionnaire avec leur traduction.

- 3) Des ambiguïtés qui sont impossibles à lever, comme : en fin de mot, (z) signifie z ou l'abréviation de (ez) et (q) signifie q ou l'abréviation de (que). Il est donc possible de traduire (coq) par coq ou coque et (riz) par riz ou riez.

Il ne peut exister de règles simples pour lever ces ambiguïtés. Elles sont en fait très rares et la solution que nous préconisons est de faire choisir l'utilisateur entre coq ou coque, riz ou riez lorsqu'il faut traduire coq ou riz.

Les conventions d'écriture en braille obligent à coller les ponctuations contre le mot précédent. Dans ce système, cette convention est impérative car prise isolément, chacune de ces ponctuations peut abréger un mot comme puis, en, etc. Cette convention permet de distinguer les ponctuations prises en tant que telles des ponctuations prises en tant qu'abréviations.

Pour pouvoir affirmer que le système d'abréviation braille est un système réversible, il faudrait pouvoir traduire avec succès un grand nombre de

textes abrégés. Cependant, à quelques exceptions près, notre analyse tendrait à prouver que c'est un système réversible. S'il existe d'autres ambiguïtés que celles mentionnées plus haut, elles seront tout à fait ponctuelles et pourront être résolues en indexant le mot litigieux avec sa traduction dans le dictionnaire.

Les caractères d'édition tels que les délimiteurs de chaîne à ne pas traduire, fin de paragraphe et fin de page sont souvent traduits par des caractères blancs. Dans le cas d'un système réversible, il faut éviter cette perte d'informations ; nous avons opté pour la solution suivante : un caractère X d'édition dans un texte intégral sera traduit par *X dans le texte abrégé. Le caractère (*) permet d'éviter toute ambiguïté avec des abréviations existantes représentées par les caractères d'édition.

Il est évident que si l'on voulait réaliser un traducteur abrégé-intégral utilisé comme outil de rédaction, le problème serait différent et il faudrait traduire *X par le nombre de caractères blancs correspondants.

CHAPITRE 3

CONTROLE DE SAISIE

Nous présentons dans ce chapitre une application du TGEF permettant de réaliser le contrôle de saisie d'un texte. Nous entendons par contrôle de saisie la détection des fautes d'orthographe d'usage et de frappe et non la détection de toutes les fautes qu'il peut y avoir dans un texte (contrôle morphologique sans contrôle syntaxique).

Ce qui signifie qu'un contrôle manuel ne sera pas exclu de la saisie, mais il sera grandement facilité par cet outil car les fautes les plus nombreuses sont des fautes de frappe ou de lecture erronée quand il s'agit de lecture automatique dans le cas de bandes photo-composeuses ou de lecture optique.

I.3.1. PRINCIPE DU CONTROLE

Pour contrôler un texte, l'automate va comparer tous les mots de ce texte à un modèle de la langue qu'il possède au moment de la lecture des données. Si un mot n'appartient pas à ce modèle, l'utilisateur est averti de la découverte de ce mot étranger au modèle ; il peut alors indexer le mot nouveau, modifier la grammaire ou le texte d'entrée ou empêcher l'automate d'analyser ce mot (mot correct mais que l'on ne veut pas voir figurer dans le dictionnaire).

Il faut remarquer que le modèle de la langue joue un rôle très important et qu'il ne peut contenir d'erreurs ; l'opération d'indexation est donc une opération délicate.

I.3.2. APPLICATION DU TGEF

Pour réaliser cette application, le TGEF est réduit au rôle d'accepteur, c'est-à-dire que le résultat n'est pas la transduction de la chaîne d'entrée mais la concaténation des mots acceptés par l'automate (ou par l'utilisateur).

Pour ceci, il a suffi de transformer la sortie de l'analyseur morphologique et de ne conserver que la partie analyse.

L'analyseur morphologique [Courl, CtDj]

Le dictionnaire contient des formes, des expressions invariables, des racines, des suffixes et des terminaisons qui font référence à des modèles morphologiques (400 modèles pour le français) et l'automate d'états finis contrôle l'application de règles de concaténation (180 règles pour le français) des éléments composant un mot (racine, suffixe, terminaison).

Si l'automate arrive dans un état final, le mot est accepté, sinon il y a redécoupage de la chaîne d'entrée pour trouver une autre solution ; si aucune solution n'est trouvée, alors il y a détection d'une erreur.

Dans le cas de l'accepteur, l'automate s'arrête sur la première solution trouvée, son rôle étant de reconnaître si un mot appartient ou n'appartient pas au français par exemple, la première solution lui suffit.

Par contre, l'analyseur morphologique qui est un préprocesseur de l'analyseur syntaxique a besoin de trouver toutes les solutions ; par exemple, (le) peut être article ou pronom.

I.3.3. ADAPTATION AUX PROBLEMES DU BRAILLE

Le laboratoire Brigitte Frybourg (J.M. Charpentier) a repris les travaux concernant l'analyse morphologique en typographie riche réalisés par l'équipe Intelligence Artificielle, pour adapter le dictionnaire et la grammaire à la codification utilisée par le terminal braille TEM 8 br de manière à être compatible avec le traducteur braille abrégé.

Cette codification, notamment, comporte (comme en braille) un caractère par lettre accentuée, même en ce qui concerne l'accent circonflexe ou le tréma.

Dans le cas de la typographie riche, l'analyse étant plus fine, le nombre de règles, de modèles et d'éléments dans le dictionnaire est plus grand.

Exemple de typographie riche et de typographie pauvre

- 1) En typographie pauvre, il existe, pour beaucoup de verbes, une ambiguïté entre la troisième personne du singulier du passé simple et la troisième personne du singulier de l'imparfait du subjonctif ; cette ambiguïté n'existe pas en typographie riche.

Exemple : le verbe falloir

	Passé simple	Imparfait du subjonctif
typographie pauvre	fallut	fallut
typographie riche	fallut	fallût

- 2) Pour analyser les dérivations du verbe haïr, on a besoin :
- en typographie pauvre, d'une seule racine "hai" pour toutes les dérivations ;
- en typographie riche, de deux racines "hai" et "haï", pour reconnaître des formes telles que : haïr, haïssons, hais, hait, ...

PARTIE II

LE SYSTEME MULTPIAF

CHAPITRE 1

RAPPELS SUR LE TRANSDUCTEUR GENERAL D'ETATS FINIS

II.1.1. INTRODUCTION

Après avoir réalisé les applications concernant le braille et produit quelques textes par ces méthodes, nous nous sommes aperçus que le principal problème pour réaliser des livres en braille n'était pas tant la traduction et l'édition en braille abrégé de textes français que leur saisie.

En effet, cette saisie peut être faite de plusieurs manières :

- soit manuellement, ce qui pose le problème des fautes de frappe et d'orthographe et du coût de la saisie ;
- soit en utilisant un système de saisie optique qui peut produire aussi des erreurs ;
- soit en récupérant les textes à partir de bandes de photo-composition utilisées pour faire l'édition des livres. Ces bandes posent un problème : il est nécessaire de les traduire pour produire un texte intermédiaire accepté par le traducteur et l'éditeur en braille abrégé.

Dans ces trois méthodes de saisie, le principal problème réside dans la détection des fautes de frappe ou d'orthographe.

En vue de réaliser une chaîne complète de traitement, saisie plus ou moins automatique, contrôle de cette saisie et enfin traduction en braille abrégé, nous avons étudié une extension du TGEF permettant l'implantation de plusieurs TGEF dans un même module.

Cette extension a donné naissance à un nouvel outil appelé système MULTPIAF. Il consiste en une structure de contrôle qui peut prendre en compte plusieurs TGEF et piloter leur fonctionnement.

Nous avons conservé pour cet automate spécialisé les propriétés du système PIAF et plus particulièrement du TGEF : généralité, interactivité et langage simple pour la mise en place de la structure.

II.1.2. PRESENTATION GENERALE

Le système MULTPIAF est une extension du TGEF du système PIAF. Il se présente comme un ensemble de TGEF reliés dans un réseau. Ce réseau appelé Surautomate permet le fonctionnement de ces automates en série (cascade) ou en parallèle (subroutine).

Chaque noeud représente un automate, éventuellement vide, et chaque transition représente un appel éventuel de l'automate courant à un autre automate.

Pour plus de commodité, nous donnerons le même nom à un noeud et à l'automate qu'il contient et par abus de langage, nous confondrons parfois le noeud et l'automate.

Il existe un point d'entrée du réseau ou noeud initial et un ou plusieurs points de sortie ou noeuds finals.
Par définition, le noeud initial contient toujours un automate vide.

Pour permettre la transduction, c'est-à-dire la transformation d'une information en une information de nature différente, un transducteur a besoin d'une zone de données et d'une zone de résultats. En conséquence, on affecte une zone de données et une zone de résultats. Une de ces zones peut être commune à plusieurs noeuds.

Ce mécanisme permet de définir complètement des automates en cascade ou en subroutine. En effet, deux automates en subroutine travailleront sur la même zone de données, alors que dans le cas de deux automates en cascade, les données du deuxième automate seront les résultats du premier automate.

II.1.3. DEFINITION D'UN AUTOMATE

Dans le système PIAF, un automate se définit par la donnée de sa grammaire, de son dictionnaire et de sa validation initiale.

Dans le système MULTPIAF, en plus des trois éléments indiqués ci-dessus, il faut pour définir un automate :

- donner une liste d'appels (éventuellement vide) qui précise quels sont les automates que l'automate peut appeler ;
- préciser les zones de données et de résultats pour cet automate ;
- préciser un certain nombre d'indicateurs pour spécifier le fonctionnement de l'automate.

Cette liste d'appels et la définition des zones de données et de résultats permettent de rattacher l'automate au réseau des automates. Il sera entièrement intégré lorsqu'il apparaîtra lui-même dans une liste d'appels.

Pour chaque automate, on précisera le dictionnaire, la grammaire et sa validation initiale. Plusieurs automates peuvent travailler sur le même dictionnaire et la même grammaire avec des validations initiales différentes ou avec des fonctionnements différents.

Par exemple, si l'on envisage une application contenant le traducteur en braille abrégé et le traducteur inverse[1], ces deux automates utilisent le même dictionnaire et la même grammaire mais avec des fonctionnements différents. On peut aussi envisager qu'un automate, dans une même application, n'accepte qu'un sous-langage d'un langage accepté par un autre automate.

Il est possible de définir un automate nul, c'est-à-dire un automate qui ne possède ni dictionnaire, ni grammaire et dont la validation initiale est nulle. Son seul rôle consistera à appeler d'autres automates et il ne fera aucun traitement sur la zone de données. C'est le cas, par exemple, de l'automate initial qui a la même fonction que la validation initiale pour le TGEF.

---

(1) Voir Chapitre 1.2.

II.1.4. MECANISME D'APPEL

Le principe général du mécanisme d'appel dans le système MULTPIAF est le suivant :
lorsque l'automate courant rencontre une chaîne étrangère à son langage, il appelle un autre automate pour analyser cette chaîne ; en cas de succès, il reprend sa propre analyse. Pour réaliser cet appel, il va consulter sa liste d'appels et lancer successivement les automates contenus dans cette liste. Dans certains cas, il s'arrêtera dès qu'un automate appelé aura obtenu un succès (fonctionnement en mode solution unique) ; dans d'autres, il lancera systématiquement tous les automates de sa liste (fonctionnement en mode solutions multiples).

Ce dernier mode de fonctionnement peut être utilisé dans le cas où les automates acceptent des langages non disjoints et où le réseau est non déterministe.

Il est à remarquer que l'ordre des automates dans cette liste est très important.

Automates non nuls :

Un automate non nul peut appeler un autre automate à n'importe quel moment de son analyse. Il utilisera, en priorité, les règles d'états finis de sa grammaire à validations-saturations. Il n'appelle un autre automate qu'en cas d'échec. Ce dernier sert à analyser la chaîne étrangère qui interdit l'analyse par l'automate appelant.

Si l'automate appelant fonctionne en mode solution unique, il se bornera à trouver dans sa liste d'appels le premier automate qui pourra analyser la chaîne obstacle ; sinon, il cherchera dans sa liste d'appels tous les automates pouvant analyser la chaîne obstacle.

Un automate non nul peut s'appeler lui-même ; nous étudierons ce cas plus précisément, dans le chapitre suivant, et nous regarderons aussi quelle conséquence cela peut avoir sur la nature du langage accepté par MULTPIAF.

Automates nuls :

Un automate nul n'acceptant aucun langage, toute chaîne est considérée comme étrangère ; l'automate nul va donc systématiquement appeler d'autres automates. Son rôle est multiple :

- en subroutine : deux automates travaillant sur la même zone de données et étant appelés par le même automate nul, joueront exactement le même rôle du point de vue de l'analyse (voir exemple) ;
- en cascade : avec le mode solutions multiples, plusieurs automates contenus dans une liste d'appel d'un automate nul dont les zones données-résultats sont en série fonctionneront en cascade.
  Il est important de noter que dans ce cas-ci, l'ordre dans lequel sont rangés les automates dans la liste d'appel est primordial.

  Cas particulier : l'automate initial est un automate nul et dont le mode est forcé au mode solutions multiples. Si sa liste d'appel contient plusieurs automates, ces derniers fonctionneront forcément en cascade.

Exemple :

AUTOMATES EN SUBROUTINE

Soit 3 TGEF :

A reconnaissant $a_1a_2...a_n$
B reconnaissant $b_1b_2...b_m$
C reconnaissant $c_1c_2...c_p$

Soit la configuration 1 :

automate initial : liste d'appel (A)
A : liste d'appel (B,C)
B : liste d'appel ()
C : liste d'appel ()

Cette configuration permet de reconnaître les chaînes suivantes : $a_1a_2...a_n$, $a_1a_2...b_1b_2...b_m...a_n$, $a_1a_2...b_1b_2...b_m...c_1c_2...c_p...a_n$, etc, mais ne reconnaîtra pas $b_1b_2...b_m$ ou $c_1c_2...c_p$ seul, car l'automate A rencontrant b_1 ou c_1 en début appellera directement B ou C ; au retour de l'appel, il sera toujours dans un état initial et ne trouvera pas d'état final pour terminer sa propre analyse.

Configuration 2 :

automate initial : liste d'appel (A0)
A0 : liste d'appel (A,B,C) automate nul
A : liste d'appel ()
B : liste d'appel ()
C : liste d'appel ()

Cette configuration ne reconnaitra que l'une ou l'autre des chaînes suivantes : $a_1 \ldots a_n$, $b_1 \ldots b_m$, $c_1 \ldots c_p$.

Il est possible de combiner les deux configurations en une seule pour reconnaître l'ensemble des chaînes.

Soit la configuration 12, combinaison des configurations 1 et 2 :

automate initial : liste d'appels(AO)
AO : liste d'appels(A,B,C) automate nul
A : liste d'appels(B,C)
B : liste d'appels()
C : liste d'appels()

Cette configuration reconnaîtra l'ensemble des chaînes reconnues par les configurations 1 et 2.

Dans ces trois configurations, le mode solutions multiples ou solution unique donne le même résultat car les langages reconnus par A, B, C sont disjoints.

