N° d'ordre : 1335

50376
1986
87

THÈSE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

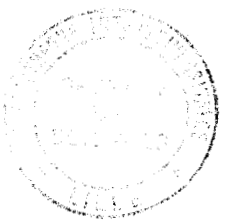
pour obtenir le titre de

DOCTEUR DE 3ème CYCLE

par

Patrick ONANGA

ETUDE D'UN EDITEUR TRANSCRIPTEUR CONDITIONNE BRAILLE PAR APPLICATION DE LA THEORIE DES RESEAUX DE TRANSITION



Soutenue le 24 Juin 1986 devant la Commission d'Examen

Membres du Jury :	L.	RACZY,	Président
	J.P.	DUBUS,	Rapporteur
	L.	AVAN,	Examinateur
	H.	VAN THIEN,	Examinateur
	M.	BONIFACE,	Examinateur
	A.	LEBRUN,	Examinateur
	Y.	MOSCHETTO,	Examinateur

A mes parents,

A tous ceux qui m'aident
sur le sentier,

R E M E R C I E M E N T S

Ce travail a été effectué au sein de l'équipe "Modélisation Instrumentation Electronique" du Centre de Recherche Sciences de Matériaux et Techniques de Construction" (CRESMAT) de l'Université des Sciences et Techniques de Lille Flandres Artois.

Je tiens à exprimer ma très sincère reconnaissance envers Monsieur le Professeur J.P. DUBUS qui m'a confié le thème de cette étude, pour son aide constante et efficace à la réalisation de ce travail. Je le remercie de la confiance qu'il a investie en moi, ainsi que du soutien qu'il m'a apporté.

Je suis très reconnaissant à Monsieur le Professeur RACZY pour l'honneur qu'il me fait de présider la commission d'examen et de juger mon travail.

Je remercie vivement Monsieur le Professeur A. LEBRUN, père du CRESMAT, dont il est resté en quelque sorte un directeur spirituel, pour sa présence à ce jury.

Messieurs les professeurs H. VU THIEN et L. AVAN du C.N.A.M. Paris, me font l'honneur de juger mes travaux. Je leur en suis vivement reconnaissant.

Mes remerciements vont également à Monsieur Y. MOSCHETTO, Directeur du C.T.B. INSERM de Lille, Monsieur P. DELECROIX, Chef du département Génie Electrique de l'I.U.T. de Lille, et Monsieur M. BONIFACE, Professeur à Lille II, pour l'intérêt qu'ils ont bien voulu porter à ce travail et pour leur participation à la commission d'examen.

Mes sympathies vont à Messieurs R. LEGAI, Directeur du Service Braille du CRDP et à son équipe, P. PARENT et F. WATTRELOT qui, par leur collaboration efficace m'ont encouragé pour l'aboutissement de ce travail.

Qu'il me soit permis d'associer à ces remerciements, Messieurs BENJELLOUN, DEVLAMINCK, WAUQUIER, DRUON, TABOURIER et WACRENIER pour la sympathie qu'ils m'ont témoignée.

J'exprime enfin ma reconnaissance à tous les membres du laboratoire ainsi qu'à l'ensemble du personnel technique et administratif, notamment à Mesdames R. CASTEGNIER, F. HENRY pour la frappe de ce document et Monsieur J.P. DEHORTER pour l'impression du manuscrit.

SOMMAIRE

-:-:-:-:-:-:-:-:-:-

N° de page

INTRODUCTION 1

CHAPITRE I : DÉFINITION D'UN SYSTÈME D'AIDE À LA COMMUNICATION ECRITE VOYANT-AVEUGLE 3

I-1- INTRODUCTION 4

I-2- RAPPEL SUR LES PRINCIPES DE L'ECRITURE EN RELIEF 4

I-2-1- Le relief Braille 4

I-2-2- Le Braille intégral 6

I-2-3- Le Braille abrégé 6

I-2-4- Notion de graphème et de codage Braille 7

I-3- DIFFERENTS SYSTEMES AUTOMATIQUES DE COMMUNICATION VOYANT-AVEUGLE... 8

I-3-1- La machine de Kurtweil..... 8

I-3-2- La machine Delta..... 9

I-3-3- I.P.E. Braille converter (Université de Stuttgart)..... 10

I-3-4- Le système Logibraille 11

I-3-5- Le système Lectobraille..... 12

I-4- DEFINITION DES FONCTIONS TEMPS REEL DE TRAITEMENT DE TEXTE BRAILLE. 13

I-4-1- Caractéristiques d'un système d'aide à la communication écrite voyant-aveugle 13

I-4-2- Définition du procédé d'édition des textes Braille..... 15

I-4-3- Les contraintes liées aux traitements de textes..... 16

I-4-4- Fonctions de traitement de texte et particularités de la syntaxe Braille intégral 17

I-4-4-1- Examen de la fonction de saisie 19

I-4-4-2- Examen des fonctions de traitements et de révision conditionnés Braille 21

I-5- CONCLUSION 23

CHAPITRE II : TRAITEMENT AUTOMATIQUE DES LANGUES NATURELLES, 24

II-1- INTRODUCTION 25

II-2- LE MODELE MORPHOLOGIQUE 26

II-2-1- La segmentation 26

II-2-2- Classification des mots 27

.../...

.../...

N° de page

II-3- LE MODELE SYNTAXIQUE 28
II-3-1- Les analyseurs généraux 29
II-3-2- Les analyseurs particuliers 29
II-3-3- L'analyse des dépendances..... 29
II-3-3-1- Le système PIAF 30

II-4- LE MODELE SEMANTIQUE 32
II-4-1- Les grammaires de cas 33

II-5- CONCLUSION 35

CHAPITRE III : LE MODÈLE SYNTAXIQUE D'ANALYSE DES TEXTES LIBRES 36

III-1- INTRODUCTION 37

III-2- DEFINITIONS PRELIMINAIRES..... 37

III-3- GRAMMAIRES, LANGAGES ET AUTOMATES..... 39
III-3-1- Grammaires et langages formels..... 39
III-3-1-1- Définition 39
III-3-1-2- Hiérarchie des grammaires..... 40
III-3-2- Les automates..... 44
III-3-2-1- Les automates d'états finis..... 44
III-3-2-2- Les automates à pile de mémoire..... 47

III-4- L'ANALYSE SYNTAXIQUE 48
III-4-1- L'Analyse syntaxique des langages réguliers 48
III-4-2- L'analyse syntaxique des grammaires context-free 50

III-4-3- Les réseaux de transition 55
III-4-3-1- Les réseaux BTN 55
III-4-3-2- Les réseaux A.T.N..... 57

III-5- LES RESEAUX SYNTAXICO-SEMANTIQUE A NOEUDS PROCEDURAUX (R.N.P.)..... 57

CHAPITRE IV : LES RESEAUX DE TRANSITION : APPLICATION A LA DESCRIPTION ET A LA MODELISATION DES FONCTIONS DE L'EDITEUR-TRANSCRIPTEUR NOIR CONDITIONNE BRAILLE 58

IV-1- INTRODUCTION 59

IV-2- LA FONCTION D'EDITION 59
IV-2-1- Le processeur de supervision 60
IV-2-2- Le processeur de gestion clavier 60
IV-2-3- Le processeur éditeur 63

N° de page

IV-2-3-1- La procédure "traitement" 63
IV-2-3-2- La procédure visualisation 73

IV-3- DESCRIPTION DE LA FONCTION DE TRANSCRIPTION BRAILLE INTEGRAL 76
IV-3-1- Examen des différents cas de conversion de syntaxe noir-Braille intégral 76
IV-3-2- Notations 79
IV-3-3- Enumération des différentes règles de transcription 80
IV-3-4- Exposé du principe d'analyse syntaxique d'une chaîne..... 84
IV-3-5- Analyse syntaxique des différents types de chaînes 90

IV-4- DESCRIPTION DE LA FONCTION DE TRANSCRIPTION BRAILLE ABREGE 95
IV-4-1- Examen des différents cas de conversion de syntaxe noir-Braille abrégé 95

IV-4-2- Analyse syntaxique d'une chaîne dictionnaire 96
IV-4-3- Analyse syntaxique d'une chaîne contraction 101

IV-5- CONCLUSION 106

CONCLUSION 107

BIBLIOGRAPHIE 109

INTRODUCTION

La formation d'un aveugle et son intégration dans la vie posent divers types de problèmes. Un des plus importants, avec celui du transport autonome, est la difficulté de "communication à mémoire" (communication écrite dans le sens voyant-aveugle).

Des travaux mettant en oeuvre des techniques informatiques ont été mené pour améliorer la communication écrite voyant-aveugle. Parmi ces travaux, ceux menés au Laboratoire de Mesure Automatique de Lille ont conduit à la réalisation d'un procédé appelé LOGIBRAILLE permettant à un voyant ne connaissant pas le Braille d'éditer par saisie dactylographique, les textes noirs et leur transcription en Braille Intégral et Abrégé.

Le mémoire que nous présentons, décrit la démarche adoptée pour définir et réaliser, pour un coût minimal, un nouvel éditeur-transcripteur dont le mécanisme de base est simulé sur le modèle des réseaux de transition.

Dans un premier chapitre, après avoir brièvement rappelé les principes de l'écriture en relief d'une part, et d'autre part certains systèmes d'aide à la communication voyant-aveugle, nous essayons de définir le type de saisie pouvant répondre au problème posé par les contraintes liées aux traitement de texte et aux particularités de la syntaxe Braille.

Nous examinons dans le deuxième chapitre, quelques modèles formels d'analyse des textes libres susceptibles de fournir une solution théorique au problème de la transcription Braille et mettons en lumière les difficultés que pose leur emploi.

Dans le troisième chapitre nous déterminons parmi les méthodes de l'approche syntaxique, la moins combinatoire s'adaptant à notre problème.

Nous décrivons enfin dans le quatrième chapitre la démarche conceptive des fonctions constituants le nouvel éditeur-transcripteur Braille qui a été réalisé.

CHAPITRE I : DÉFINITION D'UN SYSTÈME D'AIDE À LA COMMUNICATION ÉCRITE VOYANT AVEUGLE.

I-1- INTRODUCTION

Après un bref rappel du principe de l'écriture en relief et de certains systèmes automatiques d'aide à la communication voyant-aveugle, nous décrivons les fonctions temps réel de traitement de textes Braille.

Dans ce cadre, nous examinons les paramètres susceptibles d'obtenir, pour une fonction de coût minimale, un système de traitement de texte Braille utilisable par un voyant non spécialiste de cette syntaxe.

Les conditions de réalisation d'un tel système exige cependant de prendre en compte d'une part les contraintes liées au traitement de texte, et d'autre part de l'incidence des particularités de la syntaxe Braille intégral sur les fonctions de traitement de texte classique.

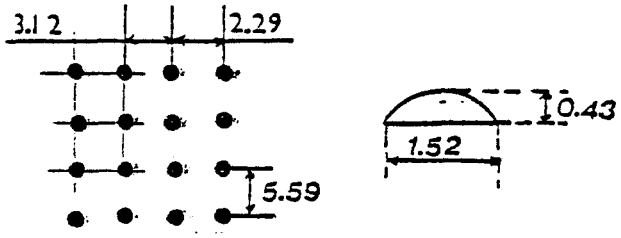
I-2- RAPPEL SUR LES PRINCIPES DE L'ECRITURE EN RELIEF

La transmission de l'information écrite par supports visuels (livres, dessins) interposés, constitue un handicap pour les déficients visuels : disponibilité d'un lecteur, interprétation des textes, etc...

Afin de pallier à ces inconvénients, des méthodes de lecture de livre en relief par saisie tactile ont été proposées. Le procédé le plus répandu de nos jours est le relief Braille.

I-2-1- LE RELIEF BRAILLE

Le relief Braille, du nom de son inventeur, est basé sur le codage des caractères orthographiques à l'aide de matrice 6 points (figure I-1-a) qui peuvent être mis en relief par diverses techniques[2]. Dans le cas de l'embossage mécanique ou manuel chaque caractère Braille est poinçonné de droite à gauche et en sens inverse de la convention numérique de repérage des points en relief (figure I-1-b).



(a):Caractère Braille embossé selon le format préconisé par Meyers, Ethington, et Ashscroft

(b):Caractère Braille compté dans le sens de l'écriture

Figure I-1

Les combinaisons de ces 6 points génèrent les 64 caractères (espaces compris) de l'alphabet Braille (figure I-2).

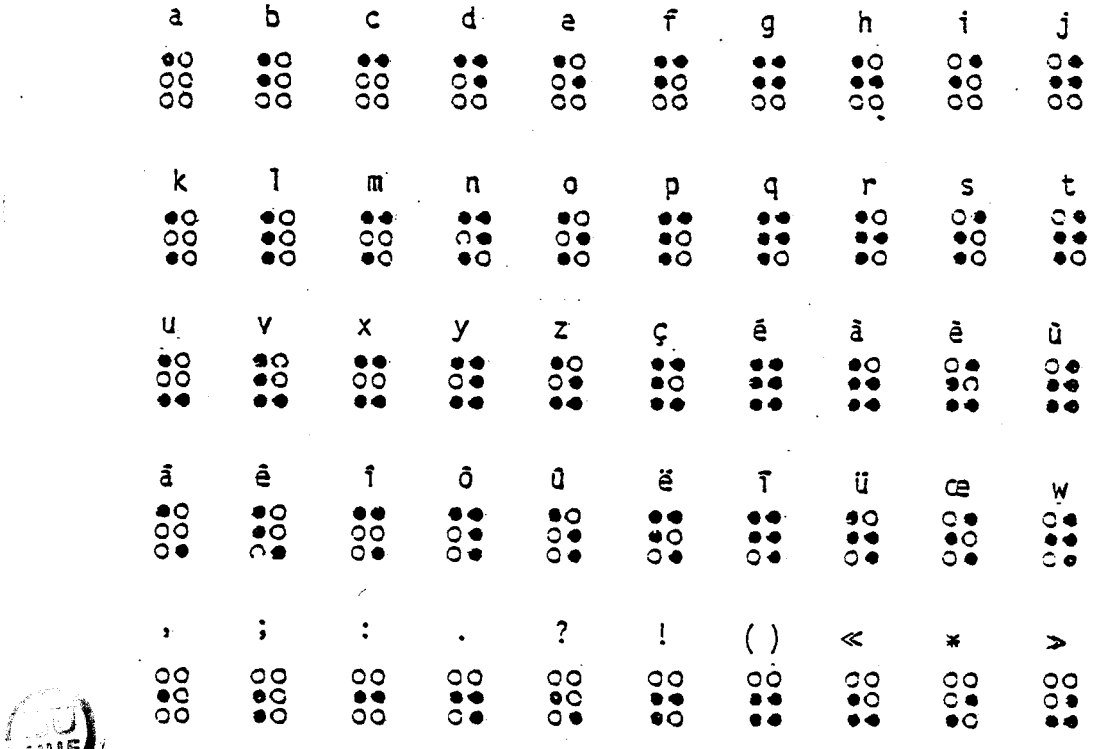


Figure I-2

Deux modes de codifications : le Braille intégral et le Braille abrégé, permettent de représenter l'ensemble de la syntaxe noire.

I-2-2- LE BRAILLE INTEGRAL

L'édition d'un texte noir nécessite plus de 64 caractères distincts. En Braille intégral, on utilise tous les éléments de l'alphabet Braille et des conventions particulières d'association de ces éléments pour représenter l'ensemble de la syntaxe noire.

Par exemple, la figure I-3 représente le codage Braille intégral des nombres.

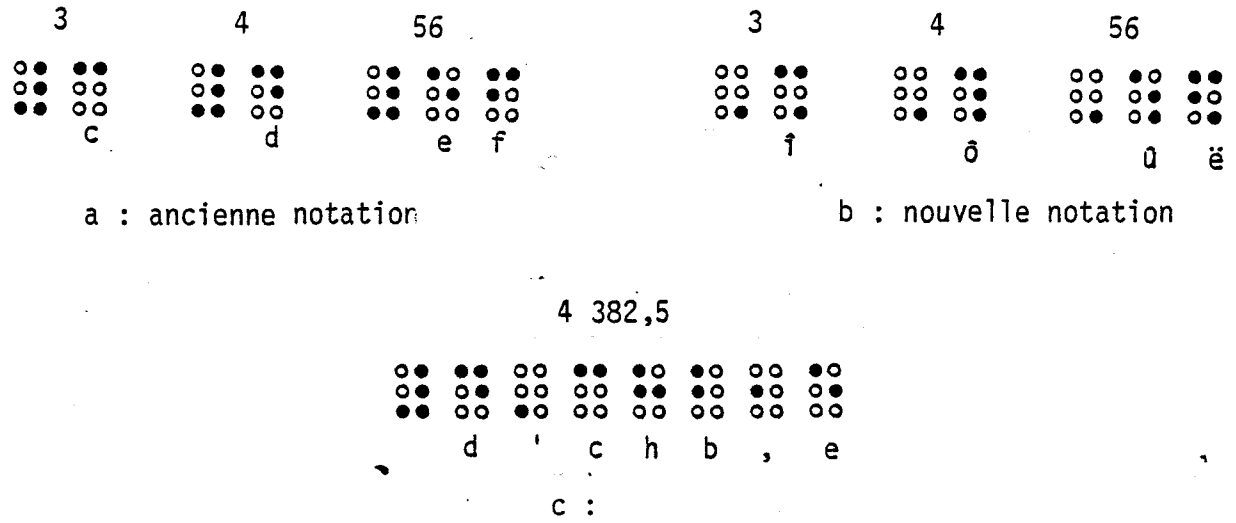


Figure I-3

Un tel procédé utilisant plusieurs caractères de l'alphabet Braille pour le même symbole noir, entraîne une augmentation de la longueur du texte Braille (on compte en général 3 pages de Braille intégral pour une page de texte noir). D'autre part, la surface occupée par le relief Braille augmente le volume des ouvrages. Pour tenter de remédier à ces inconvénients, Maurice de SIZERANNE a mis au point un procédé de compression par le codage appelé Braille abrégé.

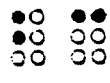
I-2-3- LE BRAILLE ABREGE

En Braille abrégé, on fait correspondre suivant des règles orthographiques déterministes, un graphème Braille à certains groupements de lettres du texte noir.

Le système d'abréviation est spécifique à chaque langue. Pour la langue française, il a été définitivement précisé dans "l'Abrégé orthographique étendu" publié par l'association Valentin Hauy[1].

La transcription en Braille abrégé nécessite de connaître :

- l'ensemble des 63 codes Braille,
- les règles de présentation d'un texte Braille,
- le dictionnaire Braille constitué de mots et locutions qui ont été abrégés, car d'un emploi fréquent (exemple figure I-4),
- un ensemble de contractions associées à leurs règles d'utilisation.

"beaucoup" :  Appartient à la liste des mots du dictionnaire

b c

"inondation" :  Décomposé en contractions

(in)(on)(d)(ation)

Figure I-4

L'utilisation du Braille abrégé par un tiers des aveugles seulement, du fait de sa complexité, nécessite la connaissance préalable du codage Braille intégral. Ce codage permet d'une part de réduire le volume des ouvrages (environ 30 % par rapport au Braille intégral), d'autre part d'augmenter la rapidité de lecture et d'écriture d'un texte Braille.

I-2-4- NOTION DE GRAPHEME ET DE CODAGE BRAILLE

A l'instar des 2 modes de codification Braille, il convient de préciser les termes qu'englobe l'écriture Braille.

Nous appellerons graphème Braille la représentation graphique Braille d'une chaîne "noire". En effet, chaque graphème Braille peut être identifié soit par le repérage numérique des points en relief qui le constituent, soit par un code ASCII traduisant de la même manière la position des points en relief (figure I-5).

Exemple :

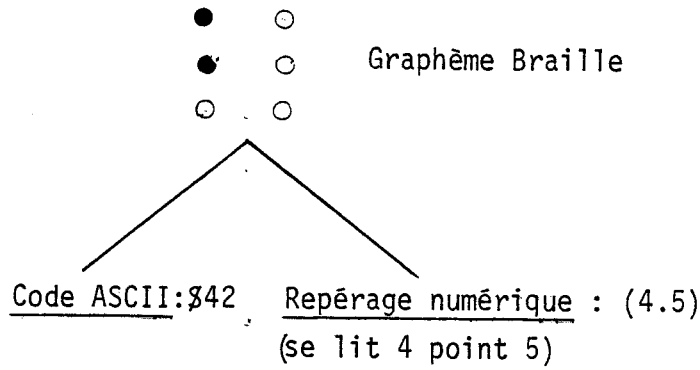


Figure I-5

Nous appellerons codage Braille les différentes règles de transcription d'une chaîne orthographique en code ASCII des graphèmes correspondants (figure I-6).

Exemple :

relief Braille	code Braille	signification après codage Braille
␣ ⠃ ␣	$42	lettre b (62) (intégral)
␣ ⠃ ␣	$42	mot (bien) (abrégé)
␣ ⠆ ⠃ ␣	$23,$42	lettre B (42)

Figure I-6

I-3- DIFFERENTS SYSTEMES AUTOMATIQUES DE COMMUNICATION VOYANT-AVEUGLE

Nous exposons dans ce paragraphe certains systèmes caractéristiques de l'évolution technologique dans le domaine de la communication voyant-aveugle.

I-3-1- LA MACHINE DE KURTZWEIL[70]

La machine de Kurtzweil convertit les textes imprimés de tous formats, tailles, styles en chaînes codées restituables sous forme visuelle ou sonore. Le système pour aveugle se compose de deux parties :

- Une partie "Reconnaissance des caractères" : le texte placé sur une vitre est saisi automatiquement ligne par ligne, par une colonne de photorécepteurs dont les déplacements dans les 4 directions sont assurés par un système mécanique.

Les caractères reconnus, par comparaison à un modèle, sont représentés par un certain nombre de caractéristiques topologiques (segments, concavité, angles...).
En cas d'ambiguité, une procédure est prévue par l'utilisation du contexte.

- Une partie "Synthétiseur" : cette partie réalise la synthèse vocale des chaînes reconnues. Elle comporte les deux modes possibles de fonctionnement en "épellation" ou en "prosodique".

La machine de Kurtzweil pour aveugle est l'aboutissement d'un ensemble de travaux sophistiqués dans les deux domaines de la reconnaissance des formes et de la synthèse vocale. Un inconvénient considérable en est le prix (400.000 F environ).

I-3-2- LA MACHINE DELTA[75]

Conçue à l'ENSHET[ ], la machine DELTA (Dispositif Electronique de Lecture de Textes pour Aveugles) est un système portatif permettant aux aveugles d'accéder directement aux textes imprimés par l'intermédiaire d'une sortie Braille ou sonore.

DELTA se compose essentiellement de trois modules :
- Un module de reconnaissance constitué d'une micro-caméra, d'un système de guidage et d'une unité de prétraitement.

La microcaméra comprend un composant CCD 100x100 assisté de circuits spécifiques pour le contrôle à grande vitesse (100 trames/seconde) et l'amélioration des performances du prétraitement.
Un couple d'aiguilles vibrantes contrôlées par l'unité de prétraitement et montées sur la microcaméra, guide l'handicapé visuel à suivre les lignes du texte.

L'unité de prétraitement réalise trois fonctions :

- codage du signal en mots binaires avant leur transfert vers le processeur central en accès direct mémoire,
- synchronisation et contrôle des transferts avec la mémoire,
- analyse des formes, sélection et localisation de chaque caractère, extraction de paramètres utiles.

Un module de traitement permet la reconnaissance automatique de plusieurs fontes de caractères, excepté les caractères script ou italique.

Un module de sortie génère les caractères reconnus soit sous forme sonore (épellation des caractères identifiés), soit en tactile (afficheur tactile de 12 caractères Braille).

I-3-3- I.P.E. BRAILLE CONVERTER (UNIVERSITE DE STUTTGART)[70]

Le schéma général de cette machine est le suivant :

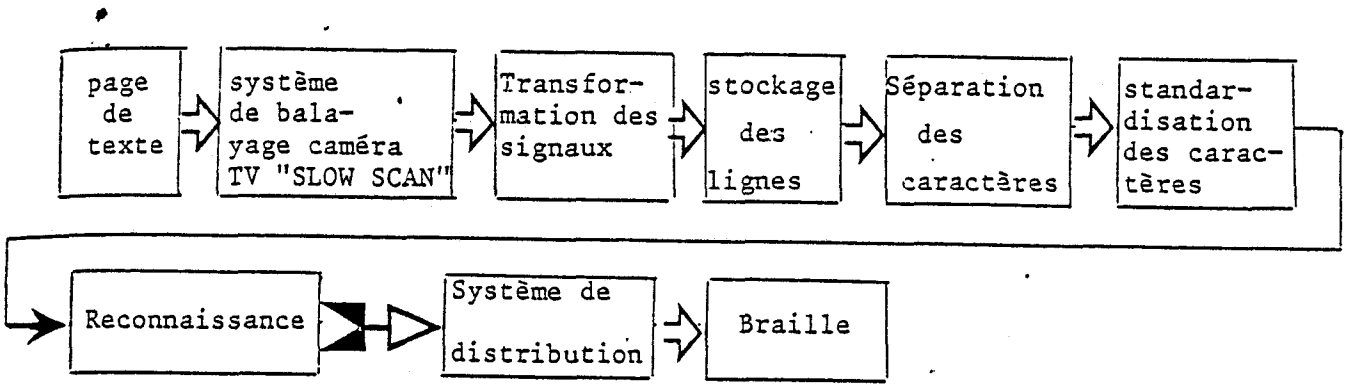


Figure I-7

Le texte est placé, immobile sur un verre spécial, l'emplacement correct étant indiqué par un signal sonore. La saisie des données s'effectue par un système de balayage électronique de la page avec une caméra TV spéciale "SLOW-SCAN". L'information contenue dans une demi-page (DIN A4) est échantillonnée en 704x1024 points binaires. Après filtrage du bruit, on procède à la détection et à la mise en mémoire d'une ligne. Chaque caractère est séparé et représenté dans une matrice 16x16. Les alphabets modèles de quelques polices sont stockés sur cassette et peuvent être mis en mémoire au moment

voulu. Les caractères lus sont ainsi reconnus par comparaison avec les modèles.

L'avantage de l'emploi d'une caméra de TV est l'élimination de la mémoire de 704x1024 bits qui est nécessaire à l'échantillonnage du texte. C'est le texte lui-même qui sert de mémoire (les inconvénients de ce système sont le coût et la lourdeur mécanique).

I-3-4- LE SYSTEME LOGIBRAILLE [2] [76] [77]

Conçu au Laboratoire de Mesures Automatiques de l'Université de Lille- Flandres-Artois, le Logibraille permet à un voyant non initié au Braille d'effectuer une saisie dactylographique d'un texte noir, ainsi que sa transcription temps réel en Braille intégral ou abrégé. Il se compose de deux modules (figure I-8) séparés :

- Un module éditeur - interpréteur qui analyse conformément à la syntaxe Braille les informations en provenance d'un clavier type machine à écrire ;

Les caractères faisant partie du texte à éditer sont affichés à l'écran dans les deux syntaxes avec ou sans demande d'informations supplémentaires, selon qu'il y a ambiguité ou non.

- Un module duplicateur qui permet à partir de l'enregistrement réalisé par l'éditeur-interpréteur de dupliquer le texte de deux manières : le texte s'inscrit soit sur une embosseuse (type SAGEM) avec possibilité de frapper recto ou recto-verso, soit sur un lecteur sensoriel moyennant une adaptation.

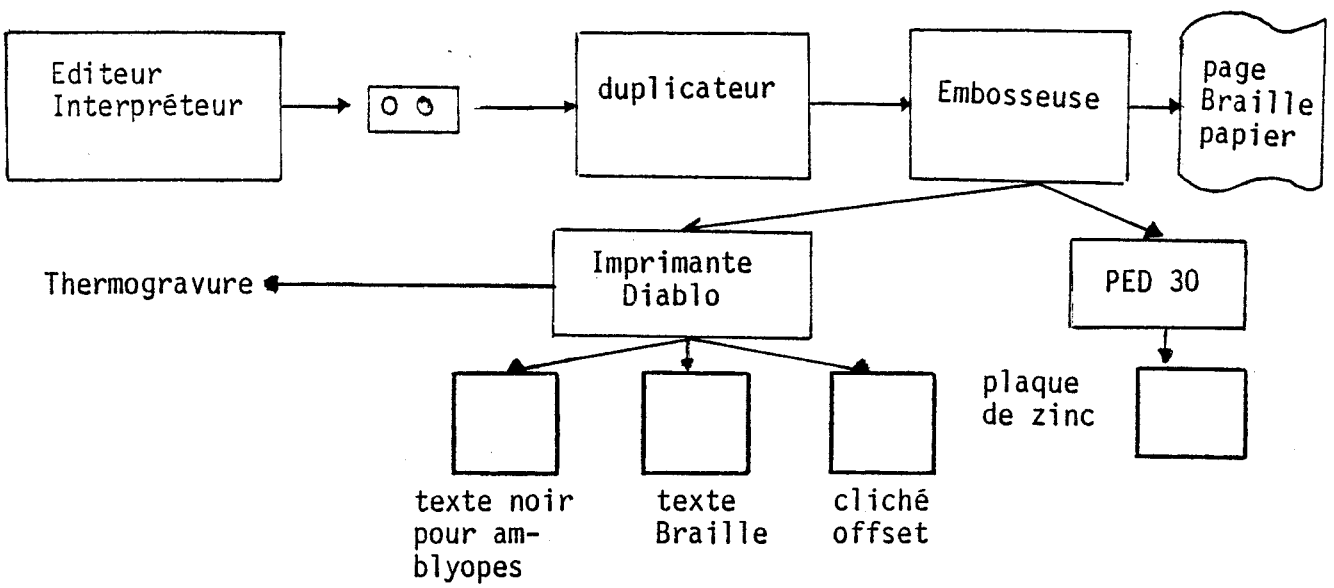


Figure I-8

I-3-5- LE SYSTEME LECTOBRAILLE [69] [78] [79]

Il permet à un voyant non initié au Braille de lire automatiquement un texte en relief Braille.

Il comporte (figure I-9) :

- un dispositif optique d'analyse d'un relief Braille, qui délivre après traitement et reconnaissance graphème par graphème les chaînes Braille codées,
- un logiciel de transcription inverse Braille intégral ou abrégé ou noir,
- un environnement microinformatique de visualisation des textes noirs après transcription et de stockage et restitution.

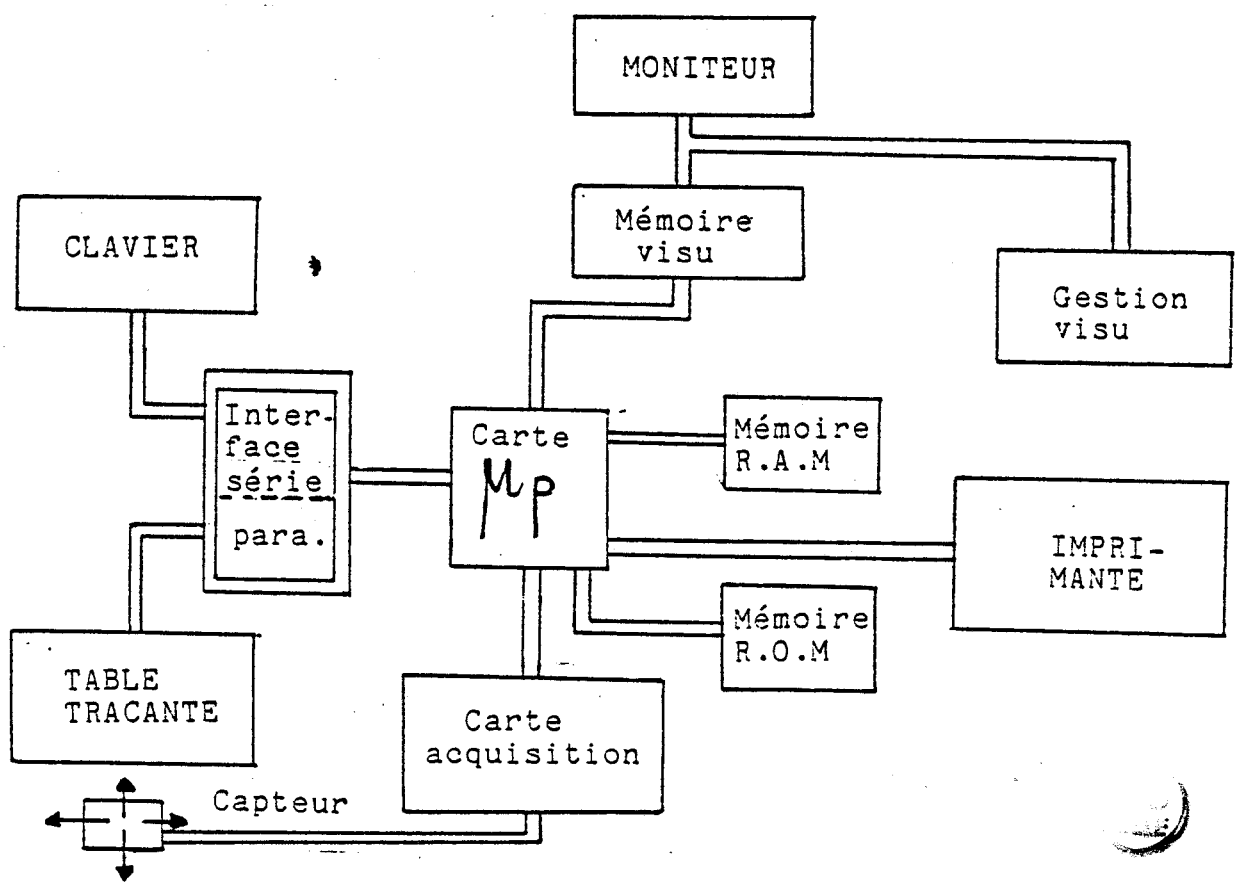


Figure I-9

I-4- DEFINITION DES FONCTIONS TEMPS REEL DE TRAITEMENT DE TEXTES BRAILLE

I-4-1- CARACTERISTIQUES D'UN SYSTEME D'AIDE A LA COMMUNICATION ECRITE VOYANT-AVEUGLE

L'examen des divers systèmes utilisant un traitement logique de l'information nous permet de dégager les paramètres intervenant lors de la conception d'une machine d'aide à la communication voyant-aveugle.

