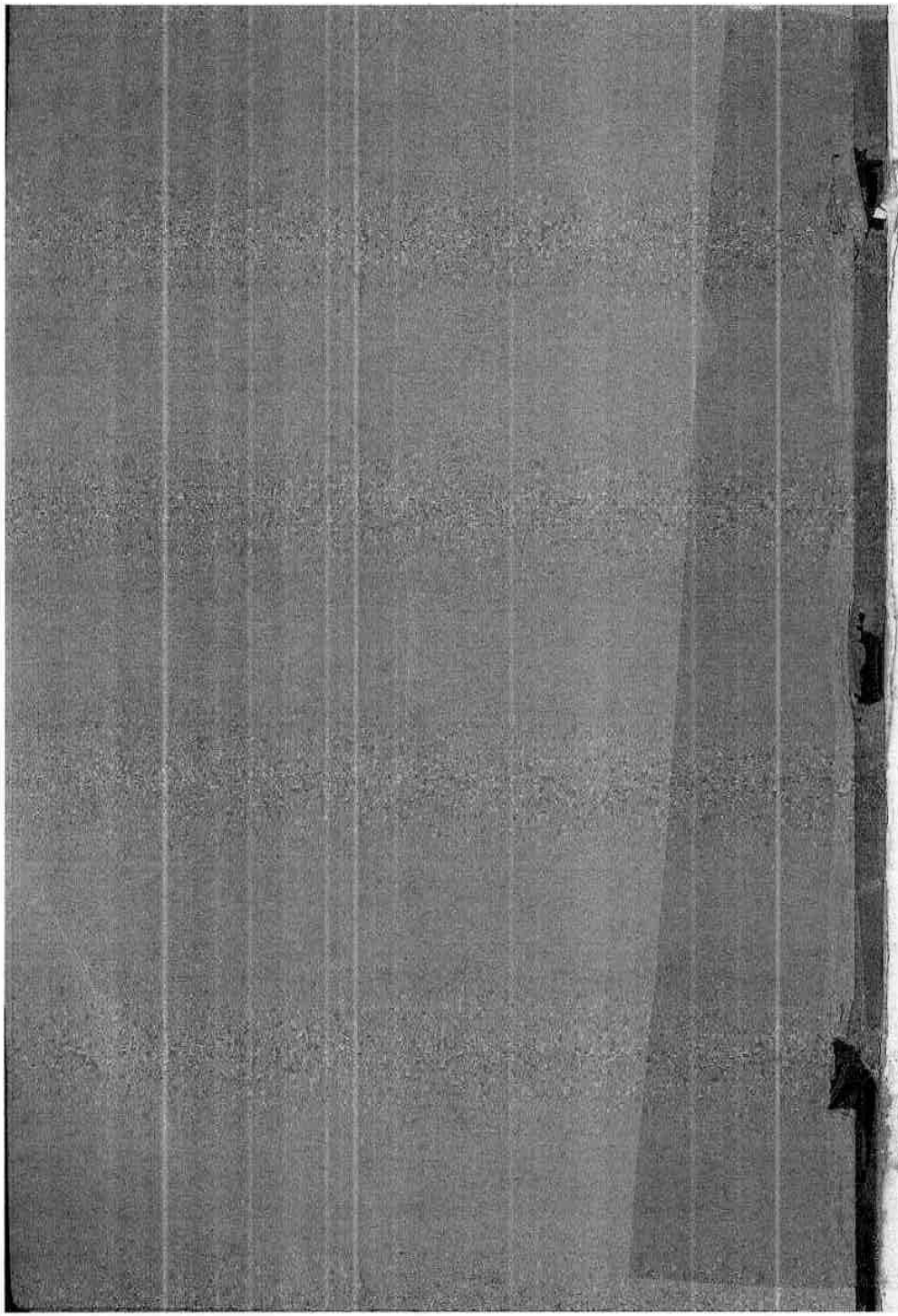
CENTRE DE RECHERCHE EN INFORMATIQUE DE NANCY

chateau du montet
54500 vandœuvre les nancy



GERALD MASINI

Réalisation d'un système de reconnaissance structurelle et d'interprétation de dessins

THESE 78-T-063

Université de Nancy I
U.E.R. de Mathématiques

REALISATION D'UN SYSTEME DE RECONNAISSANCE STRUCTURELLE ET D'INTERPRETATION DE DESSINS

THESE

présentée pour l'obtention du
doctorat de specialité
de mathématiques appliquées (informatique)

par GERALD MASINI
soutenue le 6 octobre 1978

JURY : président M. GRIFFITHS

examinateurs G. COURTIEUX
J.C. DERNIAME
J.P. HATON
R. MOHR
G. STAMON

Ce travail n'a été rendu possible qu'avec l'aide de l'I.R.I.A. par l'intermédiaire du contrat SESORI 76-039.

Je tiens à remercier ici Monsieur le Professeur M. Griffiths, directeur de l'Institut Universitaire de Calcul Automatique, de l'honneur qu'il m'a fait en acceptant la présidence de ce Jury.

Je tiens à assurer de ma reconnaissance Monsieur J.P. Haton, Maitre de Conférences à l'Université de Nancy I, pour l'intérêt manifesté tout au long de mon travail et pour la gentillesse avec laquelle il m'a prodigué ses encouragements. Je le remercie également d'avoir bien voulu faire partie de ce Jury.

Je tiens à exprimer ma gratitude à Monsieur R. Mohr, Maitre-Assistant à l'Université de Nancy I, pour m'avoir proposé ce sujet, pour sa patience et ses précieux conseils qui m'ont permis de le mener à bien, et pour sa prèsence dans ce Jury.

J'adresse aussi tous mes remerciements à Monsieur G. Courtieux, expert au SESORI, à Monsieur J.C. Derniame, Maitre de Conférences à l'Université de Nancy I, et à Monsieur le Professeur G. Stamon de l'Université de Belfort pour l'intérêt qu'ils ont porté à ce travail en acceptant de siéger à ce Jury.

Je profite de l'occasion qui m'est offerte pour remercier tous mes camarades du Laboratoire d'Informatique de l'Université de Nancy I pour leur gentillesse et en particulier A. Belaid qui a bien voulu me consacrer une partie de son temps.

Enfin je remercie Monsieur Debert qui a mis à ma disposition les moyens de l'I.R.E.M. pour mettre en page et tirer ce manuscrit.

SOMMAIRE

INTRODUCTION

Une utilisation universelle et intensive de l'informatique ne saurait être réalisée sans simplification du dialogue entre l'homme et l'ordinateur. C'est pourquoi la majorité des efforts récents dans ce domaine portent sur la mise en oeuvre de moyens de communication directs comme la parole, l'écriture ou le dessin. Si les deux premiers ont toujours été privilégiés, le développement actuel de l'audio-visuel met en évidence l'importance du dernier. En effet, une image constitue une information riche et précise qu'il est bien souvent difficile de traduire avec autant de concision.

C'est cette complexité même qui rend malaisée la mise au point d'outils appropriés. S'il existe déjà des logiciels capables de générer des schémas - notamment en conception assistée par ordinateur - les techniques actuelles de compréhension de tracés restent assez limitées. La part de l'intervention humaine y est encore très importante, voire exclusive.

En attendant la réalisation de l'outil idéal, nous proposons une méthode de reconnaissance et d'interprétation de dessins basée sur un analyseur syntaxique. Notre système permet le traitement d'une classe de figures au choix à partir de sa seule description. Ce sont les informations extraites de celle-ci qui assurent le fonctionnement de la reconnaissance.

Une figure étant considérée comme un ensemble de sous-figures, le système assemble un composant simple et facile à isoler avec ses voisins immédiats pour former une structure plus complexe. En se guidant sur la description, il procède ainsi de proche en proche jusqu'à construire la figure originelle. La suite des choix effectués au fur et à mesure par le système enrichit la connaissance de son univers d'évolution et elle sera efficacement utilisée lors des retour-arrière provoqués par de mauvaises décisions. En fin de processus, il est produit une description structurée très précise ouverte sur toute interprétation a posteriori.

Etant donné que l'acquisition du dessin peut introduire des parasites, nous avons prévu de pouvoir analyser des figures légèrement déformées ou incomplètes.

Le système fonctionne avec un minimum d'heuristiques qui ne servent qu'à résoudre les particularités greffées sur le modèle théorique(correction des images imparfaites) ou à combler ses défauts de jeunesse (traitement des suites récursives).

Le travail présenté ici est essentiellement orienté vers la réalisation pratique de l'analyseur syntaxique de dessins:

Le chapitre I expose et situe l'approche descriptive adoptée parmi les diverses tendances proposées jusqu'à ce jour. Il se termine par une critique et une justification du modèle retenu.

La stratégie générale et son fondement théorique sont expliqués dans le second chapitre. Il contient un algorithme d'analyse syntaxique combinant les deux techniques classiques ascendante et descendante, et s'adaptant à toute forme qui dépend d'un modèle syntaxique.

La troisième partie décrit l'analyseur de dessins adapté de cet algorith Sont détaillés le niveau syntaxique, c'est-à-dire l'interprétation et l'utilisation des informations extraites du système de description, et le niveau opèratoire, c'est-à-dire la manipulation et l'assemblage des composants réels du dessin. Les problèmes de saisie du tracé et de traitement de la description sont également abordés.

La réalisation effective d'un système simplifié et les résultats de son expérimentation font l'objet du chapitre IV. On y trouve la description de la classe de dessins soumise à l'analyse, les temps de reconnaissance obtenus ainsi que les motivations des restrictions apportées en cours de programmation

Enfin la dernière partie se veut une tentative de bilan d'après les premiers résultats obtenus et les observations effectuées pendant l'élaboratio du système. Une rapide mise au point sur son avenir termine cette étude.

CHAPITRE I

DESCRIPTION D'UNE FORME

En thèorie des langages, une phrase est un objet linèaire: elle est le résultat de la juxtaposition de groupes de mots (sujets, verbes, complèments...) constituant des sous-phrases. De la même façon, une forme quelconque peut être considérée comme l'assemblage de diffèrentes sous-formes à l'aide d'opèrations plus élaborées que la concatènation simple, un système approprié de notations permettant d'en opèrer la description.

Jusqu'à prèsent, en l'absence d'un système universel, ils sont tous plus ou moins adaptés à la nature de la forme traitée (image, dessin, parole) et au procédé de reconnaissance choisi (analyse syntaxique, méthode de tests, méthode statistique...). Nous allons montrer quelques solutions adoptées dans le domaine qui nous intèresse plus particulièrement: les dessins.

1-1 Etude bibliographique

Un procédé très connu est celui de LEDLEY (27)(37): pour décrire des chromosomes, il utilise des composants de base - ou primitives - répartis en classes (A: courbe convexe, B: ligne droite, C: courbe concave, D: encoche) qui sont réunis bout à bout comme dans l'addition vectorielle. Chaque chromosome est ainsi représenté par une chaine codée définie à une permutation

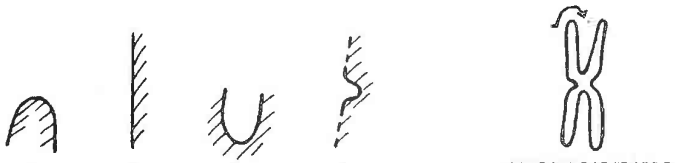
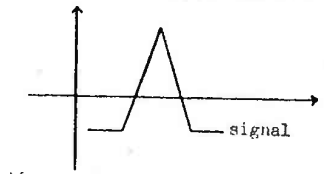


figure 1.1

circulaire près (figure 1.1). Cependant, il n'y a pas unicité: d'une part, le contour d'un même chromosome peut être décomposé en constituants élémentaires de plusieurs façons diffèrentes, d'autre part, à une même décomposition correspond plusieurs objets dissemblables. Une fois l'analyse et le classement d'un chromosome terminés, il faut avoir recours à des traitements numèriques extèrieurs pour éliminer les mauvais cas.

HOROWITZ (38) utilise un principe semblable sur des electro-cardiogramme
Le signal est découpé en segments de droite caractèrisés par leur pente



décomposition:
pente = "0/\0"
position = "-xx-"

figure 1.2

(symbolisée par "/" si elle est positiv
"\" si elle est négative, "0" si elle e
nulle) et par leur position par rapport
à l'axe de réfèrence ("-" pour au-dess
"*" pour la coincidence, "+" pour au-de
"x" pour l'intersection etc...). La description est ainsi ramenée à deux chaines de symboles donnant l'allure et la position de la courbe (figure 1.2). L'auteur leur applique ensuite une anal syntaxique à partir d'une grammaire à contexte libre pour détecter les pics du signal. Ce modèle, en plus du précéd permet la prise en compte des relations positionnelles. Néanmoins, la techniq est trop limitée pour des figures complexes.

On retrouve également ce genre de procédé - ramener un objet à deux dimensions à une chaine linèaire - parmi les nombreuses méthodes de reconnais de l'écriture manuscrite, qui peut être considérée comme un dessin d'un genre particulier. Ainsi BERTHOD (11) puis BELAID (12) décomposent chaque lettre en

T-R-T-M-T T-L-T-L-T

figure 1.3

segments de droite (T), arcs positifs (P arcs négatifs (M) avec des séparateurs qui sont le lever de crayon (L) et le rebroussement (R). Le codage de chaque caractère est déterminé par apprentissag (figure 1.3). Pendant le processus de reconnaissance, des tests numèriques permettent de lever les ambiguités lorsq plusieurs caractères possèdent un code semblable. TOU et GONZALES (39) utilisent des primitives simples (segments de de droite et arcs de cercle orientés) mais introduisent comme HOROWITZ un système de positions. La surface occupée par un caractère est découpée en huit zones(figure 1.4). Le codage exprime la prèsence des primitives dans les diffè- rentes régions, rendant compte de la structure topographique du caractère. Sa grande richesse permet d'avoir

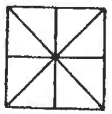


figure 1.4

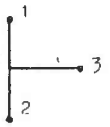
recours moins souvent à des tests permettant de lever les ambiguités.

Parmi les premiers travaux intèressants, il faut citer NARASHIMAN (23). Chaque forme élémentaire F est définie à une translation près et possède des pôles numérotés (figure 1.5). Si F a p pôles on la note F(1,2,...p). La composition des figures procède par superposition de pôles: F concaténée à F' est noté F.F' suivi de la liste des pôles superposés et des pôles conservés

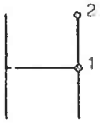
V(1,2,3)

H(1,2)

figure 1.5



F(1,2,3)=V.H(2,1:1,3;2)

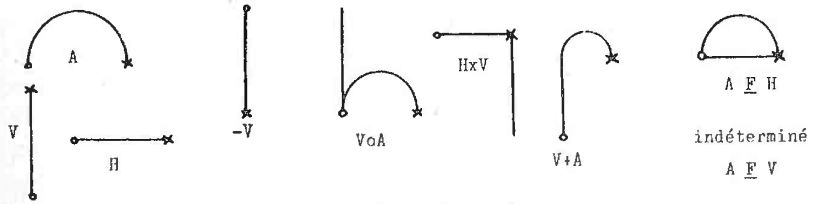


F'(1,2)=F.V(3,2:3;1)

figure 1.6

sur la figure résultante (fig. 1.6). A l'aide de onze symboles de base (segments de droite et arcs de cercle), NARASHIMAN parvient à décrire l'ensemble des caractères d'imprimerie majuscules sous forme d'une grammaire qui lui permet d'en faire une analyse syntaxique. Bien que lourd à manipuler, ce mode de description reste néanmoins attrayant puisqu'il permet d'obtenir des figures à pôles multiples, ce qui est bien utile pour décrire des molècules chimiques ou des structures de graphe par exemple.

Par contre, dans son modèle, SHAW (40) ne conserve que deux pôles privilégiés: l'origine et l'extrémité. Les formes sont réparties en classes de représentation définies à une translation près. On dispose de quatre opèrations binaires permettant diverses combinaisons de superposition des pôles et d'un opèrateur unaire qui inverse origine et extrémité (figure 1.7).

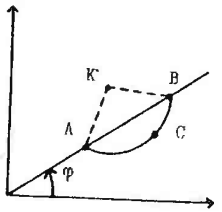


sur les schèmas: "o" symbolise l'origine
"x" symbolise l'extrémité

figure 1.7

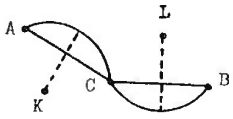
On définit $F \mathfrak{R} F'$ comme la classe des $F_i \mathfrak{R} F'_j$ avec $F_i \in F$ et $F'_j \in F'$, F et F' étant deux formes quelconques. La description d'une forme complexe se fait au moyen d'un système d'équations décrivant chaque sous-forme et analogues aux règles de réécriture employées en thèorie des langages. D'un maniement assez facile, ce modèle a été utilisé avec succès pour l'analyse de photograp de chambre à étincelles.

Cependant, s'ils restent efficients pour des figures simples, de tels formalismes sont insuffisants pour certaines applications industrielles demandant plus de richesse et de prècision dans la description (plans de pièces détachées par exemple). Il faut avoir recours à des langages plus sophistiqués tel celui de VAMOS et VASSY (41). Les diffèrentes primitives sont des segments de droite et des arcs de cercle caractèrisés par les coordonnées de leur origine (symbolisée par la lettre A) et de leur extrémité (symbolisée par la lettre B) sans oublier le centre (K) pour un arc. Elles sont assemblées à l'aide de relations géomètriques multiples et complexes telles que équivalence des extrémités, parallèlisme et orthogonalité, égalité et inégalité de longueur ou courbure etc... On peut en outre individualiser un point quelconque (C) sur un arc de cercle. La description est opèrée au



ARC

figure 1.8



ARC2

figure 1.9

moyen d'un système de règles de réécriture donnant la composition de chaque sous-forme du dessin:

nom de la sous-forme ⟶ |(liste de primitives et de sous-formes)(relations entre les symboles de droite ou leurs attributs)(définitions des attributs du symbole de gauche en fonction de ceux des symboles de droite)(liste des attributs symètriques)|

Ainsi , avec la primitive de la figure 1.8, la figure 1.9 s'écrit:

ARC2 ⟶ |(ARC,ARC)(A_1=A_2, $B_1 \neq B_2$, $K_1 \neq K_2$, $|\varphi_1 - \varphi_2| > 90°$)(A=$E_1$, B=$B_2$, K=$K_1$, L=$K_2$, C=$A_1$)()|

La grammaire ainsi constituée est assimilée à un graphe dort les noeuds représentent les sous-formes et les primitives en association avec leurs attributs. Le processus de reconnaissance fonctionne par exploration systèmatique de ce graphe en essayant de faire la synthèse de chaque sous-forme à l'aide des parties du dessin reconnues précédemment (20). A noter que la configuration admet des schèmas avec parasites. On constate que le mode de description devient vite compliqué et difficile à manier si l'on veut gagner en prècision.