Dans le cas contraire, si une même chaîne de caractères pouvait, par exemple, être reconnue à la fois par B et C en donnant des transductions différentes, alors il existerait dans chacun de ces réseaux des chemins différents pour reconnaître cette chaîne et le mode solutions multiples produirait un résultat différent du mode solution unique.

AUTOMATES EN CASCADE

Soit les TGEF suivants :

D reconnaissant $d_1 \ldots d_i$ et produisant $e_1 \ldots e_j$
E reconnaissant $e_1 \ldots e_j$ et produisant $f_1 \ldots f_k$

Configuration 3 :

automate initial : liste d'appels(D,E) lecture
D : liste d'appels() lecture zone init écriture zone D
E : liste d'appels() lecture zone D écriture zone E

Le système va transducter $d_1 \ldots d_i$ en $f_1 \ldots f_k$.
En effet, l'automate initial fonctionne implicitement en solutions multiples (appelle tous les automates de sa liste d'appels les uns après les autres) ; il lancera donc d'abord D qui transductera la chaîne $d_1 \ldots d_i$ en la chaîne

$`e_1 \ldots e_j$, puis E qui transductera $e_1 \ldots e_j$ en $f_1 \ldots f_K$.

Pour D et E, le mode solutions multiples ou solution unique est sans importance car leur liste d'appel est vide.

II.1.5. LE SURAUTOMATE

Un automate, appelé surautomate, permet de contrôler les appels des différents TGEF et leur fonctionnement. La grammaire de ce surautomate décrit en fait le réseau des automates.

Ce surautomate, comme le TGEF, est entièrement interactif (en cas d'erreur, l'utilisateur peut modifier la grammaire du surautomate ou les grammaires et dictionnaires des TGEF) ; la grammaire de ce surautomate qui est un paramètre du système s'écrit dans un langage simple.

Nous aborderons dans les chapitres suivants l'étude du langage accepté par MULTPIAF, la descritpion de l'éditeur permettant de créer ou de modifier la grammaire du réseau d'automates et, en dernier lieu, nous présenterons les principaux algorithmes de ce système et quelques exemples d'applications expérimentales.

CHAPITRE 2

ETUDE SUR LE LANGAGE ACCEPTE PAR MULTPIAF

II.2.1. INTRODUCTION

Dans ce chapitre, nous étudierons uniquement le cas des automates en subroutine. En effet, les automates en cascade ne représentent pas de difficultés particulières en ce qui concerne le langage. Si le réseau est formé uniquement d'automates en cascade, alors chaque noeud représente un TGEF et accepte un langage d'états finis. Du point de vue langage, une telle cascade peut être équivalente à une succession de langages d'états finis.

Pour les automates en subroutine, nous étudierons deux cas :

- il n'existe pas dans le réseau d'automates s'appelant eux-mêmes ou d'appels croisés ;
- il existe dans le réseau des automates s'appelant eux-mêmes ou des appels croisés.

II.2.2. PAS D'APPELS CROISES OU RECURSIFS

Le fait de ne pas avoir d'appels récursifs ou croisés implique que le réseau est sans boucle. En effet, si trois automates A, B, C sont tels que :

A appelle B qui appelle C qui appelle A, cela revient à dire que A appelle C par transitivité et comme C appelle A, c'est donc un appel croisé.

D'autre part, chaque noeud est accessible à partir d'une même racine ; le réseau est donc réduit à un arbre. Nous allons montrer que dans ce cas-là, le langage accepté par MULTPIAF est équivalent à un langage d'états finis.

Soit LM ce langage.

II.2.2.1. Langage d'états finis inclus dans LM

Le réseau peut être réduit à un seul noeud, il contient donc un seul TGEF. Nous savons, d'autre part, que tout langage d'états finis est accepté par un TGEF ; on peut donc en déduire que tout langage d'états finis est accepté par le réseau.

On a donc : langages d'états finis inclus dans LM.

II.2.2.2. LM inclus dans les langages d'états finis

Pour démontrer cette proposition, nous distinguerons deux cas :

1) LM est formé de mots de la forme $W_1W_2W_3$ avec W_1W_3 reconnu par un automate A et W_2 reconnu par un automate B. Soit $W_1W_3 = a_1a_2..a_ia_{i+1}..a_n$ et $W_2 = b_1b_2..b_m$ avec longueur $W_1 = i$, $0 \leq i < n$ et longueur $W_2 = m$, $0 \leq m$.

2) LM est formé de mots W'_1 ou W'_2 ; W'_1 reconnu par un automate A' et W'_2 reconnu par un automate B'.

Ces deux formes de LM correspondent aux deux seuls appels élémentaires possibles dans MULTPIAF :

- la première forme correspond à un automate A pouvant appeler un automate B (voir configuration 1 du chapitre précédent).
Nous étudierons deux cas pour ce langage : soit W_2 est unique dans $W_1W_2W_3$ (ce qui veut dire A n'a le droit d'appeler qu'une fois B), soit il existe plusieurs insertions de W_2 dans W_1W_3 (A peut appeler plusieurs fois B).
Il existe une option permettant de dire si A a ou n'a pas droit aux appels multiples.
Par transitivité, si la proposition est vraie pour A appelle B, elle sera vraie pour A appelle B appelle C, etc.

- la deuxième forme de LM correspond à deux automates A, B appelés par un même automate AO (éventuellement vide) (voir configuration 2 du chapitre précédent).
Par transitivité, si la proposition est vraie pour deux automates, elles sera vraie pour un nombre quelconque d'automates.

Si la proposition est vraie pour ces deux cas de figures, elle sera vraie pour n'importe quelle configuration du réseau car elle sera toujours une combinaison de ces deux cas de figures.

Mots de la forme $W_1W_2W_3$:

Nous allons montrer qu'il existe une grammaire à validation-saturation engendrant les mots $W_1W_2W_3$ sachant qu'il existe des grammaires à validation-saturation engendrant W_1W_3 et W_2.

Nous utiliserons, pour cela, le formalisme et les résultats sur les grammaires à validation-saturation décrit par J. COURTIN[1].

II.2.2.3. Rappels sur les grammaires à validation-saturation

II.2.2.3.1. Grammaire à validations

Définitions

On définit une grammaire à validations par

- la donnée d'un *vocabulaire terminal* $V_{TV} = \{a,b,...\}$
- la donnée d'un sous-ensemble E des entiers naturels $E = \{1,2,...,n\}$
- un élément particulier $SV \in P(E)$ appelé axiome
- un ensemble fini PV de productions.

Par la suite, on note $GV = (V_{TV}, SV, PV, E)$.

Production (ou règle)

Une production est un élément d'une application finie notée $\rightarrow$ entre les éléments de E d'une part et les éléments de $V_{TV} \times P(E)$ d'autre part.

Les productions sont de la forme :

$$i \rightarrow a[j_1.j_2.....j_q]$$

ou

$$i \rightarrow a[\emptyset]$$

avec $i \in E$, $[j_1.j_2.....j_q] \in P(E)$ et $a \in V_{TV}$.

NB : Par abus de langage, on note les parties de E entre crochets.

---

(1) Algorithmes pour le traitement interactif des langues naturelles.

Relation $\overset{}{\Rightarrow}_{GV}$

Soient α et $\beta \in V_{TV}^{*} \times P(E)$.

On dit que $\alpha \overset{}{\Rightarrow}_{GV} \beta$ si et seulement si il existe une chaîne $\varphi \in V_{TV}^{*}$ et $[k_1, k_2, \ldots, k_r, i, k_{r+2}, \ldots, k_q] \in P(E)$ tels que :

$\alpha = \varphi[k_1, k_2, \ldots, k_r, i, k_{r+2}, \ldots, k_q]$

$\beta = \varphi\, a[j_1, j_2, \ldots, j_q]$ (ou $\beta = \varphi a[\emptyset]$)

$i \rightarrow a[j_1, j_2, \ldots, j_q]$ (où $i \rightarrow a[\emptyset]$) étant une production de GV.

On appelle *dérivation directe* une telle relation.

Relation transitive $\overset{*}{\Rightarrow}_{GV}$

Soient α et $\beta \in V_{TV}^{*} \times P(E)$.

On dit que $\alpha \overset{*}{\Rightarrow}_{GV} \beta$ s'il existe une suite finie $\alpha_0, \alpha_1, \ldots, \alpha_n$ $(n > 0)$ $(\alpha_i \in V_{TV}^{*} \times P(E))$ telle que

$$\alpha_0 \overset{}{\Rightarrow}_{GV} \alpha_1 \overset{}{\Rightarrow}_{GV} \alpha_2 \cdots \overset{}{\Rightarrow}_{GV} \alpha_n \text{ avec } \alpha = \alpha_0 \text{ et } \beta = \alpha_n$$

On appelle *dérivation* une telle relation.

Langage défini par une grammaire GV

On appelle langage défini par GV, que l'on note L(GV), le sous-ensemble de V_{TV}^{*} tel que :

$$L(GV) = \{\alpha / SV \overset{*}{\Rightarrow}_{GV} \alpha[\emptyset],\ \alpha \in V_{TV}^{*}\}.$$

Exemple

$GV = (V_{TV}, SV, PV, E)$

avec :

$V_{TV} = \{a, b, c, d\}$

$E = \{1, 2, 3, 4\}$

$SV = [1, 2]$

PV : $1 \rightarrow a[2, 3]$

$2 \rightarrow b[2, 4]$

$3 \rightarrow c[3, 4]$

$4 \rightarrow d[\emptyset]$

Dérivation :

$[1,2] \Rightarrow a[2,3] \Rightarrow ab[2,4] \Rightarrow ab^2[2,4] \Rightarrow \ldots \Rightarrow ab^n[2,4] \Rightarrow ab^n d[\emptyset]$
$[1,2] \Rightarrow a[2,3] \Rightarrow ac[3,4] \Rightarrow ac^2[3,4] \Rightarrow \ldots \Rightarrow ac^m[3,4] \Rightarrow ac^m d[\emptyset]$
$[1,2] \Rightarrow b[2,4] \Rightarrow \ldots \Rightarrow b^p[2,4] \Rightarrow b^p d[\emptyset]$

$L(GV) = \{a\, b^n\, d,\ a\, c^m\, d,\ b^p\, d \ /\ n > 0,\ m > 0,\ p > 0\}.$

II.2.2.3.2. Grammaire à validations et saturations

Motivations

Nous allons étendre les grammaires à validations en introduisant des "saturations" qui permettent d'interdire l'application de certaines règles.

Supposons que l'on ait le problème suivant : on veut écrire une grammaire engendrant les chaînes

$W_1\ W_2\ W_3$ et $W_4\ W_2\ W_5$

avec

$W_1 = a_1\ a_2 \ldots a_p$
$W_2 = b_1\ b_2 \ldots b_l$
$W_3 = c_1\ c_2 \ldots c_q$
$W_4 = d_1\ d_2 \ldots d_m$
$W_5 = e_1\ e_2 \ldots e_r$

Pour ce faire, on peut écrire $S_1 = [1,1']$

$[1] \rightarrow a_1[2]$	$[1'] \rightarrow d_1[2']$
$[2] \rightarrow a_2[3]$	$[2'] \rightarrow d_2[3']$
---	---
$[p-1] \rightarrow a_{p-1}[p]$	$[(m-1)'] \rightarrow d_{m-1}[m']$
$[p] \rightarrow a_p[p+1]$	$[m'] \rightarrow d_m[(m+1)']$
$[p+1] \rightarrow b_1[p+2]$	$[(m+1)'] \rightarrow b_1[(m+2)']$
---	---
$[p+1] \rightarrow b_1[p+l+1]$	$[(m+1)'] \rightarrow b_1[m+l+1)']$
$[p+l+1] \rightarrow c_1[p+l+2]$	$[(m+l+1)'] \rightarrow e_1[(m+l+2)']$
---	---
$[p+l+q-1] \rightarrow c_{q-1}[p+l+q]$	$[(m+l+r-1)'] \rightarrow e_{r-1}[(m+l+r)']$
$[p+l+q] \rightarrow c_q[\emptyset]$	$[(m+l+r)'] \rightarrow e_r[\emptyset]$

On constate que l'on écrit deux fois des règles contenant l'élément terminal b_i

$$[s+i] \rightarrow b_i[s+i+1].$$

Afin d'éviter cette réécriture, on introduit, parallèlement aux validations, des saturations. Si dans une dérivation, une règle est saturée, cela signifie qu'on ne peut plus appliquer cette règle. Pour l'exemple précédent, il suffit que les règles permettant la reconnaissance des chaînes W_1 et W_4 autorisent celle de la chaîne W_2 et interdisent respectivement celles des chaînes W_5 et W_3. Par contre, les règles permettant la reconnaissance de la chaîne W_2 autorisent celles des chaînes W_3 et W_5.

Définitions

Une grammaire à validations et saturations est définie par :

- la donnée d'un *vocabulaire terminal* $V_{TVS} = \{a,b,\ldots\}$
- la donnée d'un *sous-ensemble* E des entiers naturels $E = \{1,2,\ldots,n\}$
- un élément particulier $SVS \in P(E) \times P(E)$ appelé axiome
- un ensemble fini PVS de productions.

Par la suite, on note une telle grammaire par

$$GVS = (V_{TVS}, SVS, PVS, E).$$

Production

Une production est un élément d'une application finie notée $\rightarrow$ entre les éléments de E d'une part et les éléments de $V_{TVS} \times P(E) \times P(E)$.

Les productions sont de la forme :

$$i \rightarrow a[i_1,i_2,\ldots,i_m][j_1,j_2,\ldots,j_n]$$

ou

$$i \rightarrow a[\emptyset][\emptyset]$$

avec

$$i \in E,\ a \in V_{TVS},\ [i_1,i_2,\ldots,i_m] \in P(E),$$
$$[j_1,j_2,\ldots,j_n] \in P(E) \text{ et } [i_1,i_2,\ldots,i_m] \cap [j_1,j_2,\ldots,j_n] = \emptyset$$

On appelle $[i_1,i_2,\ldots,i_m]$ les validations de la règle numéro i et $[j_1,j_2,\ldots,j_n]$ les saturations de la règle numéro i.

Relation $\overset{}{\Rightarrow}_{GVS}$

On dit que $\alpha \overset{}{\Rightarrow}_{GVS} \beta$ $(\alpha,\beta \in V^*_{TVS} \times P(E) \times P(E))$ si et seulement s'il existe une chaine $\varphi \in V^*_{TVS}$ et

$[k_1,\ldots,k_r,i,k_{r+2},\ldots,k_q][s_1,s_2,\ldots,s_q] \in P(E) \times P(E)$ tels que

$\alpha = \varphi[k_1,\ldots,k_r,i,k_{r+2},\ldots,k_q][s_1,s_2,\ldots,s_q]$

$\beta = \varphi\, a[j_1,j_2,\ldots,j_q][s_1,s_2,\ldots,s_q,\ldots,s_r]$

$i \to a[j_1,j_2,\ldots,j_1][s_{q+1},\ldots,s_r]$

étant une production de GVS

avec $[s_1,s_2,\ldots,s_r] = [s_1,s_2,\ldots,s_q] \cup [s_{q+1},\ldots,s_r]$

et

$$[j_1,j_2,\ldots,j_q] = [j_1,j_2,\ldots,j_1] \cap \overline{[s_1,s_2,\ldots,s_r]}$$

On appelle *dérivation directe* une telle relation.