Le choix de ces paramètres est en effet très important pour le résultat final. Le type de technologie ou de logiciel utilisé a une influence directe sur les performances et le coût du système.

Il convient d'abord de distinguer :

- la machine individuelle portative (qui est principalement notre sujet d'intérêt),
- la machine collective, fixe, utilisable dans une bibliothèquepublique, par exemple.

Quant à la conception du système, on peut regrouper ses caractéristiques en trois fonctions essentielles (figure I-10) :

- une fonction d'entrée assure la saisie des textes ("noir" ou Braille) par dispositif doté de capteur (voyant-clavier pour une saisie dactylographique, caméra pour une saisie optique, microphone pour une saisie sonore).

Cette fonction effectue un prétraitement de la forme ou du caractère introduit.

- Une fonction de traitement effectue le codage et la transcription directe (ou inverse) grâce à un dictionnaire et des règles de transcription.
- Une fonction de sortie permet enfin, selon l'application, d'obtenir des documents sous plusieurs formes : soit en tactile (Braille), soit en acoustique (synthétiseur vocal).

La complexité des fonctions (notamment de la fonction de traitement) augmente le risque d'erreurs et diminue en général certaines souplesses d'emploi et la portabilité de la machine.

Enfin, le prix doit entrer en considération car une telle machine ne présente un intérêt que si les contraintes liées à sa diffusion et à son utilisation sont réduites.

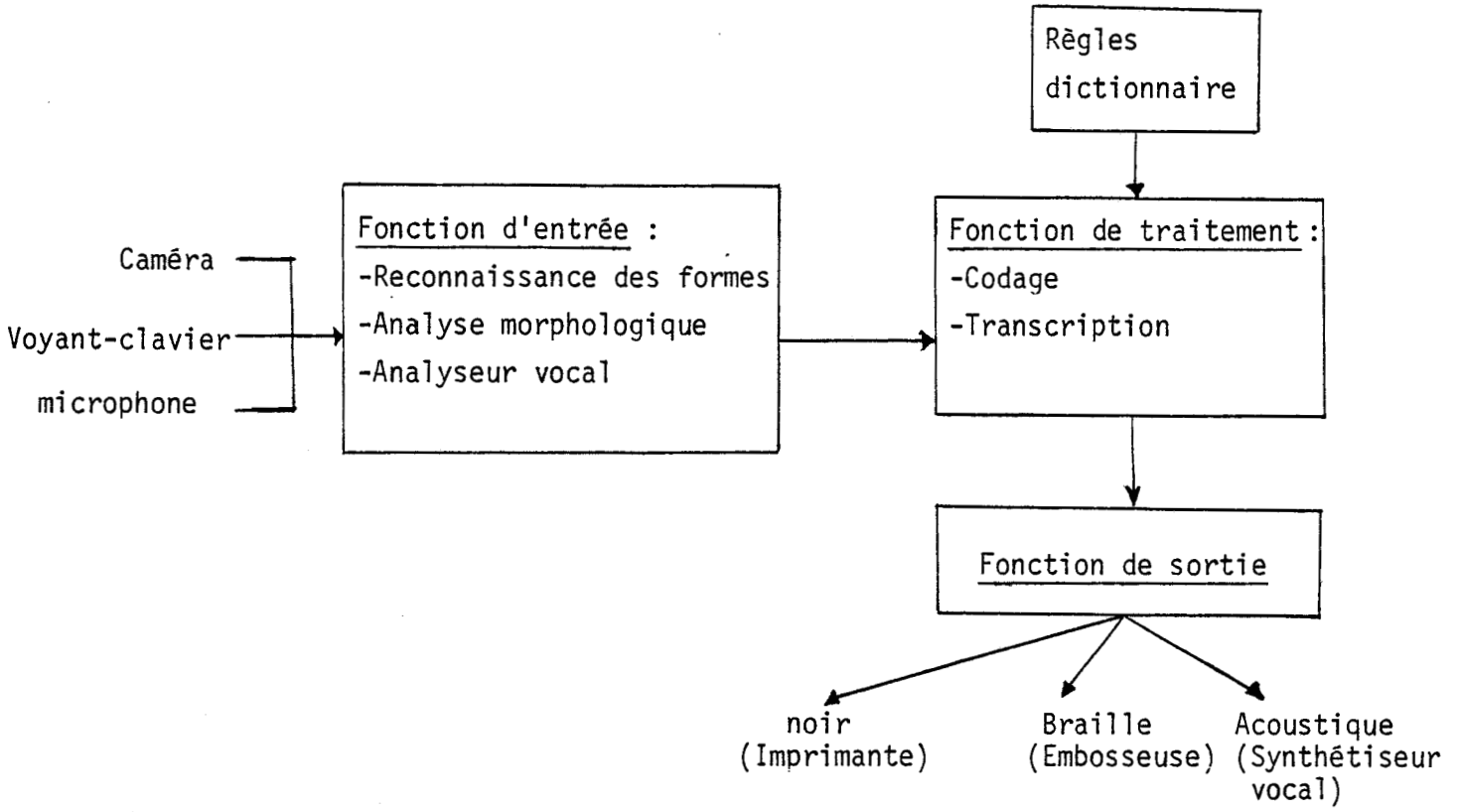


Figure I-10

Dans le cadre de l'évolution du système Logibraille, notre démarche consiste à repenser la fonction temps réel d'édition des textes Brailles. Pour cela, nous tentons compte tenu des objectifs que nous rappellons ci-dessous :

- d'optimiser les caractéristiques essentielles des systèmes ci-dessus mentionnés,
- de définir un mode d'édition qui répondrait aux contraintes d'utilisation de ce système par un voyant non spécialiste de la syntaxe Braille,
- disposer d'un modèle transportable de transcription mettant en oeuvre les connaissances issues de l'étude des langues naturelles.

Les objectifs de ce système de communication sont :

- accroître le champ de lecture des aveugles en multipliant le nombre de voyants susceptibles d'éditer des textes Braille. C'est ce qui motive l'utilisation d'un micro-ordinateur de grande diffusion,
- fournir un interface de traitement de texte "Braille" qui soit facile à employer par toute personne, quel que soit son niveau de formation,

- Rendre ce système le plusportable possible sur différentes machines et accessible pour un prix inférieur aux systèmes existants.

I-4-2- DEFINITION DU PROCEDE D'EDITION DES TEXTES BRAILLE

L'édition de textes Braille par un voyant ne connaissant pas le Braille nécessite de visualiser :

- soit à la fois le texte noir et sa transcription simultanée Braille intégral,
- soit uniquement le texte noir dont les contraintes de saisie correspondent au modèle de transcription.

La 1ère solution a l'avantage de donner à la fin de l'édition, un texte Braille conforme à la syntaxe Braille intégral et donc traduisible en Braille abrégé. Mais, elle entrainerait des solutions compliquées lors de la correction des textes.

Nous avons opté pour la deuxième solution qui comme le montre la figure I-11 consiste à :

- éditer en mode manuel le texte "noir" en vue de sa transcription Braille.

Le texte "noir" visualisé directement à l'écran durant la saisie représente alors la forme virtuelle du texte Braille. La mise en page est pilotée manuellement, en utilisant des touches spécialisées du clavier, qui déplacent un curseur visible à l'écran et définissent ainsi la position que devra occuper le caractère qui va être saisi. L'utilisateur peut tenir compte des résultats intermédiaires quand il donne une nouvelle commande,

- transcrire en mode programme le texte noir formaté Braille. Le texte Braille virtuel présent en mémoire, sert alors d'entrée au programme de transcription. Celui-ci analyse les commandes et transcrit automatiquement le document virtuel. Seul le résultat final est affiché et l'utilisateur ne peut intervenir pendant le déroulement du programme.

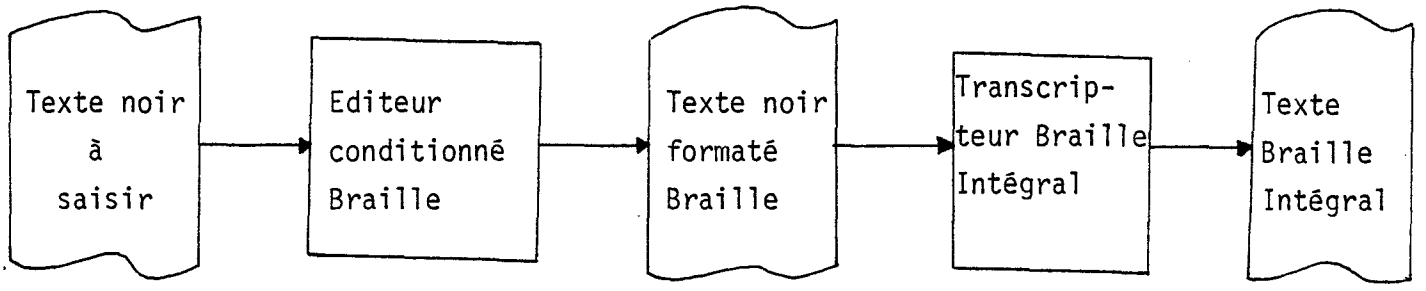


Figure I-11

Cette solution supprime les difficultés de la 1ère solution tout en gardant ses avantages. En effet, une modification sur le texte noir conditionné Braille est automatiquement reportée sur le texte Braille.

La difficulté d'une telle démarche tient aux contraintes liées aux traitements de textes, et à l'interprétation d'un code noir selon son contexte.

I-4-3- LES CONTRAINTES LIEES AUX TRAITEMENTS DE TEXTES

Les problèmes liés à l'archivage des informations textuelles et à leur traitement proviennent de la nature et de la structure des données textuelles ainsi que du volume des informations à traiter.

LE VOLUME

Le système doit gérer, outre le texte (environ 4200 caractères par page), mais aussi les informations permettant sa manipulation. Une telle quantité d'informations requière une gestion d'espace adressable suffisant, une hiérarchisation des mémoires centrales et périphériques, ainsi que l'existence de mécanismes optimaux de transfert de texte d'un point à un autre du système.

LA STRUCTURE

Sur les structures d'informations numériques interprétables par un système donné (codage et base de numération différents) sont définies directement les logiques de traitement et de manipulation ad hoc.

Par contre, les données textuelles se caractérisent par aucune structure privilégiée. Ceci entraîne souvent le recours à une logique définie sur le caractère : la logique de traitement de chaînes de caractères[22].

LA NATURE

Afin d'être communicable sous forme graphique, un texte doit être accompagné d'informations concernant sa présentation et son environnement. Ces informations secondaires aux textes doivent elles-mêmes être gérées par le système.
Nous appellerons à cet effet :

- caractéristiques externes : une liste de caractéristiques qui sont des propriétés extrinsèques, non liées au texte et à sa représentation, et pouvant être saisies et manipulées indépendamment du texte lui-même ;
- attributs : les aspects d'un document qui sont liés à sa présentation physique ;

Les attributs peuvent être partagés en deux classes :

. les attributs globaux marquent une propriété de l'ensemble du document. Ils sont liés à la structure logique de celui-ci (coupure syllabique, format ...) ;

. les attributs locaux sont les propriétés d'une sous-chaîne de caractères (soulignement, tabulation ...) indépendantes de la structure logique du texte. Ces attributs peuvent être insérés directement dans le texte.

I-4-4- FONCTIONS DE TRAITEMENT DE TEXTE ET PARTICULARITES DE LA SYNTAXE BRAILLE INTEGRAL

L'accès à un traitement de texte se fait par l'intermédiaire d'un menu hiérarchisé dont le schéma conceptuel est illustré par la figure I-12. Le menu définit le service à exécuter au moyen d'une suite d'éléments hiérarchiques dont le dernier prend ses valeurs parmi un ensemble de caractéristiques formelles et d'attributs.

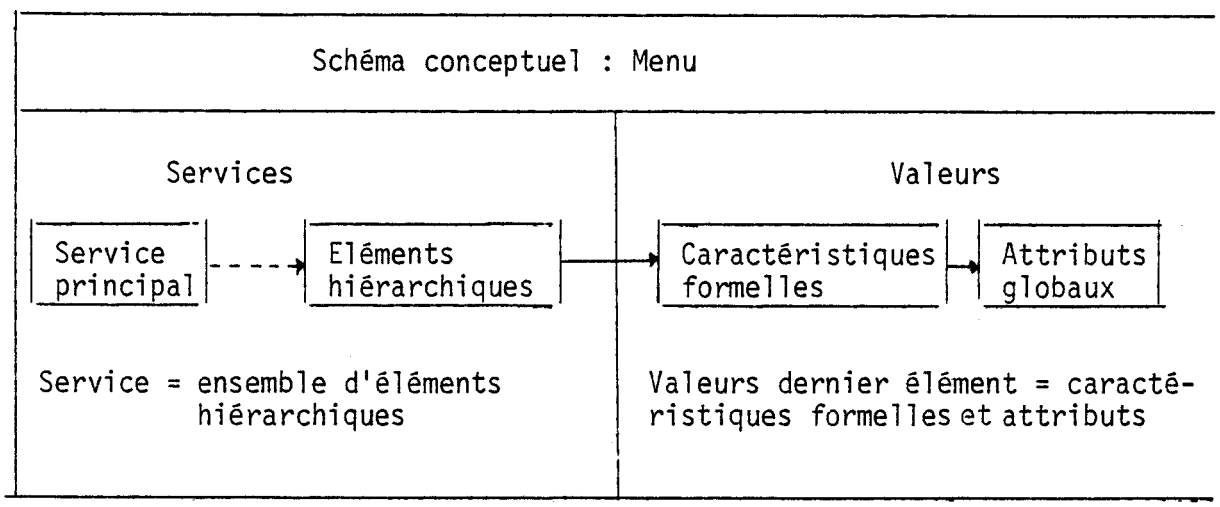


Figure I-12

La figure I-13 illustre les valeurs (titre, auteur, ...;coupure syllabique) relatives à l'édition d'un texte noir conditionné Braille.

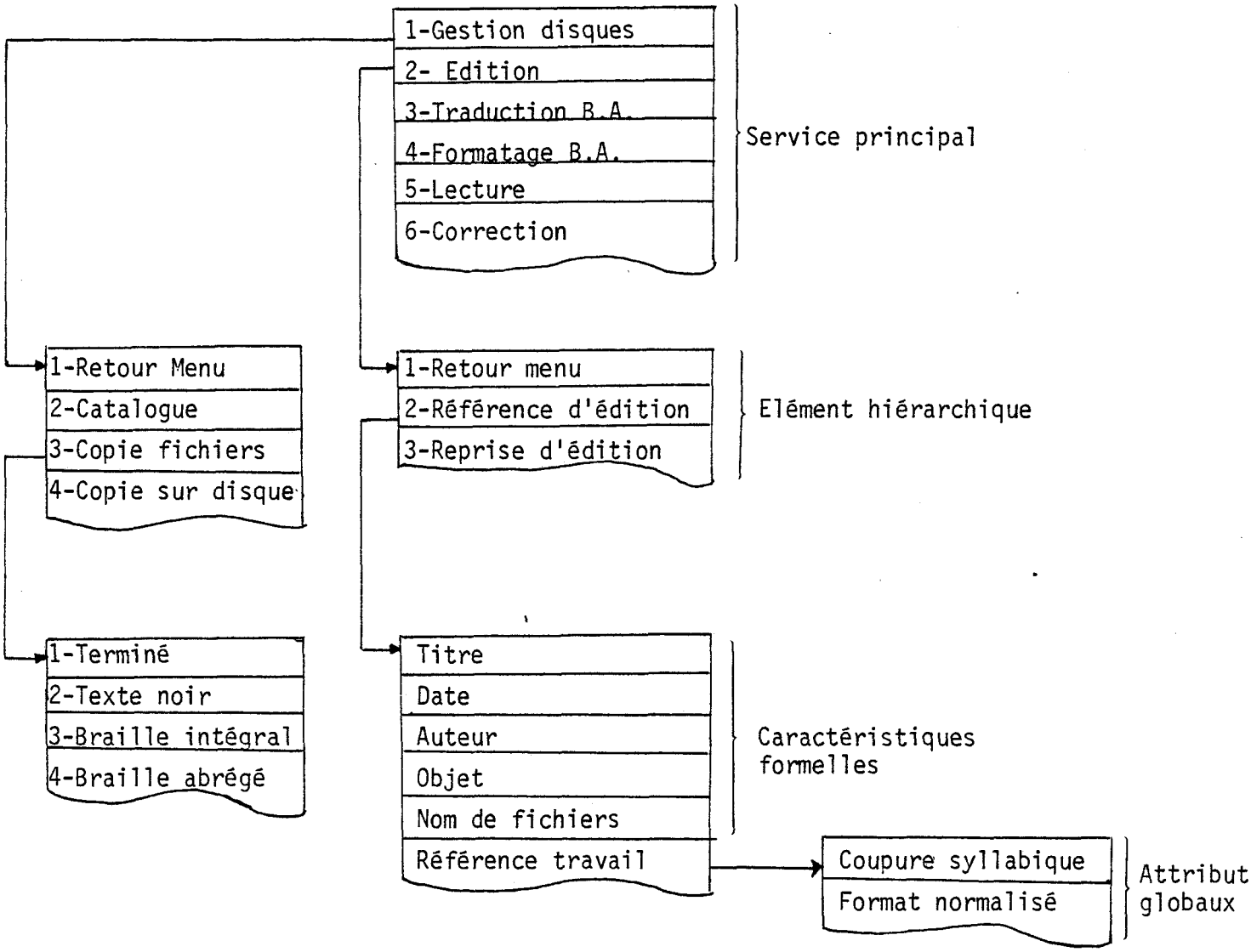


Figure I-13

Par la suite, l'utilisateur d'un traitement de texte se sert d'un certain nombre de fonctions que l'on peut répartir en 6 types (figure I-14) :

- les fonctions de saisie qui permettent de taper le texte au kilomètre et de décider les paramètres de mise en page provisoire (police de caractères , ...) ;
- Les fonctions de traitement qui concernent les différentes manipulations possibles à partir d'un document saisi et révisé ;
- les fonctions de révision qui donnent la possibilité de revenir sur le texte déjà saisi et de le corriger ;

- les fonctions d'impression qui fixent les paramètres de restitution du texte sur imprimante ;
- les fonctions de mémorisation qui rendent possible le stockage du texte présent en mémoire centrale sur disquette ;
- les fonctions de visualisation qui régissent l'édition sur écran.

Les particularités de la syntaxe Braille ainsi que les contraintes de l'architecture cible modifient essentiellement les caractéristiques des fonctions de saisie, de traitement et de révision.

I-4-4-1- Examen de la fonction de saisie

La fonction de saisie définit sur l'ensemble des codes ASCII disponible au clavier, ceux utilisables par la fonction de traitement :
- soit en code d'édition : il représente alors les caractères du texte noir (a,b,+,ñ,œ,...). Ces caractères sont tous pris en compte par la syntaxe Braille.
- soit en code d'aide à l'édition : ce sont des caractères de manipulation du texte (déplacement curseur, saut de ligne, ...).

L'accès direct au texte Braille par un spécialiste du Braille, à des fins de correction, nécessite la présence au clavier de caractères spécifiques du codage Braille. Exemple : La figure I-15 illustre pour le cas du tiret l'unicité d'une touche clavier pour trois caractères codés différemment dans la syntaxe Braille.

Compte tenu de l'absence au clavier de caractères nécessaires à certaines fonctions, il convient donc de réaliser un automate fini, transcodant les séquences d'appui des touches clavier en code ASCII du symbole recherché. Ce prétraitement d'états finis permet de réduire de nombreuses ambiguités pour la fonction de traitement. Celle-ci devient alors moins complexe.

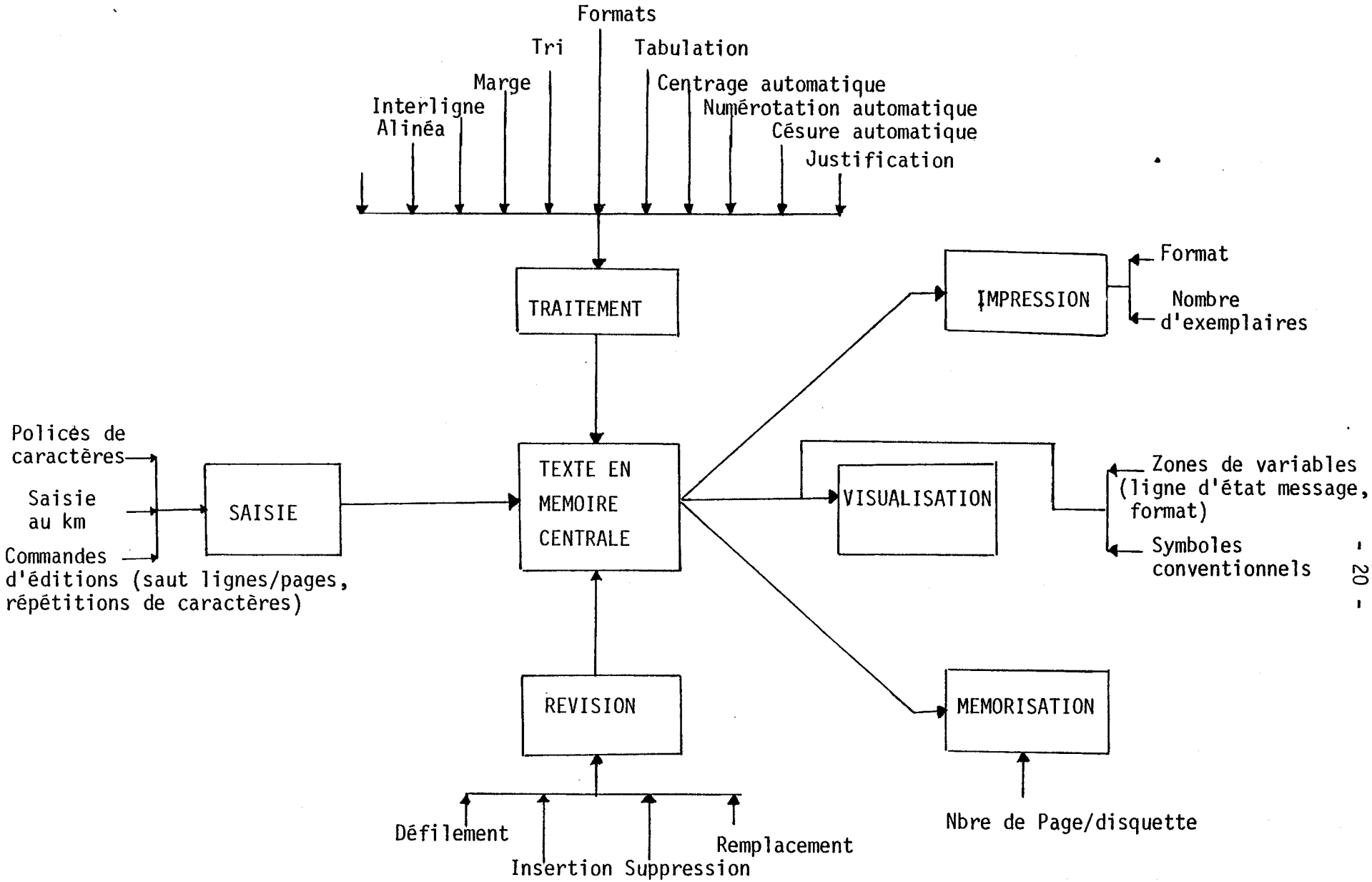


Figure I-14 : Structure fonctionnelle d'un traitement de texte.

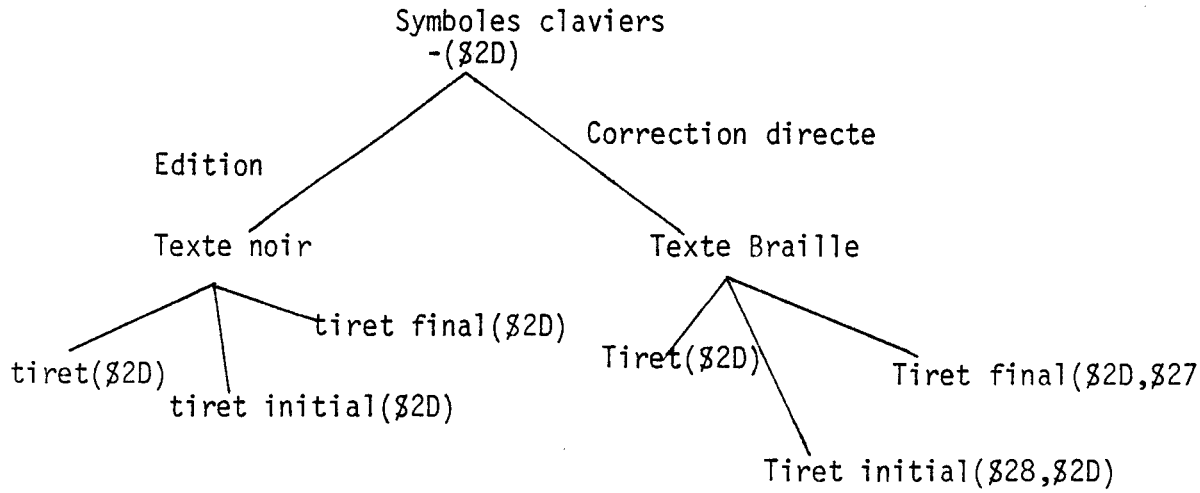


Figure I-15

I-4-4-2- Examen des fonctions de traitement et de révision conditionnés Braille

Ces fonctions interprètent les codes ASCII des caractères fournis par la fonction de saisie selon le codage du texte Braille intégral. Le format et le mode de saisie en fin de ligne de celui-ci sont fixés par les attributs globaux.

LE FORMAT : le format du texte Braille est imposé par celui des ouvrages édités. Il est variable suivant les éditeurs, mais les formats les plus couramment utilisés sont au nombre de 3. Il s'agit du format de l'embosseuse SAGEM (31 caractères par ligne, 23 lignes par page), le format mixte (29 caractères par ligne, 23 lignes par page), le format IN-OCTAVO (21 caractères par ligne et 22 lignes par page). Ce format conditionne celui du texte noir. Par ailleurs, le Braille intégral occupe en moyenne plus de place que le texte noir présent à l'écran. Par conséquent, celui-ci représentera le texte Braille virtuel présent en mémoire centrale. En particulier, aucun "caractère de contrôle" ni de "commande de présentation" n'apparaîtra dans le texte. La position du curseur indiquera l'endroit dans le texte où s'appliqueront toutes les commandes de traitement et de transcription.

CESURE DES MOTS ET JUSTIFICATION

En Braille intégral, la coupure des mots est identique à celle du texte noir. On peut donc comme sur les traitements de textes classiques, saisir un document au kilomètre. Les critères retenus pour effectuer cette coupure et sa réalisation sont exposés dans le chapitre IV.

Cependant, la conception d'un processeur de justification automatique remettrait en question, à terme : la complexité des fonctions de traitement et de révision ; la capacité mémoire des microordinateurs et le mode d'édition des textes Braille. Dans ce mode en effet, le processeur de justification automatique est conditionné par l'information de la ligne suivante (figure I-17).

Exemple : Pour souligner une liste de chaînes de caractères "noirs" en fin de ligne, l'utilisateur peut :

- soit se limiter à la ligne n pour les caractères "noirs" soulignés. Dans ce cas, le nombre de caractères Braille de cette ligne est augmenté de 2 unités,(figure I-17-a)
- soit chevaucher 2 lignes. Le nombre de caractères Braille de chaque ligne est alors incrémenté d'une unité (Figure I-17-b).

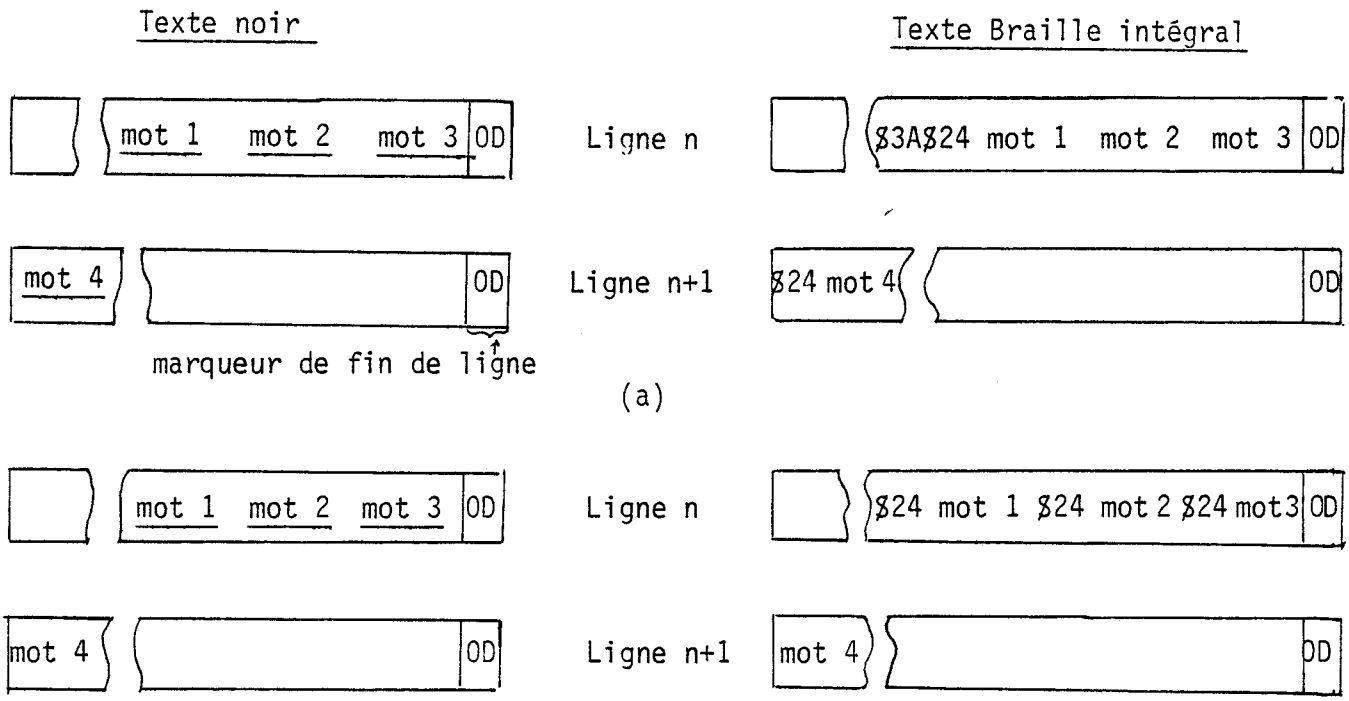


Figure I-17

Pour ces raisons, et contrairement aux traitements de textes classiques, il sera laissé à la discrétion de l'utilisateur le soin de gérer certains attributs locaux.

I-5- CONCLUSION

Dans le cadre de la communication écrite voyant-aveugle la définition des fonctions temps réel de traitement de texte Braille se caractérise par le mode d'édition des textes Braille. Ce mode consiste à :

- saisir un texte "noir" dont le profil qualitatif est celui du texte Braille,
- transcrire le texte obtenu en Braille intégral.

L'incidence des particularités de la syntaxe Braille sur les traitements de textes classiques se trouvent limitées de ce fait aux seules fonctions de saisie, de traitement et de révision.

L'identification de ces particularités permet de concevoir un éditeur pleine page de texte noir conditionné Braille. Celui-ci se présente dans ce cas, comme appartenant à la classe des interpréteurs dont l'étude fait en général appel aux connaissances issues de l'étude des langues naturelles que nous abordons dans le chapitre II.

CHAPITRE II : TRAITEMENT AUTOMATIQUE DES LANGUES NATURELLES.

II-1- INTRODUCTION

Le traitement des langues naturelles a fait l'objet de nombreuses études, dont la traduction automatique, le traitement de texte, la reconnaissance ou la synthèse de la parole sont les principales applications.

Les développements de l'intelligence artificielle initialement orientés vers la traduction automatique, permirent à celle-ci de passer de la notion de substitution des mots du texte source par les mots du texte cible, à la notion de "compréhension" du texte. Le traitement des langues naturelles est aujourd'hui considéré comme un processus cognitif de mise en relation d'un texte avec une base de connaissance.

Les nombreuses méthodes pouvant être à la base de l'analyse d'une phrase ou d'un texte peuvent être regroupées en trois grands modèles :

- le modèle morphologique,
- le modèle syntaxique,
- le modèle sémantique.

Ces trois modèles sont soit concurrents, soit complémentaires. Certains systèmes proposés mettent l'accent soit sur la syntaxe, soit sur la sémantique, soit sur une combinaison des trois types de modèles. Analyser une phrase, consiste donc à déterminer sa structure morphologique, syntaxique et/ou sémantique sous-jacente en fonction des règles syntaxiques ou de toutes autres formes de connaissances disponibles.