Tous les modèles décrits jusqu'ici font une nette dichotomie entre la reconnaissance qui est un processus indépendant et la description qui est donnée du dessin. D'où l'idée de créer un langage qui génère automatiquement le programme de reconnaissance à partir de la description. C'est ce qui a été fait avec ESP^3 (18) qui est une extension de SNOBOL4. L'approche choisie permet de disposer de primitives très élaborées. En plus des segments et arcs classiques, on trouve le cercle, la courbe, le rectangle, le carré et le triangle. A chaque primitive sont associés des points caractèristiques (origine et extrémité,

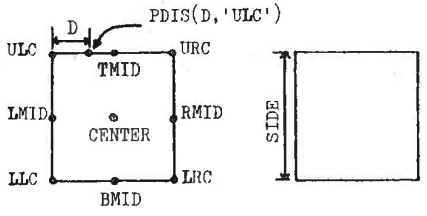


figure 1.10

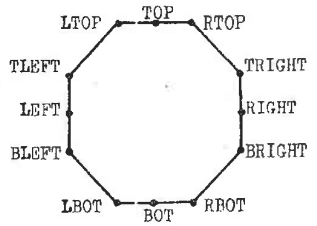
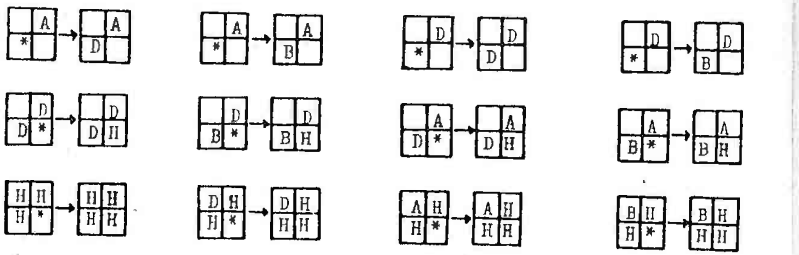


figure 1.11

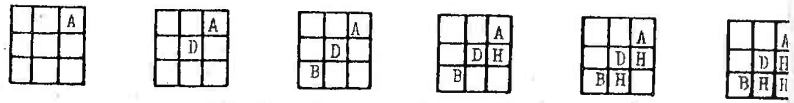
centre du cercle, sommets du triangle etc...), des valeurs caractèristiques (longueur du segment,rayon du cercle, côté du carré etc...) et des entités logiques (spècification "courbe ouverte" ou "courbe fermée" par exemple), cf figure 1.10. Une série de treize fonctions permet de rendre compte des relations entre les composants d'une figure (superposition de points, parallèlisme, intersection, au-dessus...) qui est définie selon douze points privilégiés servant de repères (figure 1.11). Il ne reste plus qu'à écrire en ESP^3 le programme effectuant la reconnaissance du dessin considéré. Du fait de la richesse de ce langage, il n'y a aucune limite de complexité à

la figure décrite, mais l'utilisateur est astreint à acquèrir la dèmarche d'esprit nécessaire à la rédaction des programmes. Il doit manier la sèmantiq qui régit la phase de reconnaissance, travail qui est réservé à la machine dans les exemples précédents.

Il faut encore signaler l'approche introduite par KIRSCH (42) et reprise par OTA (43) et ROSENFELD (44) où le symbole de base est le point. Ils utilisent cette fois un langage à deux dimensions avec des règles de réécritu qui génèrent un arrangement de symboles constituant un polygone (figure 1.12)



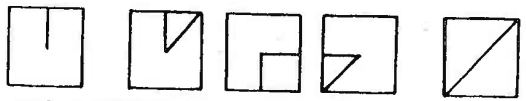
règles de production



génération d'un triangle rectangle

figure 1.12

Ce travail a été repris par DACEY (45)(46) avec des éléments plus élaborés (figure 1.13). Il définit ainsi un langage lui permettant de décrire tout polygone qui se décompose en rectangles et triangles rectangles isocèles.



quelques éléments du langage POLY de DACEY

figure 1.13

Cependant cette idée n'a pas été beaucoup développée car son utilisation reste peu intuitive et très complexe même pour des figures aussi simples que des triangles rectangles.

En fait, l'introduction de langages à deux dimensions revient à discrètiser le schèma, c'est-à-dire à le décomposer en points sur une grille prédéfinie. Ce procédé intèresse surtout le traitement d'images mais il est quelquefois

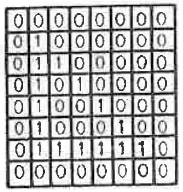


figure 1.14

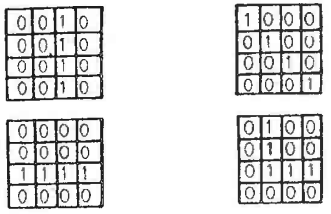


figure 1.15

utilisé pour les dessins. En particulier, les figures traitées par WILLIAMS (19) avec son langage HPL peuvent être des matrices de 0 et de 1 représentant une scène (figure 1.14). Les primitives sont toujours des segments de droite caractèrisés non plus par les coordonnées de leurs extrémités comme dans la plupart des exemples précédents mais par leur forme matricielle (figure 1.15). La décomposition en primitives se fait par comparaisons successives avec les sous-matrices qui les représentent. Par ailleurs, la description est rangée dans une structure de graphe qui reflète l'assemblage du dessin en primitives et sous-formes. Cette mèthode est à rapprocher des techniques statistiques employées en reconnaissance des caractères: chaque lettre est cadrée puis décomposée sur une grille dont chaque case contient le même nombre de points (47)(48). Elle est classée suivant le nombre de points noircis dans chaque région. On ne peut plus à proprement parler de primitives au sens où nous l'entendions jusqu'à maintenant, il s'agit plutôt de paramètres mais qui traduisent aussi la structure et son articulation. Toutefois le procédé ne saurait convenir qu'à un ensemble restreint de figures comme l'alphabet.

Nous pensons avoir présenté les modèles les plus caractéristiques existant actuellement. Nous avons insisté sur les techniques descriptives car beaucoup de méthodologies théoriquement parfaites s'avèrent inutilisables à cause d'un formalisme trop compliqué ou trop éloigné de la réalité.

1-2 Modèle choisi

Suite aux travaux de R. MOHR sur l'extension de l'utilisation des langa
à contexte libre (5)(6), nous avons défini un langage spécifique de descripti
de figures en nous inspirant de PDL (Picture Description Language) inventé pa
SHAW (40). Nous sommes ainsi à même d'appliquer les techniques d'analyse syntaxique sur des formes avec tous les avantages qui en résultent: rapidité, simplicité de mise en oeuvre et surtout ouverture sur l'interprétation sèmantique de la structure analysée.

1-2-1 Structure du modèle

a- notations

Un dessin est un assemblage de constituants élémentaires - les primitive
répartis en deux groupes: les segments de droite (T) et les arcs de cercle (A
Chaque primitive est pourvuede deux pôles: une origine (désormais symbolisée par "o" sur les schémas) et une extrémité (symbolisée par "x"). Il n'y a pas d'échelle de référence dans notre modèle: la longueur des segments et le rayon de courbure des arcs de cercle ne sont pas caractéristiques.

Les primitives sont divisées en classes suivant la direction du vecteur origine-extrémité à partir de l'horizontale dans le sens trigonomètrique, à laquelle il est adjoint une marge de tolèrance sur la mesure de cet angle: A(80, 10) représente la classe des arcs de cercle de direction $80 \pm 10°$ (voir figure 1.16). On peut ainsi regrouper les primitives par affinité: les segments de droite obliques du premier quadrant sont désignés par T(45, 40).

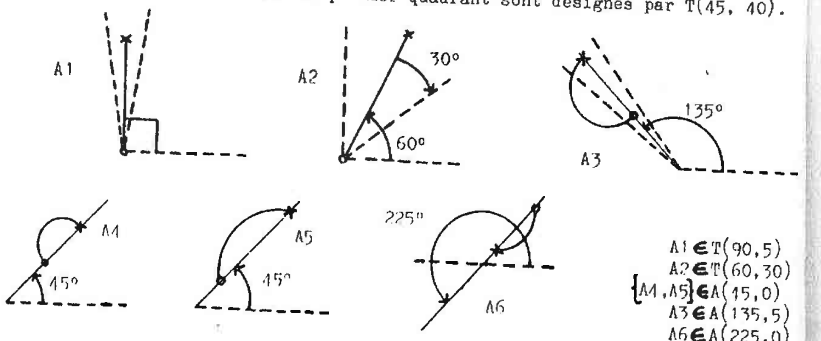


figure 1.16

La répartition en classes est choisie par l'utilisateur qui leur affecte également un nom:

par exemple: h: T(0, 5)
désormais h désigne un segment de droite horizontal avec une tolèrance de 5°.

Il est bien entendu qu'à l'intèrieur d'un même groupe, deux classes diffèrentes ont une intersection vide.

Les primitives sont concaténées par superposition des pôles. Les diffèrents opèrateurs - ou opèrateurs de concaténation - sont identiques à ceux décrits figure 1.7:

Soient f et f' deux formes quelconques.

- l'opèration f + f' fait coincider l'origine de f' avec l'extrémité de f
- l'opèration f x f' fait coincider les extrémités de f et f'
- l'opèration f o f' fait coincider les origines de f et f'
- l'opèration f F f' fait coincider les origines et les extrémités de f et f'

Pour chaque opèration, l'origine de la figure résultante est celle de l'opèrande gauche (f), l'extrémité celle de l'opèrande droit (f'). On dispose en outre de l'opèrateur unaire "-" qui inverse origine et extrémité.

Ces diverses opèrations procèdent par concaténation des pôles sans superposition partielle ou complète des primitives concernées. Aussi, toute formule n'admet pas une interprétation: si s et s' désignent deux segments de droite de classes quelconques, s F s' est indéterminé. Par contre, A_α F A_α peut avoir une signification, avec A_α: $A(\alpha, \varphi)$, cf figure 1.17.

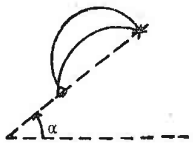


figure 1.17

Pour exprimer les relations topographiques comme "au-dessus" ou topologiques comme "à l'intérieur", nous avons été amenés à définir des opèrateurs de

concaténation topographique* et des opèrateurs de concaténation pseudo-topographique*. Ils sont tous binaires.

10 = 1 ∪ 2 ∪ 3
11 = 4 ∪ 5 ∪ 6
12 = 7 ∪ 8 ∪ 9
13 = 1 ∪ 4 ∪ 7
14 = 2 ∪ 5 ∪ 8
15 = 3 ∪ 6 ∪ 9
16 = 10 ∪ 4 ∪ 6 ∪ 12
17 = 16 ∪ 5

la sous-forme de référence s'inscrit dans la région 5

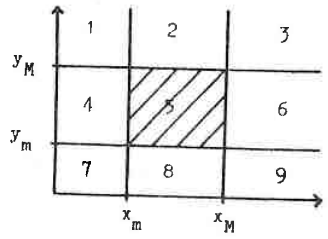


figure 1.18

Chaque sous-forme étant caractérisée par ses coordonnées extrêmes (x_m, x_M, y_m, y_M), les opèrateurs sont définis par référence à ces coordonnées. Par exemple, pour deux sous-formes f et f', "au-dessus" correspond à $y'_m > y_M$. On est ainsi amené à diviser le plan en dix-sept régions (figure 1.18) correspondant aux relations les plus évidentes. Chaque opèrateur est associé à une région et il est possible d'en introduire de nouveaux par adjonction de combinaisons inédites. Il existe pour le moment cinq opèrateurs pseudo-topographiques et huit opèrateurs topographiques prédéfinis. L'origine et l'extrémité de la figure résultante sont portés par l'opèrande gauche.

opèrateurs pseudo-topographiques: les deux opèrandes sont en contact

Soient Fi (i=1,2,3,4,5) les figures suivantes

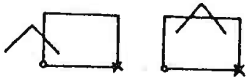
POSE SUR (ou PSUR) → région 2


PSUR (F2,F1)

POSE SOUS (ou PSOUS) → région 8

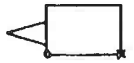

PSOUS (F3,F1

POSE DEVANT (ou PDEVANT) → région 17

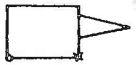

PDEVANT (F2,

* dans la suite on simplifiera en opèrateurs ou relations topographiques

POSE CONTRE A GAUCHE (ou PCTREG) → région 4


PCTREG (F4,F1)

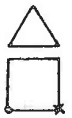
POSE CONTRE A DROITE (ou PCTRED) → région 6


PCTRED (F5,F1)

opèrateurs topographiques

Soient F1: carré et F2: triangle

SUR → région 10


SUR (F2,F1)

SOUS → région 12


SOUS (F2,F1)

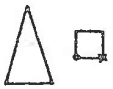
INTERIEUR (ou INT) → région 5


INT (F2,F1)

EXTERIEUR (ou EXT) → région 16


EXT (F2,F1)

AGAUCHE → région 15


AGAUCHE (F2,F1)

ADROITE → région 13


ADROITE (F2,F1)

COTE GAUCHE (ou CGAUCHE) → région 4


CGAUCHE (F2,F1)

COTE DROIT (ou CDROIT) → région 6


CDROIT (F2,F1)

Le résultat d'une opèration de concaténation topographique n'est pas unique: opèrateur (f,f') désigne une classe de figures dont chaque élément est composé d'une forme f et d'une forme f' vérifiant la relation définie par opèrateur.

b- génération d'une description

L'ensemble des noms des classes de primitives est noté P.

Les noms de classes de formes non primitives pourront également être appelés non-terminaux, par analogie avec les notations habituelles employées en théorie des langages. Leur ensemble est noté N.

Posons V = N ∪ P. La description d'une figure quelconque s'effectue au moyen d'un système de règles de réécriture décrivant l'ensemble des sous-formes utilisées pour composer le dessin. Elles ont l'allure suivante:

$$\alpha \rightarrow \varphi \ /n/ \quad \text{avec } \alpha \in N$$

φ est une expression algébrique d'opèrateurs et d'éléments de V qui donne la description de α; n est un numèro associé à la règle pour servir de repère. Lorsque plusieurs alternatives sont admises pour le non-terminal α, on adopte une notation condensée qui a la forme suivante:

$$\alpha \rightarrow \varphi_1 \ /n_1/ \ \ \varphi_2 \ /n_2/ \ \ldots \ \varphi_k \ /n_k/$$

Un non-terminal X, appelé l'axiome, désigne la classe de formes à reconnaitre, les autres n'apparaissant que comme intermèdiaires. L'ensemble des règles de description - de préférence à règles de réécriture - forme le système de description qui est noté $S = (N,P,\rightarrow,X)$.

Le schéma de télévision portative dont un exemple est donné à la figure 1.19 est décrit par le système suivant:

S = (N,P,→,TELEVISION)

N = {TELEVISION,ANTENNE,POIGNEE,CAISSE,CADRE,U,ECRAN,COMMANDES,BOUTON, BOUTONS,TIRETTE,TIRETTES,HP}

P = {hor,vert,oblg,obld,arc1,arc2,arc3,arc4}

Les définitions des primitives sont:

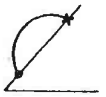
hor: T(0,5)

vert: T(90,5)

oblg: T(135,40)

obld: T(45,40)

arc1: A(45,5)



arc2: A(315,5)

arc3: A(0,5)

arc4: A(180,5)

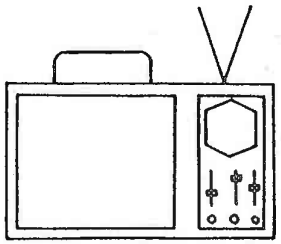


figure 1.19

Il y a quinze règles de description:

TELEVISION ⟶ POSE SUR (ADROITE (ANTENNE,POIGNEE),CAISSE) /1/

ANTENNE ⟶ oblg o obld /2/

POIGNEE ⟶ vert + arc1 + hor + (-arc2) + (-vert) /3/

CAISSE ⟶ INTERIEUR (ADROITE (COMMANDES,ECRAN),CADRE) /4/

CADRE ⟶ U F hor /5/

U ⟶ -vert + hor + vert /6/

ECRAN ⟶ CADRE /7/

COMMANDES ⟶ INTERIEUR (SOUS (SOUS (BOUTONS,TIRETTES),HP),CADRE) /8/

BOUTONS ⟶ BOUTON /9/ COTE GAUCHE (BOUTON,BOUTONS) /10/

BOUTON ⟶ arc3 F arc4 /11/

TIRETTES ⟶ TIRETTE /12/ COTE DROIT (TIRETTE,TIRETTES) /13/

TIRETTE ⟶ POSE DEVANT (CADRE,hor) /14/

HP ⟶ (vert + obld + (-oblg)) F (-oblg + obld + vert) /15/

Bien entendu, ce système n'est pas unique. Par exemple, en remplaçant les occurrences de ECRAN par CADRE, on peut supprimer la règle 7 qui ne sert qu'à rendre la description plus claire. On peut également remplacer la règle 15 par

HP ⟶ (vert + obld + (-oblg) + (-vert)) F (-oblg + obld)

ou bien HP ⟶ (obld + (-oblg) + (-vert) + (-obld) + oblg) F (-vert) etc.

Il suffit de choisir la configuration la plus pratique ou la plus significative.

1-2-2 A propos du modèle

En comparaison des modèles les plus perfectionnés décrits au premier paragraphe, le nôtre a l'avantage d'être naturel. L'utilisateur a la totale liberté du choix des classes de primitives. De plus, le mode de concaténation fonctionne sur deux pôles, d'où une grande souplesse et une grande facilité

d'emploi. Il n'est point besoin d'être familiarisé avec les techniques de programmation ou d'analyse syntaxique pour pouvoir s'en servir aisément au bout de quelques instants, alors que le modèle conserve toutes les propriétés des langages à contexte libre, primordiales pour l'analyse et l'interprétation de scènes.

Cependant cette simplicité même marque les limites de notre modèle.

L'absence d'échelle de référence ne permet pas de générer des descriptions aussi fines que celles de ESP^3 (18) par exemple. Plus grave, il est impossible

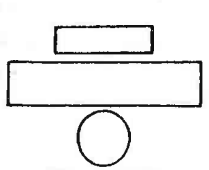


figure 1.20

figure 1.21

figure 1.22

de faire la distinction entre les figures 1.20, 1.21 et 1.22 qui appartiennent toutes les trois à la même classe, sauf par recours à des traitements numériques ultérieurs.

De même, il n'est pas possible de décrire tous les cas de figures. D'une part, nous sommes limités par notre approche basée sur les langages à contexte libre (cf la démonstration de CHO (49)): supposons que nous désirions décrire la grille de la figure 1.23. Pour relier un élément (un "carré") de cette grille à ses voisins, il y a quatre points de contact. Notre modèle est donc insuffisant. Même en adoptant un modèle à quatre pôles, la connection d'un élément avec ses voisins provoque de nouveau la création de quatre pôles supplèmentaires, soit huit au total (figure 1.23) et ainsi de suite. Il n'existe aucune grammaire à contexte libre générant l'ensemble des figures à partir de primitives prédéfinies (CHO).