Dérivation : relation transitive $\alpha \overset{*}{\Rightarrow}_{GVS} \beta$

On dit que $\alpha \overset{*}{\Rightarrow}_{GVS} \beta$ $(\alpha,\beta \in V^*_{TVS} \times P(E) \times P(E))$ si et seulement s'il existe une suite finie $\alpha_0, \alpha_1, \ldots, \alpha_n$ $(n > 0)$ telle que

$\alpha_0 \overset{}{\Rightarrow}_{GVS} \alpha_1 \overset{}{\Rightarrow}_{GVS} \alpha_2 \overset{}{\Rightarrow}_{GVS} \cdots \overset{}{\Rightarrow}_{GVS} \alpha_n$

avec $\alpha = \alpha_0$ et $\beta = \alpha_n$.

Langage défini par une grammaire GVS

On appelle langage défini par GVS, que l'on note L(GVS), le sous-ensemble de V^*_{TVS} tel que

$$L(GVS) = \{\alpha / SVS \overset{*}{\Rightarrow}_{GVS} \alpha\ [\emptyset][s_1,s_2,\ldots,s_q],$$
$$\alpha \in V^*_{TVS},\ [s_1,s_2,\ldots,s_q] \in P(E)\}.$$

Exemple :

$GVS = (V_{TVS}, SVS, PVS, E)$

$V_{TVS} = \{a,b,c,d,f,y,z,t,u\}$

$E = \{1,2,3,4,5,6,7,8,9,10\}$

$SVS = [1,2][\emptyset]$

PVS :

1 → f[2][6,10]
2 → a[3][ø]
3 → b[3,4][ø]
4 → c[5,6][ø]
5 → d[7][ø]
6 → e[5][9]
7 → y[7,8][ø]
8 → z[9,10][ø]
9 → t[ø][ø]
10 → u[ø][ø]

Dérivations :

$[1,2][ø] \Rightarrow f[2][6,10] \Rightarrow fa[3][6,10] \Rightarrow \ldots \Rightarrow fab^n[3,4][6,10]$
$\Rightarrow fab^n c[5][6,10] \Rightarrow fab^n cd[7][6,10] \Rightarrow \ldots$
$\Rightarrow fab^n cdy^m[7,8][6,10] \Rightarrow fab^n cdy^m z[9][6,10]$
$\Rightarrow fab^n cdy^m zt[ø][6,10]$

$[1,2][ø] \Rightarrow a[3][ø] \Rightarrow \ldots \Rightarrow ab^n[3,4][ø] \Rightarrow ab^n c[5,6][ø]$
$\Rightarrow ab^n cd[7][ø] \Rightarrow \ldots \Rightarrow ab^n cdy^m[7,8][ø]$
$\Rightarrow ab^n cdy^m z[9,10][ø] \Rightarrow ab^n cdy^m zt[ø][ø]$
$\Rightarrow ab^n cdy^m zu[ø][ø]$

$[1,2][ø] \Rightarrow \ldots \Rightarrow ab^n c[5,6][ø] \Rightarrow ab^n ce[5][9]$
$\Rightarrow ab^n ced[7][9] \Rightarrow \ldots \Rightarrow ab^n cedy^m[7,8][9]$
$\Rightarrow ab^n cdey^m z[10][9] \Rightarrow ab^n cedy^m zu[ø][9]$

$L(GVS) = \{fab^n cdy^m zt,\ ab^n cdy^m zt,\ ab^n cdy^m zu,$
$ab^n cedy^m zu\}$ avec $n,m > 0$.

Définitions

- VAL(i), SAT(i)

Si la ième production de GVS est de la forme :

$i \rightarrow a[i_1,i_2,\ldots,i_m][j_1;j_2,\ldots,j_n]$ alors

$VAL(i) = [i_1,i_2,\ldots,i_m]$

$SAT(i) = [j_1,j_2,\ldots,j_n]$

- VALID, SATUR

Ce sont deux applications de P(E) × P(E) dans P(E).

Soit VALSAT ∈ P(E) × P(E) :

VALSAT = $[i_1,i_2,\ldots,i_m][j_1,j_2,\ldots,j_n]$, alors

VALID(VALSAT) = $[i_1,i_2,\ldots,i_m]$

SATUR(VALSAT) = $[j_1,j_2,\ldots,j_m]$.

II.2.2.3.3. Résultats sur les grammaires à validation-saturation

On montre qu'il existe une grammaire d'états finis G équivalente à Gvs et que le nombre de règles de Gvs est inférieur ou égal à celui de G.

Soit l(Gvs) le langage formé par les α appartenant à un sous-ensemble de $Vtvs^*$ tel que Svs $\overset{*}{\underset{Gvs}{\Rightarrow}}$ α. On montre qu'il existe un automate d'états finis acceptant l(Gvs).

II.2.2.4. Définition de la concaténation I ||J

Soit I et J (I,J ∈ P(E)), on définit I|| J de la manière suivante :

I ||J = ∪ [i ||j] ∀ i ∈ I et ∀ j ∈ J.

Nous étendons la définition à I ||[ø] = [ø] et [ø] ||J = [ø].

Exemple :

Soit i = [1,2,3] et j = [6,7].

On a alors :

i ||j = [16,26,36,17,27,37]

j ||i = [61,62,63,71,72,73].

Soit GvsA la grammaire engendrant le langage l(GvsA) dont les mots peuvent se mettre sous la forme W_1W_3.

GvsA = (VtA, SvsA, PvsA, EA)

EA = {1,2...n}

SvsA = $[i_1,i_2,\ldots,i_q][\emptyset]$

i_j ∈ EA ∀ j ∈ [1,q].

Les règles de production sont de la forme :

$i \rightarrow a[i_1,i_2,\ldots,i_p][s_1,s_2,\ldots,s_q]$ $i_j \in EA$, $s_j \in EA$ et $a \in VtA$

si elles sont non terminales, sinon elles sont de la forme :

$i \rightarrow a[\emptyset][\emptyset]$.

Soit GvsB la grammaire engendrant le langage L(GvsB) dont les mots peuvent se mettre sous la forme W_2.

GvsB = (VtB, SvsB, PvsB, EB)

EB = (n+1, n+2, ...,n+m)

SvsB = $[j_1,j_2,\ldots,j_r][\emptyset]$

$j_i \in EB \ \forall \ i \in [1,r]$.

Les règles de production sont de la forme :

$j \rightarrow b[j_1,j_2,\ldots,j_k][s_1,s_2,\ldots,s_1]$ $j_i \in EB$, $s_i \in EB$ et $b \in VtB$

si elles sont non terminales, sinon :

$j \rightarrow b[\emptyset][\emptyset]$.

Soit GvsC la grammaire engendrant le langage L(GvsC) dont les mots peuvent se mettre sous la forme $W_1W_2W_3$.

GvsC = (VtC, SvsC, PvsC, EC)

VtC = Vt ∪ VtB

EC = {1,2,...,n} ∪ (EB ||EA)

valid(SvsC) = valid(SvsA) ∪ V avec V = valid(SvsB)|| valid(SvsA)

satur(SvsC) = [∅].

PvsC se construit de la manière suivante :
pour chaque règle i ∈ PvsA, on construit à partir de toutes les règles non terminales j ∈ PvsB les règles suivantes :

- si j ∈ valid(SvsB) alors j ||i → b[val(j) ||i][(sat(j)|| i) ∪ S] avec S = valid(SvsB ||EA.
- si j ∉ valid(SvsB) et est non terminale, alors j|| i → b[val(j)|| i] [sat(j) ||i].

Pour les règles terminales, j ∈ PvsB, on construit les règles j|| i → b[i][∅] pour tous les i ∈ PvsA.

A partir de toutes les règles non terminales i $\in$ PvsA, on construit :
i $\rightarrow$ a[val(i) $\cup$ (valid(SvsB) $\|$ val(i))][sat(i)]
puis on ajoute toutes les règles terminales de PvsA.

Remarques :

1) S sert à interdire l'insertion de plusieurs mots W_2 dans $W_1W_2W_3$ Si on voulait construire une grammaire engendrant un langage dont les mots W_1W_3 comporteraient éventuellement une ou plusieurs insertions de chaînes W_2, il suffirait de reprendre GvsC en supprimant S dans les règles j $\|$ i, dans le cas où j $\in$ valid(SvsB).

2) Dans GvsC, il existe n sous-grammaires semblables qui engendrent W_2. Ceci étant dû au fait que les grammaires à validation-saturation recalculent à chaque état l'ensemble des validations des règles applicables. Si ces grammaires conservaient les validations de la règle i au cours de la génération de W_2 et appliquaient ces validations à la fin de cette génération, alors une seule sous-grammaire suffirait.

On peut remarquer que le phénomène de pile apparaît dans la génération de ces grammaires, c'est pour cela que nous avons utilisé dans MULTPIAF un mécanisme analogue (conservation de l'adresse de retour) qui nous permet de n'utiliser qu'un seul automate pour accepter W_2.

Exemple :

L(GvsA) = {abcde, abf} et L(GvsB) = {xyz}
GvsA = (VtA, SvsA, PvsA, EA)
VtA = {a,b,c,d,e,f}
EA = {1,2,3,4,5,6}
SvsA = [1][$\emptyset$]
PvsA : 1 $\rightarrow$ a[2][$\emptyset$]
2 $\rightarrow$ b[3,6][$\emptyset$]
3 $\rightarrow$ c[4][$\emptyset$]
4 $\rightarrow$ d[5][$\emptyset$]
5 $\rightarrow$ e[$\emptyset$][$\emptyset$]
6 $\rightarrow$ f[$\emptyset$][$\emptyset$]

```
GvsB = (VtB, SvsB, PvsB, EB)
VtB = {x,y,z}
EB = {7,8,9}
SvsB = [7][ø]
PvsB : 7 → x[8][ø]
       8 → y[9][ø]
       9 → z[ø][ø]

GvsC = (VtC, SvsC, PvsC, EC)
VtC = {a,b,c,d,e,f,x,y,z}
EC = {1,2,3,4,5,6,71,81,91,72,82,92,73,83,93,74,84,94,75,85,95,76,86,96}
SvsC = [1,71][ø]
PvsC : 71 → x[81][71,72,73,74,75,76]
       81 → y[91][ø]
       91 → z[1][ø]
        1 → a[2,72][ø]
       72 → x[82][71,72,73,74,75,76]
       82 → y[92][ø]
       92 → z[2][ø]

        2 → b[3,6,73,76][ø]
       73 → x[83][71,72,73,74,75,76]
       83 → y[93][ø]
       93 → z[3][ø]
       76 → x[86][71,72,73,74,75,76]
       86 → y[96][ø]
       96 → z[6][ø]

        3 → c[4,74][ø]
       74 → x[84][71,72,73,74,75,76]
       84 → y[94][ø]
       94 → z[4][ø]

        4 → d[5,75][ø]
       75 → x[85][71,72,73,74,75,76]
       85 → y[95][ø]
       95 → z[5][ø]

        5 → e[ø][ø]

        6 → f[ø][ø]
```

L(GvsC) = {abcde, abf, xyzabcde, xyzabf, axyzbcde, axyzbf, abxyzcde, abxyzf, abcxyzde, abcdxyze}.

Il est donc possible de construire une grammaire à validation-saturation engendrant $W_1W_2W_3$; ce langage est d'états finis, il existe un automate d'états finis acceptant ce même langage.

Mots de la forme W_1' ou W_2'

Si GvsA' est la grammaire engendrant W_1' : GvsA' = (VtA', SvsA', PvsA', EA')
GvsB' la grammaire engendrant W_2' : GvsB' = (VtB', SvsB', PvsB', EB')
alors GvsC' engendrant des mots de la forme W_1' ou W_2' sera telle que :

GvsC' = (VtC', SvsC', PvsC', EC')

avec :

VtC' = VtA' ∪ VtB'
EC' = EA' ∪ EB'
SvsC' = SvsA' ∪ SvsB'
PvsC' = PvsA' ∪ PvsB'.

Il est donc possible de trouver une grammaire à validation-saturation engendrant des mots de la forme W_1' ou W_2'. Le langage est donc d'états finis.

Conclusion : LM inclusdans les langages d'états finis.
D'après les résultats précédents, on peut dire que LM est un langage d'états finis.

II.2.3. APPELS RECURSIFS OU CROISES

Pour pouvoir étudier ces appels, il est nécessaire de rappeler que le réseau d'automates d'états finis est un réseau dont chaque noeud est matérialisé par un automate et une liste d'appels. Pour plus de commodité, nous confondons le nom du noeud et le nom de l'automate. Dire que A appelle B, c'est dire que du noeud A qui contient l'automate A et dont la liste d'appel contient le noeud B, on peut accéder à ce noeud qui contient l'automate B.

Un appel peut donc se comprendre comme un passage d'un noeud à un autre, c'est-à-dire un changement d'automate et de liste d'appel. Pour contrôler ces appels et pour permettre le retour arrière, le système possède une pile qui contient à chaque niveau le nom d'un noeud et l'état courant de l'automate associé à ce noeud (si l'automate est nul, l'état courant est évidemment nul).

Nous avons vu que lorsqu'il n'y a pas d'appels récursifs ou croisés, le système était équivalent à un automate d'états finis. Dans ce cas, la pile sert uniquement à contrôler le fonctionnement en subroutine ou en cascade des automates d'états finis.

Si l'on considère maintenant un automate s'appelant lui-même ou deux automates réalisant des appels croisés (du point de vue réseau cela veut dire qu'il existe une boucle), alors la pile jouera le même rôle que celle d'un automate à pile, c'est-à-dire qu'elle va permettre la récursivité et l'on peut donc parler d'appels récursifs.

Le problème revient donc à étudier ce que devient un réseau d'automates d'états finis lorsqu'on adjoint une pile.

Nous utiliserons le formalisme décrit par Hopcroft-Ullmann.

II.2.3.1. Rappel sur les grammaires d'états finis et hors contexte

Grammaire d'états finis

- Une grammaire d'états finis $G = (V_T, V_N, S, P)$ avec

  V_T : vocabulaire terminal

  V_N : vocabulaire non terminal tel que :

  $V_T \cap V_N = \emptyset$, $V_T \cup V_N = V$

  S : axiome avec $S \in V_N$

  P : ensemble fini de productions de la forme :

  $A \rightarrow a$ ou $A \rightarrow aB$ avec $A,B \in V_N$, $a \in V_T$.

- Relation $\overset{}{\underset{G}{\Rightarrow}}$ entre deux chaînes

  On dit que $\alpha \overset{}{\underset{G}{\Rightarrow}} \beta (\alpha, \beta \in V^+)$ si et seulement s'il existe une chaîne $\varphi \in V_T^*$ et un non terminal A tel que :

  $\alpha = \varphi A$

  $\beta = \varphi aB$ (ou $\beta = \varphi a$)

  $A \rightarrow aB$ (ou $A \rightarrow a$) étant une production de G.

  On appelle *dérivation directe* une telle relation.

- Relation $\overset{*}{\underset{G}{\Rightarrow}}$ entre deux chaînes

  On dit que $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta (\alpha, \beta \in V^+)$ s'il existe une suite finie $\alpha_0, \alpha_1, \ldots, \alpha_n$ $(n > 0)$ telle que

  $\alpha_0 \overset{}{\underset{G}{\Rightarrow}} \alpha_1 \overset{}{\underset{G}{\Rightarrow}} \alpha_2 \overset{}{\underset{G}{\Rightarrow}} \alpha_n$ avec $\alpha = \alpha_0$ et $\beta = \alpha_n$

  On appelle *dérivation* une telle relation.