Nous donnons dans ce chapitre, sans entrer dans les détails mathématiques, des différents concepts de base liés à chaque modèle, les problèmes liés à leur emploi, ainsi que leurs limitations.

II-2- LE MODELE MORPHOLOGIQUE

La morphologie s'occupe d'une part de la segmentation d'une phrase en chaînes de caractères : les mots, et d'autre part de définir pour chacun de ces mots un certain nombre d'attributs (verbe, adjectifs, masculin ...).

II-2-1- LA SEGMENTATION

La segmentation consiste à découper une phrase en un ensemble de mots finis qui serviront ensuite aux autres modèles.

Exemple : Pour la phrase "il y a à ce propos une controverse", on pourrait déterminer quatre segmentations différentes :

Il y a à ce propos une controverse . n = 5

Il y a à ce propos une controverse , n = 7

Il y a à ce propos une controverse . n = 7

Il y a à ce propos une controverse , n = 9

où n = nombre de mots.

La détermination d'un mot peut se faire par l'intermédiaire d'un dictionnaire. On constate en effet dans la langue française que les mots sont composés de plusieurs éléments que l'on appelle suivant leur place dans les mots : préfixe, base ou racine, suffixe, désinence.

Exemple : Le mot "retrouvera" a pour :

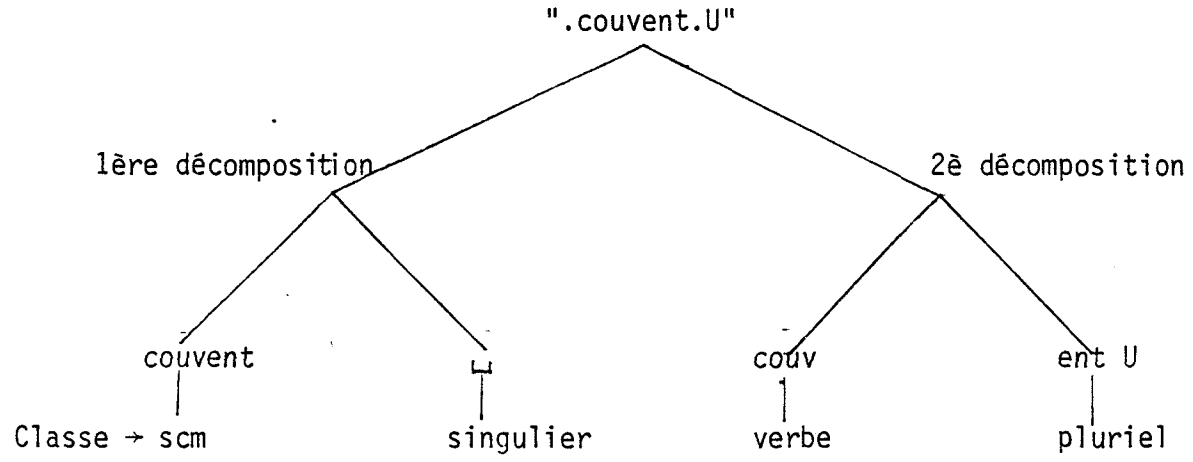
- préfixe "re" ;
- racine "trouv" ;
- suffixe "er" ;
- désinence "au" .

II-2-2- CLASSIFICATION DES MOTS

L'étude proprement dite, c'est-à-dire la définition du rôle des mots dans un texte est effectuée par les modèles syntaxiques et sémantiques. Ces modèles destinés à fournir une certaine "compréhension" du texte doivent pour cela travailler sur un ensemble fini et autant que possible réduit de symboles d'entrée.

Certains mots susceptibles de jouer un rôle identique dans leur agencements avec d'autres mots, sont donc regroupés en classes d'équivalence par le modèle morphologique. La définition des classes varie en fonction du traitement envisagé. A chaque classe, on associe une étiquette appelée catégorie lexicale (ou grammaticale). Les règles d'agencement portent alors sur ces étiquettes qui sont en nombre restreint et non plus sur les mots eux-mêmes.

Exemple : Le mot "couvent", comme le montre la figure II-1 peut soit appartenir à la classe substantif commun masculin, soit être un verbe du 1er groupe à la troisième personne du pluriel du présent de l'indicatif ou du subjonctif.



avec : scm = substantif commun masculin

Figure II-1

Si une même chaîne de caractères se présente en deux occurences dans un contexte de valeurs grammaticales différentes, on peut lui attribuer des étiquettes grammaticales différentes. Dans le cas où un mot appartient à plusieurs classes, on dit qu'il y a ambiguïté grammaticale ou homographie. Les homographies sont inhérentes au modèle car la classification et surtout le choix des classes sont arbitraires.

Exemple : La suspension de ce véhicule entre en résonnance entre 80 et 90 km/h.
Le premier "entre" est un verbe alors que le deuxième est une préposition.

La position des valeurs grammaticales des autres mots de la phrase permet seule de donner l'étiquette syntaxique du premier et du deuxième "entre".
La classification en "verbe" ou "préposition" pour le mot "entre" est totalement arbitraire. Elle est due au choix du modèle morphologique.

Le nombre d'ambiguité est lié à la finesse de définition des mots aussi bien pour la syntaxe que pour la sémantique. Le choix des classes oscille entre les trois extrêmes :

- Pratiquement une classe par mot, ce qui a l'avantage de supprimer presque toutes les ambiguités, mais devient totalement inutilisable par la suite.
- Une classification très précise. Nous multiplions alors plus ou moins les homographies.
- Une classification plus large, qui correspond à un regroupement des classes précédentes. Cette solution a le mérite de réduire le nombre d'homographies au risque de perdre de l'information.

II-3- LE MODELE SYNTAXIQUE

La syntaxe s'attache au caractère formel de la langue : l'ordre dans lequel les mots doivent être disposés pour qu'une phrase soit jugée correcte.

L'analyse syntaxique a donc pour rôle d'associer une structure à la phrase (ou toutes les structures possibles). Elle permet aussi de lever la plupart des ambiguités et si c'est nécessaire, elle contrôle la validité d'un texte. L'aspect formel de l'analyse syntaxique sera décrit au chapitre III. Nous donnons ici les

les caractéristiques des principaux analyseurs.

II-3-1- LES ANALYSEURS "GENERAUX"

Les analyseurs généraux s'appliquent à des grammaires "hors-contextes" presque quelconques. Leurs temps d'exécution est en général de l'ordre de n^3, où n est le nombre d'éléments de la phrase[68].
L'inconvénient de ce type d'analyseur, outre sa complexité, réside dans le nombre de structures proposées. Ce nombre est directement lié aux difficultés de modélisation de la grammaire d'une langue naturelle. Malgré certaines améliorations destinées à pallier ces inconvénients (grammaire à saturation[27] par exemple), la maîtrise de ce type d'analyse reste assez délicate et se pose le problème de convergence de la grammaire que l'on modifie en fonction des textes à analyser.

II-3-2- LES ANALYSEURS "PARTICULIERS"

Les analyseurs dits particuliers ne s'appliquent que sur une classe particulière de grammaire "hors-contexte"[27][67].
L'algorithme, s'il est trouvé, est alors très performant.
Dans le cas des langages de programmation, cette approche très intéressante est utilisée dans la plupart des compilateurs actuels du fait que le language que l'on désire compiler est bien définie et qu'il suffit de trouver une grammaire équivalente satisfaisant aux critères. Par contre, dans le cas des langues naturelles, le language n'est pas défini par sa grammaire. Les règles de grammaire ne sont qu'approximatives et on est amené à y apporter sans cesse des modifications à chaque analyse d'un nouveau texte. Comme il n'existe pas d'algorithme de transformation de grammaire, il est impensable dans ce cas, d'utiliser une méthode particulière.

II-3-3- L'ANALYSE DES DEPENDANCES

La structure syntaxique d'une phrase est jugée correcte lorsque les règles de grammaire du langage considéré sont vérifiées.

Les grammaires génératives (voire transformationnelles), sont des modèles intéressants pour la description et l'analyse de langues artificielles.
Par contre, dans le cas de l'analyse automatique des langues naturelles, ils posent certains problèmes. Ces difficultés de modélisation des langages naturelles ont conduit à certaines réalisations originales. Parmi celles-ci, il faut citer l'analyse de dépendances du système PIAF [27] et les travaux effectués à partir des ATN (Augmented Transition Networks) [35].
Cette dernière approche détaillée dans le 3è chapitre correspond à une présentation plus algorithmique des problèmes, la modélisation de la grammaire pouvant être représentée par un réseau d'automates.

II-3-3-1- Le système PIAF

Le système PIAF (Programme Interactif d'analyse du français) permet de faire l'analyse et la transduction d'un langage d'états finis de façon intéractive en fonction des paramètres fournis par l'utilisateur (figure II-2).

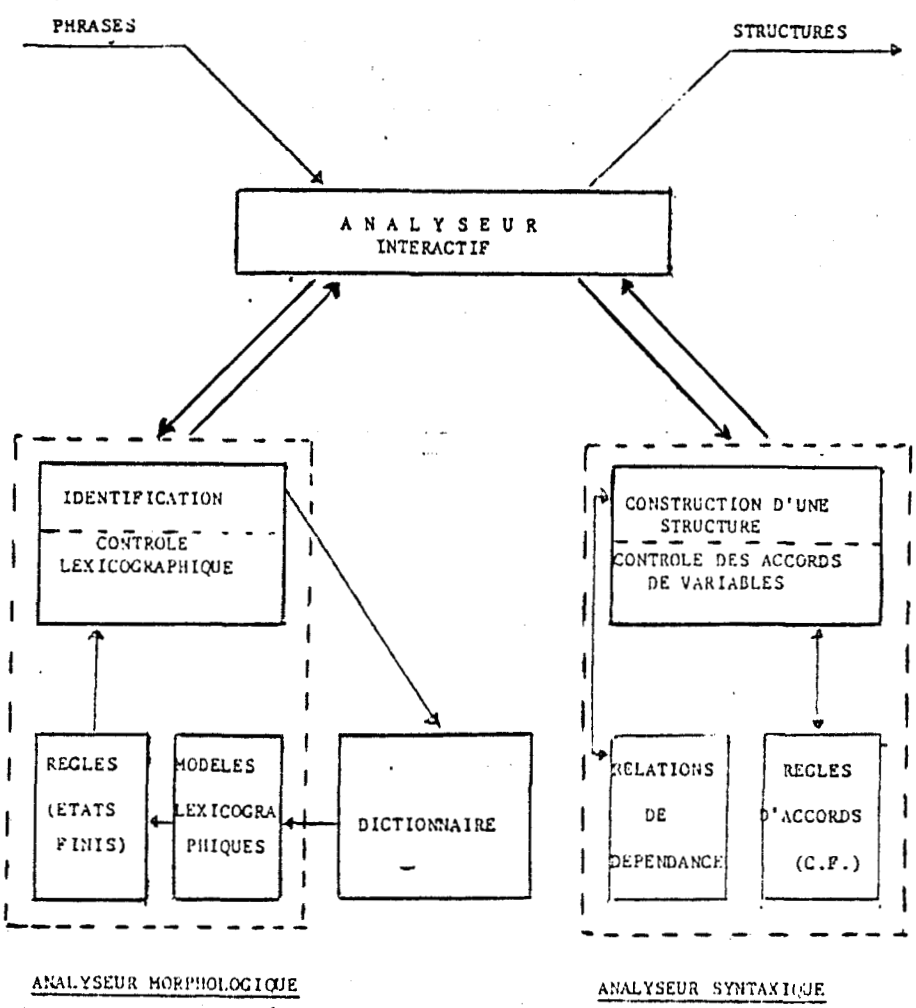


Figure II-2 : Schéma fonctionnel d'un système P.I.A.F.

Dans le système PIAF, l'analyseur n'essaie pas de construire une ou plusieurs structures représentatives du texte, mais, par contre, essaie d'éliminer toutes les structures impossibles. Ce mécanisme de filtrage permet d'exprimer simplement les relations de dépendances entre les mots, et d'éliminer une grande partie des structures "parasites" que produirait une grammaire générative.

La grammaire dans le système PIAF est représentée sous forme de relations de dépendances qui décrivent toutes les dépendances possibles entre un gouverneur et ses dépendants. Chacun de ces objets est représenté par sa catégorie lexicale.
Les dépendances sont notées sous la forme de valeurs numériques arbitraires. Ces valeurs numériques permettent d'imposer certains positionnements des différents dépendants d'un même gouverneur.

Considérons la phrase "Le petit chien noir mange la soupe", et les relations de dépendances :

(1) PHRA * VERB : =+1 ; Cette relation indique que le gouverneur principal est un verbe

(2) VERB * SUBC : =-15, +15 ; Cette relation précise qu'il peut y avoir à gauche (-15) ou à droite (+15) d'un verbe, un substantif.

(3) SUBC * ART : =-16 ; Cette relation précise qu'à gauche d'un substantif on peut trouver un article.

(4) SUBC * ADJ : =-15,+18 ; Cette relation précise qu'à gauche et à droite d'un substantif on peut trouver un adjectif qualitatif.

La structure obtenue avec les poids correspondants est décrite sur la figure II-3.

Afin de réduire le nombre de structures proposées, un module de contrôle d'accord de variables (figure II-2) complète les traitements précédents.

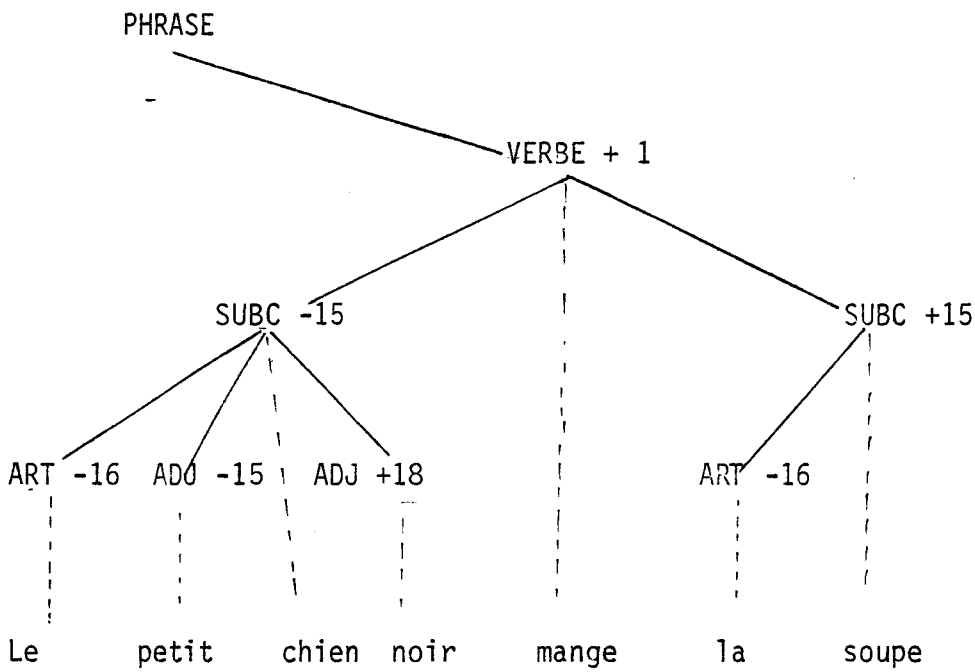


Figure II-3

II-4- LE MODELE SEMANTIQUE

L'analyse sémantique peut être considérée comme l'étape suivante dans la compréhension d'un texte. Elle permet d'extraire les concepts d'une partie d'un texte et d'en saisir ou de formaliser le sens. La difficulté de modélisation d'une langue naturelle entraîne de grandes difficultés dans le traitement général de textes libres. Certaines réalisations, dans le cadre d'une langue réduite à un domaine particulier, présentent cependant un grand intérêt ; ce sont entre autres, le système SAM et le modèle conceptuel de SHANK[73][74], le système de WILKS[71][72].

Nous décrivons les principes de représentations sémantiques basés sur la notion de grammaire de cas qui ont montré leur capacité à dégager la structure sémantique d'un énoncé en langage naturel.

II-4-1- LES GRAMMAIRES DE CAS

Introduites par FILLMORE en 1968, ces grammaires sont les descendantes des notions de "cas" de la grammaire traditionnelle.

Dans une grammaire de cas tout énoncé est traduit sous la forme d'un ensemble de cas reliés à un qualificatif central de la phrase, que l'on nomme prédicat. Généralement, ce prédicat correspond à un verbe, mais ce rôle peut éventuellement être rempli par un nom, un adjectif ou une préposition. L'essentiel du sens est contenu dans les mots pleins (nom, verbe, adjectif) ce qui permet d'éliminer un grand nombre de mots outils (article, préposition, adverbe etc...) remplacés par les qualifieurs sémantiques que sont les cas.

Une telle grammaire se caractérise principalement par le nombre de ses cas sémantiques. Ce nombre varie de 5 à 30 suivant les systèmes et la manière dont ils traitent ensuite leurs informations.

Exemple :

- Agent : ce qui cause l'évènement.
- Sujet : celui qui subit ou expérimente l'action.
- Objet ou thème : l'entité qui est au centre de l'action, le thème de l'évènement.
- Instrument : une entité qui a permis l'évènement.
- Source : l'origine de l'évènement.
- Lieu : le lieu où se situe l'évènement.
- Destination : lorsqu'il y a changement de lieu, le lieu de destination.
- But : le résultat de l'évènement.
- Trajectoire : la (ou les) étapes intermédiaires.
- Temps : Durée ou instants de l'évènement.

Le rôle de l'analyseur consiste alors à identifier le mot (ou parfois le groupe de mots) qui servira de prédicat , puis à reconnaître par leur position dans la phrase ou par les mots outils qui l'entourent, les différents cas associés à ce prédicat.

Exemple :

Dans la phrase : "La concierge ouvre la porte avec une clé"

on a :

- Prédicat : ouvrir
- Agent : la concierge
- Objet : la porte
- Instrument: une clé

Ces grammaires, permettant d'aller plus loin dans la compréhension des phrases que les analyseurs syntaxiques, sont cependant plus délicates à mettre au point.
La mise en oeuvre de ces grammaires est limitée par l'emploi de certaines contraintes sémantiques et de certaines caractéristiques syntaxiques spécifiques à chaque langue.

Exemple :

Pour la phrase : "Jean a mangé une glace à la fraise"

on a :

- Prédicat : manger
- Agent : Jean
- Objet : une glace à la fraise.

Pour la phrase : "Jean a mangé une glace à la plage"

on a :

- Prédicat : manger
- Agent : Jean
- Objet : une glace
- Lieu : la plage.

Ces deux phrases ont une même forme syntaxique mais leur structure sémantique diffère. Cette différence est déterminée par le type de concept qui suit la préposition "à" : concept de lieu pour le mot "plage" et concept de fruit pour le mot "fraise".
Ce type de contraintes permet de lever des ambiguités issues de l'emploi d'un même mot dans deux sens différents.

Les grammaires de cas sont souvent employées avec les réseaux ATN pour traduire des énoncés en structure profonde.

II-5- CONCLUSION

L'analyse sémantique de textes libres nous paraît actuellement irréaliste en raison des difficultés de modélisation et de manipulation des concepts sémantiques, de sa complexité, et donc d'un niveau de "compréhension" très insuffisant pour un traitement général. Ce type d'analyse donne cependant de bons résultats dans certains domaines particuliers, c'est-à-dire lorsque la langue est simplifiée et les concepts manipulés réduits à une application donnée, et lorsque le contexte de production peut guider ces analyses.

L'analyse syntaxique, en permettant de construire la structure d'une phrase, semble être, pour l'instant, le traitement le plus élaboré utilisable pour des textes libres. Cependant, cette analyse mise en oeuvre sur les résultats d'une analyse morphologique d'ambiguités, ne peut dans bien des cas, être employée à cause de sa complexité et de son caractère fortement combinatoire. Cette complexité est liée d'une part à la modélisation de la langue, d'autre part à la combinatoire engendrée par les ambiguités, d'origine linguistique, mais aussi générée par l'analyse morphologique. Il nous semble irréaliste d'utiliser un tel analyseur pour résoudre les homographies. L'analyse syntaxique d'un éditeur de texte conditionné Braille peut cependant être déchargée d'une partie de son travail par un modèle d'états finis. Un pré-traitement d'états finis pourrait construire déjà une bonne partie des sous-structures d'une chaîne Braille.

CHAPITRE III : LE MODÈLE SYNTAXIQUE D'ANALYSE DES TEXTES LIBRES

III-1- INTRODUCTION

Les connaissances syntaxiques sont généralement décrites sous forme d'un ensemble de règles qui définissent la structure grammaticale des expressions du langage. Si celui-ci le permet, il est avantageux de décrire les connaissances syntaxiques sous formes de réseaux .

Nous rappelons dans ce chapitre quelques notions sur la théorie des langages formels nécessaire pour aborder les problèmes d'analyse syntaxique. Ce sont d'une part des grammaires formelles, outils de description des structures telles qu'elles ont été définies par CHOMSKY. D'autre part, les automates qui constituent l'aspect reconnaissance de ces structures.

Nous exposons enfin le principe des réseaux d'automates finis que sont les réseaux de transition et notamment les réseaux ATN et RNP.

Les propriétés énoncées dans ce chapitre sont très classiques aussi nous n'en donnons pas les démonstrations.

III-2- DEFINITIONS PRELIMINAIRES

Nous introduisons ici quelques définitions qui seront utiles pour comprendre le formalisme syntaxique.

Alphabet : on appellera alphabet, un ensemble fini V contenant tous les caractères

$V = \{\sqcup, A, B, C, \ldots, Z, ., ;, \ldots, 0, 1, \ldots, 9\}$

Concaténation : On définit la concaténation "." sur V, par l'application qui à $V \times V \rightarrow V$ et tel que $m.n = mn$

Chaîne : On appelle chaîne sur V une suite ordonnée d'éléments de V représentée par simple juxtaposition (concaténation) de ces éléments.

$m = x_1 x_2 x_3 \ldots x_n$ avec $x_i \in V$, i [1,n]

On notera λ la chaîne vide

On désignera par :

V^* : l'ensemble de toutes les chaînes que l'on peut construire sur V (y compris la chaîne vide)

V^+ : L'ensemble de toutes les chaînes non vides, $V^+ = V^* - \{\lambda\}$

V^* possède l'opération interne associative de concaténation définie par :

Soit p et q G V^* avec $p = a_1\ a_2 \ldots a_n$

$$q = b_1\ b_2 \ldots b_m$$

alors $r = pq = a_1\ a_2 \ldots a_n\ b_1\ b_2 \ldots b_m \in V^*$

Cette opération est en général non commutative et possède λ pour élément neutre.

L'ensemble V est muni d'une relation d'ordre "<" et nous prenons la convention :

$v_i < v_j \Longleftrightarrow$ Code ASCII(v_i) < Code ASCII (v_j)

Soit une chaîne de V^* tel que p = ℓmno où ℓ et n $\in V^*$. ℓ représente le préfixe de m, n son suffixe et o sa désinence ou terminaison. On dit alors que m est une sous-chaîne de P.

Si P est une chaîne, P^n est P écrit n fois.

Longueur d'une chaîne : Soit m une chaîne, nous appellerons ℓ (m) la longueur de la chaîne m. On note :

$$\ell(m) = |m| = \sum_{a \in V} \ell_a(m)$$

où $\ell_a(m)$ = nombre d'occurence de a dans m.

$\ell(m)$ est une application de V^* dans N (ensemble des entiers naturels)

Dérivation directe : Soient deux chaînes de V^*, η et γ et R un ensemble de règles dites de dérivation (on dit aussi de génération ou de production).

On dit que γ est directement dérivable à partir de η, on note :

$$\eta \xrightarrow[G]{} \gamma$$

si pour $\eta = \omega_1\ \alpha\ \omega_2$ et $\gamma = \omega_1\ \beta\ \omega_2$, il existe une règle de dérivation tel que $\alpha \rightarrow \beta \in R$

Dérivation : On dit que la chaîne γ est dérivable à partir de la chaîne η , on note :

$$\eta \xrightarrow[G]{*} \gamma$$

S'il existe une séquence de chaîne $\theta_1, \theta_2, \ldots, \theta_n$ telle que :

$$\eta = \theta_1 \qquad \gamma = \theta_n \quad \text{et} \quad \theta_i \xrightarrow[G]{} \theta_{i+1} \quad \text{pour } i = 1, 2, \ldots, n-1$$

III-3-GRAMMAIRES, LANGAGES ET AUTOMATES

III-3-1- GRAMMAIRES ET LANGAGES FORMELS

CHOMSKY[55]a introduit les premières "grammaires formelles" afin de décrire la structure du langage naturel. Nous présentons ici les grammaires formelles sans leurs développements mathématiques.

III-3-1-1- Définitions

Une grammaire formelle est un quadruple

$$G = \{V_T, V_N, R, A\}$$

où

V_T est un ensemble fini de symboles terminaux,

V_N " " " " " " non terminaux (ou variables),

$V_N \cup V_T$ est l'union de V_N et de V_T constituant le vocabulaire total V de G,

R est un ensemble fini de règles de production ou de génération (on dit aussi de réécriture ou de dérivation) qui s'écrivent sous la forme suivante :

$$\alpha \rightarrow \beta$$

avec α et $\beta \in (V_N \cup V_T)^*$; le symbole "$\rightarrow$" veut dire "génère" ou "est remplacé par",

$A \in V_N$ est l'axiome (le symbole initial à partir duquel toute génération commence).

On définit le langage engendré par la grammaire $G = \{V_N, V_T, R, A\}$ comme étant l'ensemble des éléments qui dérivent de A, on le note :

$$L(G) = \{\omega \mid \omega \in V_T^* \quad \text{et} \quad A \xrightarrow[G]{*} \omega\}$$

La séquence de génération d'une chaîne $\omega \in L(G)$ est la séquence des règles de dérivation employées pour générer ω à partir de l'axiome initial.

Si on obtient ω par :

$$A \xrightarrow{R_1} \alpha_1 \xrightarrow{R_2} \alpha_2 \xrightarrow{R_3} \alpha_3 \ldots\ldots\ldots \alpha_{n-1} \xrightarrow{R_n} \omega$$

alors $R_1, R_2, R_3, \ldots, R_n$ est la séquence de réécriture ou de dérivation de ω

III-3-1-2- Hiérarchie des grammaires

CHOMSKY a classé les grammaires en 4 types (tableau III-1). Ce classement est basé sur les inclusions successives des règles de production. Une grammaire de type 3 est aussi de type 2, de type 1 et de type 0. Donc l'ensemble des langages engendrés par la grammaire de type 3 est inclus dans celui engendré par les grammaires de type 2, 1 et 0.

Nom de la grammaire	Forme de α	Forme de β
Type 0 "Structure de phrase"	$\alpha \in (V_T \cup V_N)^+$	$\beta \in (V_T \cup V_N)^+$
Type 1 "Context-sensitive"	$\gamma \,.\, N \,.\, \delta$	$\beta = \gamma.V.\delta$ $\gamma, V, \delta \in (V_T \cup V_N)^+$
Type 2 "Context-free"	N	$\beta \in (V_T \cup V_N)^+$
Type 3 "régulière" (rationnelles, de Kleene, d'états finis)	N	$a.T \quad a \in V_T$ ou $T.a \quad T \in V_N$

Tableau III-1

III-3-1-2-1- Les grammaires de type 0

Les grammaires de ce type sont non restreintes. Il n'y a aucune contrainte sur les règles de réécritures ; elles peuvent avoir des chaînes quelconques de chaque côté du symbole (la flèche) de dérivation. Ce type de grammaire est trop général pour être utile. Le problème soulevé par l'analyse d'une chaîne générée par une grammaire d'ordre 0 n'est pas résolu, c'est-à-dire que le problème est indécidable.

III-3-1-2-2- *Les grammaires de type 1*

Les règles de réécriture d'une telle grammaire ont toujours la forme suivante :

$$\mu\ A\ \eta \rightarrow \mu\ \beta\ \eta$$

où $A \in V_N$, et $\mu, \eta, \beta \in V$. En plus $|\mu\ A\ \eta| \leqslant |\mu\ \beta\ \eta|$

c'est-à-dire $|A| \leqslant |\beta|$

Un exemple d'un langage de type 1 est : $\{0^n\ 1^n\ 0^n \mid n = 1,2\ ...\}$

III-3-1-2-3- *Les grammaires de type 2*

Les grammaires "context-free" ont des règles de production de la forme :

$$N \rightarrow \beta$$

avec $N \in V_N$ et $\beta \in (V_N \cup V_N)^+$. C'est-à-dire, la partie gauche de la règle est un seul élément non terminal (sans aucun contexte) et la partie droite est une chaîne non vide. Un langage engendré par une telle grammaire sera appelé "langage context-free".

Exemple :

Soit $G = \{V_T, V_N, A, R\}$ avec :

- V_T = {a,b,c,...z,0,1,...,9,+,*,↑,;, = ,(,)}
- V_N = {A,E,T,F,P,S,L}
- R : A → S|=|E|;

  E → T|E+T

  T → F|T*F

  F → P|P↑F

  P → S|L|(E)

  S → a|b|c

  L → 1|2|...|9|0

La convention d'écriture "|" signifie "ou". Par exemple, la règle E → T|E+T signifie que E se dérive en T ou en E+T. Cette grammaire peut engendrer la chaîne suivante : a = b * (c+b↑2);

III-3-1-2-3-1- L'arbre syntaxique

Dans une grammaire context free, on peut avoir plusieurs éléments non terminaux dans une chaîne au cours de la génération, chacun étant partie gauche d'une règle de production. Le fait que l'on ait pour ce type de grammaire un seul symbole non terminal à réécrire nous permet de présenter la dérivation d'une chaîne sous la forme d'un graphisme bidimensionnel : l'arbre syntaxique.

Exemple :

La génération de la chaîne a = b * (c+b↑2); peut être représentée par l'arbre syntaxique de la figure III-1;

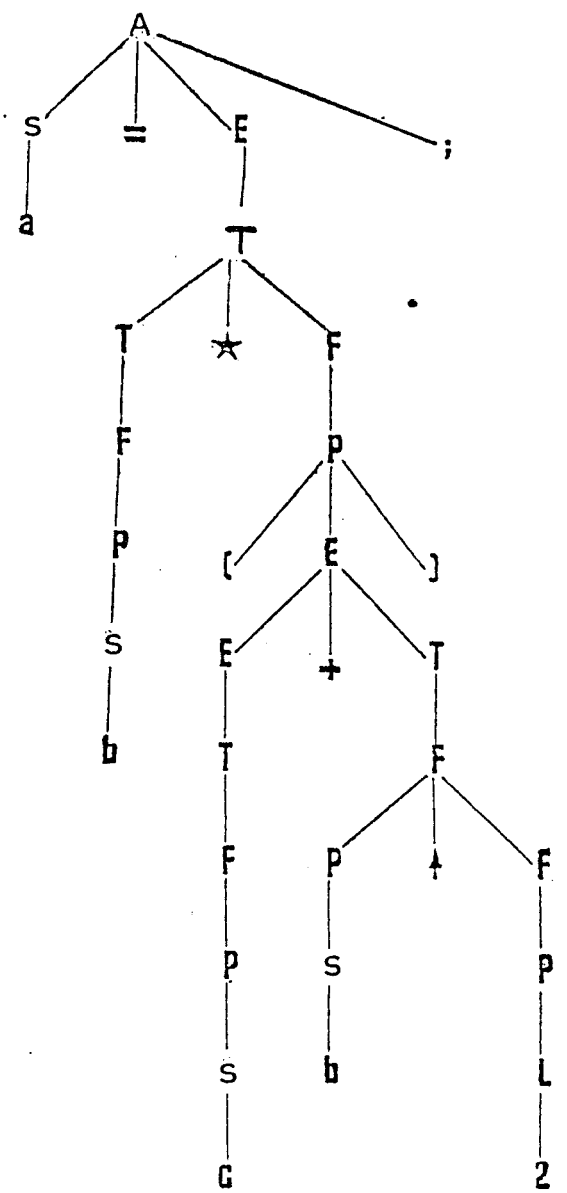


Figure III-1

Chaque sommet (sauf les feuilles) représente un élément non terminal qui donne lieu aux autres sommets descendants par application d'une règle de réécriture. La racine de l'arbre est l'axiome **A**, les symboles terminaux se trouvent sur les sommets pendants (ou feuilles) de l'arbre.

La lecture de gauche à droite des terminaux situés sur les feuilles de cet arbre permet de reconstituer la chaîne. L'existence d'un arbre syntaxique pour chaque chaîne générée par une grammaire de type 2 ou 3 définit entièrement sa structure syntaxique. S'il n'y a plus d'un seul arbre pour une chaîne cela signifie que cette chaîne peut être générée de plusieurs manières différentes ; elle peut ainsi posséder plusieurs structures syntaxiques différentes, ce qui implique une ambiguité dans le langage.

Les caractéristiques des langages algébriques sont décrits aussi par le formalisme des automates à pile de mémoire.

III-3-1-2-4- Les grammaires de type 3

Les règles de réécriture d'une grammaire régulière sont de la forme :

$N \rightarrow a.T$ ou $N \rightarrow a$

avec $N, T, \in V_n$ et $a \in V_T$

Exemple :

$G = \{V_T, V_N, A, R\}$ ou $V_N = \{A, B\}$, $V_T = \{a,b\}$

$R : A \rightarrow aA|bB$ R_1, R_2

$B \rightarrow bB|a$ R_3, R_4

Une chaîne est engendré par G selon la séquence de dérivation suivante.

$$A \xrightarrow{R_1} aA \xrightarrow{R_1} aaA \xrightarrow{R_2} aabB \xrightarrow{R_4} aaba$$

On constate aisément que le langage engendré par la grammaire G est de la forme :

$$L(G) = \{a^n b^m a | \ n > 0 , \ m > 1\}$$

Une grammaire régulière engendre un langage dit régulier.
Les propriétés des langages réguliers sont décrits plus facilement en utilisant un formalisme équivalent aux grammaires régulières : celui des automates d'états finis.