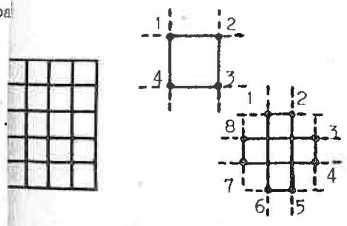


figure 1.23

D'autre part, les définitions des différentes relations (pseudo-)topographiques ne sont pas assez fines. Par exemple, si on s'en tient aux spécifications

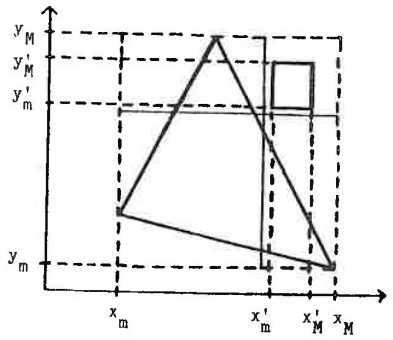


figure 1.24

du paragraphe 1-2-1 a, le carré de la figure 1.24 doit être décrit comme étan à l'intérieur du triangle. On imagine aisèment le danger représenté par de telles utilisations. Pour pallier cette carence, il faudrait amèliorer les définitions des relations topographiques en y introduisant des opèrateurs booléens et arithmètiques. Ainsi, la relation correcte pour la figure 1.24 s'écrit:

$$x'_m \geqslant x_m + \frac{2}{3}(x_M - x_m) \text{ et } y'_m \geqslant y_m + \frac{2}{3}(y_M - y_m)$$

Il serait peut-être bon également de ne plus se limiter à de simples relations binaires, pour enrichir et condenser les descriptions. Quoiqu'il en soit, notre modèle couvre largement nos exigences immèdiates et reste ouvert à tous les perfectionnements souhaités.

CHAPITRE II

NOTATIONS ET PRINCIPE

Avant de passer à l'exposé de l'algorithme de base, nous allons préciser les notations employées dans la suite. On pourra consulter (1) et (2) pour de plus amples informations.

2-1 Notations

Soit G = (N,T,::=,X) une grammaire à contexte libre.
N est le vocabulaire non-terminal.
T est le vocabulaire terminal.
on note V = N ∪ T le vocabulaire de la grammaire.
::= est la relation entre N et V^* (ensemble des mots sur V).
X est l'axiome de la grammaire (X ∈ N).
On notera ∧ la chaine vide.

La relation $\rightarrowtail$ (se réécrit) est définie sur V^* par:

$$(\beta,\gamma \in V^*)\ (\beta \rightarrowtail \gamma) \Longleftrightarrow (A \in N)(\beta',\beta'',\alpha \in V^*)(\beta=\beta' A \beta'' \text{ et } \gamma=\beta'\alpha\beta'' \text{ et } A::=\alpha)$$

La relation $\overset{*}{\rightarrowtail}$ (de...dérive...) est définie sur V^* par:

$$(\beta,\gamma \in V^*)\ (\beta \overset{*}{\rightarrowtail} \gamma) \Longleftrightarrow (\beta_0,\beta_1,\dots\beta_k \in V^* \text{ avec } \beta_0=\beta \text{ et } \beta_k=\gamma)$$
$$(\beta_0 \rightarrowtail \beta_1 \rightarrowtail \cdots \rightarrowtail \beta_k)$$

Définition 1: soit $\mathfrak{J}$ la relation dans V définie par:

$$(A,B \in V)(A\ \mathfrak{J}\ B) \Longleftrightarrow (\exists \Phi \in V^*, \exists \psi \in V^*)(A::=\Phi B \psi \text{ et } \Phi \overset{*}{\rightarrowtail} \wedge)$$

Cette relation est appelée "initiale".
Pour une grammaire sans chaine vide, elle se ramène à:

$$(A,B \in V)(A\ \mathfrak{J}\ B) \Longleftrightarrow (\exists \psi \in V^*)(A::=B\psi)$$

Définition 2: soit $\mathfrak{F}$ la relation dans V (appelée "finale") définie par

$$(A,B \in V)(A\ \mathfrak{F}\ B) \Longleftrightarrow (\exists \varphi \in V^*, \exists \psi \in V^*)(A::=\varphi B \psi \text{ et } \psi \overset{*}{\rightarrowtail} \wedge)$$

Dans le cas d'une grammaire sans chaine vide, elle se réduit à:

$$(A,B \in V)(A\ \mathfrak{F}\ B) \Longleftrightarrow (\exists \varphi \in V^*)(A::=\varphi B)$$

Définition 3: on définit la relation $\mathfrak{I}$ ("successeur") sur V par:

$$(A,B \in V)(A\ \mathfrak{I}\ B) \Longleftrightarrow (\exists C \in N, \exists \varphi \in V^*, \exists \psi \in V^*)(C::=\varphi A B \psi)$$

La fermeture réflexive et transitive de $\mathfrak{J}$ est notée $\mathfrak{J}^*$:
$(A,B\in V)(A\ \mathfrak{J}^*\ B) \Longleftrightarrow (\exists\psi\in V^*)(A \succ^* B\psi)$ pour une grammaire sans vide
De la même façon pour $\mathfrak{F}$:
$(A,B\in V)(A\ \mathfrak{F}^*\ B) \Longleftrightarrow (\exists\varphi\in V^*)(A \succ^* \varphi B)$

Définition 4: on définit la relation vs ("voisin"), sur V, par:
$$vs = \mathfrak{F}^{*-1} \circ \mathfrak{Y} \circ \mathfrak{J}^*$$
on établit la distinction entre voisin droit (vd) et voisin gauche (vg) par (3):
$$vd = \mathfrak{F}^{*-1} \circ \mathfrak{Y} \circ \mathfrak{J}^*$$
$$vg = \mathfrak{J}^{*-1} \circ \mathfrak{Y} \circ \mathfrak{F}^* = vd^{-1}$$

Les deux relations possèdent les propriétés suivantes:
$(A,B\in N)(A\ vd\ B) \Longleftrightarrow (C\in N)(\exists\varphi\in V^*, \exists\psi\in V^*)(C \succ^* \varphi BA\psi)$
$(A,B\in N)(A\ vg\ B) \Longleftrightarrow (C\in N)(\exists\varphi\in V^*, \exists\psi\in V^*)(C \succ^* \varphi AB\psi)$

Notion de ramification: les informations arborescentes sur V sont appelées ramifications sur V. Leur ensemble est noté $\hat{V}$. Considèrons celle de la figure 2.1: elle est orientée de droite à gauche. Elle possède deux racines: b et c. Chaque noeud est étiqueté par un symbole appartenant à V.

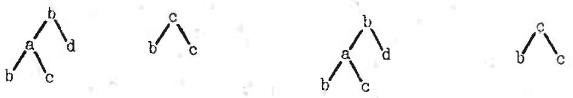
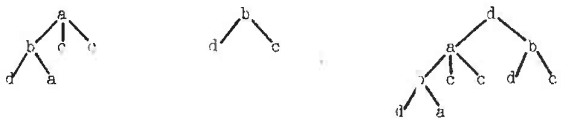


figure 2.1 figure 2.2 figure 2.3

$\hat{V}$ est muni de deux lois de composition:
- une loi de composition interne, la concatènation, qui juxtapose deux ramifications à m et n racines en une ramification à (m+n) racines. Elle est notée +.
- une loi de composition externe, l'enracinement, notée x, qui à un élément $\alpha\in V$ et une ramification r associe la ramification obtenue en enracinant r à une racine étiquetée α.

EXEMPLES: la ramification de la figure 2.1 est le résultat de la concatènation des figures 2.2 et 2.3.
la figure 2.4 montre l'enracinement de r par la racine étiquetée d.



ramification r d x r

figure 2.4

Notons $\wedge$ la ramification vide.

Définition 5: soit ρ l'application de $\hat{V}$ dans V^* telle que:
$$\begin{cases}\rho(\wedge) = \wedge \\ \rho(a\ x\ r + t) = a\rho(t)\end{cases}$$
$\rho(r)$ s'appelle le mot des racines de r. Pour la figure 2.4, $\rho(r)$ = ab.

Définition 6: soit φ l'application de $\hat{V}$ dans V^* telle que:
$$\begin{cases}\varphi(\wedge) = \wedge \\ \varphi(a\ x\ r + t) = \text{si } r=\wedge \text{ alors } a\varphi(t) \text{ sinon } \varphi(r)\varphi(t)\end{cases}$$
$\varphi(r)$ s'appelle le mot des feuilles de r. Pour la figure 2.4, $\varphi(r)$=daccdc.

2-2 Analyse syntaxique partant du milieu d'une phrase

Dans un langage destiné à la compilation, une phrase est définie par un début et une fin, immédiatement repèrables par l'ordinateur. Effectuer une analyse syntaxique à partir de l'une de ses extrémités est chose aisée. Par contre, ces concepts sont inconnus (cas d'un dessin) ou difficiles à isoler (cas de la parole) sur une forme. L'utilisation de langages de description à contexte libre étant maintenant éprouvée, il faudrait un outil permettant de conduire une analyse à partir d'un endroit quelconque de la phrase (le terme est pris ici au sens large) à traiter.

Cet algorithme a été décrit dans (3): on dèsire effectuer l'analyse syntaxique d'une chaine en démarrant avec un terminal a situé n'importe où dans cette chaine. On supposera dans la suite que la grammaire est sans chaine vide. On choisit donc une règle telle que: $A ::= \lambda a \lambda'$ avec $A\in N$, $\lambda\in V^*$, $\lambda'\in V^*$. On complète l'arbre ainsi obtenu par une analyse syntaxique descendante droite-gauche pour λ et gauche-droite pour λ'.

On réitère ensuite le processus, avec A comme départ cette fois, jusqu'à épuisement de la chaine (figure 2.5). Si, dans la dernière étape, la racine de l'arbre construit est l'axiome de la grammaire, la chaine est reconnue correcte.

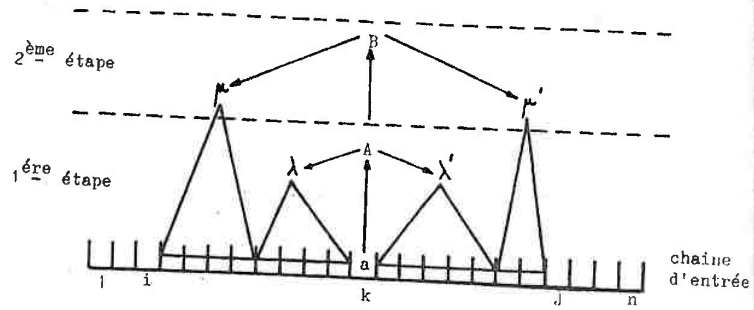


figure 2.5

Supposons que:

- r désigne l'arborescence construite pendant l'analyse:
  $r, r_1 \ldots r_q \in \hat{V}$; $A, B, C_1 \ldots C_q \in V$
- i (resp. j) pointe sur le premier terminal gauche (resp. droit) de la chaine d'entrée non encore reconnu.
- n désigne la longueur de la chaine d'entrée.
- X désigne l'axiome de la grammaire.
- ANDG (resp. ANGD) désigne une procèdure d'analyse syntaxique descendante droite-gauche (resp. gauche-droite) classique.

L'algorithme formel est présenté page suivante.

Avec la chaine de la fig. 2.5, le premier appel a la forme suivante:

```
r := a ;
i := k-1 ;
j := k+1 ;
REMONTE ;
```



```
procèdure REMONTE: A := ρ(r) ;
                   si (A = X et i = 0 et j = n+1) alors arrêt fsi* ;
                   Choix d'une règle B ::= C1...Ck A Ck+1...Cq ;
                   rk := ANDG(Ck) ; ... ; r1 := ANDG(C1) ;
                   rk+1 := ANGD(Ck+1) ; ... ; rq := ANGD(Cq) ;
                   r := B x (r1 + ... + rk + T + rk+1 + ... + rq) ;
                   REMONTE ;
fin procèdure
```

Avec un simple procédé de retour-arrière pour le choix, le temps d'exécution de cet algorithme sera exponentiel. On réduit l'indéterminisme en testant les contextes droit et gauche: la règle $B ::= C_1..C_k A C_{k+1}..C_q$ ne sera choisie que si elle permet de poursuivre l'analyse à droite et à gauche par un test sur les terminaux de rang j et i, en se restreignant à des grammaires LL(1) et LR(1) (notations de Knuth (4)); autrement dit si le terminal de rang i (resp. j) est l'initiale (resp. finale) de C_k (resp. C_{k+1}).

Le contexte droit se définit donc de la manière suivante:

$$D_1(B\ ,\ C_{k+1}\ldots C_q) = \{a \in T \mid C_{k+1}\ \mathfrak{J}^*\ a\}$$
$$D_1(B\ ,\ \wedge) = \{a \in T \mid B\ vg\ a\}$$

De même, pour le contexte gauche:

$$G_1(B\ ,\ C_1\ldots C_k) = \{a \in T \mid C_k\ \mathfrak{F}^*\ a\}$$
$$G_1(B\ ,\ \wedge) = \{a \in T \mid B\ vg\ a\}$$

La règle $B ::= \lambda A \lambda'$ n'est choisie qu'à condition que:

$$\begin{cases} a_j \in D_1(B,\lambda') \Leftrightarrow (\exists C, A \in V)(\exists \alpha,\beta,\gamma,\delta \in V^*) & C \overset{*}{\rightarrow} \alpha B \beta \rightarrow \gamma A \lambda' \beta \overset{*}{\rightarrow} \delta A a_j \delta \\ a_i \in G_1(B,\lambda) \Leftrightarrow (\exists C, A \in V)(\exists \alpha,\beta,\gamma,\delta \in V^*) & C \overset{*}{\rightarrow} \alpha B \beta \rightarrow \alpha \lambda A \gamma \overset{*}{\rightarrow} \delta a_i A \gamma \end{cases}$$

On obtient ainsi un algorithme déterministe, à condition que la grammaire correspondante soit LR(1) et LL(1), et bien sûr qu'il n'existe pas plusieurs non-terminaux possèdant des contextes identiques. Sinon, on a simplement restreint l'éventail des choix possibles, ce qui n'est déjà pas si mal.

* on supposera, pour simplifier, que l'axiome n'apparait qu'une seule fois dans la grammaire et comme premier membre d'une règle. On peut toujours se ramener à ce cas.

2-3 Application à la reconnaissance des formes

En nous restreignant au modèle retenu dans le paragraphe 1-2, nous pourrons appliquer une analyse de cette sorte à condition de généraliser les notions de contexte, initiale et finale. Ce qui va suivre est l'aboutissement d'une suite d'études menées par R. MOHR (5)(6) puis R. MOHR et J.P. HATON (7)(8). On s'y reportera utilement pour des renseignements plus précis ou d'ordre plus général.

2-3-1 Notations

P désigne l'ensemble des classes de primitives.
N désigne l'ensemble des classes de formes non primitives.
On pose $V = N \cup P$ et $-E = \{-a \mid a \in E\}$ où E désigne tout sous-ensemble de V.

2-3-2 Généralisation de la notion d'initiale et de finale

On remplace respectivement les relations "initiale" et "finale" par "prèsent à l'origine" et "prèsent à l'extrémité" (chaque forme du modèle possèdant deux pôles de concaténation - cf § 1-2). Plusieurs primitives peuvent satisfaire à l'une de ces conditions, aussi les relations s'appliqueront-elles dèsormais à l'ensemble des parties de P (en supposant qu'il y a une occurrence au plus pour une classe de primitives donnée).

Plus généralement, soit I un sous-ensemble de $-V \cup V$ dont l'origine de chaque élément coincide avec l'origine d'un nom de classe de formes non primitives A. On notera: $A \, \mathfrak{J} \, I$.

De la même façon, $-A \, \mathfrak{J} \, F$, avec F désignant l'ensemble des éléments de $-V \cup V$ dont l'origine coincide avec l'extrémité de A.

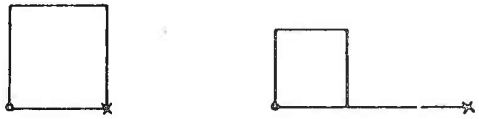
On généralise donc la relation initiale par $\mathfrak{J}^*(A)$, la relation finale par $\mathfrak{J}^*(-A)$, avec $\mathfrak{J}^*$ comme la plus petite relation sur $\mathcal{P}(-V \cup V)$ vérifiant:

- $\mathfrak{J}^*$ contient l'égalité
- $\mathfrak{J}^* \supset \mathfrak{J}$
- $\mathfrak{J}^*$ est stable par composition

EXEMPLE: soit le système de description:

$$\begin{cases} \text{FIGURE} \longrightarrow \text{CARRE} + \text{HOR} & /1/ \\ \text{CARRE} \longrightarrow (\text{VERT} + \text{HOR} + -\text{VERT}) \ \underline{F} \ \text{HOR} & /2/ \end{cases}$$

VERT désigne un segment de droite vertical.
HOR désigne un segment de droite horizontal.



CARRE FIGURE

On obtient: CARRE $\mathfrak{J}^*$ {VERT, HOR}
-CARRE $\mathfrak{J}^*$ {VERT, -HOR}
FIGURE $\mathfrak{J}^*$ {CARRE, VERT, HOR}
-FIGURE $\mathfrak{J}^*$ {-HOR}

NOTA: ceci ne reste valable que sous les deux hypothèses (cf (8) page 22):

- Pour toute forme décrite par le système, il n'y a de coincidences de pôles de primitives que celles explicitement spécifiées par la description.
- Toute formule admet une interprètation (contre-exemple au § 1-2).

2-3-3 Généralisation de la notion de voisin

Considèrons le système de description:

$$\begin{cases} X \longrightarrow a + Y \\ Y \longrightarrow b \circ c \end{cases}$$

avec $X, Y \in N$
$a, b, c \in P$

L'origine de b et celle de c coincident, comme l'indique la seconde règle. En outre, la première règle nous apprend que l'extrémité de a et l'origine de Y (qui est aussi celle de b) sont concatènées. Donc le voisinage à l'origine de b est l'origine de c et l'extrémité de a:
b vs {-a , c}.

Dans la suite, on notera vs(A) les voisins à l'origine de A et vs(-A) les voisins à son extrêmité, $A \in V$.

Le théorème suivant montre la correction de vs(cf 8):

THEOREME: soit F une forme engendrée par un système de description S,p une primitive apparaissant dans F et M son origine (resp. son extrêmité), soit E l'ensemble des primitives et opposées de primitives différentes de p ayant leur origine en M, alors: p vs E (resp. -p vs E).

Réciproquement si p vs E (resp. -p vs E), alors il existe F tell qu'il existe p dans V vérifiant que l'ensemble des primitives et opposées de primitives ayant leur origine en commun avec l'origine (resp. l'extrêmi de p soit E.

2-3-4 Les relations topographiques

Les notions décrites précédemment ne s'appliquent pas pour des règles comprenant des relations topographiques (voir § 1-2). Considèrons les trois règles de description:

B ⟶ INT (A , C)

A ⟶ AU-DESSUS (X , Y)

A ⟶ AU-DESSOUS (X , Z) $A,B,C,X,Y,Z \in N$

La sous-forme X étant reconnue, la procèdure REMONTE ne peut choisir la bonne règle à utiliser. En effet, les relations "au-dessus" et "au-desso sont trop vagues et les constituants de Y et Z (certainement) trop nombreux: un simple test local portant sur l'abscence ou la prèsence de primitives ne donnera pas la nature du contexte de X (surtout si Y et Z ont beaucoup de primitives en commun avec C).