- Langage défini par une grammaire G

  On appelle langage défini par G, que l'on note L(G), le sous-ensemble de V_T^* tel que :

  $$L(G) = \{\alpha / S \overset{*}{\underset{G}{\Rightarrow}} \alpha,\ \alpha \in V_T^*\}.$$

Grammaires hors contexte

On rappelle qu'une grammaire hors contexte est un quadruplet :

$$G = (V_T, V_N, S, P)$$

où V_T est le vocabulaire terminal

V_N est le vocabulaire auxiliaire

$S \in V_N$ est l'axiome

P est l'ensemble des règles de la forme $A \rightarrow \omega$ où $A \in V_N$ et $\omega \in V^+$

$$(V = V_T \cup V_N \; ; \; V_T \cap V_N = \emptyset)$$

Forme normale de Greibach :

Tout langage hors contexte peut être engendré par une grammaire dans laquelle les règles sont de la forme

$A \rightarrow a\varphi$ où $A \in V_N$

$a \in V_T$

$\varphi \in V_N^*$

(si $\varphi = \epsilon$ alors $A \rightarrow a$ est une règle terminale).

II.2.3.2. Démonstration

Soit une grammaire GA d'états finis GA = (VtA, VnA, SA, PA).

Les règles de productions sont de la forme :

$X \rightarrow aY$ si elles sont non terminales avec $(X,Y \in VnA)$ et $a \in VtA$

$X \rightarrow a$ si elles sont terminales avec $X \in VnA$ et $a \in VtA$.

A toute grammaire GA, on peut associer un automate d'états finis AGA et inversement.

Si dans le réseau on a AGA appelle AGA, le langage reconnu par le réseau sera engendré par la grammaire suivante, appelée GE, construite à partir de GA : GE = (VtA, VnA, SE, PE).

Les règles de productions de PE seront construites de la manière suivante :

Une règle non terminale de GA deviendra : $X \rightarrow aY$ et $X \rightarrow aSAY$.

Les règles terminales sont inchangées.

SE = SA.

Nous voyons que pour une règle non terminale de GA, nous construisons deux règles : l'une d'états finis et l'autre étant hors contexte.

Si AGA appelle les automates AG_1, AG_2, ..., AG_P dont les grammaires associées sont $G_1, G_2, \ldots, G_P$ d'axiomes $S_1, S_2, \ldots, S_P$, alors on construira dans GE à partir de $X \rightarrow aY$ les règles suivantes :
$X \rightarrow aY$, $X \rightarrow aS_1Y$, ..., $X \rightarrow aS_PY$.

La grammaire GE peut donc être hors contexte et les langages reconnus par de tels réseaux sont des langages hors contexte.

II.2.3.3. Exemples

Exemple 1

Soit une grammaire d'états finis GA engendrant ab

GA = (VtA, VnA, A, PA)

VtA = {a,b}

VnA = {A,B}

PA : A → aB

B → b

Si l'on construit un réseau tel que l'automate AGA appelle AGA, alors la grammaire équivalente GE s'écrit :

GE = (VtA, VnA, A, PE)

avec PE : A → aB

A → aAB

B → b

GE engendre a^nb^n qui est un langage hors contexte.

Exemple 2

Soit le réseau formé des automates AGA et AGB.

AGA reconnaît ab et AGB reconnaît ba.

automate initial appelé AGA, AGB

AGA appelle AGA, AGB

AGB appelle AGB, AGA.

Soit GA et GB les grammaires associées à AGA et AGB.

GA = (VtA, VnA, A, PA)

VtA = {a,b}

VnA = {A,B}

PA : A → aB

B → b

GB = (VtA, VnB, C, PB)

VnB = {C,D}

PB : C → bD

D → a

Soit GE la grammaire équivalente au réseau :

GE = (VtA, VnE, SE, PE)

VnE = {SE,A,B,C,D}

PE : SE → A

SE → C

A → aB, A → aAB, A → aCB

B → b

C → bD, C → bAD, C → bCD

D → a

Cette grammaire engendre des mots qui ont autant de a que de b. Ce langage est hors contexte.

Il ne semble pas qu'un réseau contrôlé par MULTPIAF puisse reconnaître tous les langages hors contexte ; nous n'avons pas trouvé de réseau pouvant reconnaître le langage hors contexte donné ci-dessous.

Exemple 3

Soit le langage formé des palindromes impairs sur le vocabulaire terminal a,b,c avec c symbole central.

Soit le réseau formé des trois automates AGA, AGB et AGC qui reconnaissent respectivement aa, bb et c.

On a :

automate initial appelle AGA, AGB

AGA appelle AGA, AGB et AGC

AGB appelle AGA, AGB et AGC.

Soit GA, GB et GC les grammaires associées à AGA, AGB et AGC.

GA = (VtA, VnA, A, PA)

VtA = {a}

VnA = {A,B}

PA : A → aB

B → a

GB = (VtB, VnB, C, PB)

VtB = {b}

VnB = {C,D}

PB : C → bD

D → b

GC = (VtC, VnC, E, PC)

VtC = {c}

VnC = {E}

PC : E → c.

Soit GE la grammaire équivalente au réseau :

GE = (VtE, VnE, SE, PE)

VtE = {a,b,c}

VnE = {SE, A, B, C, D, E}

PE : SE → A

SE → C

A → aB, A → aAB, A → aCB, A → aEB

B → a

C → bD, C → bAD, C → bCD, C → bED

D → b

E → c.

Cette grammaire engendre bien les palindromes impairs de la forme $vc\tilde{v}$ avec v chaîne de caractères formée sur a,b ; mais GE engendre aussi les palindromes pairs de la forme $v\tilde{v}$. Il est impossible d'obliger l'apparition du caractère c.

Or, il existe une grammaire hors contexte engendrant les palindromes impairs et seulement eux.

Soit G cette grammaire :

G = (Vt, Vn, S, P)

Vt = {a,b,c}

Vn = {S,A,B}

P : S → aSA

S → bSB

S → c

A → a

B → b.

II.2.3.4. Conclusion

Le réseau d'automates d'états finis contrôlé par MULTPIAF est capable de reconnaître des langages plus puissants que les langages d'états finis mais ne paraît pas permettre d'analyser tous les langages hors contexte.

Il ne semble donc pas possible de dire que la version actuelle de MULTPIAF est équivalente à un réseau récursif d'automates (Woods).

II.2.3.5. Exemples de langages hors contexte

Exemple d'appels récursifs

Nous reprenons le langage utilisé dans l'exemple 1.

Soit $a^n b^n$ ce langage.

Pour accepter ce langage, nous allons utiliser un TGEF A acceptant ab. Dans le réseau, nous aurons A appelle A.

Nous donnons ici le fonctionnement du surautomate analysant a^3b^3. Ce fonctionnement sera représenté par le tableau suivant où la pile est représentée par une liste bornée à droite et infinie à gauche.

A0 est l'automate initial.

Etat	Ligne d'entrée	Caractère analysé	Pile
0	aaabbb		A0
1	aabbb	a	AA0
2	abbb	a	AAA0
3	bbb	a	AAAA0
4	bb	b	AAA0
5	b	b	AA0
6		b	A0

Le surautomate est dans un état final.

Une telle configuration ne reconnaîtra pas un mot tel que : aababb car il est possible d'indiquer qu'à un état I un automate A ne peut appeler qu'une seule fois un automate de sa liste d'appel.
Ce qui veut dire, dans cet exemple, que lorsque le surautomate commence à dépiler, il n'a plus le droit de réempiler.

Après avoir analysé le premier (b) la pile sera dans l'état : AA0 et A devra rappeler A pour analyser a ; ce qui lui est interdit.

Cette option s'appelle l'appel multiple ou non multiple ; dans ce cas, l'option sera non multiple.

Exemple d'appels croisés

Nous reprenons le langage utilisé dans l'exemple 2. Ce langage est formé de mots sur le vocabulaire terminal a,b et les mots contiennent autant de a que de b.

Nous construisons deux automates A, B reconnaissant respectivement ab et ba.

Le réseau sera le suivant :

A0 automate initial
A0 liste d'appel : (A,B)
A liste d'appel : (A,B)
B liste d'appel : (B,A).

Pour reconnaître bbabaaab, le fonctionnement sera le suivant :

Etat	Ligne d'entrée	Caractère analysé	Pile
0	bbabaaab		AO
1	babaaab	b	BAO
2	abaaab	b	BBAO
3	baaab	a	BAO
4	aaab	b	BBAO
5	aab	a	BAO
6	ab	a	AO
7	b	a	AAO
8		b	AO

CHAPITRE 3

DESCRIPTION DE L'EDITEUR DE PIAF ET DE MULTPIAF

II.3.1. INTRODUCTION

Le système MULTPIAF étant une extension du TGEF du système PIAF, possède toutes les options de ce dernier, plus d'autres que nous décrirons plus tard.

Dans ces deux systèmes, pour choisir ces options, l'utilisateur répond à des questions posées par le système.

Nous allons, tout d'abord, rappeler quelles sont les options disponibles dans le TGEF et ensuite décrire précisément celles qui concernent MULTPIAF.

II.3.2. RAPPELS SUR L'EDITEUR DE PIAF

Pour pouvoir fonctionner, le TGEF a besoin d'un dictionnaire et d'une grammaire. Il possède deux compilateurs incrémentiels qui permettent à l'utilisateur à tout moment de créer ou de modifier le dictionnaire ou la grammaire.

En ce qui concerne les données, le TGEF peut fonctionner soit en prenant les données au terminal (mode INTERRO), soit en les prenant sur un fichier (mode EXPLOIT). Le premier mode est intéressant pour mettre une grammaire ou un dictionnaire au point car le TGEF analyse immédiatement la phrase qu'on lui donne au terminal, l'autre mode permet de faire du traitement proprement dit.

A l'initialisation, l'éditeur est dans un état principal appelé état "$". Les commandes disponibles sur cet éditeur sont les suivantes :

- GRAMM : accès à la grammaire
- DICT : accès au dictionnaire
- INTERRO : traitement sous le mode interactif
- EXPLOIT : traitement sur fichier des données.

GRAMM :

Cette commande permet la création, la consultation et la modification de la grammaire.

- La création se fait lors du premier appel à "GRAMM" si la grammaire n'existe pas. Si ce n'est pas le premier appel, alors la grammaire est automatiquement chargée dans le système.

- La consultation : le compilateur de la grammaire est en même temps décompilateur. Si l'on interroge une règle ou un modèle, cette règle ou ce modèle est décompilé pour apparaître dans un langage clair (langage utilisé pour la création).

- La modification : lorsque l'utilisateur modifie ou introduit un modèle ou une règle, ce modèle ou cette règle est immédiatement compilé par le compilateur : ce qui empêche d'introduire des incohérences dans la grammaire. Le langage nécessaire pour manipuler les éléments de la grammaire est simple.

La suppression d'un élément de la grammaire ne peut se faire que si cet élément ne fait pas référence à un autre élément ou réciproquement (référence entre éléments du dictionnaire et modèles entre règles et modèles).

DICT :

Cette commande permet de créer, de consulter ou de modifier le dictionnaire.

- Création : elle a lieu lors du premier appel à DICT, si le dictionnaire n'existe pas.

- Consultation : quand on est sous l'état DICT, il est possible de consulter des éléments du dictionnaire, de lister une partie ou tout le dictionnaire, ou l'arbre d'accès sur le terminal ou sur l'imprimante.

- Modification : il est possible de modifier ou d'ajouter un mot dans le dictionnaire ; le compilateur du dictionnaire compilera immédiatement ce mot en vérifiant la cohérence des informations (éléments faisant référence à un modèle inexistant, par exemple). Si l'on rentre un mot qui existe déjà dans le dictionnaire, alors le système posera les questions ad hoc pour interdire l'insertion d'informations fausses dans le dictionnaire.

Il est possible aussi de réorganiser l'arbre d'accès, calculé à partir des fréquences d'utilisations des éléments du dictionnaire, pour optimiser l'identification,et de trier le dictionnaire s'il présente un désordre physique trop important, celui-ci provenant du fait que chaque nouvel élément est ajouté en fin de dictionnaire.

Un système de chaînage permet l'accès dans l'ordre alphabétique des éléments.

Si le dictionnaire n'existe pas, le système envoie un message d'erreur et revient immédiatement sous l'état ($) pour que l'utilisateur puisse créer une grammaire et un dictionnaire sous GRAMM.

Le mode INTERRO :

En fait tout le traitement effectué par le TGEF est conversationnel ; seulement dans ce cas-là, les données sont rentrées au terminal et les résultats apparaissent immédiatement sur ce même terminal. Ce mode est très pratique pour mettre au point un dictionnaire et une grammaire, car l'utilisateur peut immédiatement visualiser un résultat et vérifier que l'analyse s'est correctement accomplie. Si ce n'est pas le cas, alors l'utilisateur a le moyen de modifier la grammaire ou le dictionnaire pour obtenir une analyse correcte.

En cas d'analyse impossible, le système signale à l'utilisateur qu'il y a une erreur et affiche au terminal ses tentatives d'analyse ; il se place ensuite dans un état permettant à l'utilisateur de se mettre dans l'état GRAMM, DICT, ou de modifier la phrase d'entrée. Selon le choix de l'utilisateur, le système reprendra l'analyse à l'endroit où il a échoué ou en début de phrase.

Le mode EXPLOIT :

Ce mode sert particulièrement à traiter des données sur fichier et à écrire les résultats sur fichier. Il permet de donner les spécifications du fichier de données (nom, type) et celles du fichier de résultats.

Dans le cas de l'application au braille, nous avons déterminé d'autres spécifications indispensables à l'édition : format des pages (longueur des lignes et de la page), initialisation du compteur de numérotation des pages et dans le cas d'impression en interpoint[1] la ventilation des pages paires et impaires sur deux fichiers différents.

---

(1) Interpoint : il est possible en braille d'imprimer les deux côtés d'une page en intercalant l'embossage des points.

Comme dans le mode INTERRO, s'il existe dans le fichier de données une chaîne de caractères non analysable, le système arrête son analyse, signale une erreur au terminal de l'utilisateur et lui laisse la possibilité de modifier le dictionnaire, la grammaire ou la chaîne de caractères qui empêche l'analyse. Le dialogue entre l'utilisateur et le système n'apparaitra pas dans le fichier de sortie.

Cette méthode est intéressante car elle permet de corriger des données tout en les traitant (correction de fautes de frappe par exemple) ; ce qui est très important pour traiter de gros fichiers.

II.3.3. EDITEUR DU SYSTEME MULTPIAF

Pour pouvoir implanter la surgrammaire, nous avons ajouté une commande de plus à l'éditeur. Cette commande est appelée surgramm. Nous savons que MULTPIAF utilise plusieurs TGEF, la différence entre l'éditeur de MULTPIAF et celui de PIAF ne réside pas seulement dans le simple ajout d'une commande, mais dans la modification de toutes les commandes existantes de l'éditeur de PIAF.