III-3-2- LES AUTOMATES

Pour chaque type de grammaire, il existe une machine théorique séquentielle qu'on appelle automate et qui caractérise le type de la dite grammaire.
Un automate reçoit en entrée une chaîne du texte source. La chaîne est acceptée par l'automate, lorsque celui-ci se trouve dans un état appelé état final. Selon le type de grammaire, les automates possèdent un mécanisme plus ou moins complexe.
L'automate le plus simple est celui que l'on appelle automate d'état fini, le plus complexe est appelée machine de Turing.

III-3-2-1- Les automates d'états finis

III-3-2-1-1- Définitions

Automate déterministe :

Un automate fini déterministe est un quintuplet :

$$A = \{X, Q, \otimes, q_0, F\}$$

où : X est un alphabet fini d'entrée,

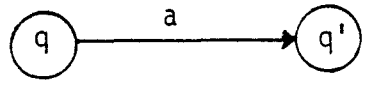
Q est un ensemble fini d'états

$q_0 \in Q$ est l'état initial,

$F \subset Q$ est l'ensemble des états finals,

$\otimes : Q \times X \to Q$ est la fonction de transition.

Graphiquement, un état q sera symbolisé par : (q), et une transition $q' = q \otimes a$ par :



On représentera l'état initial par : $(q_0)^-$

Les états finals seront schématisés par : $(q_F)^+$ avec $q_F \in F$

Automate complètement spécifié :

Un automate A est dit complètement spécifié si :

$\forall q \in Q, \forall x \in X, q \otimes x$ est définie

sinon, on construit un automate complètement spécifié équivalent à A en lui ajoutant un nouvel état p, appelé l'état puit et en prolongeant la fonction de transition de la façon suivante :

- si $q \otimes x$ n'est pas définie, alors $q \otimes x = p$
- $\forall x, p \otimes x = p$

Automates émondés :

Un automate fini déterministe est émondé si tous ses états sont accessibles à partir de l'état initial :

$\forall q \in Q, \exists \omega \in X^*$ tel que $q_0 \otimes \omega = q$

Langage accepté par un automate :

Le langage accepté par l'automate $A = \{X,Q,q_0,F,\otimes\}$ est défini par :

$L(A) = \{\omega \in X^* \mid q_0 \otimes \omega \in F\}$

Une chaîne $\omega = x_1 x_2 \ldots x_n$ est donc acceptée par A lorsque partant de l'état initial, on peut trouver une suite d'arcs portant successivement les lettres $x_1 x_2 \ldots x_n$ et menant à un état final.

Automate non déterministe :

Un automate fini non déterministe est un quintuplet

$A = \{X,Q,\otimes,E,F\}$

où : X,Q,F, sont des ensembles ayant une signification identique à celle définie dans le cas des automates déterministes.

$E \subset Q$ est un ensemble non vide d'états initiaux

$\otimes : Q \times X \to 2^Q \quad P(Q)$. A un couple (q,a) peut correspondre un ensemble d'états (figure III-2).

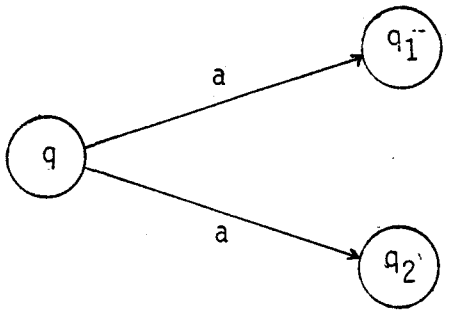


Figure III-2

Des algorithmes[67] permettent de passer d'un automate non déterministe à un automate déterministe et inversement. On démontre[53] de même qu'il existe un automate déterministe pouvant reconnaître le langage accepté par un automate non déterministe.

III-3-2-1-2- Correspondance grammaire de type 3 et automates finis

Il existe une correspondance biunivoque entre une grammaire régulière et un automate fini : l'état initial de l'automate correspond à l'axiome de la grammaire. Les deux alphabets étant identiques, les règles de production correspondent à la fonction de transition de la façon suivante (figure III-3).

où $q_F \in F$

Figure III-3

Exemple :

La grammaire régulière dont les règles de production sont :

$$R \begin{cases} A \longrightarrow aA \mid bB \\ B \longrightarrow bB \mid a \end{cases}$$

correspond à l'automate fini représenté par la figure III-4.

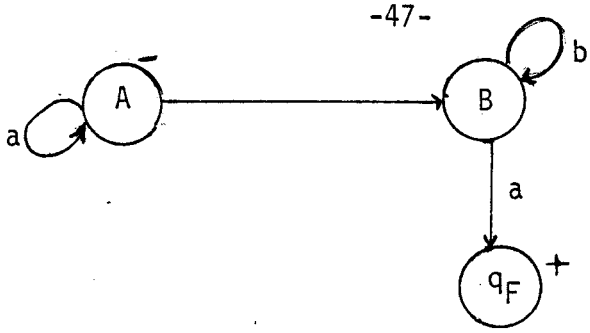


Figure III-4

On peut donc dire que la famille des langages (sur un alphabet X) reconnus par un automate fini est la même que celles des langages (sur X) engendrés par une grammaire régulière.

III-3-2-2- Les automates à pile de mémoire

III-3-2-2-1- Définitions

On appelle automate à pile le sextuplet : $Ap = \{X, \Gamma, z_0, Q, q_0, R\}$

où X est l'alphabet fini d'entrée,

Γ " " " de pile,

z_0 est le symbole de fond de pile ($z_0 \in \Gamma$),

Q est l'ensemble fini d'états,

q_0 est l'état initial,

R est l'ensemble fini de règle de mouvement de la forme : $(\alpha,q,a) \rightarrow (\beta,a,q')$
avec $\alpha \in \Gamma$, q et $q' \in Q$, $a \in X$ et $\beta \in \Gamma^*$
ou de la forme : $(\alpha,q,\varepsilon) \rightarrow (\beta,\varepsilon,q')$ avec $\alpha \in \Gamma$, q et $q' \in Q$ et
ε étant la chaîne vide.

Le langage accepté par l'automate $Ap = \{X,\Gamma,z_0,Q,q_0,R\}$ est défini par :

$$L(Ap) = \{\omega \in X^* \mid (z_0,q,\omega) \xrightarrow[Ap]{*} (\omega,z_0,q')\}$$

Une chaîne $\omega = x_1 x_2 \ldots x_n$ est donc acceptée par Ap s'il existe un mouvement $(z_0,q,\omega) \xrightarrow[Ap]{*} (\omega,z_0,q')$, c'est-à-dire un mouvement qui permet de lire complètement ω en partant de la pile vide (z_0) et aboutissant à la pile vide (z_0).

III-4- L'ANALYSE SYNTAXIQUE

L'analyse syntaxique est l'investigation de la possibilité de générer une chaîne à partir d'un axiome A en employant une grammaire G. Dans ce cas, on détermine la séquence des règles de génération (les dérivations) qui y sont employées.

Pour les grammaires context-sensitive, et les grammaires à structure de phrase, le problème n'est que partiellement résolu et par conséquent ces grammaires sont d'un maniement quelque peu incertain. Le problème est par contre bien résolu pour les grammaires context free et régulière.
Nous décrivons ici l'analyse syntaxique concernant ces deux grammaires.

III-4-1- L'ANALYSE SYNTAXIQUE DES LANGAGES REGULIERS

La décision déterminant l'appartenance ou non d'une phase donnée $\omega \in X^*$ à un langage défini par une grammaire donnée G, ou représenté par un automate est simple à effectuer. Cela revient à tester si dans l'automate fini il existe un chemin allant de l'état initial à l'état final, et portant pour suite d'étiquettes sur ses arcs la suite exacte des lettres de la chaîne à reconnaître.

Pour un automate complètement spécifié, l'algorithme est le suivant :

Soit $\omega = x_1\ x_2\ \ldots\ x_n \in X^*$

Soit $A = (W, Q, \otimes, q_0, F)$

on construit la suite d'états : $q_1 = \otimes\ (q_0, x_1)$

$q_2 = \otimes\ (q_1, x_2)$

$\vdots$

$q_n = \otimes\ (q_{n-1}, x_n)$

si $q_n \in F$, alors "$x \in L(A)$" : = vrai

si $q_n \notin F$, alors "$x \in L(A)$: = faux

Dans le cas où A n'est pas complètement spécifié, l'algorithme est alors

pour i = o à n-1 Faire :

Si $\varnothing(q_i, a_{i+1})$ n'existe pas

Alors " $x \in L(A)$": = Faux

Sinon Si i = n-1

Alors Si $q_{i+1} \in F$

Alors " $x \in L(A)$":= Vrai

sinon " $x \in L(A)$":= Faux

III-4-1-1- Distance d'une chaîne à une grammaire régulière

Afin de déterminer les erreurs surla chaîne d'entrée, un algorithme calcule la distance entre la chaîne et la grammaire. Il calcule le nombre de modifications à faire subir à la chaîne pour la transformer au moindre coût en une chaîne engendrée par la grammaire.

Soit un automate fini dont les états sont notés ici = S,T,R, etc... et numérotés de 0 à n, l'état de rang 0 étant initial, une chaîne $\alpha = \alpha_1 \ldots \alpha_n$ écritre sur un alphabet X est donnée.

On définit les modifications possibles sur une chaîne comme étant : insertion, substitution, destruction. Chacune étant affectée d'un poids unitaire.
La distance de α à l'automate sera donc le nombre minimal d'opérations à faire subir à α pour que $\alpha_1 \ldots \alpha_n$ soit la suite d'étiquettes d'un chemin menant de l'état initial de l'automate à un état final.

On définit d'abord pour tout état δ :

$G(j,S)$ pour $j = 1,\ldots,n$

comme le nombre minimal d'opérations transformant $\alpha_1, \ldots, \alpha_j$ en une chaîne menant de l'état initial à S.

Un raisonnement de programmation dynamique [43] assure alors que :

$$G(j,S) = \min_T \{G(j-1,T) + V(T,S,\alpha_j)\}$$

avec $V(T,S,\alpha_j)$ = coût minimum pour transformer le caractère α_j en une chaîne menant de l'état T à l'état S.

Si on appelle P(T,S) la longueur d'un des plus courts chemins (en nombre d'arcs) entre T et S et L(T,S,a) un indice valant 1 si la lettre a appartient à l'un des chemins, 0 sinon, on a alors : $V(T,S,a) = 1$ si $T = S$

$= P(T,S) - L(T,S,a)$ sinon

La longueur des plus courts chemins entre T et S peut se calculer par les équations :

$$p^{k+1}(T,S) = \text{Min}(p^k(T,S), p^k(T,k+1) + p^k(k+1,S)$$

où $p^k(T,S)$ est la longueur d'un plus court chemin de T à S ne passant que par les états de rang inférieur à k.

La distance cherchée entre α et l'automate vaut : Min G(n,S) avec $S \in F$

III-4-2- L'ANALYSE SYNTAXIQUE DES GRAMMAIRES CONTEXT-FREE

Pour analyser une chaîne ω, on peut parcourir l'arbre syntaxique de deux manières différentes. Le parcours s'effectue soit du bas vers le haut, il s'agit de l'analyse ascendante (bottum-up), soit du haut vers le bas, il s'agit de l'analyse descendante (les conventions ascendantes et descendantes sont définies en considérant que l'arbre syntaxique est représenté avec sa racine située plus haut que les feuilles).

L'analyse ascendante d'une chaîne ω commence donc par tous les terminaux en parallèle et tente de la réduire à l'axiome A par application successive des règles de production à l'envers. Ce type d'analyse n'est pas ensible aux erreurs, mais demande un encombrement mémoire considérable pour former et garder toutes les combinaisons locales en montant jusqu'à la racine[54][62].

L'analyse descendante, par contre commence par A et tente de trouver une séquence de règles de génération de G à appliquer pour finalement générer ω.
Ce type d'analyse est en quelque sorte prédictive car en opérant de gauche à droite à partir d'un premier élément positif recherché et trouvé, et en utilisant la grammaire, il est possible de guider la recherche en proposant les compositions possibles.
Ceci constitue un grand avantage en faveur de la méthode descendante. Par contre, le principal défaut d'une telle méthode est la conséquence grave d'une erreur, au début

de son utilisation qui malmènerait l'analyse et causerait une grande perte de temps avant sa découverte.

Nous décrivons ici deux algorithmes. Le premier est top-down et traite d'un exemple pour illustrer le mécanisme des automates à pile de mémoire. Le second est un algorithme général classique dû à Earley.

III-4-2-1- Méthode générale top-down

Cette méthode est en fait la description sous forme d'algorithme du fonctionnement de l'automate à pile de mémoire correspondant à la grammaire context-free. Nous décrivons sur un exemple, une grammaire qui génère un sous-ensemble des expressions arithmétiques d'un langage de programmation tel que Fortran.

Les règles sont étiquettées pour pouvoir être repérées. On gère durant l'analyse deux piles de mémoire : l'une dans laquelle on introduit les étiquettes des règles examinées (pile 1), l'autre dans laquelle se trouvent les symboles de la phrase (pile 2). En réalisation programmée, on mettra dans ces piles le rang des règles et le rang du symbole examiné. A noter que la première règle de la grammaire est introduite artificiellement afin de pouvoir repérer la fin de la phrase à analyser.

La mémoire h mémorise le rang du symbole en haut de la pile 2 dans la chaîne d'entrée :

Grammaire	Analyse top-down de la phrase
s : S → E ⊣	
	a + a a ⊣
e_1 : E → T	h : 1 2 3 4 5
e_2 : E → E+T	
t_1 : T → a	
t_2 : T → aT	

L'algorithme d'analyse syntaxique peut se suivre aisément en décrivant le contenu des piles en cours d'analyse (figure III-5).

Pile 1	Pile 2	h
	S	1
s	E ⊣	1
se_1	T ⊣	1
$se_1 t_1$	a ⊣	1
$se_1 t_1$	⊣	2
se_1	T ⊣	1
$se_1 t_2$	aT ⊣	1
$se_1 t_2$	T ⊣	2
$se_1 t_2 t_1$	a ⊣	2
$se_1 t_2$	aT ⊣	1
$se_1 t_2 t_2$	aaT ⊣	1
$se_1 t_2 t_2$	aT ⊣	2
$se_1 t_2$	aT ⊣	1
se_1	T ⊣	1

Pile 1	Pile 2	h
se_2	E+T ⊣	1
$se_2 e_1$	T+T ⊣	1
$se_2 e_1 t_1$	a+T ⊣	1
$se_2 e_1 t_1$	+T ⊣	2
$se_2 e_1 t_1$	T ⊣	3
$se_2 e_1 t_1 t_1$	⊣	4
$se_2 e_1 t_1$	T ⊣	3
$se_2 e_1 t_1 t_2$	aT ⊣	3
$se_2 e_1 t_1 t_2$	T ⊣	4
$se_2 e_1 t_1 t_2 t_1$	a ⊣	4
$se_2 e_1 t_1 t_2 t_1$	⊣	5

Figure III-5

L'analyse syntaxique s'arrête ici sur un succès : le symbole pointé par h et le seul élément de la pile 2 sont équivalents.

III-4-2-2- Algorithme d'Earley [30]

Cet algorithme part d'une grammaire algébrique quelconque G = (X,V,S,P) et d'une chaîne $x = a_1 \ldots a_n$ de X^*. Il construit des configurations d'analyse qui représentent les correspondances possibles entre la chaîne et la structure syntaxique induite par la grammaire G. Une configuration notée C est définie par :

$$C = [A \rightarrow X_1 X_2 \ldots X_k \bullet X_{k+1} \ldots X_m, i]$$

dans lequel :

$$A \rightarrow X_1 X_2 \ldots X_m \in P$$

Autrement dit C est une règle de production marquée par un pointeur (noté •), et associée à un chiffre i.

Les configurations construites par l'algorithme sont classées par listes I_0, I_1, ...,I_n qui ont la signification suivante :

Une configuration $C = [A \rightarrow X_1 X_2 \ldots X_k \bullet X_{k+1} \ldots X_m, i]$ appartient à la liste I_j si et seulement si il existe γ et δ dans $(X \cup V)^*$ tels que :

$$S \overset{*}{\rightarrow} \gamma A \delta \quad \text{avec} \quad \gamma \overset{*}{\rightarrow} a_1 \ldots a_i$$

$$\text{et} \quad X_1 \ldots X_k \overset{*}{\rightarrow} a_{i+1} \ldots a_j.$$

Par conséquent, les indices i (composante de la configuration) et j (numéro de liste) entourent dans x la portion de cette chaîne dérivée de $X_1 \ldots X_k$.
Autrement dit, la portion de x dérivée de $X_1 \ldots X_k$ est $a_{i+1} \ldots a_j$. Il est donc possible que la règle de production $A \rightarrow X_1 \ldots X_m$ ait servi à engendrer x.

L'algorithme construit les listes I_0, I_1, ..., I_n dans cet ordre et la décision finale vient de l'examen de I_n ; s'il contient une configuration de la forme $[S \rightarrow \alpha \bullet 0]$, la phrase x est reconnue comme appartenant au langage engendré par la grammaire G ; dans le cas contraire, elle n'appartient pas à ce langage.

III-4-2-2-1- Description de l'algorithme

1) Construction de I_0

A $I_0 = \{[S \to \bullet\, \alpha, 0]\}$

si $S \to \alpha \in P$, faire j:=0

B Tant que l'on ajoute des configurations à I_j, faire :

a) Si $([A \to \alpha \bullet B\, \beta, i] \in I_j)$ et $(B \to \gamma \in P)$

Alors $I_j = I_j \cup \{[B \to \bullet \gamma, j]\}$

b) Si $[A \to \alpha \bullet, i] \in I_j$

Alors pour toutes les configurations appartenant à I_i, pour $i < j$ de la forme $[A \to \alpha \bullet B\, \beta, j]$

Faire $I_j = I_j \cup \{[A \to \alpha\, B \bullet \beta, j]\}$

2) Construction de I_j à partir de I_{j-1}

C Pour toute configuration $[A \to \alpha \bullet a_j\, \beta, i] \in I_{j-1}$,

Faire $I_j = I_j \cup \{[A \to \alpha\, a_j \bullet \beta, i]\}$

D Faire le bloc B pour j

3) Examen de I_n

E Si $[S \to x \bullet, 0] \in I_n$

Alors $x \in L(G)$

Sinon $x \notin L(G)$

Son temps de calcul est en $O(n^3)$ dans le cas général, $O(n)^2$ si la grammaire est non ambigüe

III-4-3- LES RESEAUX DE TRANSITION [62]

Les grammaires formelles de la hiérarchie classique de Chomsky ne représentent pas le seul système de description de chaîne, et d'analyse syntaxique. Parmi les autres techniques de spécification mathématiquement équivalentes, les réseaux de transition présentent des caractéristiques intéressantes.

III-4-3-1- Les réseaux BTN

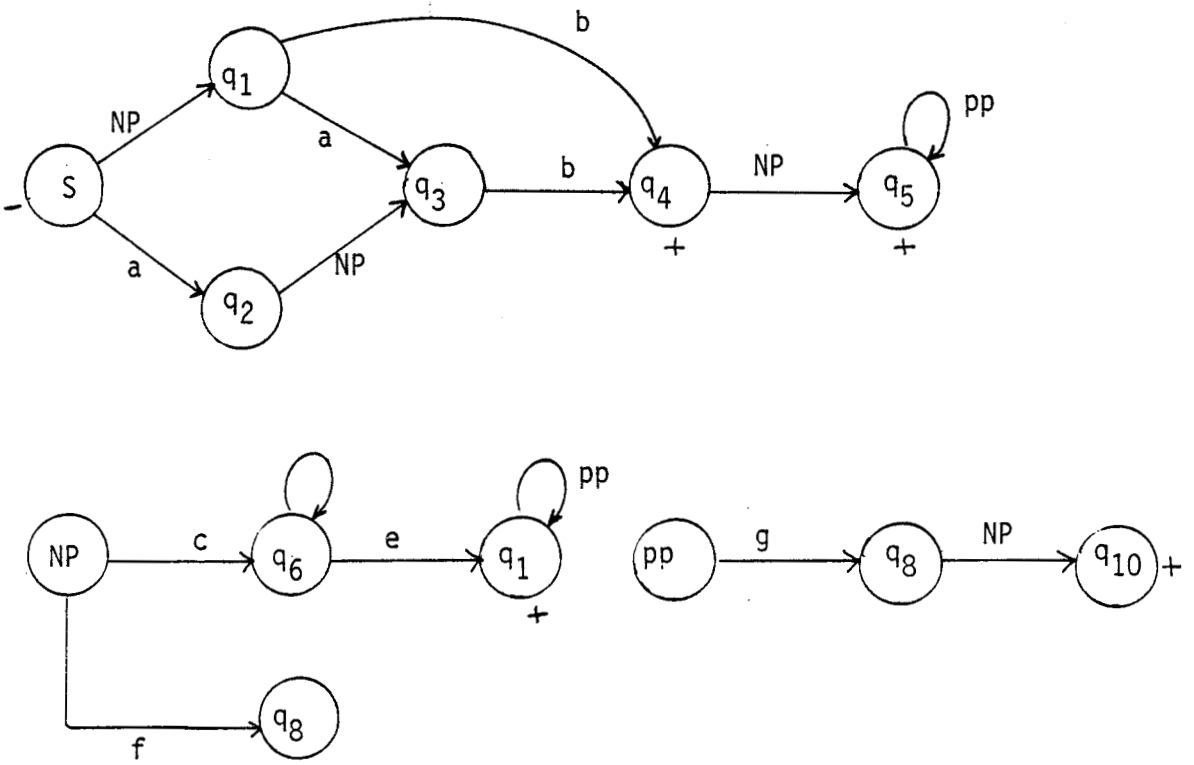
Les réseaux BTN (Basic Transition Network) constituent la version de base des réseaux de transition.

Un réseau de transition est un ensemble de graphes orientés dont les arcs et les noeuds sont étiquettés et identifiés par un nom. Les noeuds sont les états de la machine et les arcs représentent des transitions possibles d'un état à un autre. On distingue un état initial et un ensemble d'états finals. A la différence des automates d'états finis, l'étiquette d'un arc peut être soit un symbole de l'alphabet terminal (classes lexicales : article, nom, verbe, etc...), soit le nom d'un état pouvant correspondre à d'autres réseaux. Emprunter un arc étiquetté de cette dernière façon induit les actions suivantes :

- on range dans une pile le noeud où mène cet arc ("arc push") et l'analyse reprend à partir de l'état désigné par l'étiquette.
- Lorsqu'un état final est atteint, on relance l'analyse à partir de l'état qui est retiré du sommet de la pile ("arc pop").
- L'arrivée à un état final avec une pile vide assure le succès de l'analyse d'une chaîne, si la fin de celle-ci est atteinte simultanément.

La complexité de cet automate est donc celle d'un ensemble fini d'automates finis, gérés par une pile.

Exemple : La phrase "le chauffeur de DUBUS emmène Patrick dans la proche banlieue", peut être reconnue par le réseau de la figure III-5.



en interprétant :

a comme un auxiliaire,
b comme un verbe,
c comme un article,
d comme un adjectif,
e comme un nom,
f comme un nom propre
g comme une préposition

Figure III-6

Comme dans les grammaires context-free cet algorithme fait intervenir la notion de choix lorsque plusieurs arcs sont disponibles. Ce choix peut conduire à une impasse et amener le système à effectuer des retours arrière. Une version augmentée du réseau de transition BTN, le réseau de transition ATN, améliore les critères de choix.

III-4-3-2- Les réseaux ATN

Un réseau ATN ("Augmented Transition Network") est un réseau BTN auquel on associe des actions et des conditions aux arcs de transition :

- Une action est une fonction opérant sur des registres. Un registre est assimilable à une variable d'un langage de programmation associée aux noeuds de l'arbre syntaxique. Les actions ont pour effet d'affecter à ces registres une valeur calculée.

- Une condition est un prédicat sur le contenu de ces registres. Une transition ne peut être effectuée que si le prédicat est vrai à ce moment. On exécute alors l'actions associée. Les conditions augmentent les critères de sélection pour le choix d'un arc.

La construction des structures syntaxiques est améliorée par l'utilisation des registres, des conditions et des actions. Décrire une grammaire ATN revient à représenter les réseaux de transition des différents constituants syntaxiques d'une part et à spécifier la liste des conditions et actions associées à chaque transition d'autre part.

III-5- RESEAUX SYNTAXICO-SEMANTIQUE A NOEUDS PROCEDURAUX (R.N.P.)

Ce modèle est composé d'un réseau de transition intégrant une procédure à chaque noeud. Les contraintes syntaxiques traduisant la grammaire représentée sont décrites par les arcs du réseau, alors que les contraintes contextuelles sont intégrées aux noeuds grâce aux procédures qui leur sont attachées. Le rôle des procédures est de déterminer une hiérarchie pour la prise en compte des sorties possibles d'un noeud du réseau en fonction des entrées fournies par le système de reconnaissance de mots et du contexte syntaxico-sémantique. Les RNP se distinguent des ATN sur deux points essentiellement. D'une part, les procédures des RNP permettent de modifier dynamiquement le parcours du réseau compte tenu de la portion de phrase déjà analysée. D'autre part, on dissocie dans les RNP, la procédure d'analyse syntaxique et le réseau représentant la grammaire chaîne avec des tests de cohérence sémantique.

CHAPITRE IV : LES RÉSEAUX DE TRANSITION : APPLICATION À LA DESCRIPTION ET À LA MODÉLISATION DES FONCTIONS DE L'ÉDITEUR-TRANSCRIPTEUR NOIR CONDITIONNÉ BRAILLE.

IV-1-INTRODUCTION

Compte tenu des particularités de la syntaxe Braille, le programme de traitement de texte conditionné Braille qui a été réalisé est assez volumineux. Par ailleurs, lorsque ce programme est appelé, il travaille sur un texte qui occupe une partie de la mémoire centrale. Ces raisons nous ont conduit à scinder les manipulations de textes noir et Braille en plusieurs parties fonctionnellement distinctes:

- la fonction EDITION permet d'éffectuer sur le texte noir conditionné Braille, les opérations courantes de traitement de texte;
- les fonctions de transcription BRAILLE INTEGRAL et BRAILLE ABREGE pour l'obtention en mode programme des textes correspondants.

Nous décrivons dans ce chapitre la démarche adoptée pour la réalisation de ces différentes fonctions conformément au modèle d'élaboration des interpréteurs syntaxiques.

IV-2- LA FONCTION D'EDITION

Cette fonction constitue le noyau du système car elle doit permettre de produire en mémoire, les chaines codées dont le profil qualitatif indique la nature des symboles (ou séquences de symboles) rencontrés dans le texte noir, ainsi que certains paramètres d'aide à la transcription pour la fonction concernée. L'ensemble du logiciel réalisant cette fonction a donc été divisé en trois processeurs:

- un processeur de supervision;
- un processeur de gestion clavier;
- un processeur éditeur.

IV-2-1- LE PROCESSEUR DE SUPERVISION

Le processeur de supervision dont l'algorithme est schématisé figure IV-1, gère en maître les deux autres processeurs: gestion clavier et éditeur.

Son rôle consiste à:

- initialiser les périphériques et les paramètres pour les deux autres processeurs esclaves après la mise en route de l'appareil
- d'afficher les menus (états) de détermination des attributs globaux et des caractéristiques formelles en fonction des caractères frappés (alphabet d'entrée). le graphe de l'automate réalisant cette tache est représentée figure IV-2 .
- d'interroger tant qu'il y a manipulation de texte les deux processeurs esclaves. L'arrêt de cett boucle est déterminé par la reconnaissance d'un caractère spécifique de transfer page.

IV-2-2- LE PROCESSEUR GESTION CLAVIER

Ce processeur permet comme décrit sur la figure IV-3 d'éffectuer:

- d'une part l'analyse morphologique des informations en provenance du clavier. Il s'agit alors de classer ces informations en caractères de contrôles ou d'édition et de vérifier l'appartenance de ces caractères parmi ceux retenus pour saisir les textes noir conditionnés Braille.
- d'autre part un prétraitement syntaxique nécéssaire à la levée de certaines ambiguités pour la procédure de traitement. Ce prétraitement syntaxique consiste à déterminer pour certains symboles le code ASCII approprié de la syntaxe braille, soit en mode conversationnel(cas d'appui d'une touche majuscule transformée en lettre minuscule avec rappel du mode de saisie des textes), soit à la suite d'une séquence d'appui de touches prédéfinies(cas du tiret initial ou final).