Cependant, on obtiendra une probabilité de prèsence d'une forme d'une classe donnée dans une zone donnée en testant la prèsence dans cette zone des primitives caractèristiques de la forme de cette classe:

Une primitive sera dite caractèristique pour une classe A, si elle apparait obligatoirement dans toute forme de cette classe, et si elle n'apparait dans aucune sous-forme qui n'est pas obligatoirement contenue dans les sous-formes de classe A ou qui contient des sous-formes de classe

La recherche des primitives caractèristiques - lorsqu'elles existent - devient une façon simple de lever l'indéterminisme lors de choix faisant intervenir de telles relations. Désormais, il y aura distinction entre les règles comportant des relations topographiques et les autres (cf § 3-3-3) dans la procèdure REMONTE.

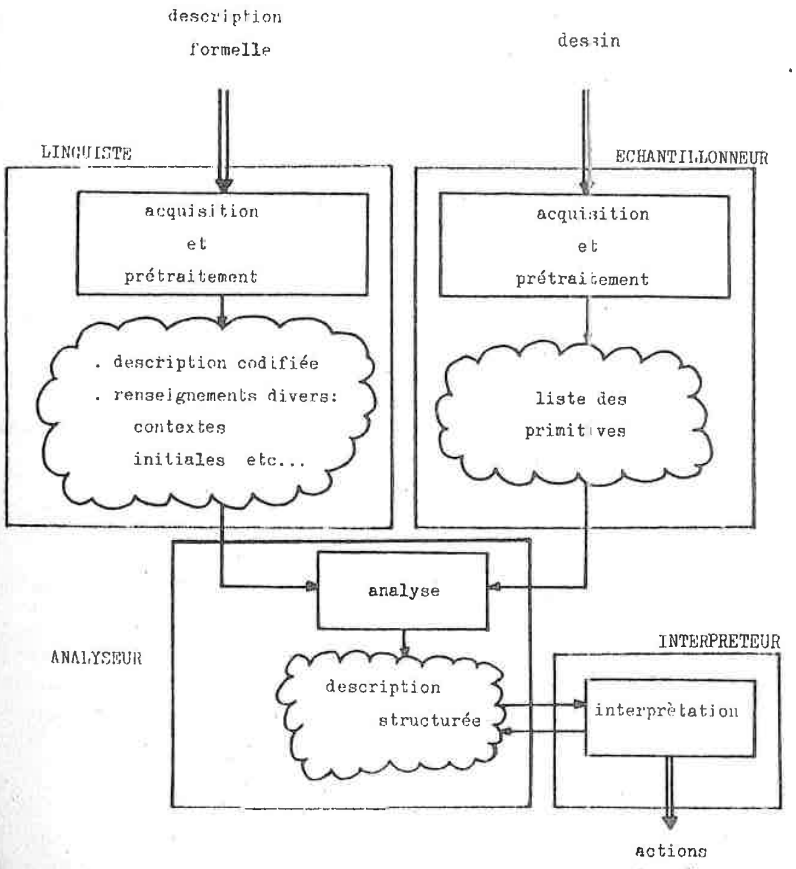
CHAPITRE III

UN ANALYSEUR DE DESSINS

3-1 Un système d'analyse de dessins

3-1-1 Présentation

L'analyseur de dessins proposé dans les chapitres suivants fait partie d'un projet de système de reconnaissance des formes (pour sa réalisation pratique: système de reconnaissance de dessins) (8), dont le schèma se trouve ci-dessous.



Le LINGUISTE est un module à deux fonctions:

. Acquisition conversationnelle du système de description de la classe de dessins et codification (dans une structure utilisable par l'ANALYSEUR). Il n'y a aucune restriction (en principe) sur le degré de complexité des règles de description et sur le "sens" des règles récursive s'il y en a (récursivité droite, gauche ou à plusieurs niveaux). Ceci peut paraitre incompatible avec l'algorithme tel qu'il est décrit dans le paragraphe 2-2: en effet, on y utilise des modules d'analyse syntaxique descendante de la droite vers la gauche et de la gauche vers la droite pour lesquels sont nécessaires, respectivement, une récursivité droite et gauche. Cependant le fonctionnement réel de l'ANALYSEUR (cf § 3-3) tolère un tel mélange.

Les groupes de règles isolées (numéros 3 et 4 dans le système de l'exemple 1) ainsi que les circuits (exemple 2) sont signalés.

EXEMPLE 1

$X \longrightarrow A \circ a$ /1/
$A \longrightarrow b + c$ /2/
$B \longrightarrow C + a$ /3/
$C \longrightarrow b \times c$ /4/

$X,A,B,C \in N$
$a,b,c \in P$

EXEMPLE 2

$A \longrightarrow (B \circ b) + a$
$B \longrightarrow (a \times b) \circ a$

$A,B \in N$
$a,b \in P$

L'utilisateur introduit ses propres classes de primitives en donnant leur nature (ARC ou TRAIT) ainsi que leur direction avec une marge. Il peut aussi introduire des opérateurs topographiques de son cru en fournissant les numéros de zone correspondants (voir § 2-3-2).

. Extraction de renseignements inhérents au système de description et qui vont faciliter et optimiser le fonctionnement de l'ANALYSEUR. Ce sont bien sûr les initiales, les voisins, les primitives caractéristiques plus une relation supplémentaire sur laquelle nous reviendrons dans le paragraphe suivant. Le processus d'analyse est notablement accéléré par cette connaissance a priori sur la forme à traiter.

L'ECHANTILLONNEUR possède également deux fonctions:

. Acquisition proprement dite du dessin (i.e. coordonnées avec levers et posers de crayon) et échantillonnage (i.e. suppression des parasites et des points non significatifs).

. Découpage en primitives: les différents points conservés sont répartis en segments de droite et arcs de cercle avec calcul de leurs caractéristiques (direction et marge) puis répartition dans les classes de primitives choisies par l'utilisateur.

L'ECHANTILLONNEUR et l'ANALYSEUR ne sont pas interactifs: ce qui veut dire que si un certain découpage en primitives semble incorrect, on ne pourra relancer l'ECHANTILLONNEUR pour le refaire. Il ne fournit pas non plus de réponses multiples valorisées. Le découpage en primitives est figé (on verra cependant que ce n'est pas tout à fait vrai au § 3-1-3).

L'ANALYSEUR appréhende le dessin pour en fournir une description structurée très précise, utilisable rapidement et efficacement par l'INTERPRETEUR qui en tire les informations nécessaires à une application ou une série d'actions voulues par l'utilisateur (sa réalisation n'est encore qu'à l'état de projet). Par contre, les renseignements donnés par l'ECHANTILLONNEUR et le LINGUISTE étant déterminants pour le fonctionnement de l' ANALYSEUR, nous allons y revenir en détail.

3-1-2 Le LINGUISTE

Sa réalisation est due à R. MOHR. Les problèmes posés par l'acquisition du système de description n'ont que peu d'intérêt pour cette étude et nous ne nous y attarderons pas (seule nous intéresse la structure de rangement des règles de description). Nous examinerons ensuite les informations extraites du système.

a - les règles de description

Elles se présentent sous forme d'arborescences binaires.

Par exemple, la règle A ⟶ -a + B o (C x D) donne l'arbre suivant:

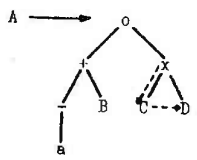


figure 3.1

On dit que o et +, - et a sont liés verticalement, + et x, C et D sont liés horizontalement. Chaque arbre est stocké dans un tableau spécial appelé matrice d'enchainement (9). Le lien vertical gauche est donné par la consécutivité dans le tableau; le lien vertical droit n'est pas représenté. L'abscence de lien est indiqué par nil. Le rangement des étiquettes des noeuds dans la matrice s'effectue de haut en bas et de gauche à droite. L'arborescence de la figure 3.1 donne la matrice suivante

étiquettes	nom	lien horizontal
107	o	nil
108	+	114
109	-	112
110	a	nil
111	nil	
112	B	109
113	nil	
114	x	108
115	C	117
116	nil	
117	D	115
118	nil	

figure 3.2

Cette représentation permet d'explorer l'arborescence dans n'importe quel sens à partir de n'importe quel noeud. Les liens horizontal et vertical gauche sont donnés par la matrice elle-même, le lien vertical droit est obtenu en composant le lien vertical gauche et le lien horizontal (flèches en pointillés sur la figure 3.1). En outre, on adjoint à cette matrice un en-tête de description qui contient les renseignements suivants:

nombre total de lignes occupées	numèro de la règle de description (négatif si la règle est récursive)
étiquette du noeud qui porte l'origine	étiquette du noeud qui porte l'extrémité
parité d'inversion pour l'origine	parité d'inversion pour l'extrémité
nom de la classe non primitive décrite	

On appelle parité d'inversion (PINV) le nombre d'opérateurs unaires moins se trouvant sur le chemin qui mène de la racine à un noeud quelconque de l'arborescence. Ce renseignement est nécessaire au fonctionnement de l'analyse (cf § 3-3-2). PINV = 1 pour un nombre pair, sinon PINV = 2.

On remarque en outre que toutes les feuilles de l'arborescence d'une règle de description sont des éléments de P ou de N et réciproquement. Ceci nous permet de supprimer les marques d'abscence de lien vertical gauche dans la matrice d'enchainement. Elle sera dèsormais indiquée par la contiguité de deux éléments de P ou N.

En supposant que l'arbre de la figure 3.1 représente la règle de description numèro 1, la matrice correcte correspondante est donnée page suivante.

étiquettes

1	12	1
2	8	12
3	2	1
4	A	

5	o	nil
6	+	10
7	-	9
8	a	nil
9	B	7
10	x	6
11	C	12
12	D	11

Toutes les matrices sont numèrotées et regroupées dans un fichier de nom FREL.

b - informations extraites du système de description

Le LINGUISTE tire du système de description le plus de renseignements possible afin d'avoir une connaissance très riche de la forme à reconnaitre et donc accèlérer le processus d'analyse. Il cherche ainsi:

. Les voisins à l'origine et à l'extrémité de chaque élément du vocabulaire pour chaque règle de description dans laquelle il apparait:

$(\forall A \in V)(\forall i$ tel que $B \longrightarrow \lambda A \lambda'$ /i/ $(B \in N, \lambda, \lambda' \in V^*))(vs_i(A), vs_i(-A))$

Ils sont regroupés dans le fichier FCONTX et utilisés lors du choix d'une règle pendant la phase ascendante de l'analyse.

. Les initiales à l'origine et à l'extrémité de chaque nom de classe de formes non primitives pour chaque définition qu'il en est donné:

$(\forall A \in N)(\forall j$ tel que $A \longrightarrow \lambda$ /j/ $(\lambda \in V^*))$ $(\mathfrak{J}_j^*(A), \mathfrak{J}_j^*(-A))$

Elles sont regroupées dans le fichier FINIT et utilisées lors du choix d'une règle pendant la phase descendante de l'analyse.

. La relation gamma (notée Γ) définie par:

$(A \in N, B \in V)$ $(A \Gamma B \Longleftrightarrow$ B est une sous-forme de A)

Le résultat est rangé dans le fichier FGAMMA sous forme d'un tableau booléen. Par exemple, pour le système de description suivant:

$X \longrightarrow B$ o a /1/ C + B /2/
$B \longrightarrow$ b /3/ b + B /4/
$C \longrightarrow$ a x c /5/ a x D /6/
$D \longrightarrow$ b /7/ c /8/

$X, B, C, D \in N$
$a, b, c \in P$

On obtient le tableau suivant:

	X	B	C	D	a	b	c
X	F	V	V	V	V	V	V
B	F	V	F	F	F	V	F
C	F	F	F	V	V	V	V
D	F	F	F	F	F	V	V

Ce n'est autre que la fermeture transitive stricte sur le graphe représentant le système de description (10).

Cette relation est utilisée lors du choix d'une règle pendant la phase ascendante de l'analyse: sachant que le résultat sera une arborescence de racine A $(A \in N)$, on ne retiendra que les règles $B \longrightarrow \lambda_1 C \lambda_2$ $(B, C \in N$, $\lambda_1, \lambda_2 \in V^*)$ telles que $A \Gamma B$, c'est-à-dire permettant effectivement d'atteindre la racine désirée.

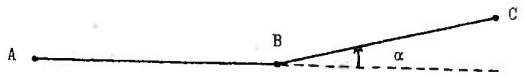
. La liste classée des primitives caractèristiques (voir définition § 2-3-4) pour chaque nom de classe de formes non primitives. En prenant comme point de départ de l'analyse d'une (sous-)forme donnée une primitive de cette liste, on est sûr que l'analyse sera la plus déterministe possible car il n'y a pas d'ambiguité sur l'appartenance du type de primitive à la sous-forme donc sur le choix des règles. Bien entendu, on utilise aussi cette liste pour déterminer les contextes dans le cas d'une relation topographique.

Toutes ces listes sont regroupées dans le fichier FPRIMD.

3-1-3 L'ECHANTILLONNEUR

a - principe

L'acquisition du dessin se fait au moyen d'une tablette graphique TEKTRONIX 4953 à principe magnètique. Elle est constituée d'un réseau de fils qui se croisent orthogonalement et dans lesquels circule un courant magnètique. Un stylo spécial repère les points sur la surface de la tablette et transmet des coordonnées à l'ordinateur à intervalles de temps réguliers. Selon la vitesse du tracé, un segment de droite ou un arc de cercle sera donc représenté par un nombre plus ou moins important de points. L'ECHANTILLONNEUR est chargé de supprimer les points non significatifs. L'algorithme utilisé a été mis au point par MARC BERTHOD (11):



Si l'angle α formé par trois points consécutifs est inférieur à un seuil donné, le point intermèdiaire (B) est supprimé. Le tracé est assimilé au segment AC. Le processus est ensuite réitéré avec A,C et le point suivant.

Il faut en outre éliminer les bavures provoquées par une mauvaise tenue du stylo (fig.3.3) ou un mauvais lever de stylo (fig.3.4): on calcule le demi-périmètre p du rectangle dans lequel s'inscrit un trait entre deux levers du stylo (fig.3.5). Si la distance séparant deux points consécutifs est infèrieure à $p\frac{k}{2}$ (k = cte), le premier point est supprimé.

Les rebroussements (fig.3.6) sont supprimés par une méthode analogue à celle de BERTHOD: si l'angle α est proche de 180°, une des deux branches (1-2 ou 2-3) est éliminée suivant que le rebroussement se situe en début ou en fin de tracé.

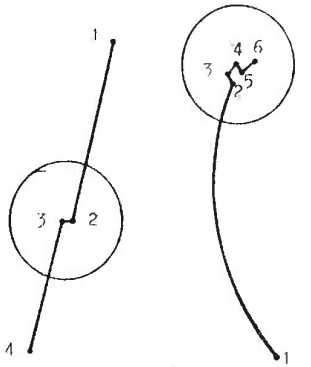


figure 3.3

figure 3.4

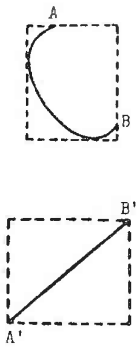
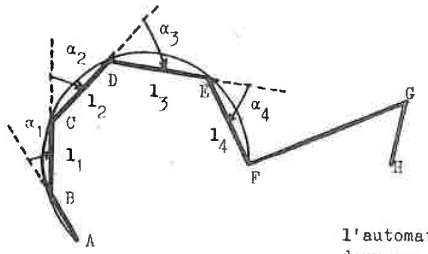


figure 3.5

figure 3.6

Le découpage en primitives est effectué à l'aide d'un automate mis au point par ABDEL BELAID pour sa thèse sur la reconnaissance de caractères manuscrits en temps réel (12). Le principe de base est le suivant: en supposant la vitesse du tracé à peu près constante, la distance sèparant deux points consécutifs émis par la tablette est aussi à peu près constante. Pour un segment de droite cela n'a aucune importance: de toute façon, l'échantillonnage ne retiendra que l'origine et l'extrémité. Par contre, pour un arc de cercle, une fois les points non significatifs éliminés, on obtiendra une succession de segments de droite de longueurs sensiblement identiques et formant entre eux des angles sensiblement égaux (fig. 3.7), sauf peut-être à son origine et à son extrémité. D'autres considèrations entrent également en ligne de compte telles que la variation du rayon de courbure, l'orientation de la courbure et le nombre de segments constituant l'arc de cercle.



$\alpha_1 \approx \alpha_2 \approx \alpha_3 \approx \alpha_4$

$l_1 \approx l_2 \approx l_3 \approx l_4$

l'automate donne l'arc AF et les deux segments FG et GH.

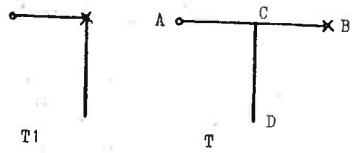
figure 3.7

A noter que la liste des primitives fournie reflète le tracé du dessin et non sa description. Prenons l'exemple de la figure 3.8: le système de description fait état de trois primitives alors que l'ECHANTILLONNEUR n'en fournira que deux si le tracé est effectué en deux traits, AB et CD. En consèquence, l'ANALYSEUR devra possèder un moyen de revenir sur le découpage fourni par l'ECHANTILLONNEUR (moyen qui sera de toute façon indispensable si l'on admet des figures imparfaites en entrée).

$T \longrightarrow T1 + HOR$
$T1 \longrightarrow HOR \times VERT$

$T, T1 \in N$
$HOR, VERT \in P$

HOR désigne un segment de droite horizontal
VERT désigne un segment de droite vertical



figures engendrées

figure 3.8

b - structure des informations

En général, les tableaux cités sont gérés en pile pour les besoins des retour-arrière.

Les coordonnées des points composant chaque primitive sont rangées dans le tableau POINT. Pour faciliter les calculs de délimitation de zones dans le plan de travail à la rencontre d'opèrateurs topographiques, on fait précèder les coordonnées d'un arc de cercle par ses abscisses et

ordonnées extrêmes (symbolisées par le rectangle en pointillés sur la figure 3.5).

Chaque primitive est caractèrisée par son vecteur d'informations appelé BLOC et qui comprend:

. un indicateur d'utilisation: à zéro lorsque la primitive n'a pas encore servi pour l'analyse.

. la direction de la primitive (en radians)

. deux pointeurs (dans le tableau POINT) sur l'origine et l'extrémité de la primitive

. deux drapeaux de modification prècisant si la primitive peut être modifiée (voir § 3-2-2 b) par son origine ou par son extrémité.

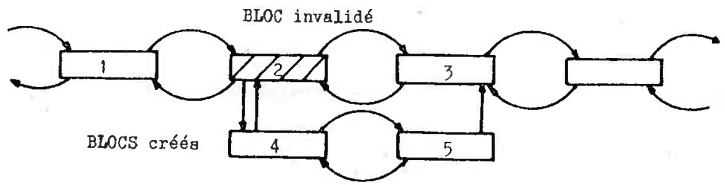
. deux liens de chainage (avant et arrière)

. un indicateur de modification prècisant si la primitive est prolongée, segmentée (i.e. coupée en deux tronçons), ajoutée (voir § 3-2-1 a) ou intacte.

. un emplacement pour le nouveau pointeur avant en cas de modification

Les primitives de même classe sont regroupées dans des listes chainées circulaires bilatères pour faciliter les inclusions de nouveaux BLOCS en cas de modification et optimiser la recherche d'une primitive. Les listes sont regroupées dans la pile PRIM.