Chaque automate est désigné par un identificateur appelé nom d'automate, cet identificateur désigne le noeud du réseau qui porte l'automate plutôt que le couple (dictionnaire, grammaire) ; en effet, plusieurs automates peuvent travailler avec le même dictionnaire et la même grammaire, mais avoir des positions différentes dans le réseau où un automate peut être le sous-automate d'un autre automate (ensemble de validations initiales différentes).

Nous allons voir, tout d'abord, de manière succincte, quelles sont les modifications apportées aux autres commandes ; ensuite, nous étudierons précisément surgramm, son fonctionnement et le langage utilisé dans cette commande.

II.3.3.1. Modifications sur DICT et GRAMM

Du fait que le système MULTPIAF peut utiliser plusieurs dictionnaires et grammaires, il est important que l'utilisateur puisse manipuler ces objets.

Les couples (dictionnaire, grammaire) portent chacun le nom d'un automate, ceci permet de les désigner plus facilement. Ces noms d'automates seront donc les seuls identificateurs connus de l'utilisateur. Il est donc impossible de créer ou de modifier un dictionnaire ou une grammaire dont le nom n'est pas celui d'un automate appartenant au réseau ; ceci peut être considéré comme un inconvénient mais facilite la tâche de l'utilisateur quand il s'agit de désigner les objets.

Pour pouvoir modifier un dictionnaire ou une grammaire, l'utilisateur doit commencer par sélectionner ce dictionnaire ou cette grammaire. Pour cette raison, dans DICT et GRAMM, le système commence par demander quelle grammaire ou quel dictionnaire l'utilisateur désire ; ce dernier les sélectionne en répondant par le nom d'automate correspondant, si ces objets sont présents nous sommes alors revenus au cas de la commande DICT ou GRAMM décrite plus haut.

II.3.3.2. Modifications sur INTERRO et EXPLOIT

Ce sont sur ces deux commandes qu'il existe le plus de modifications. En effet, ces deux commandes correspondent dans PIAF au fonctionnement du TGLF ; ici, elles correspondent au fonctionnement du réseau et de l'ensemble des automates. Nous consacrerons donc un paragraphe particulier pour exposer les modifications apportées aux algorithmes de fonctionnement.

Dans ce paragraphe, nous étudions uniquement le point de vue de l'édition.

Au lancement de ces deux commandes, le système va chercher le noeud initial du réseau, si ce noeud n'existe pas ou si le réseau est inexistant, le système lancera un message d'erreurs et reviendra sous l'état ($). L'utilisateur pourra alors indiquer le noeud initial ou implanter le réseau à l'aide de la commande SURGRAMM (définie au paragraphe suivant) avant de relancer le mode INTERRO ou EXPLOIT.

Dans le cas d'EXPLOIT, nous avons vu qu'il peut y avoir un fichier de données et un fichier de résultats ; actuellement, nous n'avons prévu qu'un seul fichier de résultats, ceci peut être gênant particulièrement dans le cas d'une cascade d'automates où l'on veut des résultats intermédiaires. Avec de petites modifications, il est simple de pallier à cet inconvénient.

En cas d'échec pendant l'analyse des données, le système se met dans un état où l'utilisateur a non seulement accès à GRAMM, DICT, mais aussi à SURGRAMM pour changer la structure du réseau. Il peut modifier également les données à analyser.

II.3.4. DESCRIPTION ET LANGAGE DE SURGRAMM

Cette commande permet d'implanter, de modifier ou de consulter la structure du réseau. Cette structure est sauvegardée automatiquement sur un fichier de manière à la conserver d'une session sur l'autre. Au lancement des commandes DICT, GRAMM, INTERRO et EXPLOIT, la structure, si elle existe, sera chargée.

Pour chaque noeud susceptible d'être créé, on garde les informations suivantes :

1) Un nom de noeud sous forme d'un identificateur de huit caractères au plus. Par la suite, nous appellerons cet identificateur nom d'automate car nous confondons le nom du noeud et le nom de l'automate porté par ce noeud.

2) Un ensemble de validations initiales de l'automate porté par le noeud (cet ensemble peut être vide si l'automate est nul).

3) Un nom d'automate représentant un dictionnaire et une grammaire. Ce nom peut être le nom du noeud ou celui d'un autre noeud. Nous rappelons en effet que plusieurs automates peuvent travailler avec le même dictionnaire et la même grammaire. Pour des questions d'optimisation de place mémoire, un seul automate possède ce dictionnaire et cette grammaire et les autres y font référence.
Le dictionnaire et la grammaire auront le même nom que l'automate qui les porte.

4) Un certain nombre d'indicateurs permettant de spécifier le comportement de l'automate, particulièrement en ce qui concerne la transduction.

5) Une zone de 256 caractères qui peut être soit une zone de données, soit une zone de résultats selon les indications de l'utilisateur.

6) Une liste d'appels qui est une liste de nom d'automates décrivant les transitions possibles à partir du noeud courant.

7) Un certain nombre de chaînages nécessaires au bon fonctionnement du surautomate. Ces chaînages sont indiqués par l'utilisateur en donnant les noms d'automates correspondants.

Lorsque l'on veut créer un réseau, il existe toujours un noeud initial qui n'a pas besoin d'être créé. Ce noeud s'appelle "ENTREE". Nous savons que ce noeud est vide ; seule sa liste d'appel et sa zone de données-résultats sont utilisées.

Lorsque le surautomate est dans un état initial, il est positionné sur ce noeud et va chercher dans la liste d'appel le prochain noeud. Si cette liste est vide, alors il ne peut démarrer.

II.3.4.1. Langage de SURGRAMM

Au lancement de SURGRAMM, le système pose la question suivante[1] :
E : "NOM D'AUTOMATE ET SES CARACTERISTIQUES ?"

La réponse de l'utilisateur aura le format suivant :

*/Champ1/Champ2/Champ3/

Le signe (*) correspond aux différentes commandes auxquelles l'utilisateur peut avoir accès. Ces commandes sont les suivantes :

1) Absence de signe : l'utilisateur veut créer un nouveau noeud.
2) ? : l'utilisateur veut consulter un noeud ou l'ensemble du réseau.
3) \ : l'utilisateur veut supprimer un noeud.
4) * : l'utilisateur veut mettre un automate et un seul dans le noeud initial.

Champ1

Ce champ a le format suivant :

NOM D'AUTOMATE(*)(*)(*)...

---

(1) Nous indiquerons les questions de l'éditeur par (E :) et les réponses de l'utilisateur par (U :).

NOM D'AUTOMATE

L'utilisateur donne le nom d'un noeud existant dans la structure s'il utilise les commandes ?, \ et *, inexistant s'il crée un nouveau noeud.

(*)(*)(*)...

Le signe (*) représente des indicateurs optionnels précisant le fonctionnement de l'automate porté par le noeud. Ces indicateurs sont : (D), (R), (E), (L), (N), (T), (M) et (S). Ils se regroupent en trois classes :

1) (D), (R) et (T) servent à la gestion des différents pointeurs des zones de données-résultats ;
2) (E), (L) et (N) servent à la transduction, c'est-à-dire indiquent comment produire les résultats ;
3) (M) et (S) indiquent les conditions à remplir pour permettre l'appel d'un automate.

L'absence de l'un de ces indicateurs signifie que la valeur de cet indicateur est à faux.

- (D) signifie que l'automate porté par le noeud est chargé de gérer le pointeur de zone de données. Ce pointeur correspond à l'indice de la ligne de données du point de vue de l'analyse. Si plusieurs automates (en subroutine) travaillent sur la même zone de données, pour éviter les conflits, un seul automate gèrera ce pointeur.
- (R) indique que l'automate porté par le noeud gère le pointeur de la zone de résultats. Ce pointeur correspond à l'indice courant de la ligne de résultats. Comme pour (D), si plusieurs automates produisent leurs résultats sur la même zone, un seul automate gère ce pointeur.
- (T) indique que l'automate porté par le noeud doit modifier la longueur de la ligne de données d'un autre automate. Si deux automates travaillent en cascade, la zone de résultats d'un automate est en même temps la zone de données de l'autre automate. Quand le premier automate produit des résultats, il augmente la longueur de la ligne de données du second automate.
  Si deux automates fonctionnent en cascade, l'indicateur (T) est obligatoire pour le premier automate. Il est inutile pour deux automates qui travaillent en subroutine.

- (L) signifie que l'automate porté par le noeud doit chercher ses données soit sur un fichier externe si l'on est en mode exploitation, soit au terminal si l'on est en mode interrogation.

- (E) signifie que l'automate porté par le noeud doit non seulement produire ses résultats dans une zone, mais aussi les sortir soit au terminal, soit sur un fichier selon le cas.

- (N). Cet indicateur est d'une très grande importance car il joue un rôle primordial du point de vue de la transduction. L'automate porté par le noeud, après avoir analysé un mot et étant dans un état final, selon la valeur de cet indicateur, saura s'il doit lui-même produire ses résultats ou laisser ce soin à l'automate qui l'a appelé. Voyons sur l'exemple 2 du chapitre 2.2 l'influence de cet indicateur :
  Soit un automate A pouvant analyser abcde ou abf ; nous ajoutons en plus que la transduction de abcde par A est a'b'c'd'e' et celle de abf est a'b'f'. Soit un automate B reconnaissant xyz et la transduction de xyz par B est x'y'z'. On a dans le réseau : A appelle B ; l'automate B est en subroutine.
  Soit à analyser la chaîne abxyzcde : si A a l'indicateur N et pas B, alors la transduction de cette chaîne sera a'b'x'y'z'c'd'e' sinon si A a l'indicateur N et B aussi, alors la transduction sera x'y'z'a'b'c'd'e'.
  Dans certaines applications, pour pouvoir être analysées, des données doivent avoir une répartition différente de leur répartition initiale.

- (S) autorise l'appel multiple décrit dans le chapitre 2.2, paragraphe 5.

- (M) permet le mode solution multiple décrit dans le chapitre 2.1, paragraphe 1.

Champ2

Ce champ a le format suivant :

VAL**

Il indique la validation initiale de l'automate porté par le noeud. Les signes ** sont remplacés par un nombre de deux chiffres conformément à l'éditeur du TGEF.

Si l'automate est vide, alors VAL** est remplacé par *****.

Il est possible que plusieurs automates aient le même VAL**, VAL00 par exemple ; s'ils ne travaillent pas sur le même dictionnaire et la même grammaire, il n'y aura pas d'ambiguïté car VAL** est lié au dictionnaire et à la grammaire d'un automate.

Champ3

Ce champ a le format suivant :

NOM D'AUTOMATE

Ce champ indique le nom de l'automate qui porte le dictionnaire et la grammaire utilisés par l'automate porté par le noeud. Selon le cas, ce nom d'automate peut être le même que celui du champs1 ou être différent.

II.3.4.2. Description des différentes commandes

Création

/NOM D'AUTOMATE(*)(*)(*).../VAL**/NOM D'AUTOMATE/

Cette commande permet de créer un nouveau noeud dans le réseau. S'il existe déjà un noeud portant le même nom, alors le compilateur refusera cette création, enverra un message d'erreur et reviendra dans l'état principal.

Le troisième champ peut être omis si :

- l'automate porté par le noeud est vide, ce qui est indiqué au compilateur par ***** à la place de VAL
- si l'automate porté par le noeud gère le dictionnaire et la grammaire qu'il utilise, alors le système chargera le dictionnaire et la grammaire à partir de fichiers externes ; si ces fichiers n'existent pas, alors il y aura lancement d'un message d'erreur et la commande sera refusée, ceci venant du fait que l'on donne une validation initiale pour une grammaire qui n'existe pas.

Si ce champ est donné et que le nom de l'automate est différent de celui du noeud, le compilateur vérifie que cet automate existe bien dans le réseau, sinon il y a message d'erreur et la commande est refusée.

E : INVERSE O/N ?

Cette question permet à l'utilisateur d'indiquer si l'automate qu'il est en train de créer doit fonctionner en mode inverse ou non. Ce mode inverse est utilisé pour des automates réversibles tels que le traducteur en braille abrégé ; si l'on traduit de l'intégral vers l'abrégé, c'est le mode

non inverse, et si l'on traduit de l'abrégé vers l'intégral, c'est le mode inverse. Tous les autres TGEF fonctionneront en mode non inverse.

E : NO DE SORTIE ?

L'expérience nous montre que chaque application risque d'avoir une sortie des résultats différente (différentes manières d'utiliser les informations dans le dictionnaire et les résultats de l'analyse).

Il est difficile de prévoir quelles seront les sorties de résultats de nouvelles applications. Pour pouvoir mettre facilement en oeuvre ces nouvelles applications, nous avons mis les procédures de sorties de résultats comme procédures externes aux algorithmes (donc façilement modifiables). Nous attribuons un numéro pour chacune de ces sorties et c'est ce numéro qui est donné ici pour indiquer quelle est la sortie utilisée par le TGEF qu'on est en train de créer. Ces numéros serviront d'identificateurs dans les procédures de sorties de résultats.

E : NOM DE L'AUTOMATE OU L'ON RANGE LES DONNEES ?

La réponse à cette question est fonction de l'indicateur (D).

Si (D) est présent, la réponse indique la zone de données-résultats où l'on range effectivement les données. En effet, un automate peut parfaitement gérer le pointeur de lecture d'une zone de données-résultats d'un autre automate. L'automate porté par le noeud ira chercher ses données dans cette zone de données-résultats. La réponse de l'utilisateur sera donc le nom de l'automate portant cette zone de données-résultats.

Si (D) est absent, l'utilisateur donne alors le nom de l'automate gérant le pointeur de lecture de la zone de données que l'automate porté par le noeud veut utiliser. Si l'automate indiqué par l'utilisateur est inexistant, le système envoie un message d'erreur et repose la question.

Si l'utilisateur répond par un retour-chariot, l'éditeur reviendra dans l'état principal et le noeud sera créé mais incomplet.

E : NOM DE L'AUTOMATE OU L'ON RANGE LES RESULTATS ?

Cette question n'est posée que si l'indicateur (R) est absent. L'utilisateur donne le nom de l'automate qui porte la zone de données-résultats utilisée par l'automate porté par le noeud pour ranger ses résultats.

Si (R) est présent, l'automate porté par le noeud range implicitement ses résultats dans sa propre zone de données-résultats.

En cas de conflits avec la zone de lecture de l'automate porté par le noeud, le système envoie un message d'erreur et repose la question.

E : NOM DE L'AUTOMATE QUI TRAITE LES RESULTATS

Cette question n'est posée que si l'indicateur (T) est présent. L'utilisateur donne le nom de'l'automate dont les données sont les résultats de l'automate porté par le noeud.

E : NOM D'AUTOMATE A APPELER ?

Cette question permet à l'utilisateur de donner le nom des automates qui constitueront la liste d'appel de l'automate porté par le noeud.

Pour cela, l'utilisateur donne un nom d'automate ou de noeud, le système regarde si ce noeud existe dans le réseau, s'il existe, il l'inscrit dans la liste et repose la question ; sinon il envoie un message d'erreur et repose la question.

Pour sortir, l'utilisateur envoie un retour-chariot ; le système reviendra sous l'état principal.

Si l'utilisateur fait précéder le nom d'automate du signe (\), cela veut dire qu'il veut supprimer un noeud de la liste d'appels ; le système vérifie que le noeud est présent avant de le supprimer. Dans le cas contraire il signale l'absence de ce noeud dans la liste d'appels.