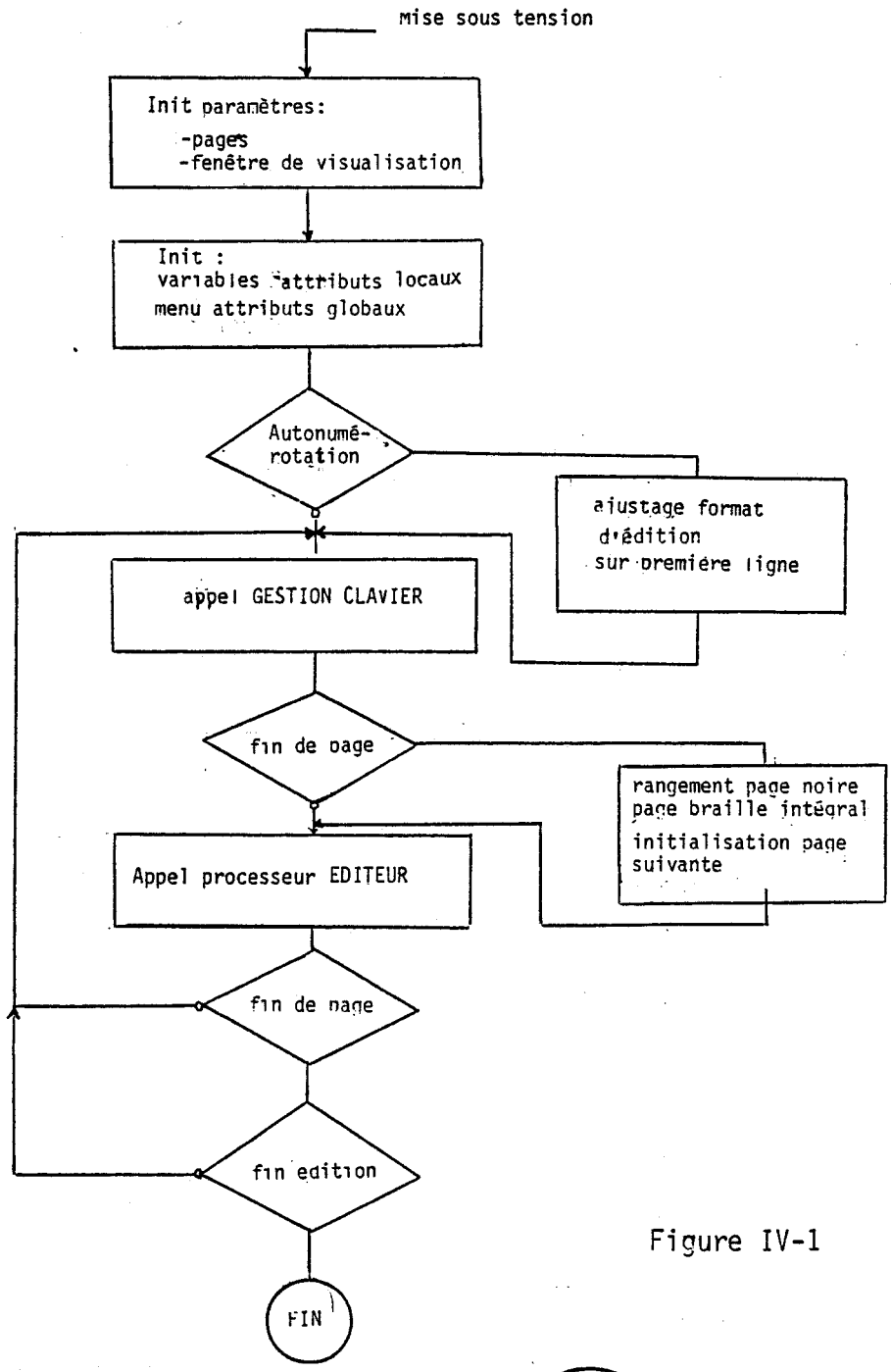


Figure IV-1

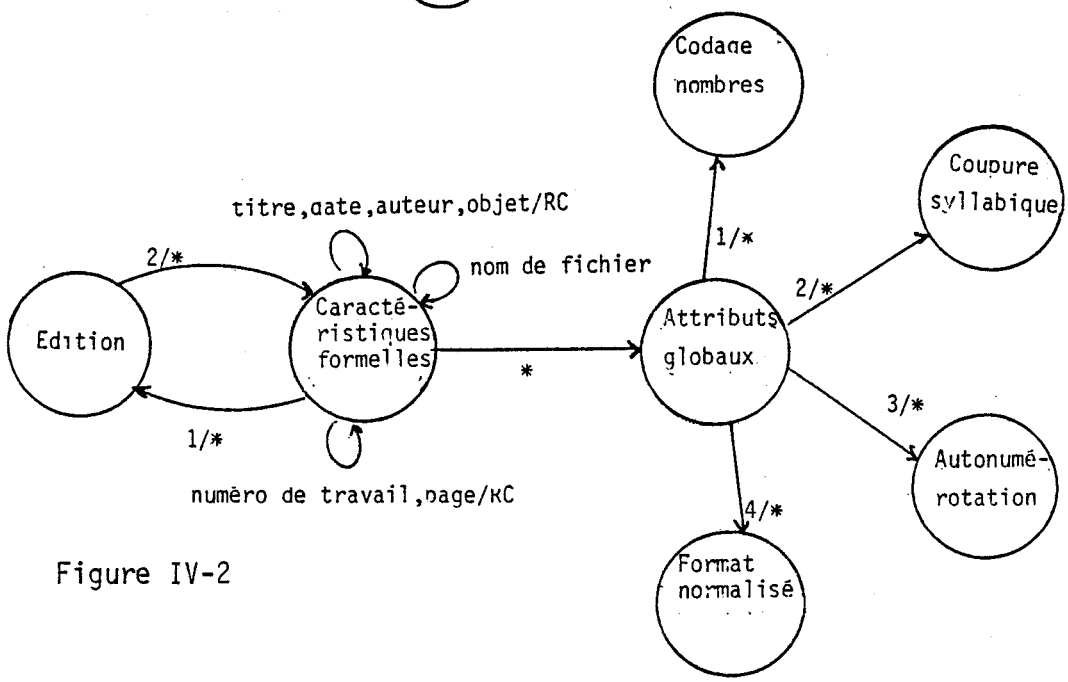


Figure IV-2

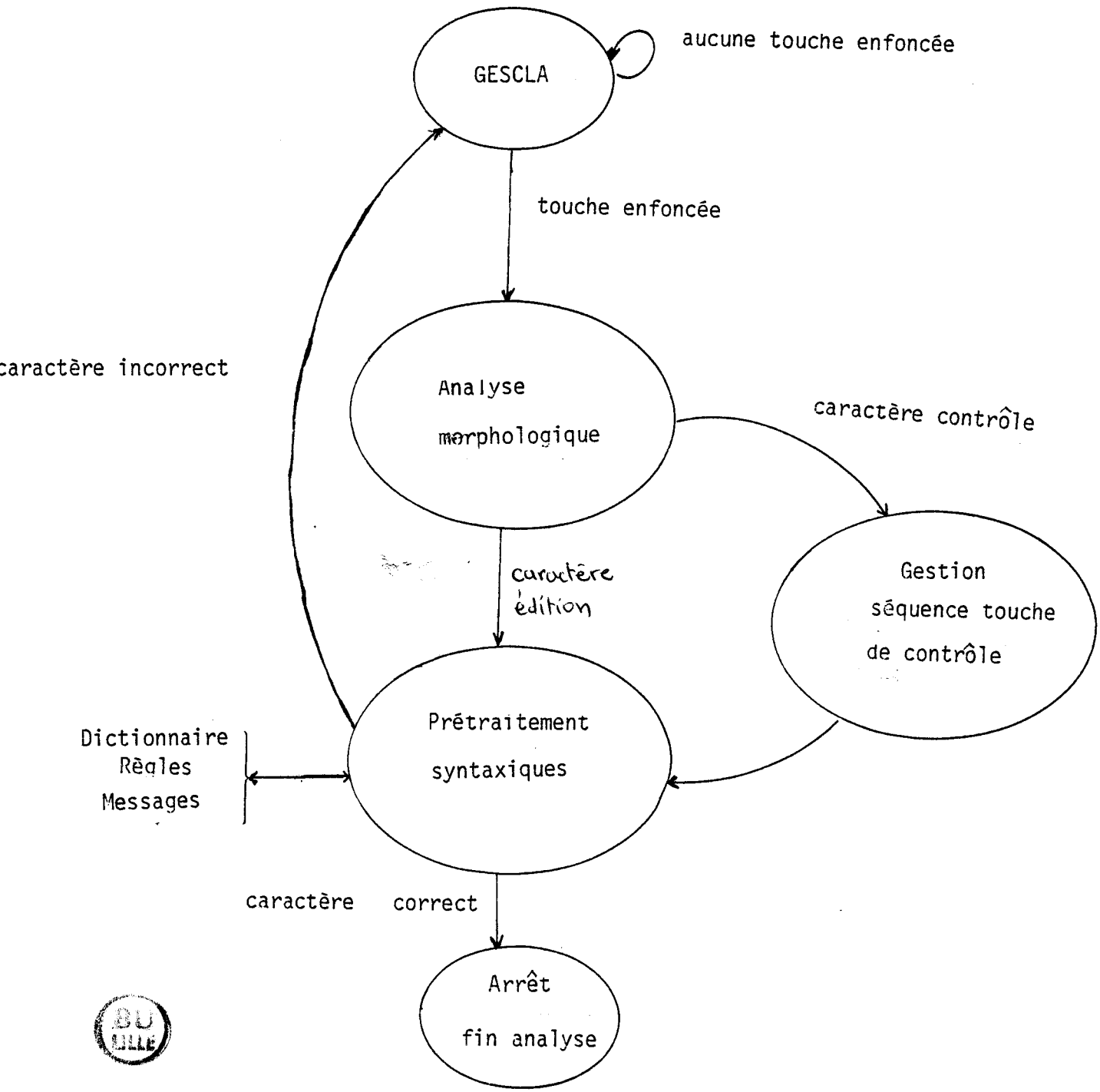


figure IV-3

IV-2-3- LE PROCESSEUR EDITEUR

Le processeur éditeur gère les données textuelles et d'aide à la mise en page conformément aux règles d'édition de texte Braille à partir des informations fournies par le processeur de gestion clavier.

Il est composé des procédures "TRAITEMENTS" et "VISUALISATION".

IV-2-3-1- LA PROCEDURE "TRAITEMENT"

Cette procédure dont le schéma descriptif est représenté par la figure IV-4 a pour but, la production et la manipulation en mémoire centrale des chaînes codées ASCII de la syntaxe noire définie sur les règles d'édition de la syntaxe Braille intégral.

IV-2-3-1-1- JUSTIFICATION DE LA CORRESPONDANCE LIGNE NOIRE-LIGNE BRAILLE

Conformément au choix du procédé d'édition, un algorithme local détermine successivement pour chaque caractère noir à éditer, sa contribution à la longueur de la ligne Braille correspondante. Cet algorithme est basé sur les principes suivants :

Soit V^* , l'ensemble des chaînes noires et B^* , l'ensemble des chaînes Brailles. Pour toute chaîne $\delta \in V$, on a :

$$\delta = m_1 m_2 \ldots m_1 \ldots m_n \text{ avec } \forall_i,\ m_i \in V$$

Soit T le procédé de transcription Braille intégral. C'est l'application associant toute chaîne de V^* , une chaîne de B^* et dont les propriétés sont les suivantes :

- La concaténation d'une chaîne noire en Braille intégral revient à concaténer la transcription de chaque caractère de la chaîne :

$$T(\delta) = T(m_1 m_2 \ldots m_i \ldots m_n) = T(m_1)T(m_2)\ldots T(m_i)\ldots T(m_n)$$

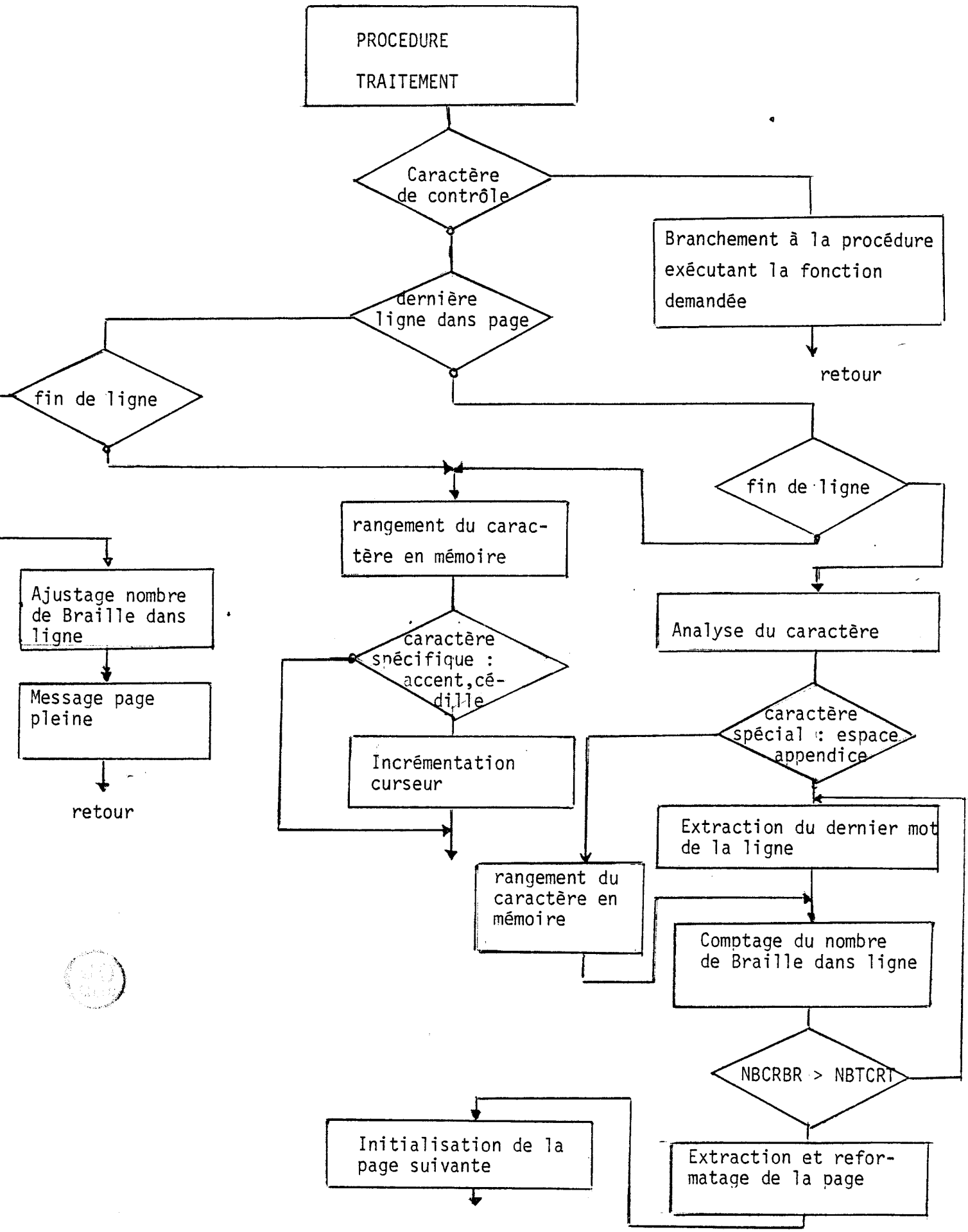


Figure IV-4 : Schéma descriptif de la procédure traitement

- La longueur d'un ensemble de chaîne transcrit revient à ajouter les longueurs de chaque chaîne transcrite:

$$l(T(\delta_1 \cdot \delta_2 \ldots \delta_n)) = l(T(\delta_1) + l(T(\delta_2)) + \ldots + l(T(\delta_n))$$

- Etant donné la possibilité pour un caractère noir d'être traduit par un ou plusieurs caractères Braille on a :

$$l(T(m_i)) \geq l(m_i) \quad \forall m_i \in V \quad \text{avec } l(m_i) = 1$$

$$\forall \beta \in B^* \quad \text{tel que } \beta = T(\delta), \text{ on a } l(\beta) \geq l(\delta)$$

La recherche par accès direct indexé dans une table permet alors de déterminer dans les cas les plus simples (minuscule, ponctuation...) la longueur Braille des chaînes noires suivies. Cette longueur est déterminée de manière optimale lorsqu'il s'agit d'une chaîne complexe, (souligné, chiffre, abréviation particulière...), après une analyse syntaxique de la chaîne analogue à celle effectuée lors de la traduction (Cf paragraphe IV-3-2). La figure IV-5 représente le schéma descriptif de cet algorithme

IV-2-3-1-2- GESTION DE FIN DE LIGNE

La longueur de la ligne Braille déterminée, celle-ci ne doit pas dépasser le nombre maximal de caractère Braille par ligne. Soit N ce nombre, cela revient à vérifier l'inéquation (1) suivante :

$$l(T(\delta_n)) \leq N - \left(\sum_{j=1}^{n-1} l(T(\delta_j)) + \gamma \right) \qquad (1)$$

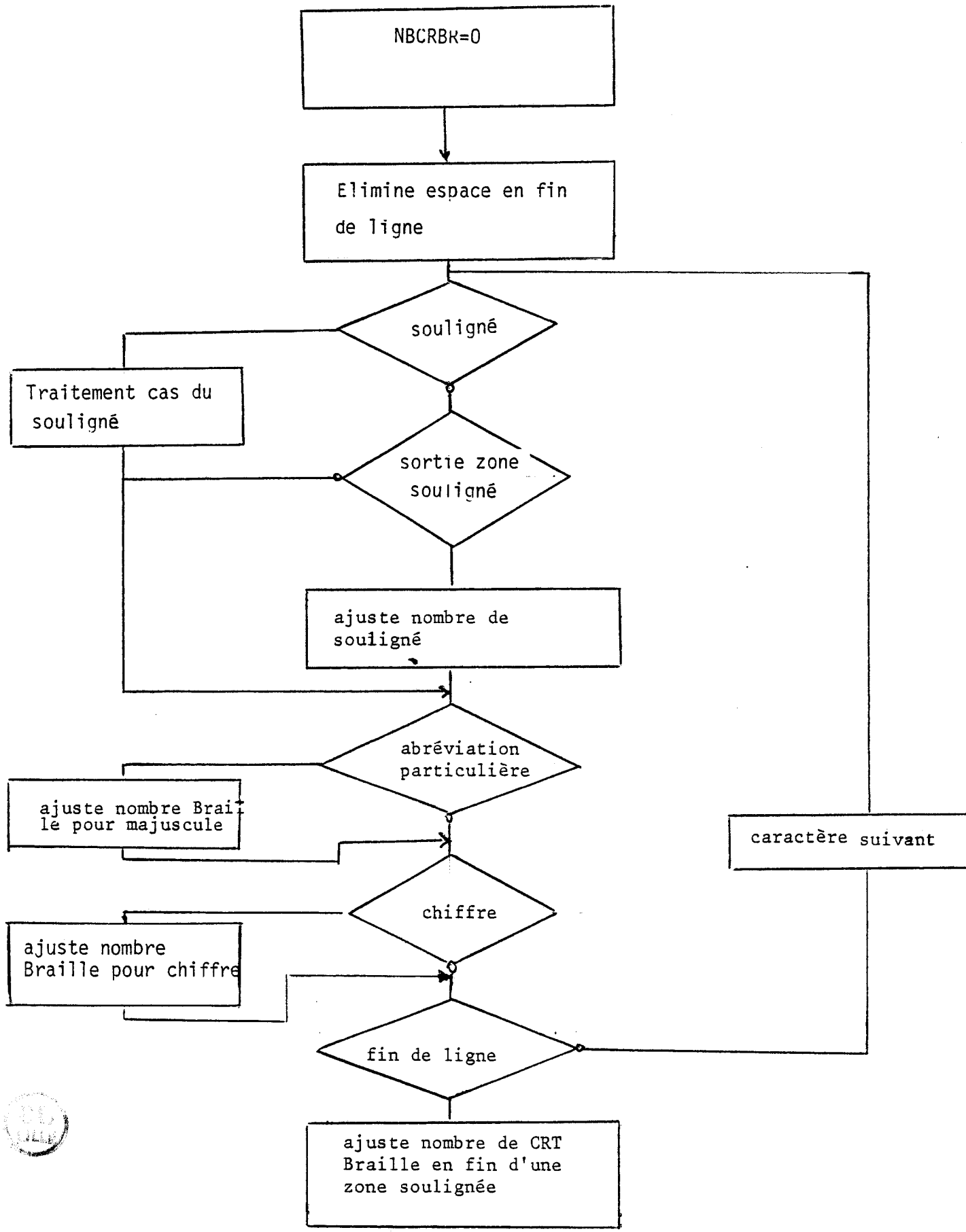


Figure IV-5 schéma descriptif du calcul du nombre de Braille

avec : $\delta_1\ \delta_2 \ldots\ \delta_n$ = ensemble de chaînes composant une ligne de texte noir ;

γ= facteur d'optimisation

γprend la valeur 1 lorsque $\delta_n$est la seule chaîne soulignée et 0 dans le cas contraire.

Les chaînes noires sont alors stockées en mémoire centrale prêtes à être visualisées et transcrites. Cette dernière opération s'effectue après acquittement de la touche de transfert page.

cependant si :

$$l(T(\delta_n) > N - \left(\sum_{j=1}^{n-1} l(T(\delta_j) + \gamma) \right) \qquad (2)$$

alors il faût envisager l'extraction de la dernière chaîne noire δ_n de façon à satisfaire l'inégalité (1).

Lorsque l'opérateur a choisi de ne pas saisir le texte au kilomètre, il est informé du remplissage de la ligne Braille par un réticule apparaissant quand la longueur de cette ligne atteint la valeur L comme indiqué sur la figure IV- 6

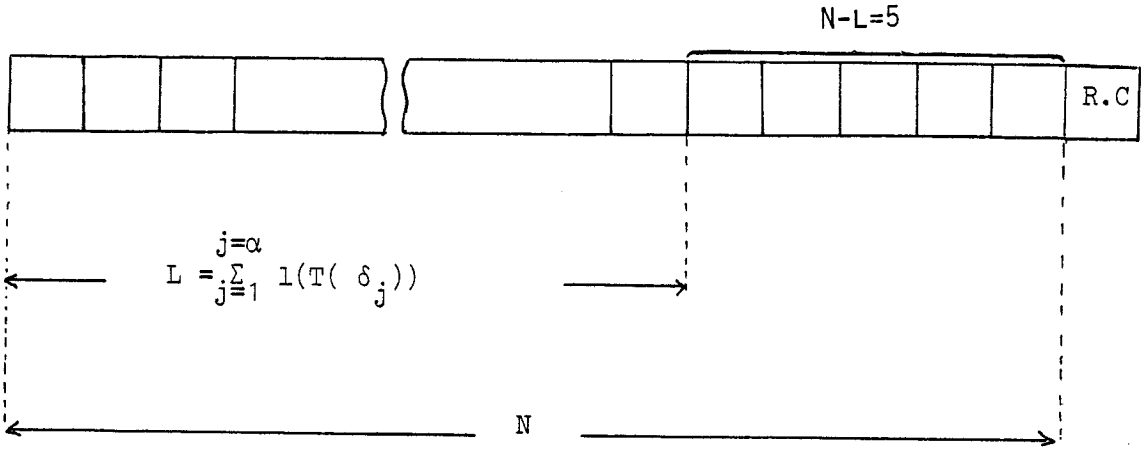


Figure IV-6 .

Mots type	Classe	offset coupure → 0	1	2	3		Probabilité P_i apparition classe
/	trivial 1	1	0	0	0	0	0,61
pousse	2	1	FF	0	0	1	0,10
souffle		1	FF	1	0	1	
halte		ℓ	1	0	0	1	
périoste		s	1	0	0	1	
extérieur		x	1	0	0	1	
infirme		r	1	0	0	1	
malgré	3	ℓ	1	1	0	1	0,11
astre		s	1	1	0	1	
extra		x	1	1	0	1	
orthographe		r	1	1	0	1	
rythme, aphteuse, technique		1	h	1	0	2	
asthme		s	1	1	1	1	
absent							
quelque		1	1	0	0	1	
arthrite		r	1	1	1	1	
constat	4	n	s	1	0	2	0,01
construire		n	s	1	1	2	
penser	5	n	s	0	0	1	0,08
combat		m	1	0	0	1	
conte		n	1	0	0	1	
branche		n	1	h	0	1	
synchrone		n	1	h	1	1	
oncle		n	1	ℓ	0	1	
contre		n	1	r	0	1	
onctueux		n	1	t	0	2	
amphore		m	1	h	0	1	
camphre		m	1	h	1	1	
comble		m	1	ℓ	0	1	
chambre		m	1	r	0	1	
compte		m	1	t	0	2	
asymptote							
prompt		m	1	t	1	FF	
être	6	1	r	0	0	0	0,09
éclisse		1	ℓ	0	0	0	
toucher		1	h	0	0	0	
ignore		1	n	0	0	0	
anachronique	6 bis	1	h	r	0	0	

Tableau IV-1

Dans le cas contraire, la procédure traite les chaînes en cours d'édition en fin de ligne selon leur nature : coupure syllabique par les chaînes alphabétiques déplacement en début de lignes suivantes dans les autres cas.

CESURE SYLLABIQUE

Pour réaliser l'automate de coupure syllabique, nous avons adopté les principes suivants :

- on ne coupe jamais un mot en laissant une lettre seule
- si le mot compte un tiret, la coupure se fait sur celui-ci
- on code le groupe de consonne compris entre le caractère test qui est le caractère situé à l'emplacement du marqueur de fin de ligne et la première voyelle rencontrée que l'on appelle voyelle pivot. (Figure IV-7)

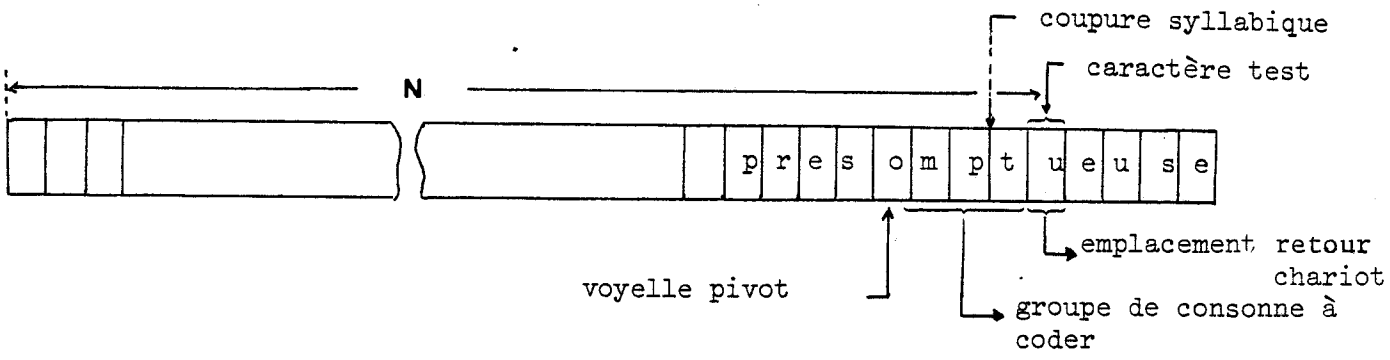


Figure IV- 7

Ce codage consiste à représenter une consonne quelconque par la valeur 1 et toute voyelle par la valeur 0. Toute consonne qui caractérise le type de la syllabe est représentée par sa valeur codée ASCII, tandis que la consonne doublée est codée par l'octet FF.

L'analyse orthographique de la langue française selon ce principe de codage permet d'une part de dénombrer 4 consonnes maximums dans un mot et d'autre part de ranger les formes de coupures syllabiques en 6 classes reportées dans le tableau n° IV-1 [80] [81]

La fonction de coût de détermination de la coupure syllabique par scrutation de la table IV-2 est donné par l'expression

$$F\ coût = \alpha_1 \left[\frac{P_1}{l_1} \sum_{i=0}^{i=l_1-1} i + \frac{P_2}{l_2} \sum_{i=l_1}^{i=l_1+l_2-1} i + \ldots + \frac{P_6}{l_6} \sum_{i=l_1+l_2+l_5}^{i=l_1+l_2+\ldots+l_6} i \right]$$

où α_1 est une grandeur qui dépend de l'écriture du programme du processeur et de la longueur des mots de la table et l_i, le nombre de codes dans la classe i.

En remarquant cependant que la plupart des coupures se font entre la première et la deuxième syllabe, on peut réduire considérablement cette fonction de coût en en adoptant l'algorithme suivant :

Algorithme de cesure syllabique

Soit n = le nombre de consonnes compris entre le caractère test ((n+1)ème caractère) et la voyelle pivot et $\delta = v_1 \ldots v_m$ la sous chaîne représentant le registre de codage

Soit FF, le code ASCII de l'offset de coupure indiquant que la césure est impossible si on ne trouve pas une autre sous chaîne δ encadrée par la précédente voyelle pivot qui devient le caractère test et une autre voyelle pivot .

```
Début

Partie A :   δ (n+1)  = voyelle

        si n = 1
        |
        |   Alors : "la coupure se fait implicitement derrière la voyelle
        |           pivot"
        |
        |           "offset coupure" : = 0
        |
        fsi

    pour 1 < n ≤4     scrutation du tableau IV-2
    |
    |   si δ ∉  au tableau n° IV-2
    |   |
    |   |     Alors "coupure derrière 1ère consonne
    |   |
    |   |           "offset coupure : =1
    |   |
    |   fsi
    |
    fpour

fin partie A


Partie B :   δ ( n+1) = consonne

        si n ≥ 2
        |
        |   Alors "scrutation tableau IV-2   "
        |
        |      si     δ ∉ tableau  IV-2
        |      |
        |      |   Alors "offset coupure": = 1
        |      |
        |      fsi
        |
        |   sinon "n < 2"
        |
        |      si δ ∈tableau  IV-2
        |      |
        |      |   Alors "offset coupure" : = valeur  tableau
        |      |
        |      |   sinon "offset coupure = FF"
        |      |
        |      fsi
        |
        fsi

fin Partie B
```

Type	0	1	2	3	offset coupure $\delta(n+1)$ = voyelle	offset coupure $\delta(n+1)$ = consonne	probabilité Pi
$3(\ell'_3)$	1	h	1	0	2	FF	0,11 = P_3
$4(\ell'_4)$	n	s	1	0	2	22	0,01 = P_4
	n	s	1	1	2	2	
$5(\ell'_5)$	n	1	t	0	2	FF	0,08 = P_5
	m	1	t	0	2	FF	
	m	1	t	1	FF	FF	
$6(\ell'_6)$	1	r	0	0	0	0	0,09 = P_6
	1	ℓ	0	0	0	0	
	1	h	0	0	0	0	
	1	n	0	0	0	0	
	1	h	r	0	0	0	

Tableau IV-2

La fonction de coût de la coupure se réduit alors à :

$$\alpha \left[\frac{P3}{1'3} \frac{(1'_{3-1})\ (1'_{3-2})}{2} + \ldots\ldots + \frac{P_6}{1'_6} \left(1'_6(1'_3+1'_4+1'_5) + \frac{(1'_6-1)\ (1'_6-2)}{2} \right) \right]$$

L'application numérique donne

13,45 α unités de coût dans le premier cas

1,73 α unités de coût dans le dernier cas

Elle permet de chiffrer un gain de temps de recherche des codes dans la table de l'ordre 8.

IV-2-3-1-3- MANIPULATION DE TEXTE ET FONCTION DE CORRECTION

Parmi les opérations de manipulation de textes et utilisé par la fonction correction on trouve :

- le déplacement du curseur dans le texte : à l'aide des quatre flèches et de séquences de touches spécifiques on peut accéder soit directement à la fin ou début d'une ligne ordinaire d'une page soit caractère par caractère en déplacement lent.
- le choix d'un format variable pour l'édition des textes selon les diverses normes existantes en Braille;
- l'insertion ligne et caractère avec extraction automatique des chaînes en bout de ligne et reformatage de la page ou déplacement de la ligne concernée.
- la suppression de caractères ou de lignes spécifiés par la position du curseur avec reconstitution automatique en bout de ligne des chaînes extraites situées sur la ligne suivante .
- la modification des caractères
- la saisie automatique du numéro des pages
- la tabulation automatique.

IV-2-3-2- LA PROCEDURE "VISUALISATION"

Le stockage en mémoire centrale des chaînes de caractères du texte noir conditionnées Braille s'éffectue conformément au paramètres de mise en page indiqués sur la figure IV-8 .

Sur cette figure la présence de zones de variables permet comme dans les traitements classiques d'afficher certaines informations d'aide à la mise en page. Par exemple la ligne système située à la nième ligne de la fenêtre de visualisation, sert pour la communication homme-machine. Elle se déplace toujours avec la fenêtre sans altérer le texte édité.

Soit f(x,y,z) la fonction représentant le contenu d'un document de n pages à éditer (figure IV-9) donné par :

$$f(x,y,z) = \sum_{i=1}^{n} z_i \text{ NBTLGN x NBTCRT} = \sum_{i=1}^{n} f_i(x,y)$$

avec x= NBTLGN = nombre total de lignes par page.

y= NBTCRT = nombre total de caractères par page

et $f_i(x,y)$= i x NBTLGN x NBRCRT = la fonction représentant une page quelconque du document.

On définit la fonction de fenêtrage comme étant l'application qui à toute page $f_i(x,y)$ associe une portion de cette page donnée par :

$$g(f_i(x,y)) = (\text{NUMLGN}(n) - \text{NUMSYS}) \times \text{NBTCRT}$$

avec NUNLGN(n) = (DEBPAG) + (n-1) (NBTCRT + 1)

pour $1 \leqslant n \leqslant$ NBTLGN

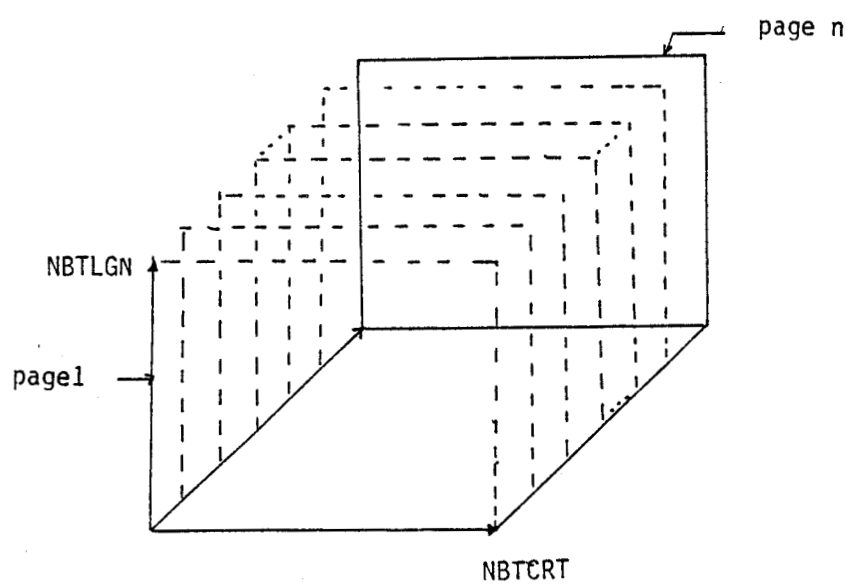
En tenant compte des zones de variables pouvant figurer dans la partie de texte encadrée par la fenêtre, l'expression de la fonction de fenêtrage devient :

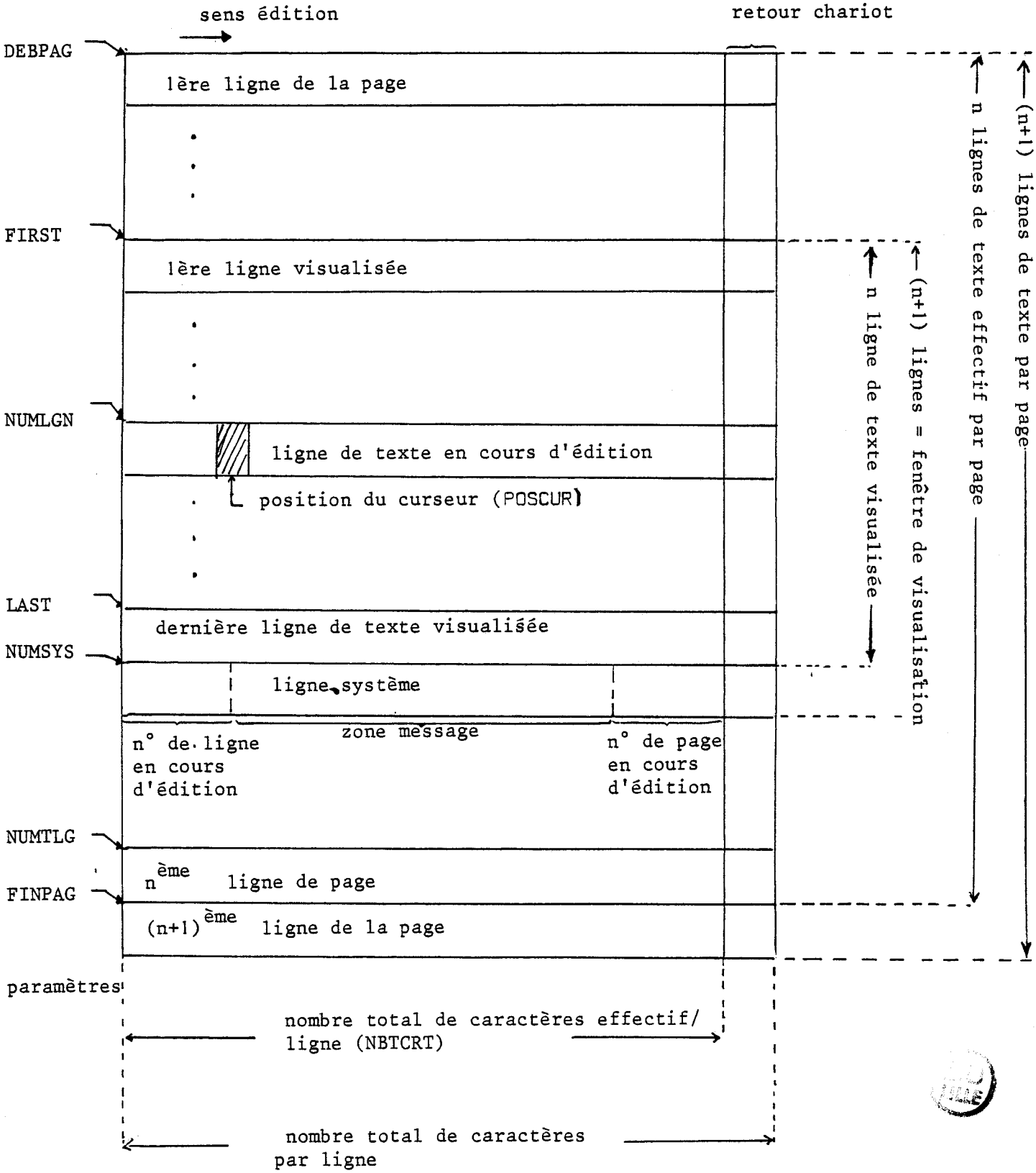
$$g(f_i(x,y) = (\text{NUMLGN}(n) - \text{NUMSYS}) \times \text{NBTCRT} - \alpha$$

avec α = longueur de la zone de variables.