En cas de segmentation ou de prolongation de la primitive, le BLOC originel est invalidé par l'indicateur de modification et un ou deux nouveaux BLOCS sont inclus dans la liste (fig. 3.9). Cette méthode permet de regrouper toutes les informations sur les modifications de primitives en couches successives dans les tableaux PRIM et POINT, et donc de les éliminer rapidement en cas de retour-arrière.



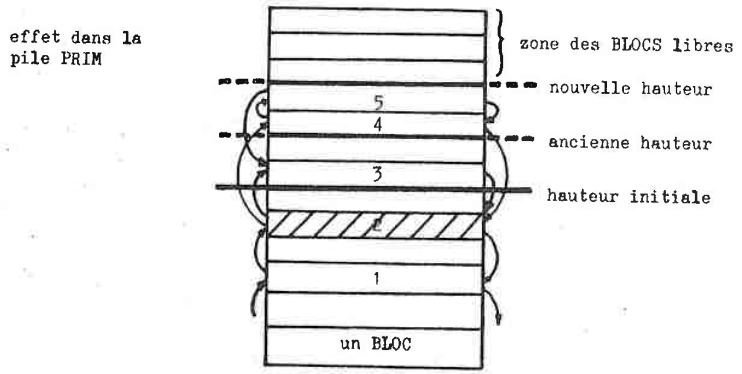


figure 3.9

3-2 Fonctions annexes

L'ANALYSEUR fonctionne sur deux niveaux interactifs: le niveau syntaxique qui donne les ordres d'après la description formelle du dessin et le niveau opèratoire qui tente de les exécuter sur les composants réels du dessin.

Le niveau syntaxique comprend un module d'analyse bien sûr, un module appelé GEOMETRE chargé du calcul et de la gestion des différentes zones du plan correspondant aux opèrateurs topographiques, le SUPERVISEUR qui dirige les choix et les retour-arrière, enfin l'ELECTEUR qui sélectionne le meilleur choix: choix de la primitive de départ pour l'analyse d'une forme ou d'une sous-forme, choix ascendant c'est-à-dire choix d'une règle de description d'après les contextes de la partie du dessin déjà reconnue (il est effectué pendant la phase ascendante de l'algorithme du paragraphe 2-2), choix descendant c'est-à-dire choix d'une règle de description d'un nom de classe de forme non primitive au cours de la phase descendante du même algorithme.

Le niveau opèratoire se compose de l'EXPLORATEUR chargé de la recherche d'une primitive dans le plan à partir d'un point prècis et du GEOGRAPHE qui situe une primitive dans le plan. Nous y rangeons également le module de construction de l'arbre de description de la forme analysée bien qu'il s'apparente aux deux niveaux par son rôle à la fois descriptif et nominatif.

L'ensemble est schèmatisé page suivante (figure 3.10).

Nous allons maintenant expliquer le fonctionnement de chaque module en commençant par l'arbre syntaxique pour faciliter le détail des modules suivants. Le module d'analyse fera l'objet d'un chapitre à part.

3-2-1 L'arbre syntaxique

a - structure

Par sa frèquence d'utilisation, l'arbre syntaxique doit se trouver en mèmoire principale, par sa grandeur, il doit se trouver en mèmoire secondaire. On l'a donc découpé en SECTIONS numèrotées de même taille, l'ensemble des SECTIONS se trouvant sur un fichier en mèmoire secondaire (FARBRE) et une seule SECTION étant prèsente à la fois en mèmoire principale.

Chaque SECTION se compose de:

. Une pile de rangement des règles de description (PREGLE) au fur et à mesure de leur choix au cours de l'analyse et son pointeur PPR.

. Une pile de rangement des vecteurs d'informations relatifs à chaque noeud des arborescences représentant les règles de description (PVECTEUR) et son pointeur PPV.

Pour une primitive, le vecteur comprend:

- un indicateur d'utilisation qui prend la valeur zéro ou la valeur du NIVEAU* de choix
- le numèro du BLOC de la primitive sélectionnée

* voir § 3-2-4 sur le problème des choix

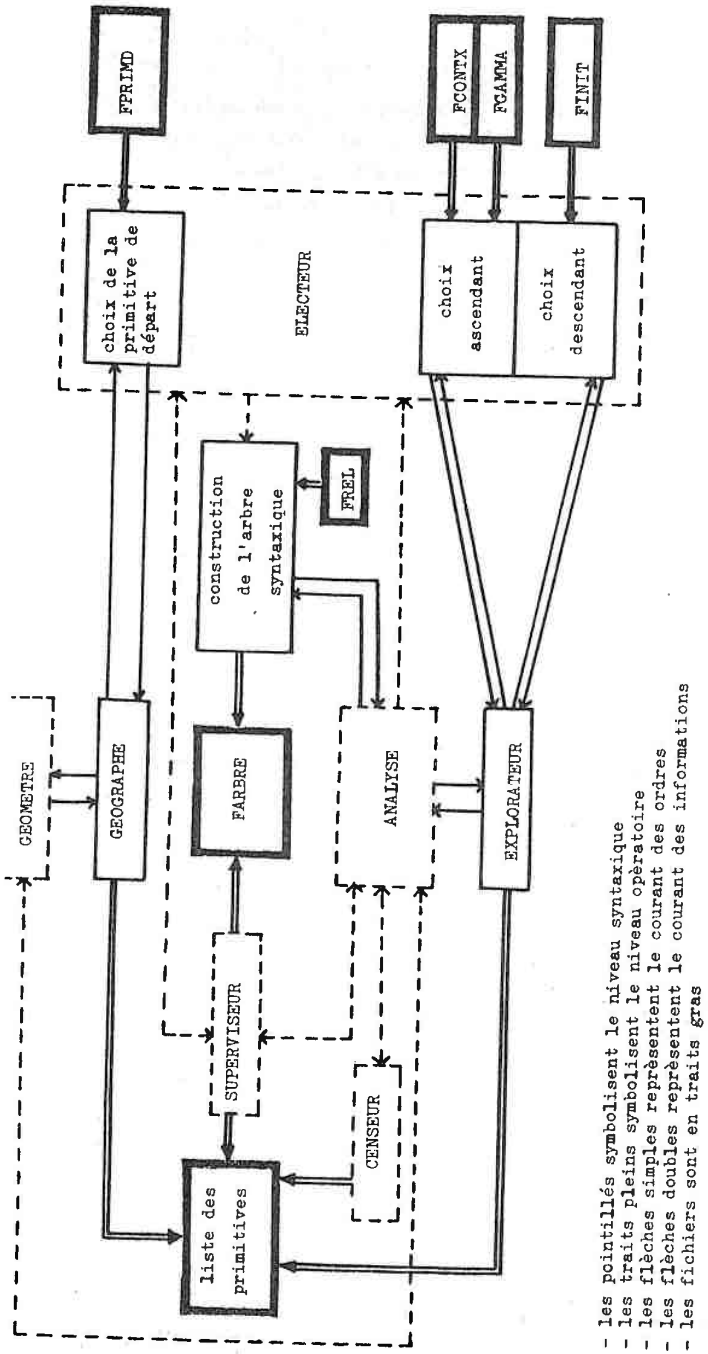


figure 3.10

Pour un nom de classe de forme non primitive ou un opérateur, le vecteur comprend:

- un indicateur d'utilisation fonctionnant comme le précédent
- un pointeur sur la liste des primitives qui composent la sous-arborescence dont ce noeud est racine, afin de pouvoir calculer à tout moment les coordonnées extrêmes de n'importe quelle sous-arborescence (notamment à la rencontre d'un opérateur topographique)
- un pointeur dans la pile PPOLE. Celle-ci donne les renseignements suivants:
  - nature de l'origine et de l'extrémité (i.e. numéro de BLOC et numéro de pôle) de la sous-arborescence dont le noeud est racine
  - si l'étiquette du noeud est une classe de forme non primitive, un pointeur sur sa règle de description dans PREGLE (ce pointeur se décompose en numéro de SECTION et adresse dans la SECTION)

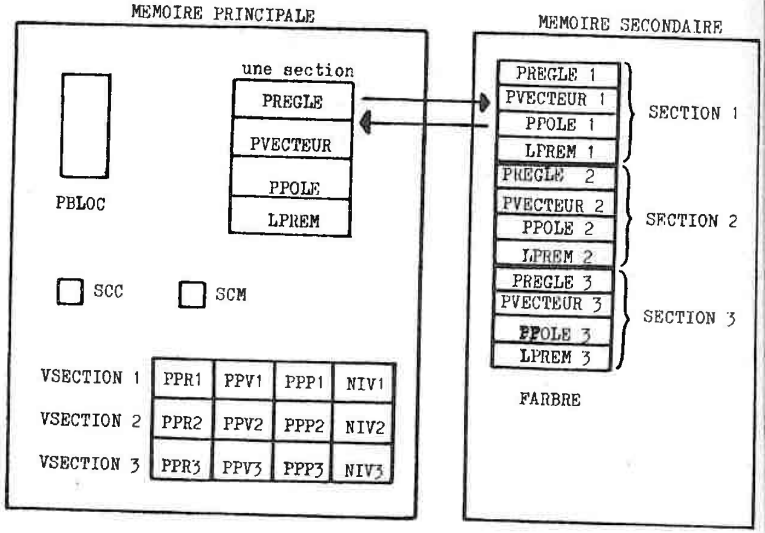
On ne peut représenter directement une (sous-)arborescence par les coordonnées de ses pôles, car celles-ci peuvent être modifiées en cours d'analyse par prolongement ou segmentation d'une ou plusieurs primitives qui la composent, d'une part à cause des limites de l'ECHANTILLONNEUR (voir § 3-1-3 a), d'autre part à cause du bruit qui peut entacher le dessin à reconnaitre.

. La liste LPREM de la première ligne occupée par chaque arbre d'une règle de description dans PREGLE

En outre, l'état de chaque SECTION est représenté en mémoire principale par un vecteur VSECTION qui donne les valeurs de PPR, PPV et PPP (pointeur de PPOLE) correspondantes, ainsi que la valeur du plus grand niveau de choix auquel il est fait référence.

On dispose également de la pile des numéros de BLOC des primitives utilisées pour l'analyse (PBLOC et son pointeur PPB), ainsi que du numéro de la SECTION courante (SCC) et celui de la SECTION en mémoire (SCM).

L'arbre syntaxique complet donne le schèma suivant:



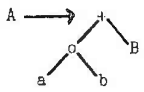
b - <u>construction</u>

Le rangement d'une matrice d'enchainement représentant une règle de description (cf § 3-1-2 a) dans la pile PREGLE s'effectue en deux temps: séparation de la matrice et de son en-tête qui est conservée en mèmoire principale pour être utilisée par le module d'analyse, puis adjonction d'un nouvel en-tête qui contient les renseignements suivants (figure ci-contre): numèro de la règle de description correspondante (A), pointeur (décomposé en numèro de SECTION et adresse dans la SECTION) sur le nom de classe de forme non primitive qu'elle définit (B), valeur du pointeur de la pile PBLOC lors du rangement dans la pile PREGLE (C) puis lorsque la matrice sera complètement utilisée par l'analyse (D), idem pour la valeur du NIVEAU de choix (E) et (F), enfin le code du nom de classe de forme non primitive dont la matrice donne la définition (G).

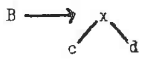
A	B
C	D
E	F
	G

En outre, il faut actualiser les liens horizontaux de la matrice en leur ajoutant une valeur de décalage calculée d'après celle du pointeur de la pile PPR, puis mettre à jour le vecteur d'informations VSECTION correspondant et éventuellement recopier la SECTION courante en mèmoire principale si elle ne s'y trouve pas. Aucune matrice ne peut se trouver à cheval sur deux SECTIONS.

<u>EXEMPLE:</u>



règle n° 90



règle n° 91

avec A,B∈N et a,b,c,d∈P

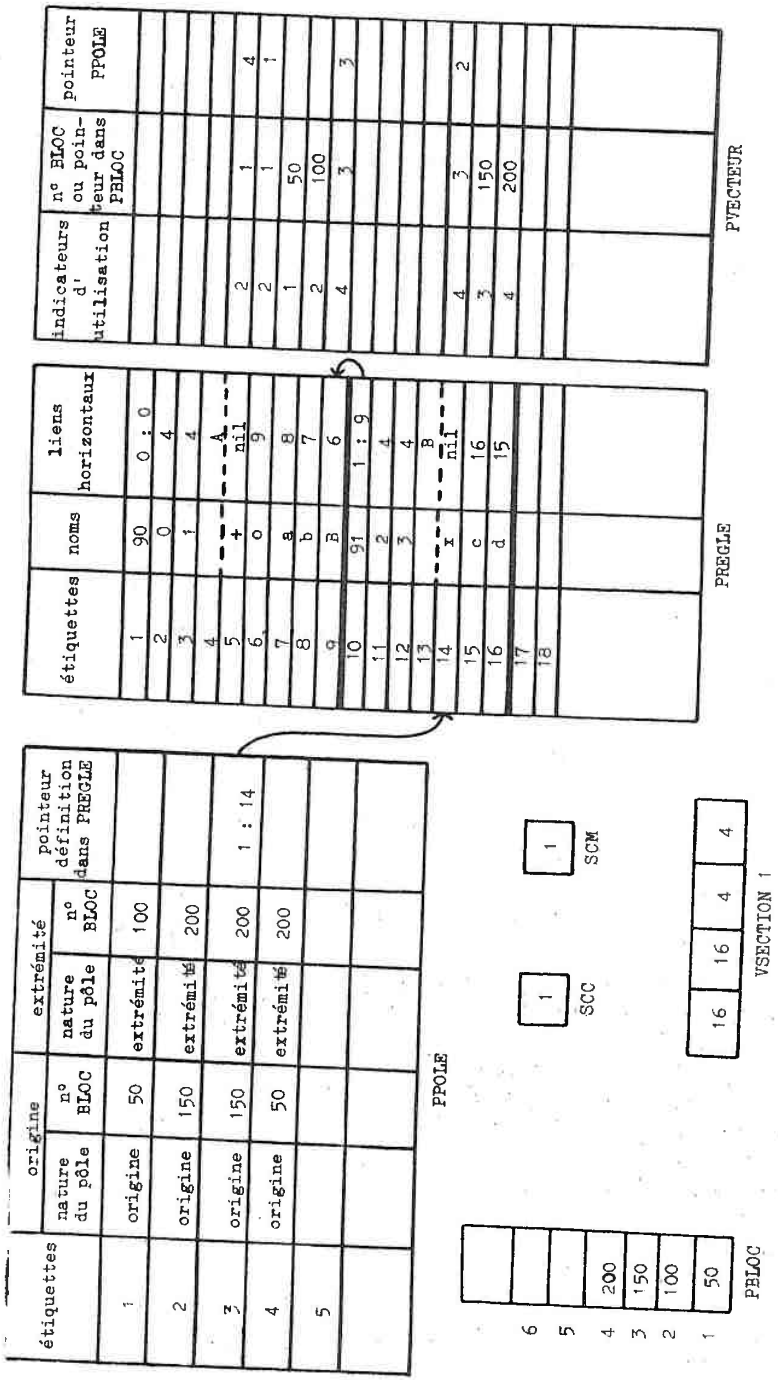
Si les numèros des BLOCS trouvés pour a,b,c,d sont respectivement 50, 100, 150, 200, après analyse on obtient l'état de la page suivante, à condition d'avoir commencé l'analyse avec la primitive de type a.

Cette représentation permet de parcourir l'arborescence dans tous les sens à partir de n'importe quel noeud (grâce aux pointeurs figurant dans les en-tête, dans PPOLE et grâce aux liens horizontaux). Le travail de l'INTERPRETEUR n'en sera que plus aisé. Elle permet en outre de minimiser le temps de calcul lors des retour-arrière car la représentation est fidèle à son évolution dans le temps du fait de sa structure de pile. Enfin, les informations qu'elle donne sont assez générales pour qu'il n'y ait aucune modification à y apporter en cas de segmentation ou de prolongation de primitives: il suffit de disposer d'une fonction donnant le numèro de BLOC correct à partir d'un numèro de BLOC invalidé (cf § 3-1-3 b).

3-2-2 <u>L'EXPLORATEUR</u>

a - <u>le problème d'une forme bruitée</u>

Une forme réelle (dans notre cas un dessin) est un ensemble complexe qu'il n'est pas facile de saisir sans le dégrader. Le système de saisie introduit des parasites (ou bruits) qui se traduisent par la disparition,



l'apparition ou la distorsion de primitives. Dans certains cas particuliers ou extrêmes, il peut y avoir mauvaise décomposition en primitives. Il faut encore tenir compte d'une maladresse possible de l'opérateur. Le problème est donc de rétablir dans sa conception originelle la forme bruitée ou plutôt d'essayer de trouver parmi toutes les corrections possibles celles qui rétabliront la forme d'origine avec le plus de vraisemblance .

Bien que le procédé existe (13), nous ne tiendrons pas compte ici du processus contraire c'est-à-dire des erreurs introduites par un système de description incorrect, où il y aurait remplacements, omissions ou adjonctions de symboles. Nous supposerons que le système de description coincide parfaitement avec les dessins fournis.

Il y a deux façons de prendre en compte les erreurs: au niveau syntaxique ou au niveau de la forme elle-même.

Au niveau syntaxique, un moyen de pallier les erreurs d'omission est d'utiliser des grammaires génèrant des langages contenant ces omissions. On introduit de nouvelles règles de description qui donnent toutes les unités syntaxiques possibles avec omissions. Cette méthode alourdit considèrablement la grammaire pour un seul type d'erreur possible et donne un langage indéterministe s'il ne l'était pas au départ (14). Il existe aussi des grammaires stochastiques, proposées par FU (15). Les règles de description contiennent toutes les possibilités d'erreurs. A chaque règle est associé un coefficient de probabilité, les règles introduisant des erreurs ayant les coefficients les plus faibles. Cela permet d'attribuer une probabilité à chaque forme reconnue, qui est écartée pour une valeur trop basse. Outre le fait qu'il est pratiquement impossible, vu la grande variété des erreurs, d'engendrer de telles grammaires, le principal défaut de ces deux méthodes est de nécessiter la mise en oeuvre d'algorithmes généraux complexes et coûteux en temps de calcul. Pour l'application en temps réel qui est la nôtre, on ne retiendra donc pas ce genre de solutions.

Nous nous cantonnerons dans des corrections au niveau de la forme: trouver une forme déduite de la forme proposée par corrections (i.e. adjonctions, modifications ou mises à l'écart de primitives) et affectée d'un coefficient de vraisemblance; si celui-ci est trop faible, essayer de trouver une autre forme déduite avec un meilleur coefficient.