Interrogation

Format de la question :

U : ?/NOM D'AUTOMATE/

Format de la réponse :

E : /NOM D'AUTOMATE(*)(*)(*).../VAL**/NOM D'AUTOMATE/

Inverse : (O/N)

N° de sortie : *

Données : NOM D'AUTOMATE

Résultats : NOM D'AUTOMATE

Transferts : NOM D'AUTOMATE

Liste d'appels : NOM D'AUTOMATE, NOM D'AUTOMATE ...

E : VOULEZ-VOUS COMPLETER LE NOEUD ? O/N

U : O ou N

Si l'utilisateur répond par oui (O), l'éditeur reviendra à la question : "nom de l'automate où l'on range les données" de la commande création. L'utilisateur pourra alors compléter le noeud interrogé.

E : NOM D'AUTOMATE A APPELER ?

U : NOM D'AUTOMATE ou Retour-chariot.

L'utilisateur peut compléter la liste d'appels de l'automate demandé. Pour ceci, voir paragraphe précédent.

"Données"indique le nom de l'automate qui gère la zone de données, "résultats"le nom de l'automate qui gère la zone de résultats et "transferts" le nom de l'automate qui traitera les résultats. Selon les cas, certaines zones ne seront pas remplies : par exemple, s'il s'agit d'un automate vide, seul le nom du noeud et la liste d'appels auront une signification.

Une extension de cette commande permet d'afficher l'ensemble du réseau. L'utilisateur tape "?/****/" et l'éditeur affiche tous les noeuds du réseau avec le format donné plus haut. L'utilisateur dans ce cas n'a pas la possibilité de modifier les spécifications de chaque noeud.

Suppression

Format de la commande :

U : \/NOM D'AUTOMATE/

Cette commande supprime le noeud désigné, s'il existe, dans le réseau et toutes les apparitions de ce noeud dans les différentes listes d'appels du réseau.

S'il existe une référence de la zone données-résultats du noeud désigné dans le réseau, alors la commande est refusée ; ceci pour éviter qu'un automate travaille avec une zone de données-résultats n'existant plus.

Par contre, si l'automate du noeud désigné porte un dictionnaire et une grammaire et que d'autres automates y font référence, lors de la suppression du noeud le dictionnaire et la grammaire seront portés par un autre automate et les références seront automatiquement changées.

La commande *

Format de la commande :

U : */NOM D'AUTOMATE/

Cette commande facilite l'initialisation du réseau. Elle permet de nettoyer la liste d'appels du noeud initial appelé "ENTREE" et d'initialiser la liste d'appels par le noeud désigné par l'utilisateur.

Il existe une autre manière de le faire, c'est en interrogeant "ENTREE".

II.3.4.3. Exemples

Exemple 1

Cet exemple montre comment construire un réseau comprenant un seul TGEF. Nous l'appellerons "TGEF" et il utilisera un dictionnaire et une grammaire du même nom. Il lira ses données et écrira ses résultats sur des supports externes. Il utilisera pour la lecture la zone de données-résultats du noeud initial et pour l'écriture sa propre zone de données-résultats. Il ne fonctionnera pas en mode inverse et il n'y aura qu'un type de sortie.

Voici le dialogue susceptible d'exister entre l'éditeur et l'utilisateur pour créer ce réseau :

U : surgram
E : NOM D'AUTOMATE ET SES CARACTERISTIQUES
U : /TGEF(d)(r)(e)(l)(n)/val00/
E : INVERSE O/N
U : N
E : N° DE SORTIE ?
U : 1
E : NOM DE L'AUTOMATE OU L'ON RANGE LES DONNEES ?
U : entrée
E : NOM DE L'AUTOMATE OU L'ON RANGE LES RESULTATS ?
U : TGEF
E : NOM D'AUTOMATE A APPELER ?
U : RC
E : NOM D'AUTOMATE ET SES CARACTERISTIQUES ?
U : */TGEF/
E : NOM D'AUTOMATE ET SES CARACTERISTIQUES ?

U : ?/****/

E : /ENTREE/*****/

APPELS : TGEF

/TGEF(d)(r)(e)(l)(n)/VALOO/TGEF/

INVERSE : n

N° DE SORTIE : 1

DONNEES : ENTREE

RESULTATS : TGEF.

Exemple 2

Nous donnons ici la représentation d'un réseau permettant de faire du contrôle de saisie sur un texte français (détection des fautes d'orthographe d'usage et des fautes de frappe) et de le traduire en braille abrégé.

Le contrôle de saisie : Pour traduire et éditer un texte en braille abrégé, il est nécessaire d'inclure dans le texte source un certain nombre de signes spéciaux utilisés pour l'édition. Ces signes n'étant pas considérés comme des mots français, ils ne rentreront pas dans un dictionnaire et une grammaire détectant des fautes de frappe et d'orthographe d'usage. Nous aurons donc deux automates en subroutine, l'un appelé "accent" qui accepte seulement les mots français, l'autre appelé "typo" qui accepte seulement les signes spéciaux. Ces deux automates sont des accepteurs, c'est-à-dire qu'ils ne font qu'accepter le texte source sans faire de transduction ; ce sont simplement des filtres.

La traduction en braille abrégé : nous utilisons un automate dont les données sont les résultats du contrôle de saisie. Cet automate appelé "braille" utilise le dictionnaire et la grammaire du braille abrégé.

Dans cette application, nous aurons deux types de sortie :

- pour la détection de fautes ; si une chaîne de caractères quelconque est acceptée, elle est intégralement recopiée comme résultat ;
- pour la traduction en braille abrégé, les résultats de l'analyse et certaines informations dans le dictionnaire sont utilisés pour réaliser la traduction.

Caractéristique des automates :

- DETECT : automate vide, gère la zone de données-résultats de ENTREE et sa propre zone, s'occupe de lecture du fichier de données, appelle ACCENT et TYPO

- ACCENT : prend ses données dans ENTREE et met ses résultats dans DETECT, marche en mode non inverse, mode de sortie 1
- TYPO : lecture des données ENTREE, écriture des résultats DETECT, mode non inverse, type de sortie 1
- BRAILLE : lecture des données DETECT, gère sa zone de données-résultats, écrit les résultats sur un support externe, fonctionne en mode non inverse, type de sortie 2.

Le réseau étant défini, on obtient en tapant la commande suivante :

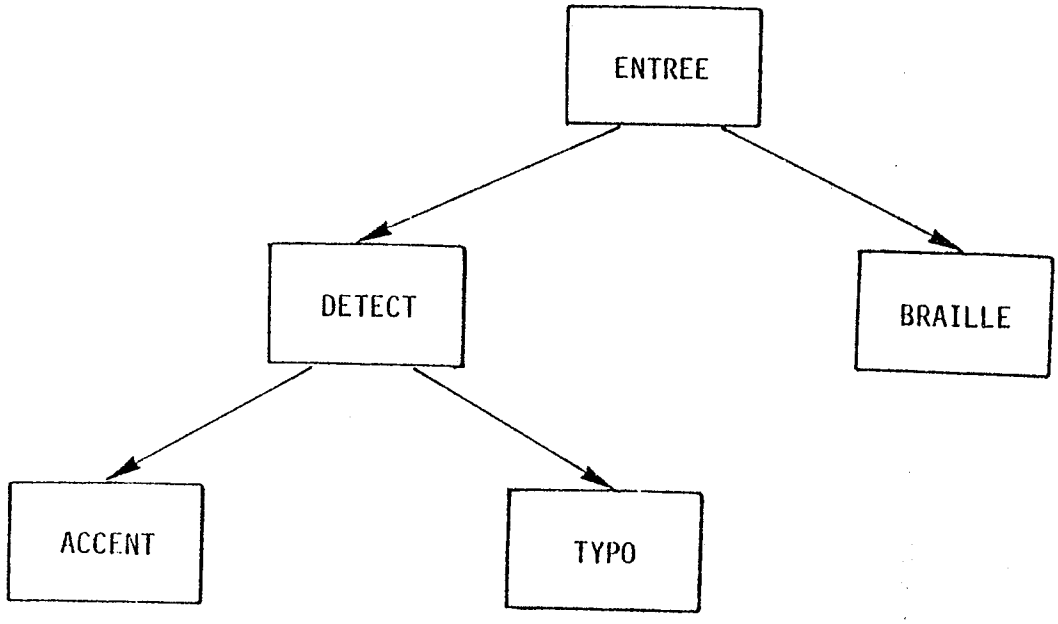
```
U : ?/****/
```

La structure suivante :

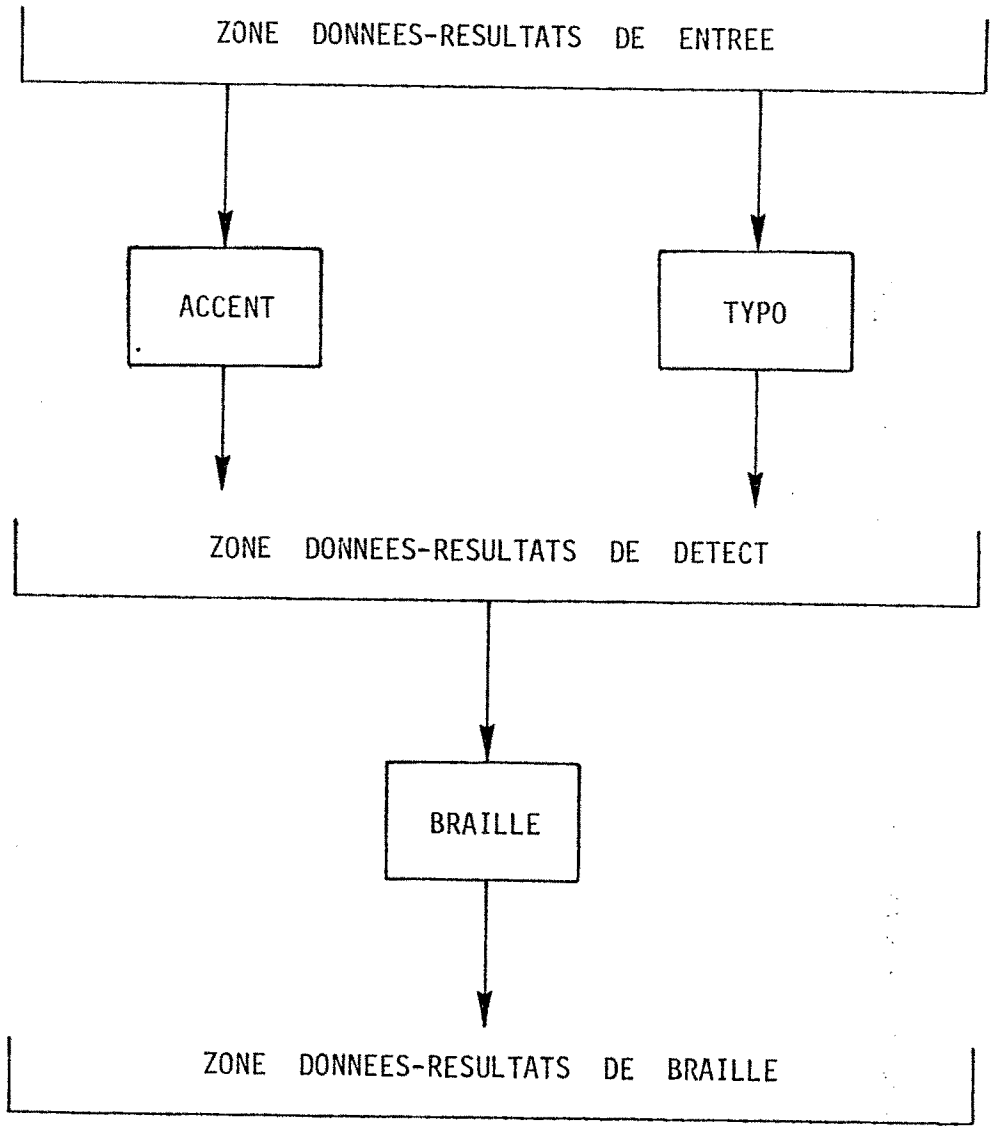
```
E : /ENTREE/*****/
        APPELS : DETECT BRAILLE

    /DETECT(d)(r)(l)(t)/*****/
        DONNEES : ENTREE
        TRANSFERT : BRAILLE
        APPELS : ACCENT TYPO

    /ACCENT(n)/VALOO/ACCENT/
        INVERSE : N
        N° DE SORTIE : 1
        DONNEES : DETECT
        RESULTATS : DETECT

    /TYPO(n)/VALOO/TYPO/
        INVERSE : N
        N° DE SORTIE : 1
        DONNEES : DETECT
        RESULTATS : DETECT

    /BRAILLE(d)(r)(e)(n)/VALOO/BRAILLE/
        INVERSE : N
        N° DE SORTIE : 2
        DONNEES : DETECT
```

On trouve ci-dessous les schémas correspondant au réseau des appels et au réseau des zones de données-résultats de l'exemple précédent.

Réseau des appels



Réseau des zones de données-résultats



CHAPITRE 4

SCHEMAS D'ALGORITHME ET IMPLANTATION DE MULTPIAF

II.4.1. SCHEMAS D'ALGORITHMES

Nous donnons ici les schémas d'algorithmes des deux procédures qui concernent le fonctionnement du réseau des automates. La procédure "analyser" est appelée à partir des commandes INTERRO et EXPLOIT du programme principal.

Le paramètre réseau d'automates est un paramètre contenant toutes les informations concernant le réseau d'automates, notamment listeinitialed'appels qui est la liste d'appels de l'automate initial.

```
PROCEDURE analyser (entrée : réseau d'automates) ;
début
    déclaration : succès, bavard, automatetrouvé ;
    chercherautomate (listeinitialed'appels, automatetrouvé) ;
    si automatetrouvé = nil
        alors
        début
            écrire(message d'erreur) ;
            sortir ;
        fin ;
    lire (zonedonnées) ;
    tantque ¬ fin de fichier faire
            début
                tantque automatetrouvé ¬ = nil faire
                    début
                        tantque ¬ fin de segment de données faire
                            début
                                succès ← faux ;
                                bavard ← faux ;
                                lancerautomate(automatetrouvé,
                                    étatinitial, bavard, succès) ;
```

```
                                commentaire : s'il y a succès, le pointeur de
                                              données dans le segment de données
                                              avance de la longueur de la
                                              chaîne de caractères analysée ;
                                        si ¬ succès alors
                                             début
                                                  bavard ← vrai ;
                                                  lancerautomate(automate-
                                                    trouvé, étatinitial,
                                                    bavard, succès) ;
                                                  modification ;
                                             fin ;
                                    fin ;
                               chercherautomate (listeinitialed'appels,
                                 automatetrouvé) ;
                          fin ;
                     démarquer (listeinitialed'appels) ;
                     chercherautomate (listeinitialed'appels, automatetrouvé) ;
                     lire (zonedonnées) ;
               fin ;
finprocédure ;
```

Le paramètre automate est une structure d'informations concernant l'automate et comprenant plusieurs champs, notamment listed'appels, appelé et étatc.

Appelé est un booléen permettant de savoir si l'automate correspondant a déjà été appelé et étatc permet de conserver le dernier état dans lequel se trouve l'automate correspondant.