On peut alors définir la fonction de visualisation d'une partie de texte comme étant l'application qui à toute chaîne du plan mémoire g(f(x,y) associe son image sur l'écran. En pratique cette tâche est réalisée par l'utilisation des routines du micro-ordinateur hôte

Figure IV-9





E(c) = NUMLGN(n) + POSCUR = emplacement d'un caractère quelconque dans la page

Figure IV- 8

IV-3- DESCRIPTION DE LA FONCTION DE TRANSCRIPTION BRAILLE INTEGRAL

L'interpréteur syntaxique Braille intégral assure en mode programme, la transcription d'un texte noir en Braille intégral. La grammaire de transcription constituant cette fonction est une grammaire d'états finis dont le formalisme, analogue à celui des automates d'états finis, nous permet d'exposer la démarche conceptive de cet interpréteur.

Celle-ci consiste à analyser de façon aussi exhaustive que possible les différentes règles de conversion de syntaxe, d'établir la structure la plus compacte des automates réalisant les règles de conversion et d'assembler ces automates sous formes de réseau de transition pour constituer l'automate complet.

IV-3-1- Examen des différents cas de conversion de syntaxe noir-Braille intégral

Les colonnes 2 et 3 du tableau IV-3 résument, à titre d'exemple, les différents cas de conversion de syntaxe à réaliser entre un texte noir et un texte Braille intégral.

L'ensemble de définition des règles de transcription est donc constitué de différents types de chaînes "noires" énumérées à la colonne 1 ainsi que des symboles représentés dans la colonne intitulée "classe lexicale" de ce même tableau. Les symboles de cette dernière colonne appartiennent à l'alphabet d'entrée de chaque automate, et donc au contexte d'analyse de chaque règle.

En particulier, la première classe relative à la syntaxe des chaînes numériques met en jeu les symboles "chiffres" et le symbole "espace" en tant que séparateur de milliers et le symbole "virgule" en tant que séparateurs entre les parties entières et décimales.

La classe relative à la chaîne "ponctuation" met en évidence le fait qu'en syntaxe noir, les caractères de ponctuation simple (virgule, point) doivent normalement être accolés en mot, contrairement aux caractères de ponctuation composés (; ,:,...) qui doivent être séparés du mot par un espace, tandis qu'en Braille

le mot doit être accolé à la ponctuation quelle qu'elle soit. Une erreur éventuelle de syntaxe noir du type "espace entre fin de mot et ponctuation" doit donc être corrigé par l'automate correspondant à ce type de chaîne.

On constate par ailleurs qu'il existe une intersection non vide entre les différentes classes des divers types de chaînes pris séparément. Par exemple, on voit que le point pris en tant que fin de chaîne alphabétique peut séparer aussi, soit les lettres d'un sigle , soit les chaînes numériques pour les dates. Nous traiterons donc les problèmes que posent ces intersections selon leur contexte soit par un même automate, soit par le passage d'un automate à un autre.

Type de chaîne	Exemple de syntaxe dactylographique	Syntaxe Braille intégral	Classe lexicale
<c.n>	nombres		ch = {01,2,...9}
	4 382,5	PN 4' 382,5	s_1 = {␣} s_2 ={,}
<c.n.c.>	dates		ch ={01,2,...,9}
	1.04.85	PN1-PN04-PN85	s_1 = {␣} s_2 {,} s_3 = {.}
	Opération arithmétique		s_4 = op = {+,-,1,*}
	1/2	PN 1.2	
	+4 -3 = +1	+PN4 -3 = +1	
<c.a>	Minuscule		s_5 = {; ! ? : .}
	faire	faire	s_1 = {␣} S_3 = {.} s_2 = {,}
	Majuscule		1 = {a,b,...z}
	Faire	PM faire	<L> = {A, B, ..., Z}
	FAIRE	PM faire	
	ILE	PM île	
<c.a.p>	Sigle		s_3 = {.} s_1={␣}
	A.V.H.	PMA ' PMV ' PMH '	ℓ_1 = {r, m, ℓ, e}
	Abréviations particulières		
	M ou M.	PM Mr	
	Mr ou Mr.	PM Mr	
	MM ou MM.	PM Mrs	
	Mme	PM Mme	
	Mlle ou Melle	PM Mlle	
	Mlles ou Melles	PM Mlles	

.../...

.../...			
<Pt>	Ponctuation simple chaîne . , Ponctuation composée chaîne 4 . , ; : ? !	chaîne . , ; : ? !	s_2 = {,} s_3 = {.} s_5 = {;!?: ...}
<c.i>	guillemet "mots" Souligné n = nombre mots < 3 n = nombre mots > 3	"mots" PS mot 1 PS mot 2 PS mot 3 3A PS mot 1 ...PS mot( )	
<c.p.>	Pour cent : 25 % Pour mille : 25 % Tiret initial : - Tiret final : - numéro et degré : n° 23 23° Nième 1er 2ème ou 2me	PN 25 (ϕ # ϕ PN 25 (ϕ # ϕ ϕ PN - - ▼ n * 0 PN 23 PN 23 0 PN 1 * er PN 2 * me	"%" ‰ tiret "°" <"er"> <"me"> <"ème">

Tableau IV-3

IV-3-2- Notations

Nous noterons $\delta(i,j)$ la chaîne de caractères $m_i\ m_{i+1} \ldots m_j$ avec $i < j$ et $j \in [i+1,n]$ tel que :

$$\ell\ [\delta(i,j)] = j-i+1$$

Pour $j = i$, $\delta(i,j) = \delta(i,i) = \delta(i)$

Afin de symboliser le mieux possible le rôle dans la transcription des éléments constitutifs d'une chaîne noire, celle-ci pourra être décomposée de la manière suivante :

$$[CG] + [CT] + [CD] = CB$$

avec :

CG = contexe gauche ; il s'agit de symboles m_i de $\delta(i,j)$ avec $i \in [1, i-1]$ correspondant à la partie gauche de la chaîne déjà analysée ;

CT = caractère(s) à transcrire;ce sont les caractères pointés par l'analyseur syntaxique Braille ;

CD = Contexte droit ; ce sont les symboles de m_i de $\delta(i,j)$ avec $i \in [i+1, n\ ]$ de la partie droite de la chaîne restant à analyser ;

CB = caractère(s) Braille transcrit(s).

Une règle de transcription Braille intégral s'applique à une chaîne noire, lorsqu'il y a identité entre la chaîne et la règle pour les termes de CT et identité ou appartenance pour les termes CG et CD.

La connaissance des contextes gauche et droit du (des) caractères à transcrire même si ceux-ci correspondent à la chaîne vide, permet de déterminer les informations morphologiques et linguistiques nécessaires au décodage de la chaîne Braille par l'aveugle (figure IV-10).

Exemple : Syntaxe noire — 4 878,5

Syntaxe Braille intégral — PN 4 878 , 5

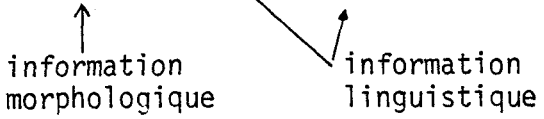


Figure IV-10

Nous utiliserons également les notations suivantes :

<c.n> : chaîne numérique

<c.n.c> : chaîne numérique composée

<c.a> : chaîne alphabétique

<c.a.c> : chaîne alphabétique composée

<pt> : ponctuation

<c.i> : chaîne indiquée

<c.p> : chaîne particulière

sp : caractère espace

<sep> : séparateur

<ct.q> : caractère quelconque

R : ensemble des règles de transcription

Rg[$\delta(i,j)$] : numéro de la règle à vérifier pour la chaîne $\delta(i,j)$

H [$\delta(i,j)$] : le procédé de vérification des règles de transcription de la chaîne $\delta(i)$

H[$\delta(i,j)$] est une application de R dans 0,1 . Elle prend la valeur 1 Lorsque les règles s'appliquant à $\delta(i,j)$ sont vérifiées et 0 dans le cas contraire.

Min[$\delta(i,j)$] : la fonction qui à toute chaîne alphabétique noire associe sa correspondance Braille en minuscule

IV-3-3- Enumération des différentes règles de transcription

Nous donnons ici les règles de transcription relatives à l'analyse de la transcription des différents types de chaînes.

Le formalisme utilisé pour la description des règles est assez classique : les symboles entre "<" et ">" sont des non-terminaux de l'analyse. Chaque classe lexicale est représentée par son mnémonique. Les chaînes entre crochets ("[", "]") constituent les règles à vérifier. Dans certains cas de spécifité des caractères, ou de risque d'ambiguité, nous noterons entre guillemets (") les symboles pouvant apparaître dans la règle.

Rg	$\delta(i,j)$	H $[\delta(i,j)]$ =	CB
01	<c.n.>	Soit m = nombre de caractères à partir du ième caractère i = 1 en début de chaîne ns_1 = nombre de séparateurs de milliers (<"␣">) <u>Pour $m - ns_1 < 3$</u> $[\delta(i-1) \cap \delta(i-2) \neq ch]$ $[\delta(i,n) = \langle ch \rangle \cup \langle s_2 \rangle]$	PN $\delta(i,n)$
		<u>Pour $m - ns_1 > 3$</u> $[\delta(i-1) \neq \langle ch \rangle] \cap [\delta(i,m-ns_1) = \langle ch \rangle \cup \langle s_2 \rangle]$ $\cap [\delta(i+1+4\,(ms_1-1)) = \langle s_1 \rangle]$	PN $\delta(i,ms_1\ \alpha)$ $[\delta(i+1+4\,(ms_1-1) = ']$
02	<c.n.c.>	<u>Date :</u> n_{s3} = nombre de séparateur "." dans les dates $[\delta(i-1) \cap \delta(i-2) \neq ch] \cap [\delta(i,m-n_{s3}) = ch]$ $\cap [\delta(i+2+3\,(n_{s3}-1)) = s_3]$	PN $\delta(i,m-n_{s3})$ $[\delta(i+2+3\,(n_{s3}-1)) =$ "_"]
		<u>Expression arithmétique</u> α = nombre de chaîne numérique ns_4 = nombre d'opérateur arithmétique dans une expression $[\delta(i-1) \cap \delta(i-2) \neq ch.] \cap$ $[\delta(i,\alpha - n_{s4}) = \langle c.n. \rangle] \cap$ $[\delta(i+1+2\,(n_{s4}-1)) = s_4]$	PN $\delta(i,\alpha - n_{s4})$ $\delta(i+1+2\,(n_{s4}-1))$

Tableau IV-4

Rg	δ (i,j)	H [δ(i,j)] =	CB
03	<c·a·>	Majuscule : $\delta(i) = M_j$ β = le nombre de caractère de la chaîne numérique $[\delta(i-1) = sp] \cap [\delta(i+1,\beta) \quad S_3]$	"PM" Min $[\delta(i, \beta)]$
		Minuscule : $\delta(i) = l$ = lettre minuscule quelconque $[\delta(i-1) = s_1] \cap [\delta(i+1,\beta) = s_3]$	Min $[\delta(i, \beta)]$
04	<c.a.p>	Sigle $[\delta(i-1) = s_1] \cap [\delta(i,\beta) = Mj \cap s_3]$	PM $\delta(i,\beta - ns_3)$ $[\delta(i+1+2(ns_3-1)) = " \ "]$
		Abréviations particulières $\delta(i)$ = "M" $[\delta(i-1) = s_1] \cap$ $[\delta(i+1,\beta) = s_1 \cup s_3 \cup "r\sqcup" \cup "r"]$	PM $\delta(i)$ "r"
		$[\delta(i-1) = s_1] \cap [\delta(i+1,\beta) = "M\sqcup" \cup "M."]$	PM $\delta(i)$ "rs"
		$[\delta(i-1) = s_1] \cap [\delta(i+1, \beta) = "me "]$	PM $\delta(i,\beta)$
		$[\delta(i-1) = s_1] \cap [\delta(i+1, \beta) = "elle" \cup "lle"]$	PM $\delta(i)[\delta(i+1, \beta) =$ " lle"]

Tableau IV-5

Rg	$\delta(i,j)$	H $[\delta(i,j)]$ =	CB
05	< pt >	Ponctuation simple : $\delta(i) = s_1 \cup s_3$ $[\delta(i-1) = 1] \cap [\delta(i+1) = s_1]$ Ponctuation composée : $\delta(i) = s_5$ $[\delta(i-1) = s_1] \cap [\delta(i+1) = s_1]$	$\delta(i-1, i+1)$ $\delta(i-1, i) = \delta(i, i+1)$
06	< c.i >	souligné: soit m= nombre de chaîne souligné m_1 = nombre total de caractère compris entre le début du 1er mot et la fin du dernier mot souligné. n_1 = longueur de la première chaîne n_m = longueur de la nième chaîne $m \leq 3$ $[\delta(i, m_1 - n_m + 1) = "\cup"] \cap [\delta(i, m_1 - (m-1)) = < c.q. >]$ $m > 3$ Guillemet soit m = nombre total de chaîne compris entre les guillemets $[\delta(i) = <">] \cap [\delta(i+m) = <">]$	PS T $<c, q_m>$ $\delta(i, n_1 - n_m + 1)$ 3A PS T $[\delta(i, n_1)]$ T $< c.q_{m-2} >$ PS T $[\delta(i, n_m)]$ $\delta(i, m_1 - n_m + 1)$ $[\delta(i) = "]$ $[\delta(i+m) = "]$

Tableau IV-6

IV-3-5- EXPOSE DU PRINCIPE D'ANALYSE SYNTAXIQUE D'UNE CHAINE

La méthode employée consiste d'une part à construire l'automate vérifiant les différentes règles de transcription relative à une chaîne, d'autre part à réduire sous une forme minimale le graphe représentant cet automate.

Nous décrivons le principe de la méthode sur l'exemple de la 1ere chaîne.

soit A = (X,Q,⊗,qo,S), l'automate d'analyse syntaxique d'une chaîne numérique.

où X= alphabet d'entrée = $\{ch, s_1, s_2\}$

Q = l'ensemble des états

Qo = l'état initial

S = l'ensemble des états de sortie

⊗ = la fonction de transmition telle que $Q \otimes X \longrightarrow Q$

L'analyse syntaxique d'une chaîne numérique qui s'effectue par l'examen de toutes les formes possibles que peut prendre cette chaîne conduit à l'automate décrit par le graphe de la figure IV-11 .

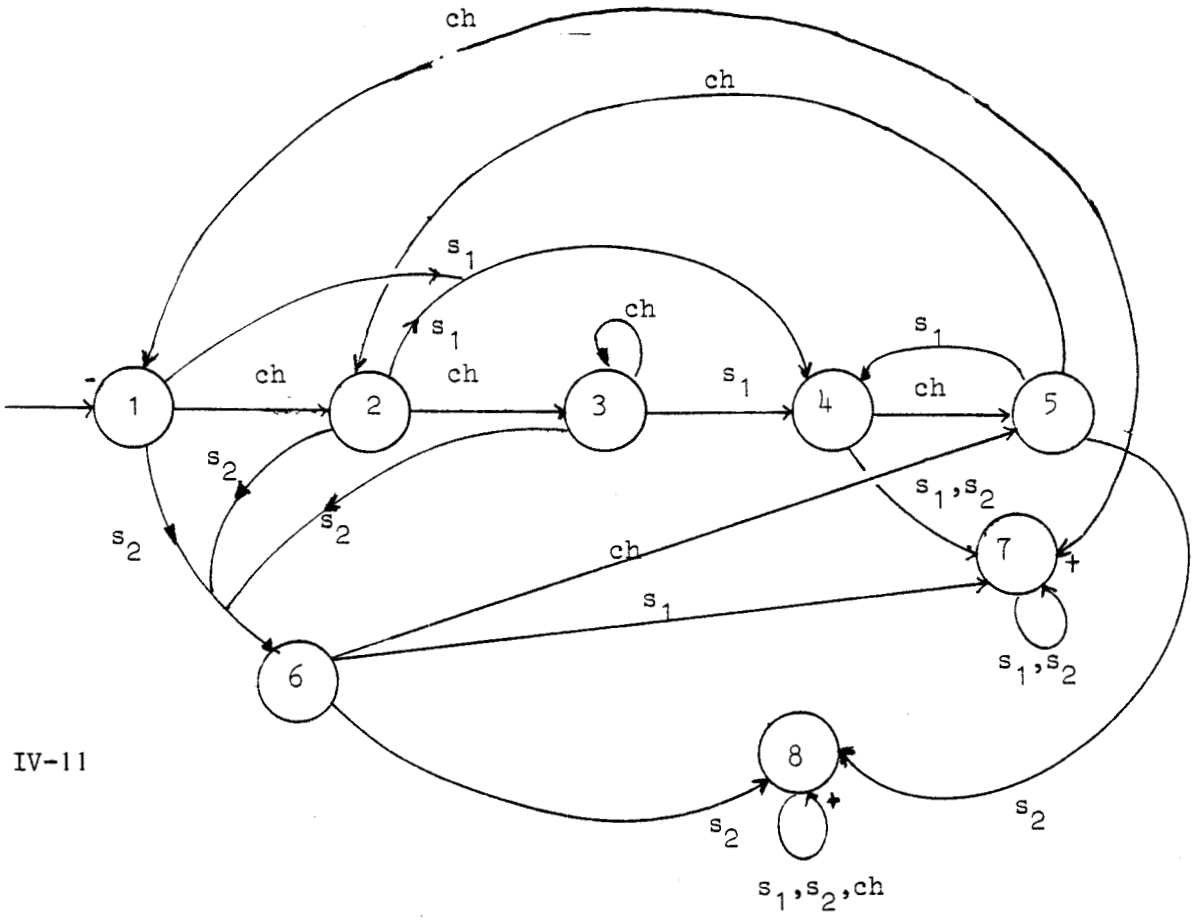


Figure IV-11

Rg	$\delta(i,j)$	H $[\delta(i,j)]$ =	CB
05	< pt >	Ponctuation simple : $\delta(i) = s_1 \cup s_3$ $[\delta(i-1) = 1] \cap [\delta(i+1) = s_1]$ Ponctuation composée : $\delta(i) = s_5$ $[\delta(i-1) = s_1] \cap [\delta(i+1) = s_1]$	$\delta(i-1, i+1)$ $\delta(i-1, i) = \delta(i, i+1)$
06	< c.i >	souligné: soit m= nombre de chaîne souligné m_1= nombre total de caractère compris entre le début du 1er mot et la fin du dernier mot souligné. n_1= longueur de la première chaîne n_m =longueur de la nième chaîne $m \leq 3$ $[\delta(i, m_1 - n_m + 1) = "\cup"] \cap [\delta(i, m_1 - (m-1)) = <c.q.>]$ $m > 3$ Guillemet soit m = nombre total de chaîne compris entre les guillemets $[\delta(i) = <"> \cap [\delta(i+m) = <">]$	PS T $<c, q_m>$ $\delta(i, n_1 - n_m + 1)$ 3A PS T $[\delta(i, n_1)]$ T $< c.q_{m-2} >$ PS T $[\delta(i, n_m)]$ $\delta(i, m_1 - n_m + 1)$ $[\delta(i) = "]$ $[\delta(i+m) = "]$

Tableau IV-6

Rg	δ (i,j)	H [δ(i,j)] =	CB
07	<c.p>	pour cent : [δ(i) = "%"]	[δ(i,5)=(ϕ ≠ ϕ]
		pour mille : [δ (i) = "‰"]	[δ (i,j) = (ϕ # ϕ ϕ]
		Tiret initial et final: m= nombre total de chaîne compris entre les tirets [δ(i-1) = δ(i-2) = δ(i-3) ≠ ch] ∩ [δ (i+1) = δ (i+2) ≠ ch] ∩ [δ (i+m) = δ (i) = "-"]	[δ (i) = (-] [δ (i+m)= - ▼]
		Symbole degré et numéro δ(i) = "o" numéro: [δ(i-1) = "n"] degré : [δ(i-1) = ch]	δ(i-1) [δ(i)] = * o T [δ(i-1)] δ(i)
		Nième [δ(i,j) = "er" ∪ "ième" ∪ "eme"] [δ(i-1) = "1"] ∩ [δ(i,j) = "er"] [δ(i-1) = ch] ∩ [δ(i,j)="me" ∪ "ème"]	δ(i,j) δ(i,j)

Tableau IV-7

IV-3-5- EXPOSE DU PRINCIPE D'ANALYSE SYNTAXIQUE D'UNE CHAINE

La méthode employée consiste d'une part à construire l'automate vérifiant les différentes règles de transcription relative à une chaîne, d'autre part à réduire sous une forme minimale le graphe représentant cet automate.

Nous décrivons le principe de la méthode sur l'exemple de la 1ere chaîne.

soit A = (X,Q,⊗,qo,S), l'automate d'analyse syntaxique d'une chaîne numérique.

où X= alphabet d'entrée = $\{ch, s_1, s_2\}$

Q = l'ensemble des états

Qo = l'état initial

S = l'ensemble des états de sortie

⊗ = la fonction de transmition telle que $Q \otimes X \longrightarrow Q$

L'analyse syntaxique d'une chaîne numérique qui s'effectue par l'examen de toutes les formes possibles que peut prendre cette chaîne conduit à l'automate décrit par le graphe de la figure IV-11 .

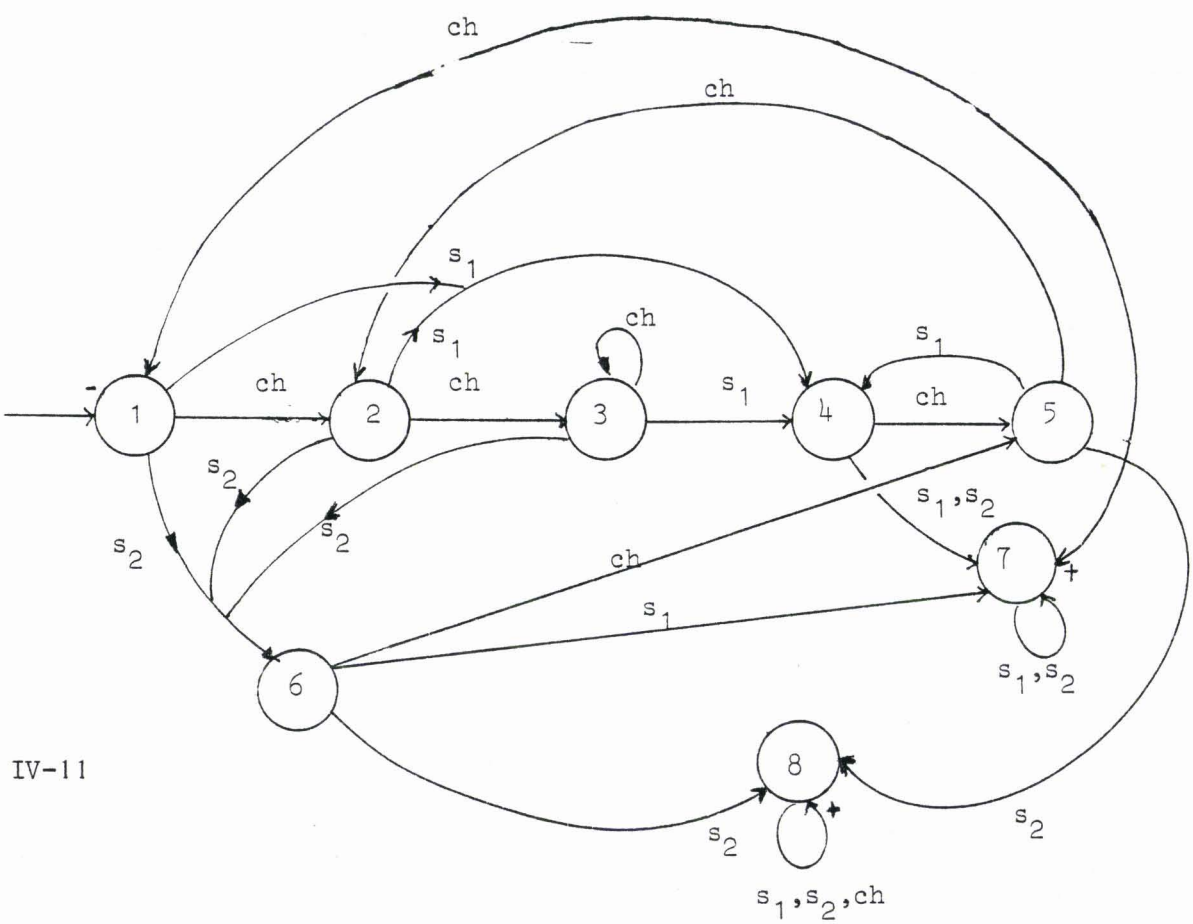


Figure IV-11

Sur ce graphe, les noeuds représentent les différents états de l'automate, les arcs représentent les transitions possibles d'un état à un autre concrétitées par l'apparition d'un élément de l'alphabet d'entrée. Les différents états ont les significations suivantes :

1 : c'est l'état initial ou le début d'analyse. Il représente l'apparition du premier chiffre d'un nombre, c'est à dire précédé d'un marqueur de fin d'analyse (deux espaces ou tout autre caractère autre qu'un opérateur suivi d'un espace).

2 : réception d'un deuxième chiffre (ou deuxième chiffre d'une série de millier).

3 : réception d'un nième chiffre :

- n=3 pour une série de milliers
- n=4 quatrième chiffre consécutif, il faut placer le séparteur de milliers :

si $\delta(i,n-4) = s_1 \cup s_2$

alors $\delta(n) := sp \cap \delta(n+1) := \delta(n)$

4 : réception d'un espace si :

- précédé de chiffre et espace uniquement et:
  - suivi de 3 chiffres ⟶ cas d'un séparateur de milliers
  - pas suivi de 3 chiffres ⟶ début d'un marqueur de fin de chaîne numérique

$\delta(n-\alpha-1) := \delta(n-\alpha-2)$

$\delta(n-\alpha-2) := s_1$

α = nombre de chiffres entre les deux derniers espaces.

- précédé d'un séparateur de partie entière ⟶ erreur

5 : réception d'un chiffre après :

- un espace précédé d'un chiffre début du concept millier
- une virgule précédé d'un chiffre début du concept de partie décimale.

6 : réception d'une virgule précédé d'un chiffre (séparateur de partie entière et décimale)

7 : sortie chaîne numérique correcte

8 : sortie chaîne numérique incorrecte.

Le graphe d'un tel automate n'a pas en général une forme minimale. La réduction peut se faire par trois méthodes : méthode de la table de transition, méthode de l'algorithme de la partition minimale, algorithme de l'automate miroir [53] . Nous décrivons les deux méthodes les plus utilisées.

METHODE DE LA TABLE DE TRANSITION

Le tableau IV-8 représente la fonction de transition de l'automate, Dans ce tableau, les éléments de X sont disposés sur la première ligne tandis que les éléments de Q figurent sur la première colonne. Chaque case du tableau représente l'état atteint par l'automate, après reconnaissance de l'élément de X située sur la colonne correspondante: $Q_{i\ell} = Q_i \otimes X_\ell$

Tableau IV-8

Q \ X	ch	s_1	s_2
1	2	4	6
2	3	4	6
3	3	4	6
4	5	7	7
5	2	4	8
6	5	7	8
7	1	7	7
8	8	8	8

Sur cette table on remarque immédiatement que :

$\forall x \in X \qquad Q_i \otimes x = Qj \otimes x \quad$ modulo $\otimes$

entraîne Q_i équivalente à Q_j que l'on note $Q_i \equiv Q_j$ (modulo $\otimes$)

L'équivalence des états 2 et 3 du tableau IV- 8 permet de remplacer $Q_{11} = 2$ par $Q_{11} = 3$ et de mettre en évidence l'équivalence des états 1, 2 et 3 que l'on réduit à un seul état (1,2,3). Il en résulte graphe réduit de la figure IV-12

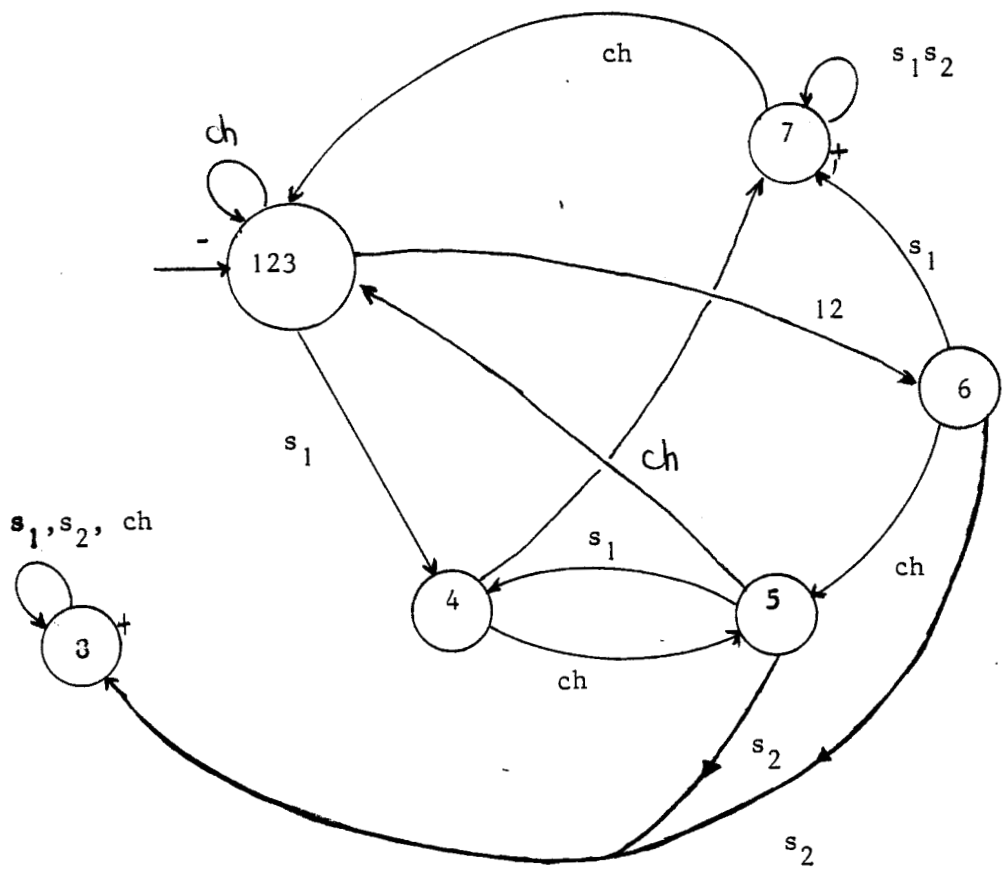


Figure IV-12

Méthode de la partition minimale

Dans cette méthode on cherche à aboutir à une partition régulière de l'ensemble Q des états en un nombre de sous ensembles N inférieur ou égal au nombre des éléments de Q. On obtient alors un automate réduit à N états.

soit $A = (X, Q, \otimes, q_o, S)$, l'automate précédent.

Si l'on écrit : $Q = \{ P_1, P_2 \}$ avec

$P_2 = \{1,2,3,4,5,6\}$ $\qquad P_1 = \{7,8\}$

En s'aidant du tableau de la figure 3 on remarque que :

$$\forall\, q_i \in P_1,\ q_i \otimes s_1 = Q_{i1} \in P_1,\ q_i \otimes s_2 = Q_{i2} \in P_1$$

mais $q_i \otimes s_1$ = soit $\{1\} \in P_2$, i=7

soit $\{8\} \in P_1$, i=8

On peut donc partitionner P_1 en deux sous ensembles irréductibles:

$P_1 = \{7\}$ et $P_3 = \{8\}$

Si à nouveau on effectue :

$$\forall\, q_i \in P_2 \ ,\ q_i \otimes ch = Q_{i1} \in P_2$$

mais $\forall\, q_i \in P_2$, $q_i \otimes s_1$ = soit $\{4\} \in P_2$, i=1,2,3,5

soit $\{7\} \in P_1$, i=4,6

donc $P_2 = \{1,2,3,5\}$ $\quad P_4 = \{4,6\}$ $\quad P_1 = \{7\}$ $\quad P_3 = \{8\}$

En poursuivant ce cheminement pour P_2 et P_4 par s_2 on obtient l'arborescence de la figure IV-13 .