Le modèle proposé est le suivant (8): la probabilité d'avoir une forme f' alors qu'on observe une forme f est le produit des probabilités affectées à chacune des opèrations de la transformation de f en f'. Par exemple, à une substitution d'une primitive a en une primitive b , on associera la probabilité pour que a soit détectée alors que b devrait l'être. Ceci nous amène au calcul d'un score de confiance par moyenne géomètrique:

$$S = \sqrt[k]{\prod_{i=1}^{n} P_i}$$

k = nombre de primitives de f'

n = nombre de corrections

P_i = probabilité associée à chaque correction

En pratique, pour des raisons de simplicité et de rapidité de calcul, on retiendra l'équivalence d'une moyenne arithmètique:

$$S = \frac{k_1 A + k_2 P}{k_3 N}$$

A = nombre de primitives ajoutées

P = nombre de primitives prolongées

N = nombre de primitives de f'

k_1, k_2, k_3 = constantes

Le calcul de la valeur de ce score de confiance est assuré par un module appelé DESSINATEUR qui peut être consulté à tout moment, en particulier avant l'exécution d'une correction. On évite ainsi d'effectuer des corrections entrainant un abaissement du score de confiance vers une valeur trop faible, et donc menant vraisemblablement à une forme f' à écarter.

Les corrections sont effectuées par l'EXPLORATEUR, indépendamment du module d'analyse mais en accord avec le DESSINATEUR, toujours dans l'optique de faire coincider la structure réelle du dessin avec la structure indiquée par l'ANALYSEUR.

b - principe

L'EXPLORATEUR est chargé de trouver une primitive d'une classe donnée à partir d'un point précis du plan dans lequel s'inscrit le dessin à reconnaitre. Il est commandé par le module d'analyse. A cause du bruit qui peut entacher le dessin, il comporte plusieurs phases interdépendantes sur un modèle similaire à l'algorithme employé par Y. SHIRAI pour la reconnaissance de polyèdres (16).

Il convient également de distinguer différents cas, selon que la primitive cherchée est un segment de droite ou un arc de cercle, car si le découpage d'un segment de droite en deux tronçons donne deux segments du même type, il n'en est pas de même pour un arc de cercle puisque sa classe est définie par la direction du vecteur origine-extrémité, nonobstant la courbure (fig. 3.11). Idem pour la primitive à laquelle appartient le point de départ de la recherche.

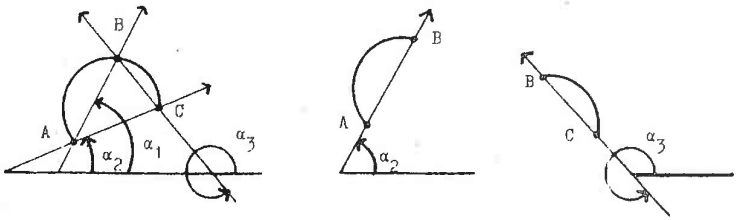
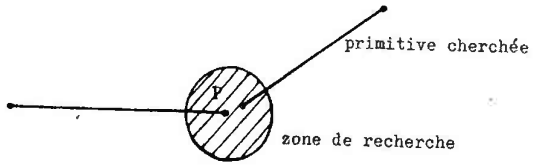


figure 3.11: découpage d'un arc de cercle en deux tronçons et arcs résultants

* La primitive cherchée est un segment de droite
Le point de départ appartient à un segment de droite
(dèsormais ce dernier sera symbolisé par la lettre P)

phase 1: la recherche circulaire

Le relevé des coordonnées effectué par la tablette graphique étant très précis (5 unités par millimètre), après un lever de crayon, l'opérateur croira reprendre le tracé au même endroit de la tablette alors que les coordonnées transmises seront sensiblement différentes. On recherche donc un pôle de la primitive demandée dans une zone de faible rayon autour du point P.



phase 2: la segmentation

Le tracé du dessin sur la tablette ne correspond pas toujours à sa description dans le système (cf § 3-1-3 a dont nous reproduisons le schèma fig. 3.12). Ainsi par exemple, il faut être capable de trouver le segment BD à partir du segment AC puis éliminer le tronçon AB ou BC en trop. Le cas est résolu en définissant un faisceau parallèle à la direction de la primitive cherchée et centré sur la primitive p à laquelle appartient le point P.

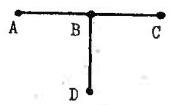
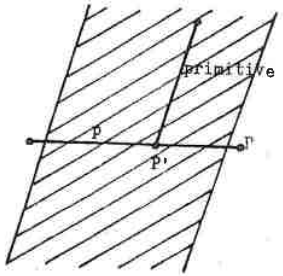


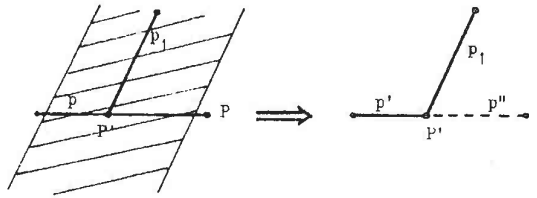
figure 3.12



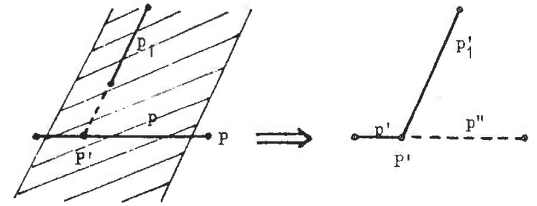
Toutes les primitives détectées dans cette zone sont classées en fonction de la distance sèparant le pôle cherché de la primitive p. Elles pourront être essayées l'une après l'autre lors d'éventuels retour-arrière.

On est amené à effectuer trois types de corrections suivant la disposition relative des primitives en cause et l'importance du bruit ayant pu déformer le tracé:

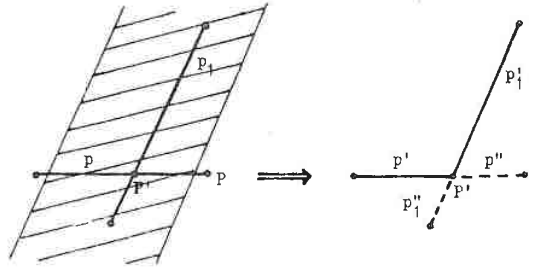
1 - segmentation de la primitive de départ uniquement:



2 - segmentation de la primitive de départ et prolongation de la primitive cherchée:



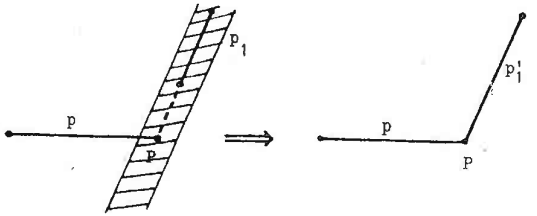
3 - segmentation de la primitive de départ et de la primitive cherchée:



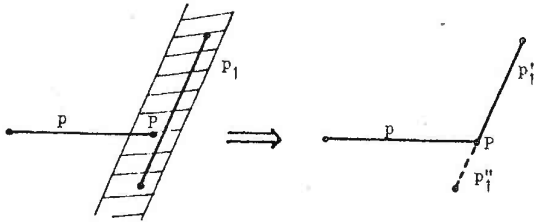
phase 3: la recherche en faisceau

Elle est conçue pour corriger les anomalies provoquées par l'introduction de bruits dans le dessin. La zone de recherche circulaire est étendue à un faisceau parallèle à la direction de la primitive cherchée, de même largeur. Le procédé est le même que pour la phase précèdente. Deux types de corrections peuvent être nécessaires (la suppression d'un morceau de longueur inférieure au diamètre de la zone de recherche circulaire n'étant pas considérée comme significative):

1 - prolongation de la primitive cherchée:



2 - segmentation de la primitive cherchée:



phase 4: la prolongation par tronçons

Son but est analogue à celui de la recherche en faisceau. Si aucune des phases précèdentes ne donne de résultat, la primitive p est prolongée d'une longueur fixée (égale au diamètre de la zone de recherche circulaire), puis on essaie de nouveau une recherche circulaire et une recherche en faisceau à partir du nouveau point de départ obtenu. Le processus est réitèré jusqu'à trouver une primitive ou abandonner la recherche à cause d'un nombre trop important de prolongations ou parce-qu'il y a débordement du plan d'analyse.

EXEMPLE: soit la figure 3.14, image bruitée de la figure 3.13:

figure 3.13

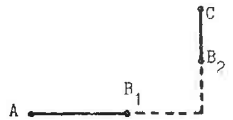


figure 3.14

A partir de AB_1, deux prolongations par tronçons puis une recherche en faisceau permettent de trouver le segment B_2C (fig. 3.15). Les deux primitives sont ensuite corrigées et rétablies dans leur forme originelle.

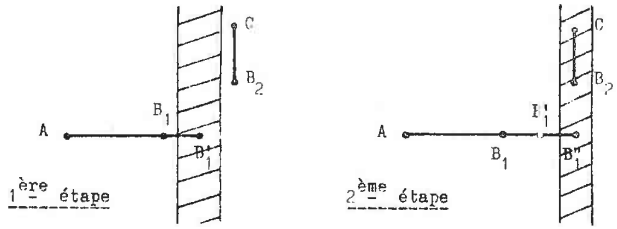


figure 3.15

phase 5: adjonction

Si aucune des phases précèdentes ne donne de résultat, on invente une primitive de la classe cherchée que l'on place au point de départ d'origine. Sa longueur est calculée proportionnellement à la taille globale de la sous-forme en cours d'analyse. De toute façon, elle sera corrigée ultérieurement par l'une ou l'autre phase. Bien entendu, cette correction n'est effectuée qu'à condition que le DESSINATEUR le permette.

Supposons que:

- PX,PY désignent les coordonnées du point de départ
- PX+PXT,PY+PYT représentent les nouvelles coordonnées après une prolongation par tronçons
- boolseg = vrai indique que la phase de segmentation doit être effectuée

On obtient l'algorithme suivant:

```
fonction booléen EXPLORATEUR (boolseg, PX, PY):

        booléen fonction DESSINATEUR, RECHERCHE CIRCULAIRE,
                         SEGMENTATION, RECHERCHE EN FAISCEAU ;
        booléen boolseg ;
        EXPLORATEUR := faux ;

        si RECHERCHE CIRCULAIRE alors EXPLORATEUR=vrai ; retour fsi
        si boolseg alors si SEGMENTATION alors EXPLORATEUR=vrai ;
                                               retour fsi fsi
        si RECHERCHE EN FAISCEAU alors EXPLORATEUR=vrai ; retour fsi
        EXPLORATEUR = EXPLORATEUR (faux, PX+PXT, PY+PYT) ;
        si DESSINATEUR alors ADJONCTION ; EXPLORATEUR=vrai ;
                             retour fsi
        retour ;

fin fonction
```

L'appel initial est: EXPLORATEUR (vrai, PX_0, PY_0) ;

Au niveau des BLOCS, l'utilisation d'une primitive se traduit par l'affectation de la valeur du NIVEAU de choix en cours à l'indicateur d'utilisation et au(x) drapeau(x) de modifications selon que la primitive a été connectée par son origine, son extrémité ou les deux en même temps. Ainsi, il ne sera pas question de procèder à une modification sur une primitive par un pôle donné si ce dernier est commun avec une autre primitive. Par exemple, sur la figure 3.16, on ne peut prolonger ou segmenter la primitive p_1 en B si elle est déjà connectée à p_2. Par contre, cela est possible en A, ou en B si la connection avec p_2 n'est pas encore établie.

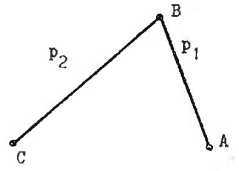


figure 3.16

Pour une modification de primitive, les opèrations sont plus complexes: création de un ou deux nouveaux BLOCS et chainage avec le BLOC d'origine; mise à jour des indicateurs d'utilisation et des drapeaux de modification des nouveaux BLOCS; calcul des coordonnées des nouvelles primitives, rangement dans la pile POINT et mise à jour des pointeurs des nouveaux BLOCS; enfin mise à jour de l'indicateur de modification du BLOC d'origine. La figure 3.17 schèmatise le rèsultat d'une segmentation.

* La primitive chorchée est un arc de cercle
Le point de départ appartient à un segment de droite

Comme nous l'avons déjà expliqué auparavant, on ne peut procèder à des corrections sur un arc de cercle sans presque toujours modifier son appartenance à une classe donnée.

Prenons par exemple la figure 3.18 représentant l'arc AB de direction ω et de marge ε.

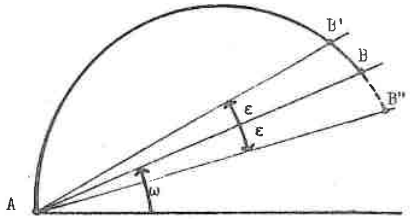


figure 3.18

On peut couper cet arc en n'importe quel point de B'B ou le prolonger jusqu'à n'importe quel point de BB" sans changer sa classe, à cause justement de la marge de tolèrance. En deçà de B' et au-delà de B", on obtient un (ou deux) arc(s) de classe(s) diffèrente(s) sans rapport avec la classe de départ. Encore n'est-on même pas certain d'avoir deux arcs. En effet, il ne faut pas oublier que le découpage en primitives a été effectué par l'ECHANTILLONNEUR d'après une liste de points significatifs, i.e. un nombre restreint de points. Ainsi pour l'arc de la figure 3.19, la segmentation

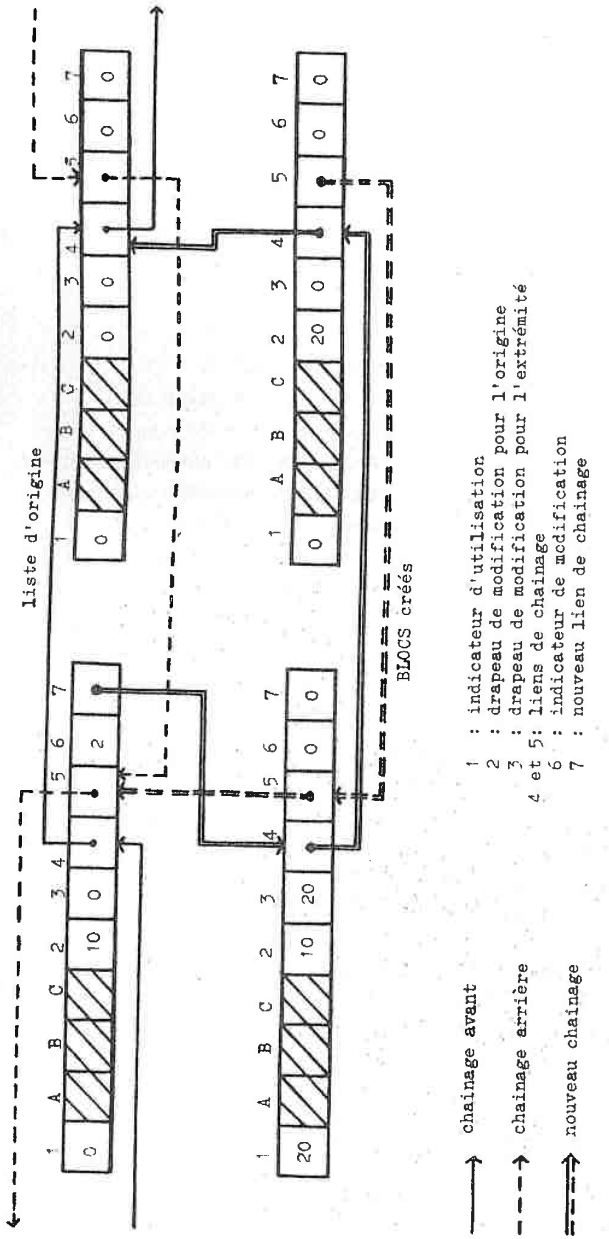


figure 3.17

au point C donnera certainement deux arcs si l'on réapplique l'algorithme de découpage en primitives à AC et CB. Par contre, on obtiendra presque certainement deux droites dans le cas de la figure 3.20, les angles φ_1 et φ_2 étant trop proches du seuil d'

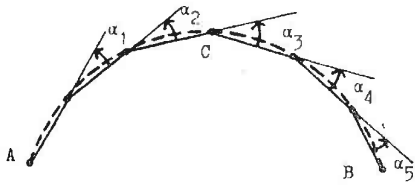


figure 3.19

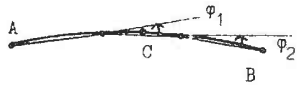


figure 3.20

échantillonnage. Il faut aussi tenir compte du tracé dans lequel s'inscrit la primitive: supposons qu'à la figure 3.21 le découpage obtenu soit l'arc AB et les deux droites BC et CD. En divisant l'arc au point B', l'ECHANTILLONNEUR donnerait-il un arc et trois

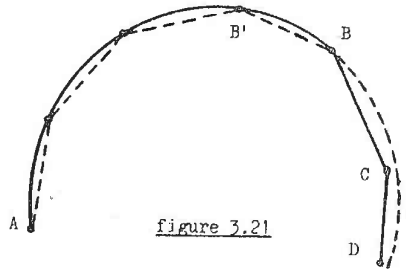


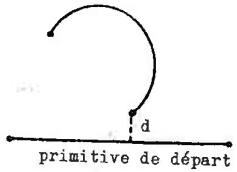
figure 3.21

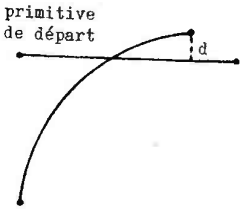
droites (AB',B'B,BC,CD) ou deux arcs (AB',B'D)? On pourrait ainsi introduire des variantes à l'infini. Les mêmes problèmes se posent en cas de prolongation, une fois admis que l'on sait rajouter des points à la liste de ceux qui figurent l'arc de telle manière que le résultat de l'ECHANTILLONNEUR soit encore un arc. Aussi dorènavant ne pourra-t-on pas appliquer toutes les phases comme elles étaient décrites pour le cas précédent.

Dans le cas qui nous intèresse, on se réduira à:

- une recherche circulaire avec une zone de recherche un peu plus importante pour résoudre les cas les plus évidents de "dérapage" ou de mauvais posers de crayon.

- une segmentation réduite au cas n° 1, à savoir segmentation de la primitive de départ uniquement. On tolèrera une distance d entre le segment et l'arc de l'ordre du diamètre de la zone de recherche circulaire (figures ci-dessous).





- une prolongation par tronçons, bien sûr, ne pose aucun problème.

Deux phases sont à écarter: la recherche en faisceau puisqu' elle nécessite la prolongation de la primitive cherchée (et encore ne serait-on même pas sûr de la la détecter effectivement dans le faisceau - figure 3.21) et l'adjonction, qui reste un cas plus délicat: on pourrait supposer détenir en mèmoire des modèles types d'arcs de chaque classe décrite par l'utilisateur. Mais notre système ne tenant pas compte des notions de longueur et de courbure, des tentatives d'ajustement risquent de perturber indéfiniment l'analyse.

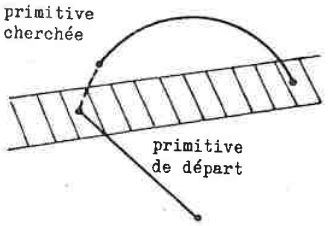


figure 3.21

* La primitive cherchée est une droite
Le point de départ appartient à un arc de cercle

Les phases provoquant une modification de la primitive de départ sont donc à écarter, c'est-à-dire la segmentation et la prolongation par tronçons. Il reste alors:

- la recherche circulaire (sur le modèle précèdent)

- la recherche en faisceau

- l'adjonction

* La primitive cherchée est un arc de cercle
Le point de départ appartient à un arc de cercle

Ne pouvant corriger aucune des deux primitives en prèsence, seule la recherche circulaire est permise.