Ces deux éléments sont utilisés pour éviter l'indétermination en cas d'automates récursifs.

```
PROCEDURE lancerautomate (entrée : automate, étatcourant, bavard sortie :
                          succès) ;
début
    déclaration : succèsappel, chaîned'entrée, listap, automatetrouvé,
                  lrègleapl, règle ;
    chaîned'entrée ← vide ;
    listap ← listed'appels (automate) ;
    succès ← faux ;
    appelé (automate) ← vrai ;
    étatc (automate) ← étatcourant ;
    si bavard alors
        écrire (nomd'automate (automate)) ;
    tantque étatcourant ¬ = nil faire
        début
            si automate ¬ = vide alors
                début
                    ident (chaîned'entrée) ;
                    si chaîned'entrée ¬ = vide alors
                        début
                            lrègleapl ← calcul (modèle, validation-
                              courante) ;
                            sélectionner (lrègleapl, règle) ;
                            tantque règle ¬ = nil faire
                                début
                                    si bavard alors
                                        écrire(chaîned'entrée) ;
                                    appliquer(règle, étatsuivant) ;
                                    lancerautomate (automate,
                                      étatsuivant, bavard, succès) ;
                                    sélectionner (lrègleapl, règle) ;
                                fin ;
```

```
                                si étatcourant = étatfinal alors
                                    début
                                        succès ← vrai ;
                                        appelé (automate) ← faux ;
                                        si indicateur N alors
                                            écrirerésultats (automate)
                                            sinon
                                            conserverrésultats(automate);
                                    fin ;
                            fin ;
                    fin ;
            si (¬ succès ou solutionmultiple) et listed'appels non toute
               marquée alors
commentaire : solutionmultiple correspond à l'indicateur (M) mentionné
               dans le chapitre précédent ;
                début
                    si analysemultiple alors
            commentaire : analysemultiple correspond à l'indicateur (S)
                        mentionné dans le chapitre précédent ;
                        démarquer (listap) ;
                    succèsappel ← faux ;
                    chercherautomate (listap, automatetrouvé) ;
                    tantque automatetrouvé ¬ = nil et ¬ succèsappel faire
                        début
                            si ¬ appelé (automatetrouvé) ou étatc
                              (automatetrouvé) ¬ = étatinitial alors
                    commentaire : test interdisant l'indétermination
                                dans le cas d'appels récursifs ;
                                début
                                    lancerautomate(automatetrouvé,
                                      étatinitial, bavard, succèsap-
                                      pel) ;
                                    appelé(automatetrouvé) ← faux ;
                                fin ;
```

```
                                    si succèsappel alors
                                        début
                                            chaîned'entrée ← vide ;
                                            si automate = vide alors
                                                succès ← vrai ;
                                            si ¬ solutionmultiple alors
                                                marquer(listap) ;
                                        fin
                                    sinon
                                        chercherautomate(listap, automate-
                                            trouvé) ;
                                fin ;
                        fin
                    sinon
                        étatcourant ← nil ;
                fin ;
finprocédure ;
```

Il est à remarquer que la procédure "analyser" privilégie les automates appelés à partir du noeud initial "ENTREE". En effet, les automates dont les noms sont contenus dans la liste d'appels de "ENTREE" sont considérés comme les noeuds initiaux de sous-réseaux (sous-réseaux pouvant être réduits à un seul automate). Ces sous-réseaux seront appelés successivement et après chaque appel, on contrôle que le sous-réseau appelé n'a pas eu d'échec ; s'il y a eu échec, alors l'utilisateur peut intervenir.

Dans le cas d'un traitement de données en cascade, cette méthode permet de faire un contrôle et d'intervenir éventuellement à chaque pas de la cascade. Dans le cas de la traduction en braille abrégé avec contrôle de saisie, il est possible d'intervenir entre le contrôle de saisie et la traduction en braille abrégé si cette dernière échoue.

II.4.1.1. Procédure utilisée dans les algorithmes

PROCEDURE ident (entrée-sortie : chaîned'entrée) ;

Cette procédure permet d'affecter chaîned'entrée de la manière suivante : si chaîned'entrée est vide alors chaîned'entrée ← identification entre la plus longue chaîne de caractères à partir du pointeur de la zone de données et les éléments du dictionnaire.
Si la chaîne d'entrée n'est pas vide, alors redécoupage de la chaîne d'entrée en sélectionnant, dans le dictionnaire, le premier fils de l'élément contenu dans chaîned'entrée.
S'il n'existe aucune coïncidence entre la zone de données et les éléments du dictionnaire, alors chaîned'entrée sera vide.

FONCTION calcul (modèle, validationcourante) ;

Cette fonction calcule la liste de règles à appliquer à partir du modèle référencé par l'élément du dictionnaire et la validation courante.

PROCEDURE sélectionner (entrée : liste de règles, sortie : règle) ;

Cette procédure sélectionne dans une liste de règles une règle qui n'a pas encore été appliquée. S'il n'en existe pas, la procédure délivre NIL.

PROCEDURE appliquer (entrée : règle, sortie : étatsuivant) ;

Cette procédure applique une des règles validées à l'état courant et calcule l'état suivant, c'est-à-dire principalement les règles applicables de l'état suivant.

PROCEDURE chercherautomate (entrée : listed'appels, sortie : automate) ;

Cette procédure sélectionne dans la liste d'appels d'un automate le premier automate non marqué de cette liste. Elle marque cet automate et délivre son nom en sortie.
Si tous les automates de la liste d'appels sont déjà marqués, alors la procédure délivre nil à la sortie.
Ces marques permettent de savoir si l'on a déjà appelé un automate ou pas.

PROCEDURE marquer (entrée-sortie : listed'appels) ;

Cette procédure marque tous les automates d'une liste d'appels.

PROCEDURE démarquer (entrée-sortie : listed'appels) ;

Cette procédure enlève les marques sur les automates d'une liste d'appels.

PROCEDURE écrirerésultats (entrée : automate) ;

Cette procédure est chargée de sortir tous les résultats qui concernent cet automate, c'est-à-dire ceux produits par l'automate lui-même et ceux produits par des automates qui ont confié la sortie de leurs résultats à l'automate considéré.

PROCEDURE conserverrésultats (entrée : automate) ;

Cette procédure permet de confier les résultats de l'automate courant dans la liste des sorties à faire d'un autre automate.

NB : Ces deux dernières procédures correspondent au fonctionnement de l'indicateur N décrit dans le chapitre 2.3.

PROCEDURE écrire (entrée : chaîne de caractères) ;

Cette procédure écrit "chaîne de caractères" sur le terminal.

PROCEDURE lire (entrée : zonedonnées) ;

Cette procédure lit des données au terminal ou sur un fichier et remplit zonedonnées avec un segment de phrase.
Ce segment de phrase peut être défini en fonction de l'application ; en général, il sera défini comme une chaîne de caractères comprise entre deux ponctuations.

PROCEDURE modification ;

Cette procédure permet à l'utilisateur d'accéder aux commandes DICT, GRAMM et SURGRAM. De plus, elle lui permet de modifier la zone de données.

Si l'utilisateur ne sait que faire, il peut soit demander au système des informations sur les commandes disponibles sous cet état, soit relancer l'analyse après le mot qui a provoqué l'appel de cette procédure.

II.4.1.2. Terminaisons des algorithmes

PROCEDURE analyser

- 1ère boucle :
  condition de fin : fin de fichier ; cette condition se réalisera toujours.
- 2ème boucle :
  condition de fin : automatetrouvé = nil, c'est-à-dire tous les automates de listeinitialed'appels ont été appelés. La liste étant finie, cette condition se réalisera toujours.
- 3ème boucle :
  condition de fin : fin du segment de données ou contrôle donné à l'utilisateur. Le segment de données ayant toujours une longueur finie, cette condition se réalisera toujours.

En cas d'échec à l'analyse, l'utilisateur peut effectuer des modifications ou forcer la sortie de la boucle.

PROCEDURE lancerautomate

- 1ère boucle :
  condition de fin : étatcourant = nil ;
  Cette condition est réalisée si l'on a :
  (succès et ¬ solutionmultiple) ou listed'appels toute marquée. L'une au moins de ces deux conditions se réalisera toujours.
- 2ème boucle :
  condition de fin : règle = nil ; toutes les règles de la liste de règles applicables ont été appliquées. Cette liste étant finie, cette condition se réalisera toujours.
- 3ème boucle :
  condition de fin : automatetrouvé = nil ou succèsappel, c'est-à-dire tous les automates de la liste d'appels ont été appelés ou un automate appelé a réussi son analyse.
  La liste d'appel étant finie, cette condition se réalisera toujours.

II.4.2. IMPLANTATION DU SYSTEME MULTPIAF

Nous avons réalisé l'implantation du système MULTPIAF sur IBM 360-67 système CP-CMS.

Cette réalisation pratique a été faite de manière à être compatible avec le système PIAF (un dictionnaire et une grammaire créés avec le système PIAF sont utilisables dans le système MULTPIAF et réciproquement).

D'autre part, il existe des différences entre la réalisation pratique et le modèle théorique décrit jusqu'à présent. Nous exposons ci-dessous les principales différences :

1) L'éditeur :

Il est impossible de créer un nouveau noeud en deux phases :

- création du nom de l'automate, de la référence au dictionnaire et à la grammaire, de la validation initiale, etc ;
- définition des zones de données-résultats et de la liste d'appels.

L'obligation de créer un noeud en une seule phase donne moins de souplesse à l'éditeur car beaucoup d'informations données à la création du noeud font références à d'autres noeuds ; il est donc nécessaire que ces noeuds existent.

L'ordre dans lequel doivent être créés les noeuds d'un réseau n'est donc pas indifférent. Dans la réalisation pratique, seule la liste d'appels peut être complétée ou modifiée en dehors de la création du noeud.

Les indicateurs (M) et (S) n'ont pas été implantés et ne sont pas actuellement utilisables. Les automates fonctionnent en mode analyse non multiple et solution non multiple. Ce dernier point influence aussi le fonctionnement du système.

Il ne faut pas beaucoup de modifications de programmes pour obtenir un éditeur correspondant à la description faite dans le chapitre précédent.

2) Implantation du système :

A part le mode analysemultiple et solutionmultiple, le fonctionnement du système est sensiblement le même que celui décrit dans les chapitres précédents. Ce qui diffère un peu, c'est la manière dont il a été programmé.

En effet, dans les algorithmes exposés dans les paragraphes précédents, il n'apparaît qu'une pile (récursivité de la procédure lancerautomate). Si on utilise le système pour une application où il y a peu d'appels, alors beaucoup d'informations redondantes sont conservées à chaque niveau de la pile. Pour éviter cet inconvénient, nous avons utilisé deux piles, l'une servant à gérer les différents appels et l'autre servant à contrôler le fonctionnement des différents automates. Ces deux piles conservent d'autres informations pour permettre leur utilisation en parallèle.

Il nous semble que l'exposition d'un algorithme ne comportant qu'une seule pile est plus claire.

II.4.3. EXEMPLE DE SESSION

Nous utilisons le réseau décrit dans l'exemple 2 du chapitre 2.3 et nous traitons le fichier HIST2 DATA dont nous donnons, ci-dessous, le contenu :

```
$*_"LA #FRANCE EN (1789."$*
_[ LA FIN DU "XVIII+ME" SI\CLE, LA #FRANCE /TAIT COMPARABLE [ LA
#FRANCE D'AUJOURD'HUI PAR SON /TENDUE ET SES LIMITES. SA
SUPERFICIE /TAIT DE &525'000 KM2, SES FRONTI\RES SUIVAIENT [ PEU
PR\S LE TRAC/ ACTUEL. TOUTEFOIS, #MULHOUSE /TAIT UNE R/PUBLIQUE
IND/PENDANTE; #MONTB/LIARD, LA #SAVOIE ET #NICE APPARTENAIENT
[ DES PUISSANCES /TRANG\RES; LE PAPE POSS/DAIT EN #FRANCE #AVIGNON
ET LE #COMTAT #VENAISSIN, TANDIS QUE #LANDAU /TAIT UNE ENCLAVE
FRAN@AISE DANS LE #PALATINAT.MAIS, [ L'INT/RIEUR DE CES FRONTI\RES,
LE PAYS /TAIT TR\S DIFF/RENT DE LA #FRANCE DE NOTRE /POQUE, PAR
LE GOUVERNEMENT ET LE FONCTIONNEMANT DE L'ADMINISTRATION, PAR
LA VIE /CONOMIQUE ET PAR L'ORGANISATION DE LA SOCI/T/.
```

Signification des codes utilisés

$*	début ou fin de page
#	ne pas abréger le mot suivant
(	la chaîne de caractères qui suit est un nombre
[	code de à
\	code de è
/	code de é
&	la chaîne de caractères qui suit est un nombre
"	le texte entre " est un titre

```
multpiaf                                                    appel du système
EXECUTION BEGINS...
ACTIVER PROGRAMME ? (GRAM,DICT,INTERRO,EXPLOITATION,SURGRAM)
$
exploit                                                     on va traduire un fichier en braille
DONNEES ? (TERMINAL OU NOM DE FICHIER)                      abrégé en le contrôlant
%
hist2
TYPE DU FICHIER?
>
data
RESULTATS?(TERMINAL OU NOM DE FICHIER)
>                                                           terminal

TAILLE DE LA LIGNE A ECRIRE
>
80

   LA HFRANCE EN ( 1789 .                                   traduction du titre

  [ ' F) <  XVIII+ME  SC, ' (FRANCE /T3 -PAR% [ ' (FRANCE D'K'H P 9 /T?DUE ] SS
A PARTIR DE :SUPERFICIE /TAIT DE $5                         traduction première phrase
SU

S                                                           superficie est un mot absent du
U                                                           dictionnaire du contrôle de saisie
U

-
dict
NOM DU DICTIONNAIRE?
>
accent                                                      spécification du dictionnaire
>
/superficie/voiture/                                        indexation de ce mot sur le modèle
>                                                           "voiture"

A PARTIR DE :KM2 ,                                          KM2 est une chaîne existante
-
ortho
>
/km2/" km2"/                                                correction de la faute de frappe
LIMITS. SA SUP<FICIE /T3 D (525'000  KM2, SS !^TI\R5 S;V*2 [$P$! L >AC/ ACTUEL.
TF, (MULHOUSE /T3 UN R/PU%IQ )D/P?D,TE; (MONTB/LIARD, ' (SAVOIE ] (NICE          Traduction
APP"TEN*2 [ DS PCS />,G\R5; L PAPE PO\/D3 ? (FRANCE (AVIGNON ] L (CONTAT
(VENAISSIN, T$Q (LANDAU /T3 UN ?3AVE 1,@*SE 4 L (PALATINAT. X, [ L')T D CS
A PARTIR DE :FONCTIONNEMANT DE L'AD
```

```
FONCTION
FONC                              le mot "fonctionnemant" est mal orthographié et ne peut
FON                               donc être accepté par le système
FO
F
F
-
ortho
>
/nemant/nement/                   correction de la faute d'orthographe
1^TI\R5, L PAYS /T3 >S D? D ´ (FRANCE D N> /POQ, P L GV ] L F^M D L´ADMINIS>1, P      traduction
´ VIE /CONOMIQ ] P L´+GANIS1 D ´ SOCI/T/.
$
interro                                   changement de mode
>
nous eeHssayons le syst\me, _             entrée d'une phrase
A PARTIR DE :EEHSSAYONS LE SYST\ME        détection de l'erreur
-
ortho
>
/eehssayons/essayons/                     correction de l'erreur
A PARTIR DE :SYST\ME , _
S
                                          "système" est un mot absent du
-                                         dictionnaire servant au contrôle
dict
NOM DU DICTIONNAIRE?
>
accent                                    spécification du dictionnaire
>
/syst\me/homme/                           indexation de ce mot
>

0 5SAY^S L SY,                            traduction de la phrase
>
$fin
```

CONCLUSION

I - APPLICATIONS DU TGEF

Traduction en braille abrégé

L'application du TGEF du système PIAF à la traduction en braille abrégé de textes français a montré l'intérêt d'utiliser un outil de traitement des langues naturelles pour résoudre un tel problème.