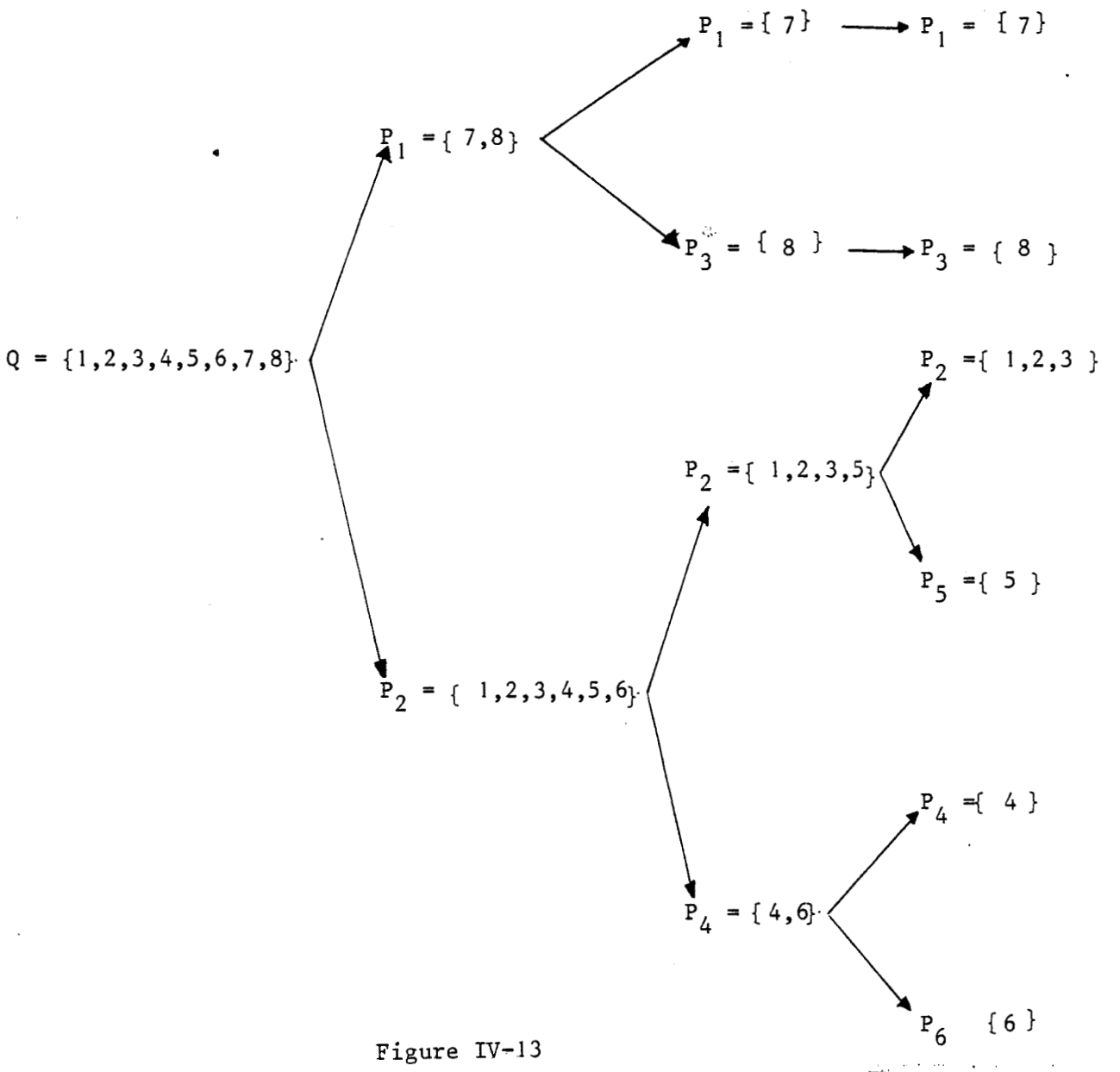


Figure IV-13

irréductibles

On déduit l'algorithme de la partition minimale :

soit $Q_q = \{ S_i, \ldots, S_k, \ldots, S_p \}$ une partition régulière de Q en P sous ensembles (P = 2 au début) avec

$S_1 = \{q_1, \ldots q_a\} \ldots\ldots S_k = \{ q_1, \ldots q_3\} \ldots\ldots S_p = \{ q_j, \ldots, q_n\}$

Tant qu'il existe un sous ensemble S irréductible, alors :

Début : $\forall x \in X$, $\forall q_i \in S_i$ et $q'_i \in S_i$

si $q_i \otimes x = q_l$ avec $q_l \in S_i$

$q'_i \otimes x = q_k$ avec $q_k \in S_j$

alors "partitionnement"

$S_i := \{ q_i \}$

$S'_i := \{ q'_i \}$

sinon "irréductibilité"

$\forall x$, $q_i \otimes x = q_l$ avec $q_l \in S_j$ et $j = i$ ou $j \neq i$

fsi

fin.

L'application systématique de cet algorithme conduit au graphe réduit de la figure IV-12.

IV-3-6- ANALYSE SYNTAXIQUE DES DIFFERENTS TYPES DE CHAINES

Il apparaît après examen des tableaux IV-4 à IV-7 l'intersection et/ ou l'inclusion entre les éléments appartenants aux classes lexicales des différents types de chaînes. Par exemple une expression arithmétique est l'association de plusieurs chaînes numériques avec divers opérateurs. De même le point apparaît dans les dates et les ponctuations.

Au terme de cet examen, nous avons convenu de décrire la fonction de transcription par un réseau de transition. Pour cela nous tentons :

- d'établir l'ensemble des automates optimalisés pouvant figurer comme étiquettes associées aux arcs d'automates plus complexes
- de regrouper les différents cas présentants une inclusion dans la classe lexicale des résultats. C'est à dire au niveau de l'espace de la syntaxe Braille.

L'application systèmatique de cette démarche conduit au réseau de transition décrit par les figures suivantes.

GRAPHE DE L'AUTOMATE D' ANALYSE DES CHAINES NUMERIQUES DATES ET EXPRESSIONS ARITHMIQUES

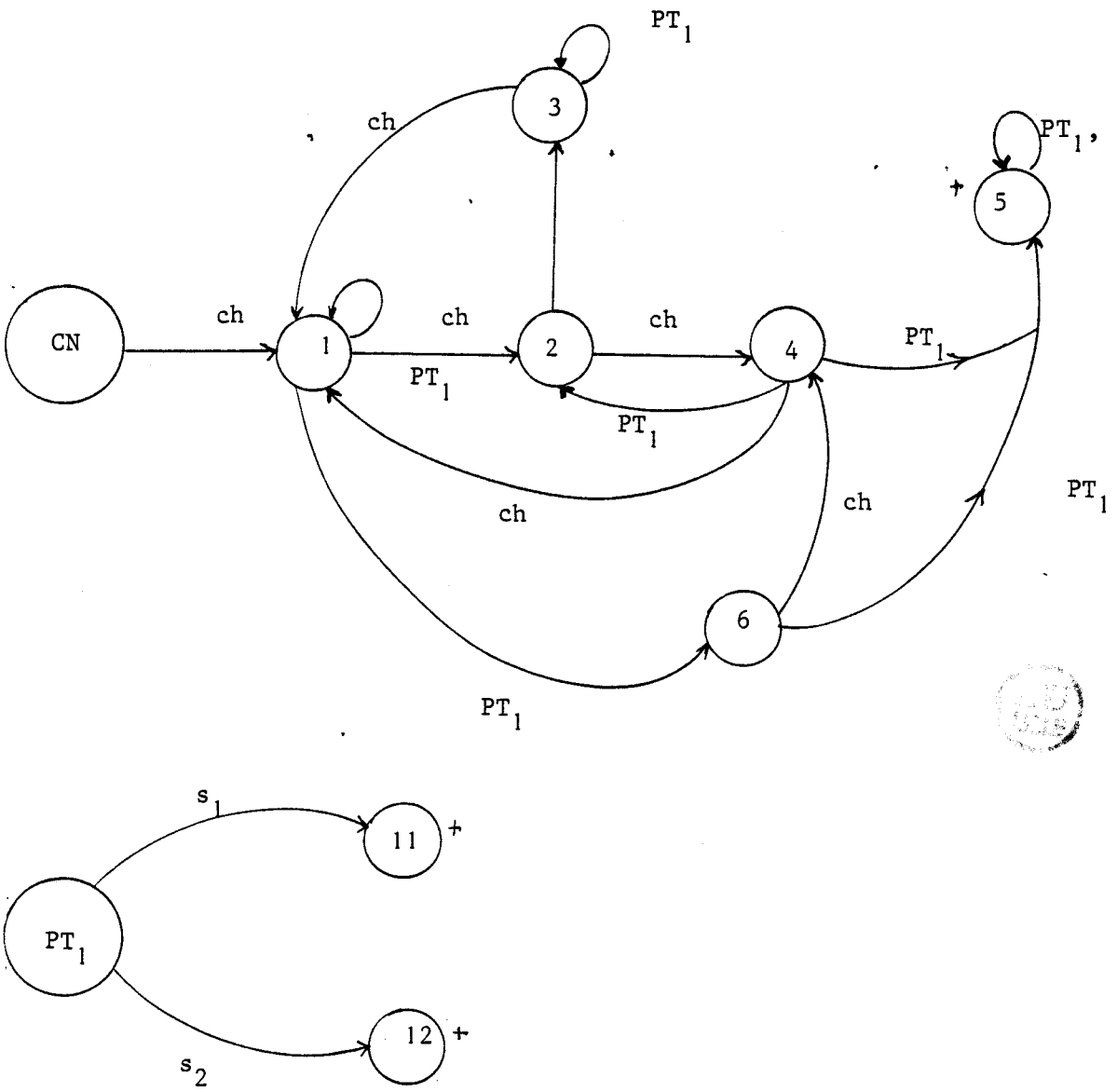


Figure IV-14-a

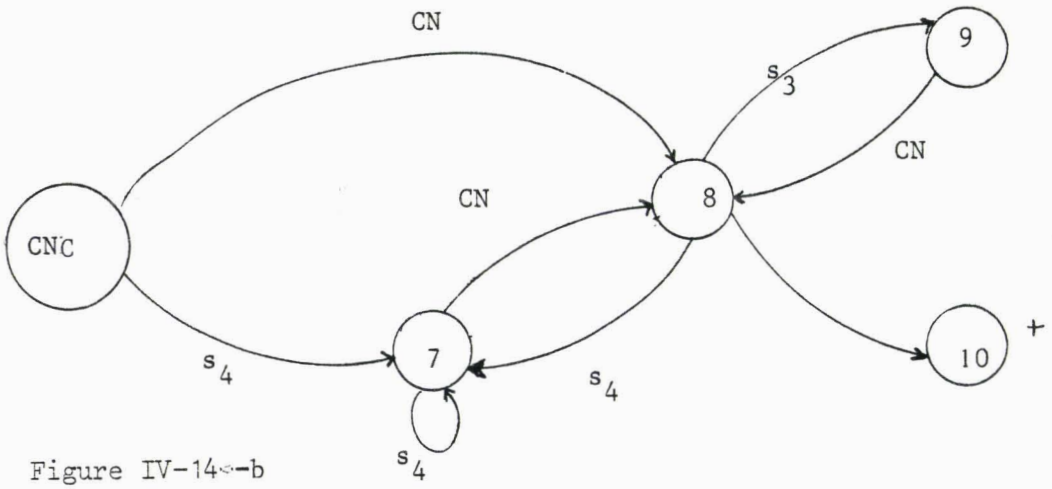


Figure IV-14-b

GRAPHE DE L'AUTOMATE D'ANALYSE DES CHAINES ALPHABETIQUES SIGLES ET PONCTUATIONS

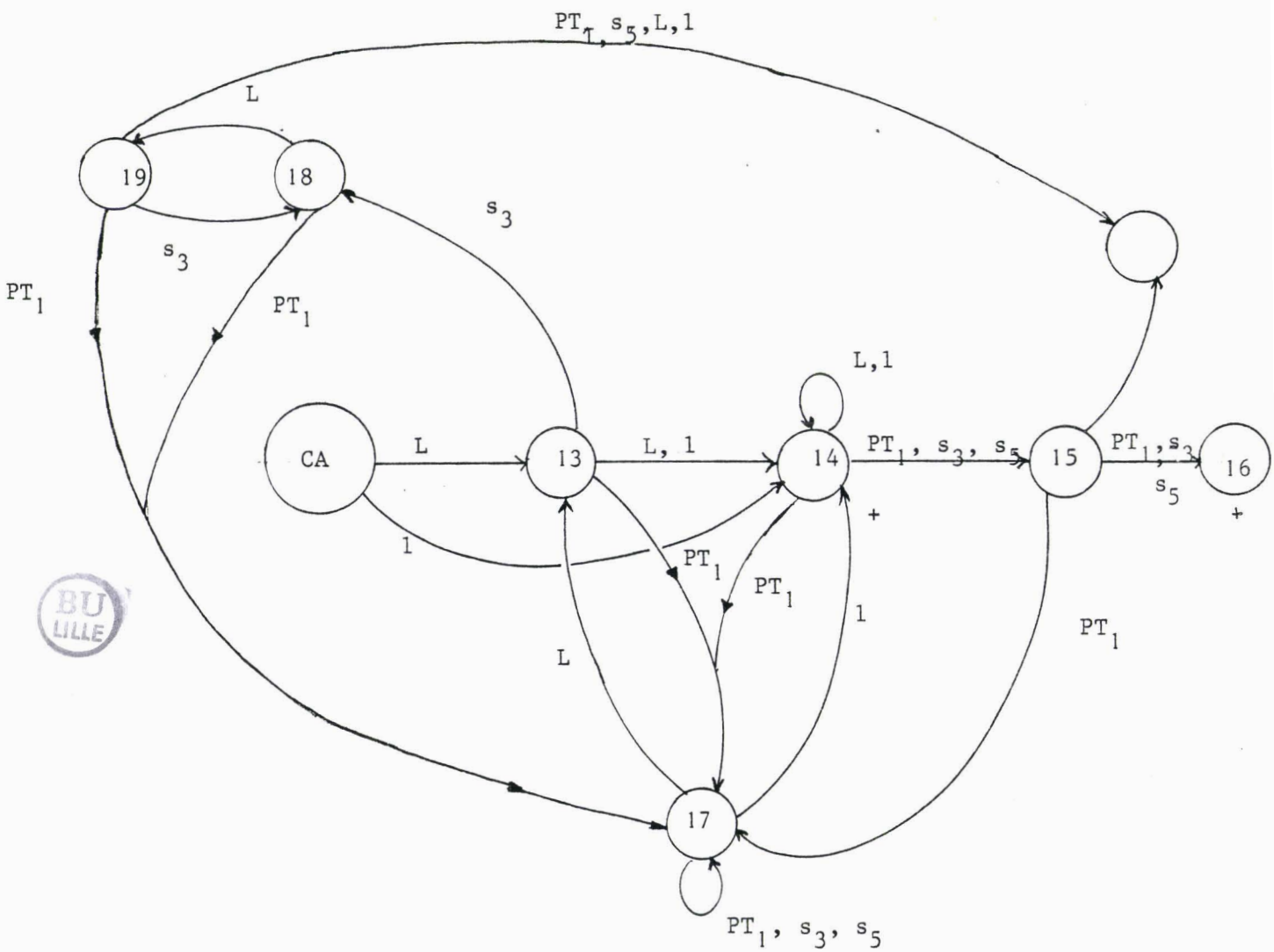


Figure IV-15

GRAPHE DE L'AUTOMATE D'ANALYSE DES ABREVIATIONS PARTICULIERES

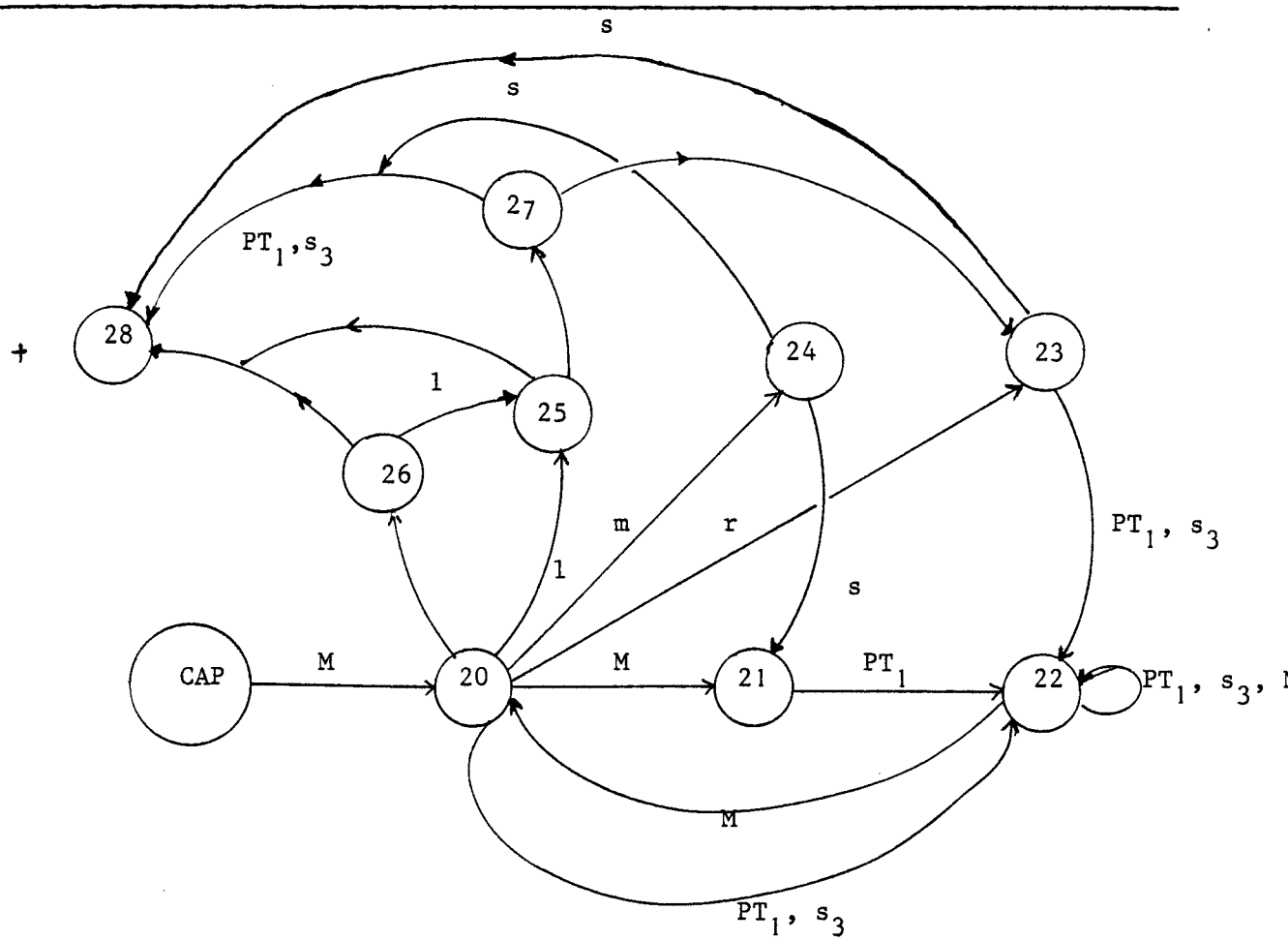


Figure IV-16

Toutes les autres transitions non spécifiées conduisent à l'état de rejet 13

GRAPHE DE L'AUTOMATE D'ANALYSE DES CHAINES PARTICULIERES ET INDIQUEES

SOULIGNEMENT

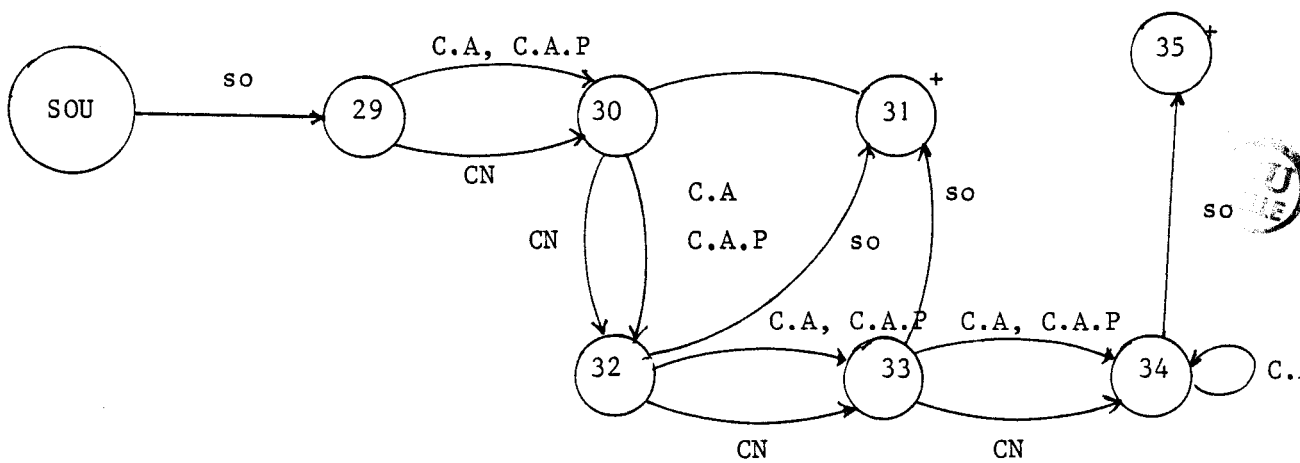
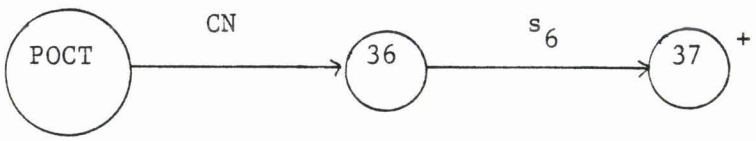
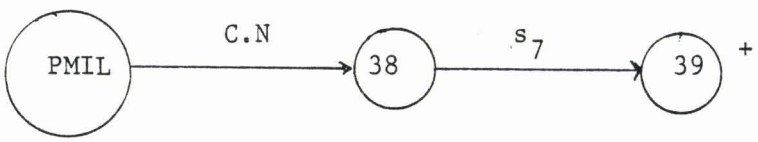


Figure IV-17

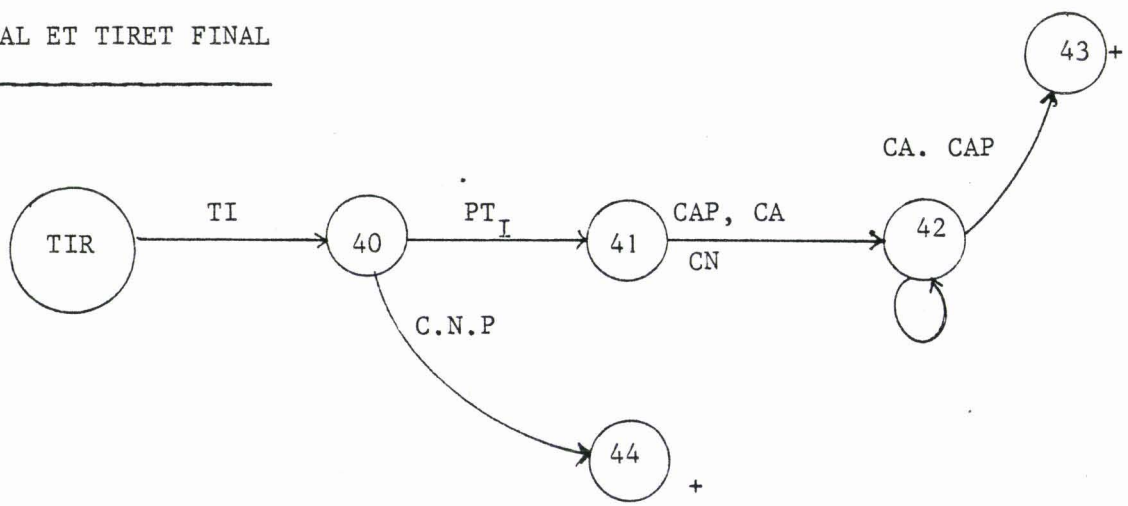
POUR CENT s_6 = %



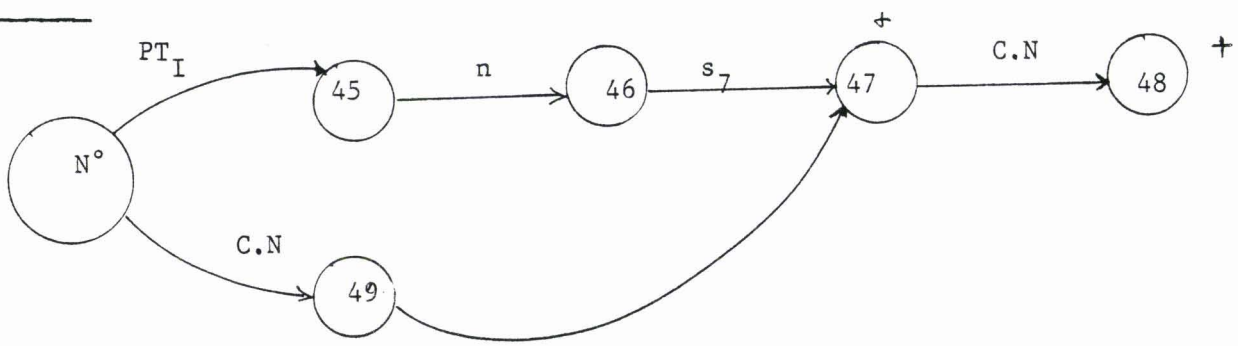
POUR MILLE s_7 = %.



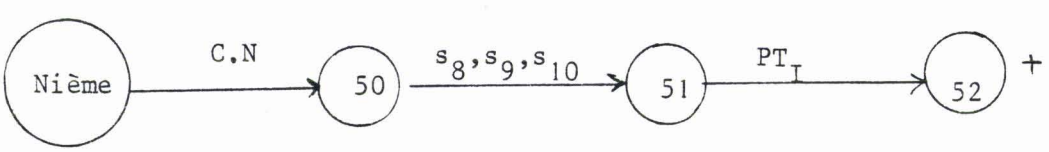
TIRET INITIAL ET TIRET FINAL



NUMERO ET DEGRE



Nième



s_8 = "er" s_9 = "me" s_{10} = "ème"

Figure IV-18

IV.4. Description de la fonction de transcription Braille Abrégé

La transcription en braille Abrégé s'effectue à partir du texte Braille intégral en mode programme. La démarche conceptive de l'analyseur syntaxique réalisant cette fonction est analogue à celle adoptée pour la transcription en Braille intégral.

IV.4.1. Examen des différents cas de conversion de syntaxe noir Braille Abrégé

On distingue deux types de chaînes noires pour la transcription en Braille Abrégé [2] [69] . La chaîne dictionnaire et la chaîne contraction.

Les chaînes dictionnaires appartiennent à un index d'environ 900 mots préabrégés que nous noterons V_1^* Soit T_1, le procédé de transcription en Braille Abrégé. C'est une application qui a tout élément de V_1^* associe un élément de B^*_1 et dont les propriétés sont énumérées ci-après :

Propriété 1 (désinence)

$\delta \in V^*$ avec $\delta = r(\delta)\ s(\delta)$ et $r(\delta) \in V_1^*$ tel que

$T_1(\delta) = T_1(r(\delta))\ T_1(s(\delta))$ si et seulement si

$s(\delta) \in ds_1 = \{e, s, x, es\}$

ds_1 = dictionnaire de désinence
$r(\delta)$= racine de δ
$s(\delta)$= suffixe de δ

Propriété 2 (Association chaîne dictionnaire avec des préfixes)

$\delta \in V^*$, $\delta = P(\delta)\ r(\delta)$ et $r(\delta) \in V_1^*$ tel que

$T_1(\delta) = T_1(P(\delta))\ T_1(r(\delta)$ si et seulement si

a) $P(\delta) \in D_p = \{in, \ldots\}$ = dictionnaire des préfixes

b) δ et $r(\delta)$ garde le même sens sémantique.

Propriété 3 (Association racine chaîne dictionnaire - suffixe)

$\delta, \lambda \in V_1^*$ avec $\delta = r(\delta)\ s(\delta)$ et $\lambda = r(\lambda)\ s(\lambda)$

tel que pour $T_1(\delta) = T_1(r(\delta))\ T_1(s(\delta))$ et $T_1(\delta) = T_1(r(\delta)\ T_1(s(\lambda))$

on a :

1) $T_1(s(\delta)) = T_1(s(\lambda)) \quad \forall\ r(\delta), r(\lambda)$

2) $T_1(r(\delta)) = T_1(r(\lambda)) \quad \forall\ s(\delta), s(\lambda)$

L'ensemble des chaînes contractions est constitué par toutes les chaînes n'appartenant pas à l'index. Ces chaînes sont transcrites après vérification de règles déterministes pour les contractions dites simples tandis que pour les contractions composées et les cas particuliers, une analyse sémantique hors ligne est nécessaire.

IV.4.2. Analyse syntaxique d'une chaîne dictionnaire

Pour optimiser l'algorithme d'analyse syntaxique d'une chaîne dictionnaire, l'index a été structuré en un dictionnaire des préfixes D_p, un dictionnaire de racine D_r, un dictionnaire de suffixe D_s et un dictionnaire de désinence D_{si}.

IV.4.2.1. Structure du dictionnaire des préfixes

Une analyse sémantique des chaînes de V* [2] a permis de regrouper les préfixes vérifiant la propriété 2 en sous ensemble ε_i (figure IV.19) tel que :

- $\varepsilon_i = \cup_j P_{ji}$
- $\cup_i \varepsilon_i = D_p$ et $\cap_i \varepsilon_i \neq \{\phi\}$

Il en résulte que la structure d'un élément du dictionnaire des préfixes est alors celle d'un n-uple représenté par :

$$\left\{ \underbrace{P\ ,\ T_1(P)\ ,}_{\text{Octet du préfixe P et de sa transcription}} \quad \underbrace{\varepsilon_i\ ,\ \varepsilon_k\ ,\ \varepsilon_e\ ,\ \ldots\ ,\ \varepsilon_m}_{\text{Octet représentant les paramètres d'association racine-suffixe}} \right\}$$

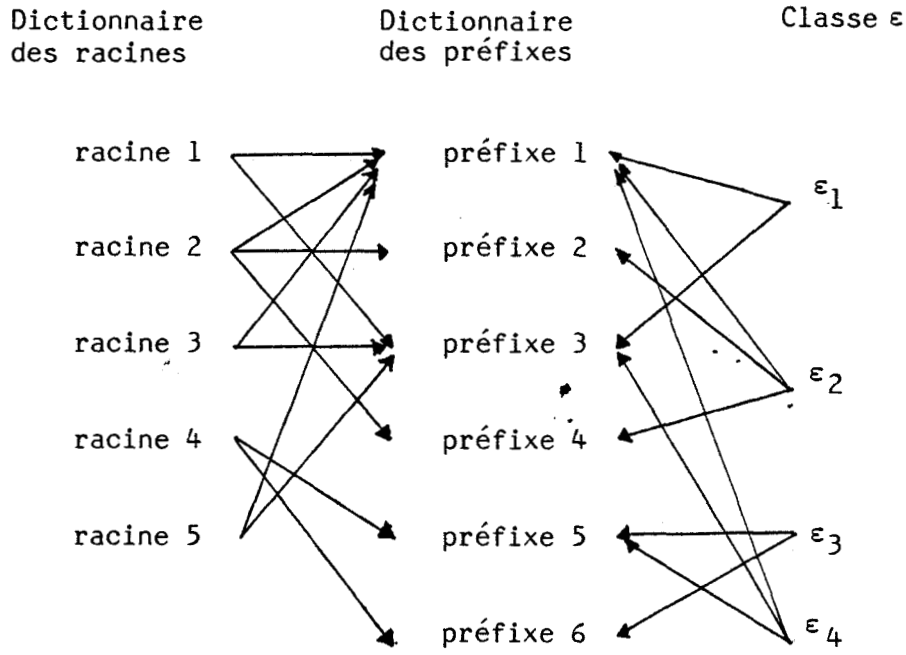


Figure IV.19.: Diagramme d'association des racines et des préfixes en transcription directe.

IV.4.2.2. Structure du dictionnaire des suffixes et des racines

L'invariance à la transcription des racines ou des suffixes de certaines chaînes de l'index (Propriété 3) conduit à l'élimination de leur redondance et par voie de conséquence à la réduction de la taille de l'index. Le tableau (IV.9.) illustre les deux cas de figure présents pour le choix des éléments de D_r et D_s.

Tableau IV.9. : Cas de découpe racine-suffixe en transcription directe.