3-2-3 Le GEOMETRE

Son rôle consiste à définir, à la rencontre d'une opèration de concatènation topographique, la région du plan dans laquelle va se dérouler la suite de l'analyse.

Pour cela, au fur et à mesure de l'analyse, chaque morceau du dessin analysé est défini par ses abscisses et ordonnées maxima (M_x,M_y) et minima (m_x,m_y). Pour calculer ces valeurs il suffit de se reporter, dans l'arbre syntaxique construit, à la racine de l'arborescence représentant la sous-forme qui nous intèresse. La seconde colonne de la pile PVECTEUR nous donne la tête de liste dans PBLOC des numèros de BLOCS composant la sous-forme. La queue de liste est donnée soit par le pointeur de pile PPB, soit par le quatrième mot de l'en-tête (cf figure 3.10). Cette gymnastique est nécessaire car une primitive peut être corrigée après avoir été utilisée par l'ANALYSEUR (dans le cas d'une segmentation par exemple): pour chaque BLOC invalidé donné par la liste, une fonction nous

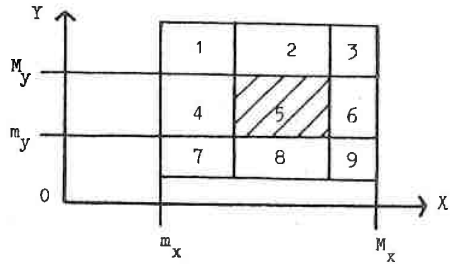
fournit le numèro de BLOC de la primitive corrigée (pour une segmentation il s'agit du BLOC immèdiatement suivant - cf figure 3.17).

Par exemple, sur le schèma de la page III-18, si on veut connaitre la liste des BLOCS composant la sous-forme de nom A, le quatrième mot de l'en-tête nous donne 4(ligne 2 dans PREGLE) et la seconde colonne de PVECTEUR (ligne 5) nous donne 1. La liste correspondante est donc 50, 100, 150 et 200.

Le plan d'analyse est divisé en dix-sept zones à partir des m_i et M_i (cf § 1-2). Un tableau de réfèrence donne la région correspondant à chaque relation pour son opèrande gauche et pour son opèrande droit. Par exemple, pour l'expression SUR(A,B), à partir de A on se reportera dans la région 12 pour trouver B, mais on cherchera A dans la région 10 en connaissant B. La figure 3.23 résume toutes les possibilités.

10 = 1 U 2 U 3
11 = 4 U 5 U 6
12 = 7 U 8 U 9
13 = 1 U 4 U 7
14 = 2 U 5 U 8
15 = 3 U 6 U 9
16 = 10 U 12 U 4 U 6
17 = 16 U 5

5 : région dans laquelle s'inscrit la sous-forme connue



	PSUR	PSOUS	PDEVANT	PCTREG	PCTRED
opèrande gauche	12	10	17	15	13
opèrande droit	2	8	5	4	6

	SUR	SOUS	INT	EXT	AGAUCHE	ADROITE	CGAUCHE	CDROIT
opèrande gauche	12	10	16	5	15	13	6	4
opèrande droit	10	12	5	16	13	15	4	6

figure 3.23

Les frontières de chaque région sont définies à un coefficient de tolèrance près, proportionnellement aux tailles de la zone d'analyse initiale et de la partie du dessin concernée. Un second tableau associe opèrateurs et coefficients.

Chaque nouvelle région calculée est sauvegardée dans la pile PZONE.

En résumé, le travail du GEOMETRE comprend trois étapes:
- calcul des coordonnées extrêmes de l'opèrande gauche ou droit de l'opèrateur topographique donné.
- calcul de la région du plan où se trouve l'autre opèrande.
- empilement dans PZONE des coordonnées de cette région.

3-2-4 Le SUPERVISEUR

a - le problème des choix

Comme nous l'avons déjà dit le dessin à analyser peut être imparfait du fait de l'introduction de bruits. Le rôle de l'EXPLORATEUR est d'essayer de rétablir le tracé initial. Il comporte plusieurs phases successives de recherche, chaque phase qui a abouti posant pour hypothèse que la primitive selectionnée est bien celle qui convient. Mais, dans une même phase, il peut exister plusieurs primitives satisfaisant aux conditions requises. Pour trouver la bonne, il n'y a guère d'autres solutions que d'en choisir une pour poursuivre l'analyse et, en cas d'échec, revenir sur ce choix jusqu'à aboutir.

Ce n'est pas la seule source de retour-arrière: si notre algorithme de base est thèoriquement déterministe (§ 2-2), dans la pratique il est difficile de construire une grammaire sans obtenir des noms de classes de formes non primitives ayant mêmes initiales ou mêmes contextes. Il existe donc encore une possibilité de choix, restreinte certes, au cours des phases ascendantes et descendantes de l'analyse. Il en est de même pour les primitives caractèristiques qui servent de départ au fonctionnement de l'algorithme: accorder le modèle thèorique avec la pratique n'est pas aisé; nous y reviendrons d'ailleurs.

La gestion des choix et des retour-arrière est confié au SUPERVISEUR.

b - les choix de la primitive de départ

Nous avons déjà parlé dans le paragraphe 2-3-4 de la notion de primitive caractèristique pour la détermination des contextes dans le cas d'une relation topographique: elle garantit la prèsence de la sous-forme dont elle est, prècisèment, la caractèristique. Cependant, pour amorcer l'analyse, on aimerait également disposer d'autres critères: identification certaine de la primitive et déterminisme pour la suite de l'analyse.

Le choix de la primitive doit être effectué avec certitude pour éviter que le score de confiance ne se dégrade rapidement, provoquant un grand nombre de retour-arrière. Son identification doit être certaine au point de vue appartenance à une classe, et on évitera donc de choisir des arcs de cercle dont le découpage par l'ECHANTILLONNEUR est plus souvent sujet à caution que celui des segments de droite. En outre, il faut que la recherche de cette primitive soit rapide: par définition les primitives caractèristiques appartiennent à des classes assez peu répandues pour que cette condition soit d'elle-même respectée.

Bien sûr, on veut surtout éviter les retour-arrière dus à des possibilités multiples de choix pour les phases descendantes et ascendantes de notre algorithme, et on va rechercher les primitives nous donnant un début d'analyse le plus déterministe possible. Prenons $A \in N \cup P$:

A est dit déterministe ascendant si la connaissance du contexte d'une sous-forme de nom A suffit à rendre déterministe la phase ascendante de la procèdure REMONTE.

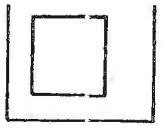
De même, A est dit déterministe descendant si la connaissance du contexte suffit à rendre déterministe une analyse descendante de A.

Ainsi, avoir un début d'analyse déterministe à partir d'une primitive p équivaut à construire une arborescence admettant au moins une feuille de nom p, telle que:

- Tout noeud situé sur le chemin de la racine à toute feuille de nom p soit déterministe ascendant.
- Tout noeud non situé sur ce chemin soit déterministe descendant.

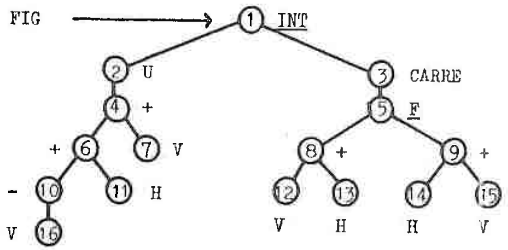
Par exemple, considèrons la forme et le système de description suivants:

(1)
FIG ⟶ INT (U,CARRE) /1/
U ⟶ -V + H + V /2/
CARRE ⟶ (V + H) F (H + V) /3/



V symbolise un segment de droite vertical
H symbolise un segment de droite horizontal

L'arborescence correspondante est:



Pour un noeud de type V, toute l'arborescence correspond aux critères précédents. Par contre, pour un noeud de type H, ce n'est pas le cas: on ne peut effectuer une analyse déterministe qu'à partir du noeud 13. Le dessin le montre bien puisque pour les segments verticaux tous les contextes sont différents:

Au contraire, pour les segments horizontaux, il y a deux contextes identiques pour deux règles différentes (2 et 3):

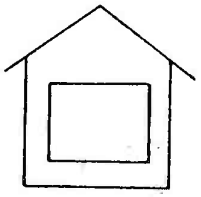
Dans ce cas la primitive de départ idéale est du type V.

Considèrons maintenant cette autre description possible:

(2)
FIG ⟶ INT (U,CARRE) /1/
CARRE ⟶ U F H /2/
U ⟶ -V + H + V /3/

Cette fois, tous les noeuds rèpondent aux critères demandés car s'il y a toujours deux contextes identiques pour un segment horizontal il n'y a qu'une seule règle possible: la numèro 3. La primitive de départ est alors indiffèrente.

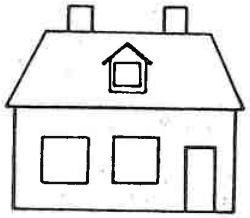
Complètons maintenant la description:



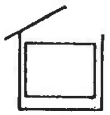
CHIEN-ASSIS ⟶ POSE SOUS (FIG,TOIT)
TOIT ⟶ OBLD + OBLG
FIG ⟶ ...

OBLD désigne un segment oblique droit
OBLG désigne un segment oblique gauche

Que l'on complète avec le système (1) ou (2), une analyse effectuée à partir d'une oblique reste déterministe. De plus, dans un environnement tel que ci-dessous, un segment de droite oblique est plus caractèristique



qu'une verticale ou qu'une horizontale, et sa recherche est moins coûteuse. Dans ce cas, on prendra comme primitive de départ de préfèrence une oblique, à défaut une horizontale ou une verticale car il ne faut pas oublier que la forme peut être parasitée et comporter des segments manquants ou déformés, de cette façon par exemple:



On ne peut choisir une oblique comme primitive de départ pour deux raisons: le contexte n'est pas assez riche et surtout, comme il faut ajouter un segment oblique pour pouvoir continuer l'analyse, le score de confiance prend une valeur en-dessous du seuil de tolèrance.

Pour toutes ces raisons, le LINGUISTE fournit pour chaque nom de classe de forme non primitive, la liste classée des primitives de départ possibles (fichier FPRIMD). Ces primitives doivent possèder les qualités suivantes:

- faciles à localiser
- identifiables avec certitude
- caractèristiques de la forme à reconnaitre
- rendre le début d'analyse le plus déterministe possible

Qualités qui ne sont pas toujours compatibles en pratique.

Les primitives caractèristiques sont utilisées pour amorcer l'analyse globale et, par un usage récursif de la procèdure REMONTE, pour commencer l'analyse de chaque partie du dessin non connexe (décrite par relation topographique).

c - choix des règles dans la phase ascendante

Le choix des règles s'effectue à travers deux filtrages: connaissance des contextes et application de la relation gamma (cf § 3-1-2 b).

Il s'agit d'abord de déterminer les voisins à l'origine et à l'extrémité de la partie du dessin déjà analysée et d'en déduire la règle de description à sélectionner. On compare le voisinage trouvé avec les voisinages possibles fournis par le LINGUISTE. Les règles à appliquer sont classées par ordre décroissant suivant un score qui mesure le degré de concordance contextes trouvés-contextes thèoriques. Le calcul de ce score (en pourcentage) s'effectue de la façon suivante:

$$S = 100 \frac{N2}{k_1\ N3 + N1} \quad \text{pour } N1 \neq 0 \text{ et } k_1 = \text{cte}$$

avec N1 = nombre total thèorique de primitives formant le voisinage
N2 = nombre de primitives appartenant au voisinage et prèsentes sur le dessin
N3 = nombre de primitives n'appartenant pas au voisinage mais prèsentes sur le dessin

Pour le cas des relations topographiques où les voisins à l'origine et à l'extrémité sont des ensembles vides, la formule est ramenée à:

$$S' = 50\left(\frac{1}{k_2\ N1 + 1} + \frac{N3}{N2}\right) \qquad \text{avec } k_2 = \text{cte}$$

avec N1 = nombre de primitives formant le voisinage sur le dessin

N2 = nombre de primitives caractèristiques prèsentes thèoriquement dans l'entourage

N3 = nombre de primitives caractèristiques effectivement prèsentes

De cette manière, on pénalise les primitives prèsentes dans un voisinage dont elles devraient être absentes. Il semble que $k_1 = 1$ et $k_2 = 1$ donnent de bons résultats. Par exemple, pour N1 = 3 et N1 = 4, voici les diffèrentes valeurs possibles de S:

N3 \ N2	0	1	2	3
0	0	33	66	100
1	0	25	50	75
2	0	20	40	60
3	0	16	33	50
4	0	14	28	43
5	0	12	25	37
6	0	11	22	33

N1 = 3

N3 \ N2	0	1	2	3	4
0	0	25	50	75	100
1	0	20	40	60	80
2	0	16	33	50	66
3	0	14	28	43	57
4	0	12	25	37	50
5	0	11	22	33	44
6	0	10	20	30	40

N1 = 4

Les valeurs des scores des contextes complets augmentent avec N1, alors que les scores des contextes incomplets diminuent

Bien sûr ne sont retenues pour ce classement que les règles utiles à la construction de la sous-arborescence visée: si l'on veut construire un sous-arbre ayant pour racine un nom de classe de formes A, on éliminera de la liste fournie par FCONTX toutes les règles $B \longrightarrow \varphi$, $\varphi \in V^*$ ne satisfaisant pas à la condition $A \Gamma B$, c'est-à-dire ne permettant pas d'atteindre A. Par exemple pour la grammaire du § 1-2, si une sous-forme CADRE est reconnue au cours de l'analyse de la sous-forme COMMANDES, les règles 4 et 7 sont écartées du choix dans la phase ascendante suivante

puisque les relations COMMANDES Γ CAISSE et COMMANDES Γ ECRAN ne sont pas vérifiées. Il n'existe aucun sous-arbre possible de racine COMMANDES contenant les noeuds étiquetés CAISSE ou ECRAN.

d - <u>choix des règles dans la phase descendante</u>

Au cours de la phase d'analyse syntaxique descendante, à la rencontre d'un nom de classe de formes non primitives A, l'ANALYSEUR doit choisir une règle de desription de A parmi celles proposées par le système de description. On dresse alors la liste des primitives formant le voisinage du point où doit se poursuivre l'analyse. Cette liste est comparée à celle donnant les initiales à l'origine ou à l'extrémité de A (fichier FINIT) suivant le sens gauche-droite ou droite-gauche de l'analyse descendante. Puisque thèoriquement notre grammaire est LL(1) et LR(1), cela devrait suffire pour en déduire <u>la</u> bonne règle à appliquer. Cependant d'une part parce-que la construction d'une telle grammaire n'est pas toujours aisée ni pratique, d'autre part à cause d'éventuels parasites, on est obligé d'attribuer un score à chaque règle possible, dont on dresse ensuite une liste par scores décroissants. Le score (en pourcentage) est donné par la formule suivante:

$$S = 100\ \frac{N1}{N2}$$

avec N1 = nombre de primitives des initiales prèsentes sur le dessin

N2 = nombre total de primitives formant les initiales

On ne tient pas compte des primitives prèsentes mais n'appartenant pas à la liste du fichier comme dans la phase ascendante, car la liste fournie dans ce cas était exhaustive alors qu'ici il ne s'agit que de faire coincider la liste du fichier avec un sous-ensemble des primitives détectées sur le dessin.

e - <u>le retour-arrière</u>

La gestion des retour-arrière est confiée au SUPERVISEUR. Il a deux fonctions: mèmorisation des informations nécessaires pour effectuer un retour-arrière et exécution proprement dite de l'opèration.

Chaque fois que l'ELECTEUR procède à un choix, il transmet au SUPERVISEUR la liste classée correspondante (pour le choix de la primitive de départ, le choix d'une règle dans la phase descendante ou ascendante) ou bien l'étape de l'EXPLORATEUR qui a donné un résultat. Le SUPERVISEUR y associe toutes les informations susceptibles d'être modifiées du fait de ce choix et vitales pour le fonctionnement de l'analyse: en particulier l'état de toutes les piles du système (PRIM, POINT, PBLOC...), l'état de l'arbre syntaxique (les vecteurs VSECTION) et l'état d'avancement de l'analyse. Chacun de ces ensembles spécifiques est nommé un NIVEAU. Il est repèré par un indicateur de NIVEAU incrèmenté de un en un à partir de zéro et qui donne également la nature du choix correspondant.

Lorsque l'analyse aboutit à un échec (impossibilité de trouver ou d'ajouter la primitive demandée), le SUPERVISEUR rétablit un NIVEAU prècédent, passe au choix suivant grâce à la liste correspondante et relance l'analyse sur ces nouvelles bases.

Toutefois, il faut procèder à quelques mises à jour indispensables dans la liste des BLOCS et dans l'arbre syntaxique:

- Mise à zéro des indicateurs d'utilisation et des drapeaux de modification dont la valeur est supèrieure au NIVEAU de retour.
- Même chose pour les noeuds de chaque SECTION de l'arbre syntaxique dont le vecteur d'état VSECTION donne une valeur supèrieure au NIVEAU de retour. A l'intèrieur d'une même SECTION on ne met à jour que les matrices d'enchainement dont l'en-tête (cf § 3-2-1 b) porte une valeur supèrieure au NIVEAU concerné.
- Remise à zéro de l'indicateur de modification des primitives qui ne sont plus invalidées du fait du retour-arrière.

Le retour-arrière est dirigé c'est-à-dire que ce n'est pas toujours le NIVEAU immédiatement prècédent qui est régénéré, mais ceux qui ont le score le plus élevé en priorité, sans toutefois éliminer les scores faibles, toujours à cause des parasites. Ainsi, supposons que deux retours possibles donnent des scores de 10 et 80. Le second sera essayé d'abord.

3-2-5 Fonctions diverses

a - le GEOGRAPHE

Son rôle est analogue à celui de l'EXPLORATEUR: il consiste à chercher une primitive donnée dans une zone du plan donné (au début de l'analyse par exemple) et non plus à partir d'un point donné. Il n'y a donc aucune difficulté due aux parasites si ce n'est une déformation possible de la primitive, problème qui sera résolu ultèrieurement dans l'analyse.

b - le CENSEUR

Lorsque l'analyse est correctement terminée, la figure reconnue n'est acceptée que si le nombre de primitives utilisées est supèrieur à une fraction du nombre total de primitives fournies (soit 97 pour cent, ce qui correspond au taux le plus élevé de parasites observé jusqu'à prèsent). De cette manière, on peut éliminer certains cas où l'arbre syntaxique final ne correspond pas avec le dessin:

- Par un jeu d'adjonctions et de prolongations de primitives, il y a coincidence entre un fragment du dessin et le système de description.
- Par un mauvais choix de règles, une partie du dessin n'est pas utilisée.

3-3 L'ANALYSEUR

Au contraire des techniques d'analyse syntaxique classiques, notre algorithme doit être capable de fonctionner en démarrant n'importe où dans la phrase (au sens large) à analyser. De plus, un algorithme classique traite des informations à une dimension: le passage d'un élément (primitive) de la phrase à un autre est d'autant plus aisé qu'ils ne sont liés que par une relation de concatènation. Dans notre cas, la relation entre deux primitives est plus complexe car elle dépend de la suite de relations (topographiques ou non) qui les lient.