Le système d'abréviation du braille dans une langue donnée étant construit à l'aide des particularité linguistiques de cette langue (fréquence des mots et des groupes de lettres), il était normal d'utiliser les principes définis pour l'analyse morphologique.

Les outils utilisés résident principalement dans la segmentation de la chaîne d'entrée, la construction et l'utilisation d'un automate d'états finis et l'accès rapide à un dictionnaire contenant les informations nécessaires à la segmentation et à la marche de l'automate.

Le TGEF étant construit pour résoudre ces difficultés et étant en plus entièrement interactif, il nous a été facile de réaliser rapidement un traducteur en braille abrégé de textes français.

Pour les mêmes raisons, il est possible de réaliser, à l'aide du TGEF, un traducteur en braille abrégé pour une autre langue que le français pourvu que cette langue ait un système d'abréviation bien défini.

Le TGEF et ses applications ont été réalisés en PL360 (SU) sur IBM 360-67 système CP-CMS et peut fonctionner sur IBM 370 système CMS-VM.
Nous signalons que l'implantation du TGEF sur micro-ordinateur en langage PASCAL est en cours de réalisation.

Le coût de production de textes en braille abrégé devant être le plus réduit possible, il est très important d'utiliser un micro ou un miniordinateur pour réaliser cette traduction.

Le terminal braille SAGEM TEM 8 br utilisé par le Laboratoire Brigitte Frybourg pour tester le traducteur en braille abrégé réalisé à l'aide du TGEF ne nous paraît pas assez rapide pour répondre au besoin d'un grand centre de production de livres en braille.

A notre connaissance, il n'existe pas, autrement que sous forme de prototypes, d'imprimantes braille rapides. L'absence d'organe de sortie rapide sera toujours une limite à la vitesse de production du système.

Traduction inverse du braille abrégé

Le développement de la bureautique, des microordinateurs et des terminaux utilisant un affichage en braille éphémère va permettre aux non-voyants de pouvoir utiliser des outils de rédaction très puissants et très efficaces qui faciliteront la communication entre personnes voyantes et non-voyantes.

Dans cette optique, un traducteur braille abrégé - braille intégral peut être un composant intéressant de ces outils de rédaction (beaucoup de non-voyants préfèrent écrire en braille abrégé).

Malgré les ambiguïtés des symboles d'abréviation du braille, nous sommes arrivés, à quelques exceptions près, à rendre inverse le traducteur en braille abrégé par l'ajout de règles dans la grammaire et d'éléments dans le dictionnaire.

Les ambiguïtés restantes sont peu nombreuses mais très difficiles à lever automatiquement ; à la rencontre de l'une d'entre elles, le traducteur demande à l'utilisateur de faire un choix.

Cette application bénéficiera beaucoup de l'implantation du TGEF sur micro-ordinateur.

Le contrôle de saisie

Cette application, qui dérive de l'analyseur morphologique, peut servir dans tous les domaines où la saisie d'informations intervient.

Elle est particulièrement indiquée dans le cas d'un centre de production de textes en braille car la saisie correcte des informations est une des opérations les plus coûteuses.

Ce contrôle de saisie ne détectant que les fautes de frappe et d'orthographe d'usage n'exclut pas le contrôle manuel, mais il facilite beaucoup la tâche de ce dernier (les fautes citées ci-dessus sont les plus fréquentes).

Nous n'avons pas inclu de contrôle de type syntaxique (dérivé de l'analyseur syntaxique du système PIAF), car ce contrôle aurait été trop lourd par rapport à l'amélioration apportée.

II - LE SYSTEME MULTPIAF

Il existe bien d'autres applications du système MULTPIAF que celles décrites jusqu'à présent. Nous donnons ici les propriétés qui caractérisent ce système et le désignent tout particulièrement pour certaines applications :

1) Compatibilité avec le TGEF :

Il possède toutes les propriétés du TGEF et est entièrement compatible avec ce dernier. Une application déjà réalisée à l'aide du TGEF peut être considérée comme un élément d'un réseau contrôlé par MULTPIAF.

2) Analyse d'un langage complexe :

Le fait de pouvoir faire fonctionner plusieurs automates en subroutine permet de décomposer un problème complexe en plusieurs problèmes plus faciles à résoudre.

Des langages tels que ceux utilisés dans les bandes de photocomposition sont des langages d'états finis mais si l'on veut contrôler globalement ces bandes, le problème devient complexe ; en contrôlant séparément les informations textuelles et les informations typographiques, le problème devient plus simple.

3) Indépendance des grammaires et des dictionnaires :

L'indépendance des grammaires et des dictionnaires nous semble l'une des propriétés les plus importantes de MULTPIAF.

En effet, une grammaire et un dictionnaire peuvent être utilisés dans plusieurs applications différentes ; c'est le cas, par exemple, de l'analyseur morphologique ou du contrôleur de saisie qui présente un intérêt assez

général pour participer à de nombreuses applications.

Or, si l'on utilisait le TGEF seul, il faudrait pour chacune de ces applications construire une grammaire et un dictionnaire intégrant à la fois le contrôleur de saisie et le transducteur propre à l'application.

Avec MULTPIAF, il suffit de créer un automate analysant les éléments du langage propre à l'application et d'utiliser cet automate dans un réseau avec le contrôleur de saisie.

C'est le cas du contrôle et de la traduction des bandes de photocomposition qui ont pour chaque type de bandes un langage spécial de commandes d'édition.

4) Fiabilité des dictionnaires et des grammaires utilisés :

Dans certaines applications, il est indispensable de conserver des dictionnaires et grammaires sans aucune erreur ; souvent, la mise à jour de ces dictionnaires et grammaires demande une connaissance particulière de leur structure et du langage analysé, ce qui n'est pas toujours le cas de l'utilisateur.

Pour éviter l'introduction définitive d'une erreur dans un de ces automates, nous proposons de construire un réseau comportant, notamment, deux automates en subroutine dont l'un est l'automate, dit définitif, qui ne doit pas comporter d'erreurs et l'autre étant un automate, dit annexe, dans lequel l'utilisateur non spécialiste peut introduire des nouvelles chaînes de caractères qu'il veut voir analysées par le réseau.

Cet automate annexe pourra être repris périodiquement par un spécialiste pour transférer, avec correction si nécessaire, les chaînes de caractères dans l'automate définitif.

Cette méthode a l'intérêt de mieux préserver l'automate définitif, tout en permettant à l'utilisateur non spécialiste de répondre au système lors d'un échec à l'analyse.

L'automate annexe peut avoir une grammaire très simple n'acceptant, par exemple, que des formes invariables.

Nous préconisons cette solution en ce qui concerne la mise à jour du dictionnaire et de la grammaire du contrôleur de saisie ; cette mise à jour étant délicate à faire proprement (voir chapitre 1.3).

III - POINTS A DEVELOPPER

Les études qui ont été faites sur la fréquence des mots en langue française tendraient à prouver que les cent mots les plus fréquents occupent cinquante pour cent de la totalité des textes (KDF).

L'introduction de ces cent mots avec leur traduction dans le dictionnaire permettrait d'optimiser considérablement le traducteur braille abrégé.

A la rencontre de l'un de ces mots, le traducteur le traiterait directement sans réaliser de découpages superflus.

Comme cela a été fait pour le TGEF, il serait intéressant de développer le fonctionnement du système MULTPIAF en génération.

Cette possibilité permettrait de connaître les chaînes engendrées par un réseau d'automates et de détecter celles qui sont indésirables. Cette méthode faciliterait beaucoup la mise en place d'un réseau, surtout s'il est un peu complexe.

L'utilisateur devra pouvoir limiter la génération comme il le désire (génération à partir d'une racine, d'une classe de racines, de l'ensemble des racines) et empêcher la génération des chaînes et des appels infinis.

Il serait intéressant, peut-être, d'étudier plus à fond ce qu'il faudrait ajouter pour que MULTPIAF puisse accepter tous les langages hors contexte. Nous pensons que cette faiblesse provient d'un contrôle trop simple au retour de l'appel d'un automate.

Pour terminer, nous suggérons deux applications où MULTPIAF serait intéressant à utiliser :

Traduction assistée de textes pris en sténotypie

Pour ceci, il serait nécessaire d'utiliser un réseau comportant deux automates en cascade :

- le premier servirait à produire toutes les possibilités orthographiques obtenues à partir d'une chaîne de caractères sténotypiques ;

- le deuxième étant le contrôleur de saisie servirait à filtrer les résultats du premier en ne gardant que les mots appartenant au français.
S'il existe un problème après ce filtrage (ambiguïté sur une chaîne de caractères sténotypiques ou impossibilité de trouver un mot français correspondant), l'utilisateur sera appelé à intervenir soit en faisant un choix en cas d'ambiguïtés, soit en modifiant les grammaires, les dictionnaires ou le texte d'entrée.

Nous ne prétendons pas que ce réseau serait un traducteur automatique de sténotypie, mais il apporterait une aide considérable à la traduction et à l'édition de textes pris en sténotypie (stG).

Mise en oeuvre d'une présyntaxe

Le système MULTPIAF pourrait être utilisé pour réaliser une présyntaxe.

Nous entendons par présyntaxe une analyse sommaire et linéaire d'une phrase délivrant en résultat une phrase en partie parenthésée, en regroupant les mots de la phrase en éléments grammaticaux tels que groupe substantif (article + substantif, par exemple).

Cette présyntaxe aurait l'intérêt de faciliter le travail de l'analyseur syntaxique en permettant de lever facilement certaines ambiguïtés sur les catégories ("le" peut être considéré comme article ou pronom) et en évitant des tentatives de construction de structures erronées.

Une présyntaxe se justifie par le fait qu'une langue naturelle possède peu de structures imbriquées, c'est un langage pratiquement linéaire.

Cette présyntaxe ne devra pas délivrer des résultats erronés car l'analyseur syntaxique ne peut pas se permettre de vérifier le travail de l'analyseur présyntaxique.

Cet analyseur présyntaxique utiliserait la liste de catégories et de variables fournies par l'analyseur morphologique et il pourrait se composer de plusieurs TGEF travaillant en subroutine et contenant dans leur dictionnaire des suites types de catégories auxquelles l'on associe un nom de groupe grammatical.

Les possibilités d'appels récursifs des TGEF, dans un réseau, devraient permettre de résoudre les problèmes des structures imbriqués (propositions relatives).

BIBLIOGRAPHIE

[AVH] Association Valentin HAUY
"Abrégé Orthographique français Etendu".
Paris, 1965.

[Cha] C. CHASSAGNE
"Etude et réalisation d'une méthode de transport : traduction de programmes PL360 en LP80".
Thèse Docteur-Ingénieur, Institut National Polytechnique de Grenoble, Janvier 1979.

[ChCGV] Y. CHIARAMELLA, J. COURTIN, E. GRANDJEAN, G. VEILLON
"Utilisation des techniques en parallèle au contrôle de données ambiguës. Applications aux sciences humaines".
Congrès AFCET, Panorama des nouveautés informatiques en France. Paris, Novembre 1976.

[Chia] Y. CHIARAMELLA
"Détection automatique des variations orthographiques sur les noms propres. Définition d'un transducteur morphologique interactif".
Conférence internationale sur le traitement automatique des Langues. Ottawa, Juin-Juillet 1976.

[Chom] N. CHOMSKY
"Aspects of theory of syntax".
The M.I.T. Press, 1965.

[Cour1] J. COURTIN
"Algorithmes pour le traitement interactif des langues naturelles".
Thèse d'Etat, Université de Grenoble, 1977.

[Cour2] J. COURTIN
"Utilisation des redondances pour l'analyse et le contrôle automatique d'énoncés en langue naturelle".
Conférence Informatique sur le traitement automatique des langues naturelles, Ottawa, 1976.

[CtDj] J. COURTIN, D. DUJARDIN
"Paramètres linguistiques pour la morphologie française dans le système PIAF".
Document interne, Grenoble, 1976.

[CGV] J. COURTIN, E. GRANDJEAN, G. VEILLON
"PIAF : Un système de manipulation de données en langue naturelle".
Colloque international de terminologie, Paris, 1976.

[GiHu] J.H. GILL, J.B. HUMPHREYS
"An analysis of braille contraction".
Warwick research unit for the blind, UK.

[Grj1] E. GRANDJEAN
"Application d'un système de traitement de langues naturelles pour l'indexation automatique".
Avril 1978.

[Grj2] E. GRANDJEAN
"Conception et réalisation d'un dictionnaire pour un analyseur interactif de langues naturelles".
Mémoire CNAM, Université de Grenoble, 1975.

[HoUl] J.E. HOPCROFT, J.D. ULLMAN
"Formal languages and their relation to automata".
Addison-Wesley, 1969.

[KDF] D. KEEPING, M.S. DOYLE, P.A. FORTIER
"Braille large print and voice synthesis".
International conference on *"Computerised Braille Production - to day and to morrow"*, Londres 1979.

[Jol] V. JOLOBOFF
"Unification d'arborescences. Evaluation sémantique d'énoncés en langue naturelle".
Thèse de Docteur-Ingénieur, INPG, Grenoble, Septembre 1978.

[Mat1] B. MATHIEU
"Utilisation d'un transducteur général d'états finis pour la saisie de textes et leur traduction en braille abrégé".
Congrès AFCET, Paris, Novembre 1978.

[Mat2] B. MATHIEU
"Applications of PIAF program".
International conference on *"Computerised Braille Production - to day and to morrow"*, Londres, 1979.

[stG] Sténotype Grandjean
"Méthode de sténotypie".

[Su] D. SUTY
"Le langage PL360".
Janvier 1971.

[Vauq] B. VAUQUOIS
"Calculabilité des langages".
Cours, Université de Grenoble.

[Woods] W.A. WOODS
"Transition network grammars for natural language analysis".
CACM, Octobre 1970.

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 Avril 1974,

VU les rapports de présentation de Messieurs :

- J. COURTIN, Professeur à l'Université des Sciences Sociales de GRENOBLE
- V. JOLOBOFF, Ingénieur à la Société Européenne de Mini Informatique et de Systèmes - GRENOBLE -

Monsieur Blaise M A T H I E U

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Génie Informatique".

Grenoble, le 9 Septembre 1980

Le Président de l'I.N.P.G.

Ph. TRAYNARD
Président
de l'Institut National Polytechnique

P.O. le Vice-Président,