Type	$\alpha \in V_1^*$	$S(\alpha) \in D_S$	$l(T_1(r(\alpha))$	$l(T_1(S(\alpha)))$	$l(T_1(\alpha))$
I	avantage		3	0	3
	avantage	use	3	2	5
	avantage	ux	3	1	4
	avantage	usement	3	2	5
II	facile		2	0	2
	facil	ité	2	1	3
	facile	ment	2	1	3

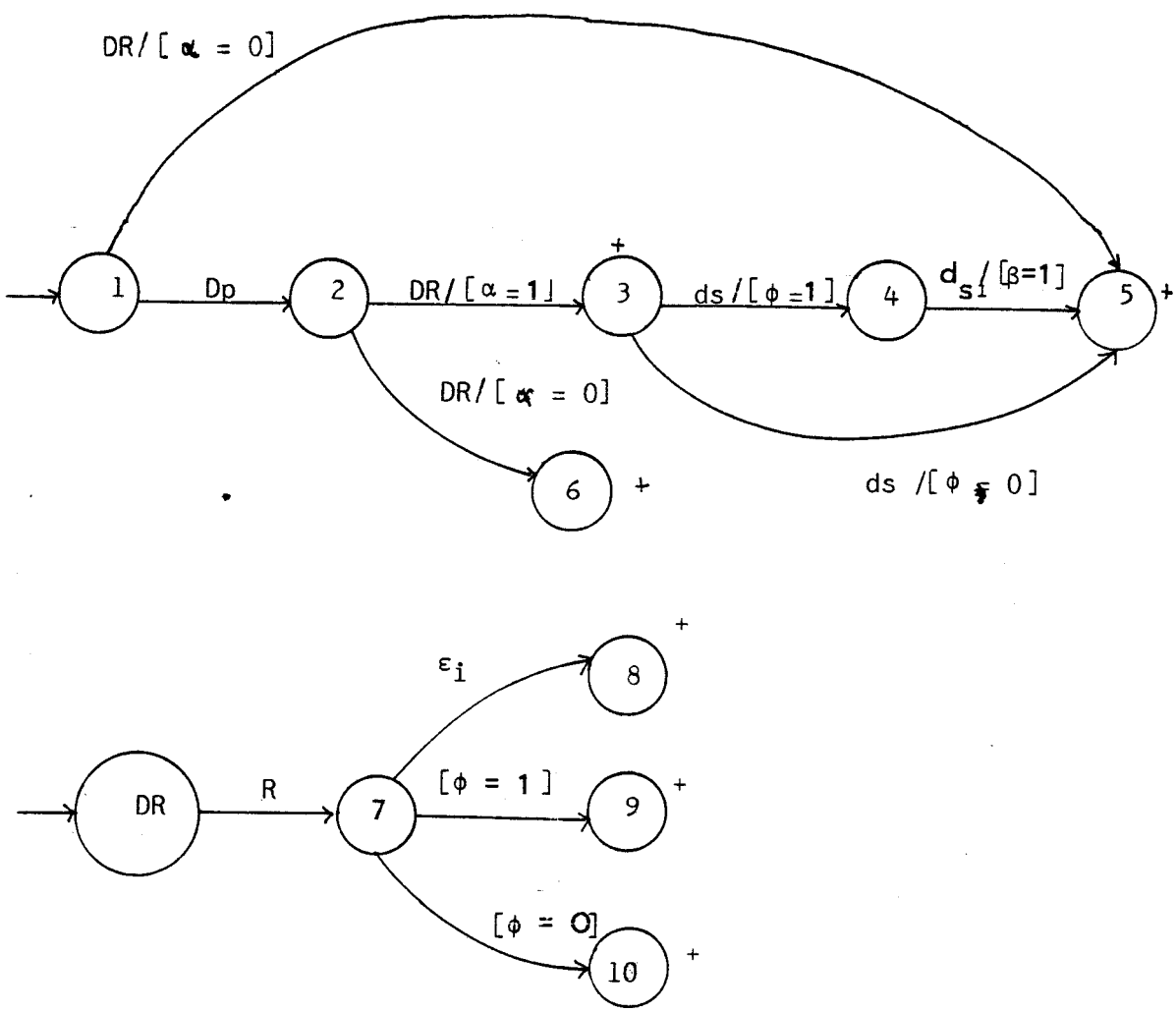
Les éléments du dictionnaire des suffixes sont classés dans un ordre croissant des indices (ordre alphabétique) et pour un indice donné ils sont ordonnés par ordre décroissants des longueurs.

La structure d'un élément de D_r compatible avec ceux de D_p et D_s est la suivante :

$D_r = \{\varepsilon_i , R , \phi , T_1(R)\}$ ou $\phi \in [0,1]$ est une variable prenant la valeur 0 lorsque des racines ne s'associent pas à un suffixe par simple concaténation (Type II).

IV.4.2.3. Réseau ATN pour l'analyse syntaxique d'une chaîne dictionnaire

Le graphe du réseau ATN d'analyse syntaxique d'une chaîne dictionnaire est illustrée par la figure IV.20 .



Avec α = condition d'association racine préfixe

ϕ = condition d'association racine suffixe

β = condition d'association avec une terminaison

Figure IV.20

(1) : début d'analyse d'une chaîne dictionnaire.

(2) : La première chaîne isolée est un préfixe. La reconnaissance du plus long préfixe par scrutation séquentielle de D_p est certaine si $\forall P_i, P_j \in D_p$ et $\lambda, \alpha \in V^*$ tel que $P_i = P(P_i, \alpha)$, $P_j = P(P_j, \alpha)$ et $P_j = P(P_j, P_i)$ (c'est-à-dire que $P_i = P_j \lambda$) avec $l(P_j) < l(P_i)$ on organise D_p de façon à avoir $P_i < P_j$.

(3) : La deuxième sous-chaîne isolée est une racine et la condition d'association racine-préfixe est vérifiée.

(4) : La sous-chaîne restante est un suffixe et la condition d'association racine suffixe est vérifiée. Le procédé de reconnaissance du plus long suffixe est identique à celui des préfixes.

(5) : Sortie chaîne dictionnaire correcte

(6) : La chaîne n'est pas dictionnaire, voir la chaîne contraction

(8) : Reconnaissance du sous-ensemble εi associé à une racine
$\alpha = 1$ si $\varepsilon i \in$ au P_j - uplet, $\alpha = 0$ sinon

(10): $\phi = 1$ condition d'association racine suffixe vérifié

(11): $\phi = 0$ condition d'association racine suffixe non vérifiée

IV.4.3. Analyse syntaxique d'une chaîne contraction

L'échec de l'analyse syntaxique d'une chaîne noire par l'automate précédent, conduit à sa découpe en contraction (Figure IV.21.)

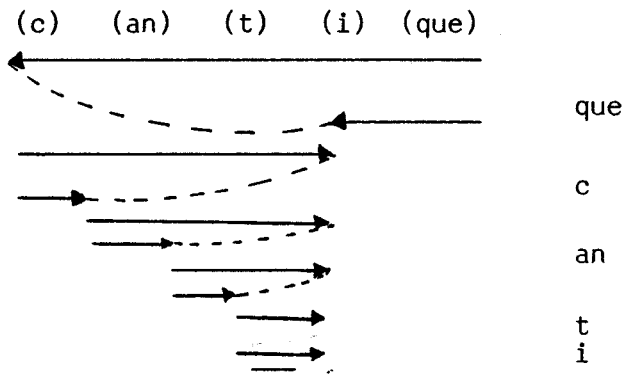


Figure IV.21. : Algorithme de découpe d'un mot en contractions.

Pour réaliser une découpe optimale, ainsi que la vérification de règles particulières de traduction Braille, l'analyse des formes de contraction [2] a permis de distinguer trois cas de figures :

- Les contractions simples (cs) = Elles vérifient certaines règles énumérées dans le tableau (IV.10.)

- Les contractions composées (cc) = Elles ont été créées (Tableau IV.11) pour solutionner certaines particularités de la découpe auxquelles auraient conduites, les règles classiques de contraction.

- Les contractions particulières (CP) = sont des solutions apportées à l'automate de transcription par contraction pour éviter une ambiguïté de lecture du non-voyant (Tableau (IV.12)

Règle	$H[\delta(i,j)] =$	
01	$= 1\ \forall\delta(j+1)$ et $\delta(i-1)$	
02	$= [\delta(j+1) = <\text{sp}>] \cup [\delta(j+1) = s] \cap [\delta(j+2) = <\text{sp}>]$	
03	$= [\delta(j+1) = <\text{v}l>]$	
04	$= [\delta(j+1) = <\text{cn}>]$	
05	$= [\delta(j+1) = <\text{v}l>] \cap [\delta(i-1) = <\text{v}l>]$	
06	$= [\delta(i-1) = <\text{sp}>]$	
07	$= [\delta(i-1) = <\text{sp}>] \cap [\delta(j+1) = <\text{cn}>]$	
08	$= [\delta(i-1) = <\text{sp}>] \cup [\delta(j+1) = <\text{sp}>]$	
09	$= [\delta(j+1) = <\text{cn}>] \cap [\delta(j+1) = <\text{sp}>] \cup [\delta(j+1) = s] \cap [\delta(j+2) = <\text{sp}>]$	
10	$= [\delta(i-1) = <\text{sp}>] \cup [\delta(j+1) = <\text{cn}>] \cup [\delta(j+1) = <\text{sp}>]$	
11	$= [\delta(l-4, l) = <i\text{ent}>] \cap [\delta(1, l-5) = D_{ient}]$ avec D_{ient} = ensemble des mots se terminant par ient avec le son ($j\ \tilde{\alpha}$)	convient
12	$= [\delta(j+1) = <\text{sp}>] \cap [\delta(j+1) \neq <\ ''\ >] \cup [\delta(j+1) = s] \cap [\delta(j+2) = <\text{sp}>] \cup [\delta(j+1) = <\text{cn}>] \cap [\delta(i-1) \neq <\text{sp}>]$	
13	$= [\delta(i-1) = <\text{sp}>] \cap [\delta(j+1) = <\text{v}l>]$	
14	$= [\delta(j+1) = <\text{cn}>] \cup [(\delta(j+1) = s] \cap [\delta(j+2) = <\text{sp}>] \cup [\delta(j+1) = <\text{sp}>] \cap [\delta(j+1) \neq <\ .\ >]$	
15	$= [\delta(j+1) = <\text{cn}>] \cup [\delta(j+1) = s] \cap [\delta(j+2) = <\text{sp}>] \cup [\delta(j+1) = <\text{sp}>] \cap [\delta(j+1) \neq <\ ?\ >]$	
16	$= [\delta(i-1) \neq <\text{sp}>] \cap [\delta(j+1) = <q>]$	logi
17	$= [\delta(i-1) = <\text{sp}>] \cap [\delta(i-1) \neq <\ -\ >] \cup [\delta(j+1) = <\text{cn}>]$	
18	$= [\delta(j+1) \neq <\text{sp}>] \cap [\delta(j+1) = <\text{cn}>]$	
19	$= [\delta(j+1) = <\text{cn}>] \cup [\delta(j+1) = <\text{sp}>] \cap [\delta(j+1) \neq <\ *\ >]$	
20	$= [\delta(j+1) \neq <\text{sp}>] \cap [\delta(j+1) = <\text{cn}>]$	
21	$= [\delta(j+1) = <\text{cn}>] \cup [\delta(j+1) = <\text{sp}>] \cap [\delta(j+1) \neq <\ *\ >]$	

Tableau IV-10 — Formalisation des règles de découpe en contractions.

$\delta(i, j)$	H ($\delta(i, j)$	Caractères Brailles
ain, aim, oin ren, rem, ess	$\delta(j+1) = \langle v \rangle$ $\delta(j+1) = \langle cn \rangle$	$T_1[\delta(i,i+1]\ T_2[\delta(j)]$ $T_2[\delta(i)]\ T_1[\delta(i+1,j)]$
ques, ables...	$\delta(j+1) = \langle sp \rangle$	$T_1[\delta(i,j-1)]\ T_1(\delta(j))$
dîo, prop	$\delta(j+1) = \langle cn \rangle \cup \langle v \rangle$	$T(\delta(i))$

TABLEAU IV.11.

Type de chaîne particulières	Exemples de contraction	Solution
Terminaison (ent) ils convient il convient expédient ils expédient	 (con) (V) (l) (ent) (cons) (U) (en) (t) (ex) (P) (è) (d) (ien) (t) (ex) (P) (é) (d) (i) (ent)	Communication homme-machine
Symboles inférieurs eu, er, ...	(ev) , (er)	Contraction la plus du mot en Braille intégral
Autres cas inouïe		Mot entier en intégral

TABLEAU (IV.12)

Ces différents cas ont conduit à organiser le dictionnaire des contractions en 7 sous-ensembles contenant des contractions de même longueur (cf [2]) et dont les éléments ont la structure suivante :

$$\left\{\delta(i, j),\ \text{if},\ Rg\ (\delta(i, j)),\ T(\delta(i, j))\right\}$$

où if est un indicateur prenant la valeur 0 lorsque T ($\delta(i, j)$ contient au moins un symbole supérieur et 1 dans le cas contraire

La figure (IV.22) représente le graphe de l'automate d'analyse d'une chaîne contraction.

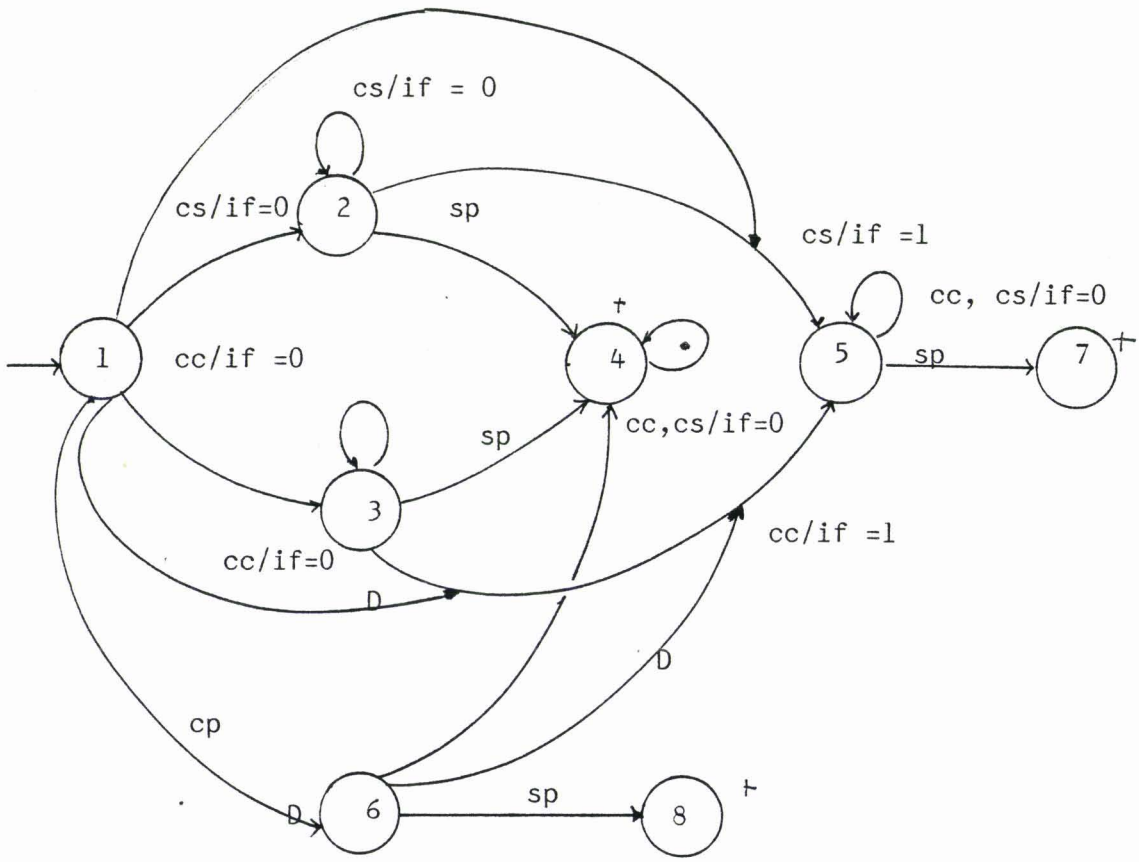


Figure (IV.22)

Signification des états

(1) : Début d'analyse d'une chaîne contraction

(2) : Reconnaissance d'une contraction simple dans un sous-ensemble de Dc dont la traduction comporte au moins un symbole supérieur

(3) : Reconnaissance d'une contraction composée dans D_c et dont la traduction comporte au moins un symbole supérieur

(4) : Fin d'analyse d'une chaîne contraction dont la traduction ne comporte que des symboles inférieurs. Donc transcription de la contraction la plus courte en Braille intégral.

(5) : Reconnaissance d'une contraction dont la traduction ne comporte que des symboles inférieurs

(6) : Traitement des chaînes particulières.

(7) : Fin d'analyse d'une chaîne contraction dont la traduction comporte au moins un symbole supérieur.

(8) : Fin d'analyse d'une chaîne particulière.

IV-4-3- Conclusion

La méthode d'analyse syntaxique présentée dans ce chapitre permet d'une part de modéliser entièrement la fonction de transcription Braille et d'autre part de décrire systématiquement par un réseau d'automates la structure des différents processeurs qui constituent le nouvel éditeur-transcripteur de texte noir conditionné Braille qui a été réalisé.

Cet éditeur-transcripteur, caractéristique d'une nouvelle génération de système, fait partie intégrante du logiciel du LOGIBRAILLE implanté sur micro-ordinateur. Sur ce système sont permises différentes opérations assurant la souplesse du traitement de texte temps réel simultanément dans les deux syntaxes. On peut citer parmi ces opérations, la saisie pleine page, la césure syllabique, le formatage, la correction automatique de certaines règles syntaxiques, la manipulation de fichiers (copie, effacement, sélection, insertion,...).

En outre, de par la modélisation totale de la fonction de transcription en une grammaire d'états finis spécifique à cette application, on peut envisager l'intégration d'un tel procédé dans un système de gestion de base de données textuelles, en prenant soin de déterminer dans le cas de la transcription Braille abrégé, les outils de définition et de manipulation du dictionnaire et de l'ensemble des chaînes contractions.

CONCLUSION

-:-:-:-:-:-:-:-:-:-:-:-:-

En résumé des études présentées dans ce mémoire, trois points essentiels nous paraissent devoir être soulignés.

Le premier point est relatif à la mise en évidence de différents problèmes posés par l'intéraction des textes noirs et Braille , en vue d'une édition pleine page par saisie dactylographique de texte Braille par un voyant ne connaissant pas cette syntaxe.

L'analyse des particularités des deux syntaxes ainsi que de l'influence des traitements de texte permet de définir pour s'affranchir de ces contraintes, et dans un objectif de coût minimal, un mode d'édition basé sur le conditionnement du format source par le format cible.

Dans un second temps, nous avons étudié un certain nombre de modèles d'analyse des textes libres susceptibles de fournir à l'utilisateur une souplesse de traitement envisageable dans les machines présentes et futures.
On a pu ainsi remarqué que devant l'insuffisance d'autres modèles, l'analyse syntaxique reste le traitement de base de plus haut niveau adaptable à une analyse des textes libres. Il présente cependant certains inconvénients liés à sa complexité : le modèle hors-contexte (donc non linéaire) et la présence de nombreuses ambiguités impliquent une combinatoire incompatible avec un temps de réponse rapide et une fonction de coût minimale.

Enfin dans une dernière étape, nous avons défini un prétraitement syntasique dans l'acquisition des chaînes noires, pour la levée des ambiguités syntaxiques noir et Braille.
Ce prétraitement, associé au modèle d'analyse syntaxique que sont les réseaux de transition, permet d'une part de modéliser entièrement la fonction de transcription et d'autre part de décrire les automates constitutifs de la fonction d'édition qui a été réalisée.

.../...

Un tel procédé permet d'éditer et de manipuler les textes Braille avec la souplesse des traitements de texte sans contraintes excessives.

Dans une optique d'évolution future, il reste à déterminer les outils de définition et de manipulation des chaînes dictionnaires et contractions. On pourrait de la sorte envisager l'intégration d'un tel procédé dans un système de base de données textuelles et diminuer de ce fait la spécificité de l'édition de texte Braille à l'aide d'un système informatique.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

BIBLIOGRAPHIE SUR LE BRAILLE

[1] ASSOCIATION VALENTIN HAUY
"Abrégé orthographique étendu"
Edition 1964

[2] A. MANDAR
"Etude théorique et réalisation d'un système autonome intéractif de transcription Braille"
Thèse de Doctorat-Ingénieur, Lille 1982

[3] J.P. DUBUS, A. MANDAR, M. MORTREUX
"Modélisation des règles de transcription Braille : Application à l'étude de processeurs de communication individuelle entre voyant et aveugle"
Annales des Télécommunications, Tome 38, n° 3-4, Mars-Avril 1984

[4] R. RUBINSTEIN and J. FELDMAN
"A controller for a Braille Terminal"
C.A.C.M.

[5] GARY B ANDERSON AND DAVID W. ROGERS
"An inexpensive Braille Terminal device"
C.A.C.M., Vol. 11, n° 6, Juin 1968

[6] WALTER J. WELLER, VIRGINIA C. KLEMA
"A rapid Braille Translitération technique for certain IBM machines"
C.A.C.M., Vol. 8, n° 2, Février 1965

[7] M. TRUQUET
"Transcription en Braille intégral ou abrégé"
Thèse d'Etat, 30 novembre 1979

[8] KARI LARSEN
Handitext. An interactive reacting system for the blind
Colloque international "Informatique et Braille", 23-26 septembre 1981

[9] J. MEEKEL, J. FRONTIN, D. LEVY, M. TRUQUET
"Transcription du Braille abrégé français"
Colloque international "Informatique et Braille", 23-26 septembre 1981

[10] J. KAPLIN
"Methods of inputing text for Braille translation at the RNIB"
Colloque international "Informatique et Braille", 23-26 septembre 1981

BIBLIOGRAPHIE CONCERNANT L'EDITION DE TEXTE

[11] R.S. SCOWEN
"A survey of some texte editors"
Software-Practice and Experience, Vol,11, pp. 883-906, 1981

[12] T.A. CARGILL
"Full-sreen editing in a hostile environment"
Software-Practice and experience, vol. 11, pp. 975-981, 1981

.../...

.../...

[13] M.A. MACLEAN
"Chef : a versatible portable text editor"
Software-Practice and Experience, Vol. 11, pp. 467-477, 1981

[14] J. ALAN HUNTER, NIGEL P. HALL
"A network screen editor implementation"
Software-Practice and Experience, Vol. 12, pp. 843-856, 1982

[15] CHRISTOPHER W. FRAZER
"A programmable text editor"
Software-Practice and experience, Vol. 12, pp. 241-250, 1981

[16] J.C. BOUSSARD, J. COHEN, Ph. JORRAND
"Etude et réalisation d'un langage d'édition"
Note technique I.M.A.G.

[17] J.C. VERGES-ESCUIN, J.P. VERJUS
"Reconnaissance automatique des structures des textes en vue de l'édition"
RAIRO, Oct. 1973

[18] O. STRÖMFORS, L. JONES JÖ
"The implementation and experiences of a structured-Oriented text editor"
Proc. of the ACM Sigplan Sigoa symposium on texte manipulation,Portland (Ore), pp. 22-27, juin 1981

[19] C.F. GOLLDFARD
"A generalized approach to document markup"
Proc. of the ACM Sigplan Sigoa symposium on text manipulation, pp. 68-73, Juin 1981

[20] TODD ALLEN, ROBERT NIX, ALAN PERLIS
"Pen : A hierarchical document editor"
Proc. of the ACM Sigplan Sigoa symposium on text manipulation, pp. 74-81, Juin 1981

[21] DONALD D. CHAMBERLIN, JAMES O. KING
"Janus : an interactive system for document composition"
Proc. of the ACM Sigplan Sigoa symposium on text manipulation, pp. 82-91, Juin 1981

[22] KOWARSKI Irène
"Les bases de données textuelles : étude du concept de document et application à deux réalisations"
Thèse 3è cycle, Grenoble, juillet 1983

[23] ALLEN B. TUCKER, Jr
"Text processing algorithms languages, and applications"
ACADEMIC PRESSING

.../...

BIBLIOGRAPHIE SUR LES METHODES D'ANALYSES DES TEXTES LIBRES

[24] GREGORY F. JOHNSON, CHARLES N. FISCHER
"Non-syntatic attribute flow in language based editors"

[25] THOMAS REPS
"Optimal-time Incremental semantic analysis for syntax-directed editros"
C.A.C.M, Vol. 6, pp. 169-176, 1982

[26] J.P. DUBUS, P. ONANGA, M. BENJELLOUN
"Application de la théorie des automates à la réalisation d'un éditeur de textes Braille sur micro-ordinateur"
En attente de publication dans ITBM

[27] J. COURTIN
"Algorithmes pour le traitement interactif des langues naturelles"
Thèse d'état en mathématiques, Grenoble, 1977

[28] K. SUGBARA
"Range-data analysis guided by a junction dictionnary"
Artificial intellignece, Vol. 12, n° 1, Mai 1979

[29] BARABRA J. GROSZ
"Natural language processing"
Artificial Intelligence, Vol. 19, n° 2, Oct. 1982

[30] L. MICLET
"Méthodes structurelles pour la reconnaissance des formes"
Edition Eyrolles et CNET-ENST, 1984

[31] ADNAN ARIN, GERALD MASINI, J.M. MATON
"Un système de compréhension et de traduction automatique de l'arabe manuscrit avec sortie vocale en français"
4è Congrès R.F. et I.A., Paris, Janvier 1984

[32] R.J. BOBROW, B.L. WEBBER
"Knowledge representation for syntatic/semantic processing"
AAAI-80, Stanford, pp. 316-332, 1980

[33] J.K. BAKER
"The dragon system : an overview"
I.E.E.E. T.A.S.S.P., Vol. 23, pp. 24-29, 1975

[34] M. MINSKY
"A framework for representing knowledge"
The psychology of computer vision, p. Winston ed., Mc Graw-Hill, New-York, 1975

[35] W.A. WOODS
"Transition network grammars for natural language processing"
C.A.C.M., Vol. 13, n° 10, pp. 591-602, Octobre 1970

[36] FATHI DELIBI
"Analyse syntaxico-semantique fondée sur une acquisition automatique de relations lexicales-sémantiques"
Thèse d'état, Paris XI, Janvier 1982

[37] WINOGRAD
"Understanding naturel language"
Academic press (1972)

[38] Brigitte GRAU
"Analyse et représentation d'un texte d'après le thème du discours"
4è congrès R.F. et I.A., Paris, Janvier 1984

[39] Anne DALADIER
"Traitement de coréférences dans une représentation applicative des textes"
4è Congrès R.F. et I.A., Paris, Janvier 1984

[40] Célestin SEDOGBO
"Evaluation comparative de méthodes d'analyse syntaxique partielle"
Thèse 3ème cycle, Université de Paris-Sud, Mars 1983

[41] TSEITING G.S.
"Algorithme d'analyse syntaxique simplifiée"
Pb. cyb., n° 24, pp. 227-242, 1971

[42] VAUQUOIS B.
"La traduction automatique à Grenoble"
Documents de linguistique quantitative n° 24, Dunod

[43] R.S. BOYER, J.S. MOORE
"A fast string cherching algorithm"
C.A.C.M., Vol. 20, n° 10, Oct. 1977

[44] R.A. WAGNER, M.J. FISCHER
"The string to string correction problem"
J.A.C.M., Vol. 21, n° 1, pp. 168-173, janvier 1974

[45] M.A. HARRISON
"Introduction to formal language theory"
Addison-Wisley, 1978

[46] W.A. WOODS
"Transition networks grammars for natural language analysis"
C.A.C.M., Vol. 13, n° 10, Oct. 1970

[47] K.S.FU, T.L. BOOTH
"Grammatical Inference : Introduction and survey"
IEEE Tr S.M. , Vol S.M.C.5 n° 1, janvier 1975 et Vol. S.M.C.5 n° 7, juil.85

[48] A. PYSTER, H.W. BUTTELMANN
"Semantic-Syntax directed transliteration"
Information and control, Vol. 36, n° 3, Mars 1978

[49] D.E. KNUTH
"Semantics of context-free languages"
Mathematical system theory, Vol. 2, n° 2, pp. 127-146, 1968

[50] D.E. KNUTH
"Semantics of context free languages : correction"
Math. Syst. Theor., Vol. 5, n° 1, pp. 95-96

[51] D.E. KNUTH
"The art of computer programming"
Vol. 1 et 3, Addison Wesley, 1973

[52] D.E. KNUTH
"On the translation of languages from left to right"
Information and control, Vol. 8, n° 6, pp. 607-639, 1965

[53] HOPCROFT J.E., ULLMANN J.D.
"Formal languages and their relation to automata"
Addison Wesley, 1969

[54] H. GALLAIRE
"Technique de compilation"
CEPADUES Edition

[55] M. GROSS, A. LENTIN
"Notion sur les grammaires formelles"
Gauthiers-Villars, Paris

[56] M. GROSS
"Méthode en syntaxe"
Heimann, 1975

[57] R. MAHL, J.C. BROUSSARD
"Algorithmique et structures de données : Programmation avancée"
Eyrolles, 1983

[58] M. GONDRAY, M. MINOUX
"Graphes et Algorithmes"
Eyrolles, 1979

[59] J.C. SIMON
"Représentation et traitement de certaines structures d'information"
Institut de programmation, Université de Paris VI, 1974

[60] R. FAURE, B. LEMAIRE
"Mathematiques pour l'informatique.Tome 1 : Ensemble, relations, graphes, monoïdes, automates"
Gauthier-Villars, Collection "Programmation", 1973

[61] G. VEILLON
"Modèles et algorithme pour la traduction automatique"
Thèse d'état, Grenoble 1970

[62] J.M. CHOU, K.S. FU
"Inférence for transition network grammars"
Computer Language Analysis, C.A.C.M. Vol. 4, n°2, 1979

[63] W.R. LALONDE
"Regular right part grammars and their parsers"
C.A.C.M. Vol. 20,n°10, octobre 1977

[64] T. PAVLIDIS
"Structural pattern recognition"
Spinger Verlag,1977

[65] R.C. GONZALES,M.G. THOMASON
"Syntactic pattern recognition: an introduction"
Addison-Wesley,1978, Chapitre 3 et 4

[66] K.S. FU
"Syntactic pattern recognition and application"
Prentice-Hall 1982

[67] A.V. AHO? J.E. HOPCROFT? J.P. ULLMAN
"The design and analysis of computer algorithms"
Addison-Wesley, octobre 1975

[68] A. MERLE
"Un analyseur pré-syntaxique pour la levée des ambiguïtés dans des documents écrits en langue naturelle: application à l'indexation automatique"
Thèse de Docteur-Ingénieur,Grenoble 1982

[69] M. MORTREUX
"Etude et conception d'un lecteur optique de relief Braille avec transcription temps réel des chaînes Braille en noir"
Thèse 3ème cycle ,Lille 1 1985

[70] J.L. KRAHE
"Etude de méthodees de reconnaissance automatique de caractères imprimés: application à un système de lecture pour aveugles"
Thèse de doctorat d'université, PARIS 1979

[71] Y. WILKS
"Knowledge structures and language boundaires"
IJCAI MIT 1977

[72] Y. WILKS
"An intelligent analyser and understander of english"
ACM, Vol. 18, May 1975

[73] R.C. SHANK
"Identification of conceptualisation underlying natural language. Computer models of thought and language"
Freeman Suns Francisco,1973

[74] R.C. SHANK, K.M. COLBY
"Senpts, plans and knowledge"
IJCA, 1975

[75] J. CONTER. M. VALOBRA. A. BRUEL
"Un dispositif électroniquede lecture de texte pour aveugles:DELTA".
Bulletin INRIA,novembre 1981, pp 67-70

[76] J.P. DUBUS, A. MANDAR
"Demande adoptée pour établir la faisabilité d'un traducteur autonome temps réel Braille abrégé, à l'aide d'un microprocesseur.
Description détaillée d'une partie des solutions adoptées".
I.T.B.M., vol. 1 n° 2, 1980.

[77] J.P. DUBUS, A. MANDAR
"Méthode d'élaboration des algorithmes de traduction en Braille abrégé par dactylographie d'un texte.
Optimisation de la taille mémoire et du temps de traduction pour implantation sur microprocesseur".
I.T.B.M., vol. 1, n° 4, 1980.

[78] J.P. DUBUS, M. MORTREUX, M. DURAND
"Méthode d'élaboration des algorithmes de transcription automatique Braille abrégé-Noir en vue de leur implantation dans une architecture de transcripteur autonome.
I.T.B.M., vol. 3, n° 4, 1982.

[79] J.P. DUBUS, M. MORTREUX, P. VINCKE, C. SION
"Etude et réalisation d'un lecteur optique de relief Braille avec transcription automatique en texte "noir"".
Onde électrique, Mai - Juin 1985, vol. 65, n°3.

[80] J.P. DUBUS, F. WATTRELOT
"Etude de la visualisation de textes alphanumériques sur écran de télévision à format variable à usage des amblyopes".
I.T.B.M., vol. 1, n° 6, 1980.

[81] J.P. DUBUS, F. WATTRELOT
"A Study of the visualisation of alphanumérical texts on a t;v; screen for the use of the partially sighted persons".
Microprocessing and Microprogramming, 9 (1982), pp 133-141.

RESUME

Ce travail présenté dans le cadre de l'aide à la communication voyant-aveugle porte sur l'étude et la mise au point d'un éditeur-transcripteur Braille.
Dans un premier chapitre, nous donnons un aperçu des systèmes existants et définissons compte tenu des particularités de la syntaxe Braillle et des contraintes dues au traitement de texte, une approche à coût minimum de l'édition de texte Braille par un voyant non spécialiste de cette syntaxe.
Nous examinons dans le second chapitre le problème de l'analyse d'un texte libre en vue de sa transcription. Différents modèles y sont exposés et discutés.

L'approche syntaxique, en raison de son caractère faiblement combinatoire pour notre problème et de sa meilleure adaptabilité aux contraintes de coût minimum, fait l'objet du troisième chapitre.
Enfin, le quatrième chapitre est consacré à la description d'un modèle formel des règles de transcription d'un texte noir en Braille intégral et Braille abrégé Ce modèle, associé à l'outil d'analyse syntaxique que sont les réseaux de transition et à une structure de données ad hoc permet d'obtenir des algorithmes performants implementésdans le nouvel éditeur-transcripteur.

MOTS CLES : Automate d'étatsfinis - grammaires formelles - Analyse syntaxique - Réseau de transition - Braille intégral - Braille abrégé - Traitement de texte.