Aussi le recours à des méthodes d'analyse syntaxique descendante connues (17) est-il sinon impossible du moins difficile et surtout inapproprié. Le démarquage analyse descendante gauche-droite et analyse descendante droite-gauche de la procèdure REMONTE (chapitre II) n'a plus de raison d'être. Elles seront mêlées dans le procédé pratique pour un résultat et une démarche strictement équivalents avec, en plus, un gain appréciable en simplicité, clarté et rapidité d'exécution.

3-3-1 Notations

Nous avons déjà expliqué que les règles de définition étaient représentées par des structures arborescentes binaires. Par exemple B ⟶ (a o b) + (c x d) donne l'arbre suivant:

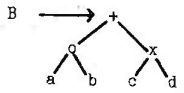


figure 3.24

Le noeud étiqueté par + est la racine de l'arborescence.

B est appelé le nom de l'arborescence ($B \in N$).

Par abus de langage, seront confondus dans la suite le noeud et son étiquette.

On dit que a est le frère gauche de b: $a = \mathcal{F}_g(b)$. De même, b est le frère droit de a: $b = \mathcal{F}_d(a)$. Plus généralement, si deux noeuds sont frères, on notera $a = \mathcal{F}(b)$ ou $b = \mathcal{F}(a)$. Sur le dessin, il y a également $o = \mathcal{F}(x)$ et $c = \mathcal{F}(d)$.

On dit aussi que x est le père de c et le père de d: $x = \mathcal{P}(c)$ et $x = \mathcal{P}(d)$. Egalement $+ = \mathcal{P}(o)$, $+ = \mathcal{P}(x)$ etc... Par analogie, o est le fils droit de + et x son fils gauche: $o = f_d(+)$ et $x = f_g(+)$. Enfin, le calcul de l'adresse d'un noeud n dans la représentation mèmoire de l'arborescence s'effectue à l'aide de la fonction $\mathcal{A}d(n)$.

3-3-2 Cheminement dans une arborescence

A partir d'un noeud quelconque d'une arborescence donnée, il faut être capable de trouver le noeud suivant à analyser ainsi que l'opèrateur qui les relie, puis réitèrer le processus jusqu'à exploration complète de l'arbre.

Intèressons-nous pour l'instant aux arborescences dont tous les noeuds sont des primitives et où ne figure aucune relation topographique. Reprenons la figure 3.24 et considèrons les sous-arbres A1 et A2 de racines o et x. Supposons que la sous-forme représentée par A1 soit analysée: comment poursuivre? L'opèrateur + nous apprend que l'extrémité de A1 se superpose

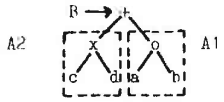


figure 3.25

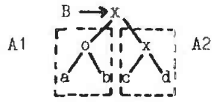


figure 3.26

à l'origine de A2: il faut donc trouver le noeud portant l'origine de la sous-forme décrite par A2. Inversons maintenant A1 et A2 (figure 3.25): cette fois, c'est l'origine de A1 qui se superpose avec l'extrémité de A2. Il faut chercher le noeud représentant l'extrémité de A2. Si l'on change + en x, il faut encore trouver l'extrémité de A2. Dèsormais (figure 3.24), on dira que + est le noeud directeur, A1 le sous-arbre source et A2 le sous-arbre arrivée. On constate que le pôle de connection du sous-arbre arrivée dépend du noeud directeur et du sens du lien horizontal unissant le sous-arbre source au sous-arbre arrivée. On peut construire le tableau donnant ce pôle en fonction du noeud directeur et du lien horizontal (un opèrateur topographique ne pouvant pas être noeud directeur comme on le verra par la suite, ni l'opèrateur moins puisqu'il est unaire):

lien horizontal \ noeud directeur	+	x	o	F
droit	∅	E	∅	∅ ou E
gauche	E	E	∅	∅ ou E

∅ pour origine
E pour extrémité

TRANS

Reste à trouver le moyen de calculer le noeud origine ou le noeud extrémité d'une arborescence quelconque en partant de sa racine. Nous avons vu que tous les opèrateurs employés étaient binaires: l'origine du résultat d'une opération est toujours portée par le terme gauche, l'extrémité par le terme droit (traduits en fils gauche et fils droit sur l'arborescence), à l'exception des opèrateurs topographiques où l'origine et l'extrémité sont toujours portées par le terme droit. Il suffit donc de partir de la racine et de toujours choisir la bonne transition en fonction du pôle cherché jusqu'à arriver à une feuille, sans oublier que l'opèrateur moins inverse les pôles donc les transitions. Dorènavant on appellera ce noeud le successeur (noté $\mathcal{G}$). L'algorithme s'écrit:

A désigne l'arborescence
d désigne le noeud directeur
lh désigne le sens du lien horizontal
TRANS désigne le tableau de la page précédente
P désigne l'ensemble des classes de primitives

```
procèdure SUCCESSEUR (A,d,lh) retourne (n) :
    tableau TRANS (2,4) ;
    pôle = TRANS (lh,d) ; n = racine (A) ;
    tant que n ∉P faire
        si n = '-' alors si pôle = origine alors pôle = extrémité
                                           sinon pôle = origine fsi
                  n = f(n) ;
              sinon si n = 'opèrateur topographique'
                    alors  n = f_g(n)
                    sinon si pôle = origine alors n = f_g(n)
                                           sinon n = f_d(n) fsi
                    fsi
              fsi
    fin tant que
fin procèdure
```

On trouvera page suivante (fig. 3.27) deux exemples de cheminement.

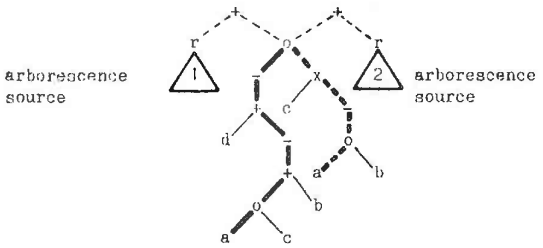


figure 3.27: deux exemples de cheminement:
traits pleins: cas 1 (noeud directeur + et lien horizontal droit)
pointillés: cas 2 (noeud directeur + et lien horizontal gauche)
dans les deux cas $\mathcal{G}(r) = a$

En supposant les pôles de l'arborescence source connus, il faut maintenant déterminer les pôles de connection: origine ou extrémité de l'arborescence source se superposant avec l'origine ou l'extrémité du successeur. Reprenons la figure 3.24. Le noeud directeur (+) nous apprend qu'il faut superposer l'extrémité de b avec l'origine de c: on notera $\mathcal{G}_e(o) = c_o$. Par contre, pour la figure 3.25, $\mathcal{G}_o(o) = d_e$, à cause de l'inversion du lien horizontal. Dans le cas de la figure 3.26, le changement du noeud directeur donne $\mathcal{G}_e(o) = d_e$. Introduisons des opèrateurs moins: pour la figure 3.28, $\mathcal{G}_e(o) = d_e$ alors que $\mathcal{G}_e(o) = d_o$ pour la figure 3.29.

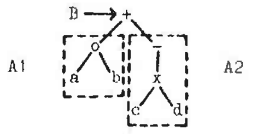


figure 3.28

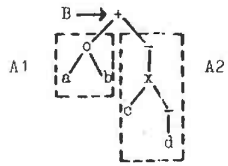


figure 3.29

On constate donc que les pôles de connection dépendent du noeud directeur et du sens du lien horizontal. Mais, en plus, le pôle de connection du successeur dépend de la parité du nombre de moins unaires prèsents dans le chemin de la racine au successeur.

On la nommera dèsormais parité d'inversion. Il suffit alors de réunir tous les cas possibles dans les deux tableaux ci-dessous.

lien horizontal \ noeud directeur	+	x	o	F	parité d'inversion
droit	∅	E	∅	∅ ou E	paire
	E	E	E	∅ ou E	impaire
gauche	E	E	∅	∅ ou E	paire
	∅	E	E	∅ ou E	impaire

tableau ARRIVEE: il donne la nature du pôle de connection du successeur

lien horizontal \ noeud directeur	+	x	o	F
droit	E	E	∅	∅ ou E
gauche	∅	E	∅	∅ ou E

tableau DEPART: il donne la nature du pôle de connection de l'arborescence source

EXEMPLE: considèrons la figure 3.30 et l'arborescence qui la reprèsente:

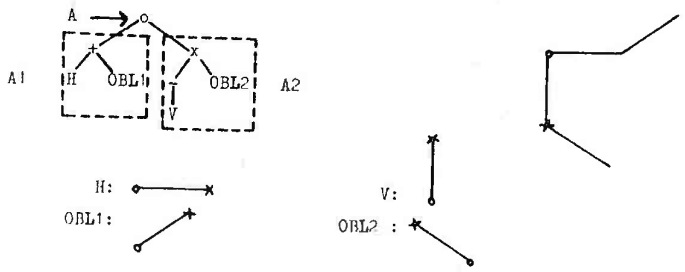


figure 3.30

1er cas: A1 est l'arborescence source.

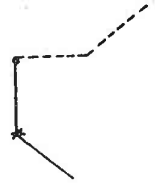
noeud directeur: o
lien horizontal: droit
successeur: V avec une parité d'inversion impaire
ARRIVEE (droit, o, impaire) = E
DEPART (droit, o) = ∅

Donc à l'origine de A1 (i.e. à l'origine de la primitive de type H), il faut rechercher l'extrémité d'une primitive de type V.

2ème cas: A2 est l'arborescence source.

noeud directeur: o
lien horizontal: gauche
successeur: H avec une parité d'inversion paire
ARRIVEE (gauche, o, paire) = ∅
DEPART (gauche, o) = ∅

Donc à l'origine de A2 (i.e. à l'extrémité de V), il faut rechercher l'origine d'une primitive de type H.



3-3-3 Algorithme d'analyse

a - principe général

En considèrant une arborescence à traiter à partir d'une feuille quelconque, il suffit de calculer son successeur et de réitèrer le processus jusqu'à atteindre la racine en décomposant au fur et à mesure en arborescences source et arrivée.

Deux cas peuvent se prèsenter:

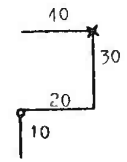
.La racine de l'arborescence arrivée n'est pas traitée: on calcule le successeur et on continue.

. La racine de l'arborescence arrivée est traitée: il y a en prèsence deux sous-arbres de même père complètement traités. On procède alors à l'enracinement et on poursuit avec l'arborescence résultante comme nouvelle arborescence source.

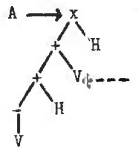
Le noeud à traiter étant trouvé, il faut le marquer c'est-à-dire lui attribuer ses caractèristiques dans l'arbre syntaxique (cf § 3-2-1): valeur du NIVEAU d'analyse, origine, extrémité et coordonnées extrêmes du sous-arbre dont il est racine (sous-arbre pouvant se rèduire à un noeud unique). Si le noeud est une feuille, donc un nom de forme primitive, tous ces renseignements sont fournis par l'EXPLORATEUR. Si le noeud est un opèrateur, le marquage rèsulte d'une opèration d'enracinement et les valeurs caractèristiques dépendent des deux sous-arbres enracinés. Aussi, tout au long de l'analyse, on empile l'origine et l'extrémité (i.e. numèro de BLOC et nature du pôle) de chaque sous-arborescence traitée. Lors d'un enracinement, on calcule l'origine et l'extrémité du sous-arbre résultant en dépilant deux fois et en tenant compte de la nature de l'opèrateur d'enracinement ainsi que du sens du lien horizontal. L'opèrateur moins étant unaire, son cas est particulier: il suffit de reporter les caractèristiques du noeud dont il est le père en inversant origine et extrémité.

Considèrons une arborescence A et une feuille quelconque n, l'algorithme s'écrit:

```
procèdure ANALYSE (A,n):
    tant que n ≠ racine(A) faire
        si 𝔖(n) = '-' ou 𝔉(n) est marqué
        alors n = 𝔖(n) sinon n = 𝔜(n) fsi ;
        MARQUER (n) ;
    fin tant que
fin procèdure
```

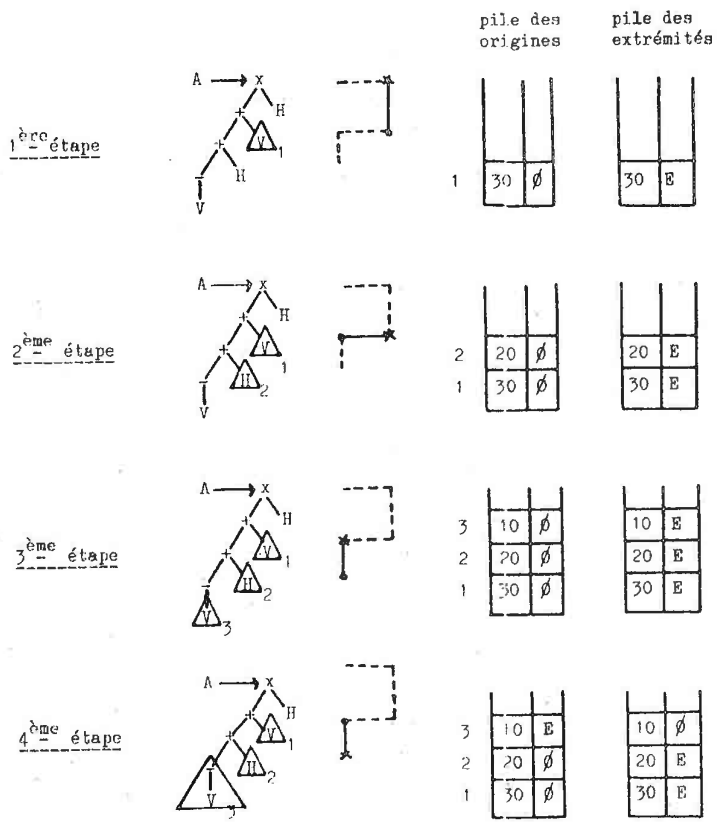
les numèros des BLOCS sont portés sur les primitives

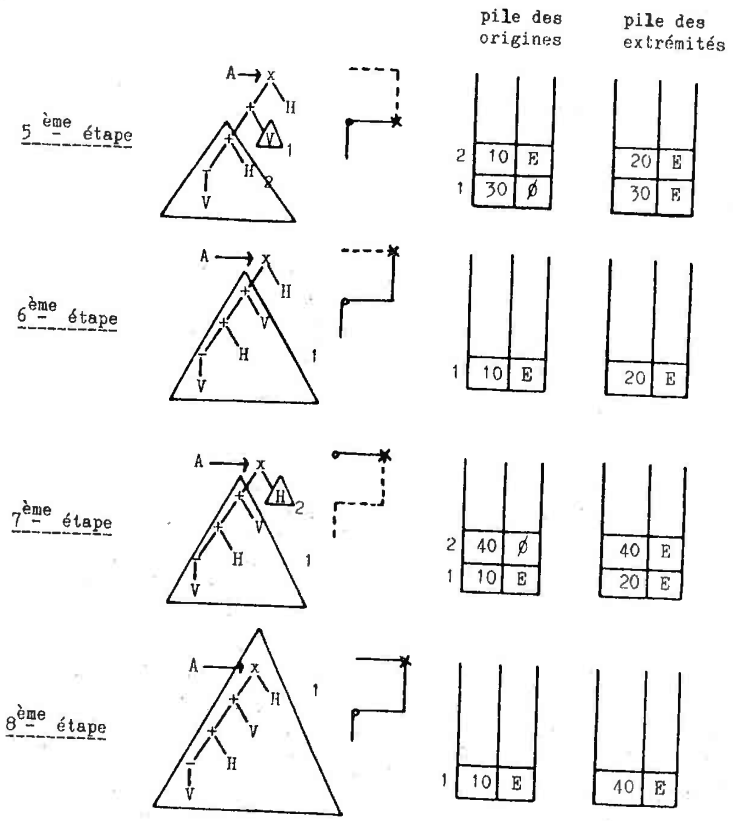


H désigne un segment horizontal
V désigne un segment vertical

figure 3.31

Par exemple, sur la figure 3.31, à partir du noeud désigné par une flèche, le dèroulement sera le suivant:





b - l'ANALYSEUR

Il ne reste plus qu'à appliquer ce principe à chaque règle (donc arborescence) choisie par l'ELECTEUR lors de la phase ascendante, en associant les modules décrits précédemment aux différentes étapes. Cependant les arborescences considérées jusqu'à présent ne comportent pas de noms de formes non primitives ni d'opérateurs de concatènation topographiques.

A la rencontre d'un nom de classe de forme non primitive, il suffit de remplacer son noeud, une fois le choix de l'ELECTEUR effectué, par la structure arborescente qui le définit, puis de poursuivre le traitement avec la nouvelle arborescence obtenue. Les valeurs contenues dans l'en-tête de description de la matrice d'enchainement correspondante (cf § 3-2-1 a) combinées avec celles déjà calculées permettent de trouver le plus rapidement possible le successeur et les pôles de connection.

Lorsque le noeud directeur est une relation topographique, on applique récursivement l'algorithme à l'arborescence arrivée après avoir calculé la zone du plan dans laquelle se poursuit l'analyse. Il ne faudra pas oublier, lors de l'opération d'enracinement par cet opérateur, de rétablir la zone d'analyse d'origine (en utilisant la pile PZONE - cf § 3-2-3).

Considèrons la grammaire G = (→,N,P,X).

Le texte de l'algorithme figure page suivante.

Les commentaires sont indiqués par /* ... */

La procèdure ELECTEUR effectue les choix; ses paramètres sont, dans l'ordre:

- nature du choix:
  0: choix de départ
  1: choix dans la phase ascendante
  2: choix dans la phase descendante
- nature de la racine de la sous-arborescence en construction
- résultat du choix

Le paramètre de la procèdure ANALYSE désigne la racine de l'arborescence que l'on veut construire.

fin désigne l'adresse de cette racine.

noeud désigne l'adresse du noeud en cours d'analyse.

MARQUER est la procèdure qui effectue les opèrations de marquage du noeud fourni en paramètre. Toutes les autres procèdures utilisées ont été décrites dans les paragraphes précédents de ce chapitre.

Le premier appel s'écrit: ANALYSE (X) ;

```
procèdure ANALYSE (Z) :

    /* choix de la primitive de départ p */
    ELECTEUR (0,Z,p) ; A := p ;

    tant que A ≠ Z faire

        /* choix d'une règle B → λ1Aλ2 d'après vs (A) et vs(-A) */
        ELECTEUR (1,Z,B) ; fin = 𝔉d(B) ; noeud = 𝔉d(A) ;

        /* traitement d'une arborescence */
        tant que fin ≠ noeud faire

            si 𝔉(noeud) est marqué ou 𝔓(noeud) = '-'

            alors /* enracinement */ noeud := 𝔉(noeud) ; MARQUER (noeud) ;
                 si noeud = 'relation topographique'
                 alors rétablir l'ancienne zone d'analyse fsi

            sinon si 𝔓(noeud) = 'relation topographique'

                 alors /* calcul de la nouvelle zone d'analyse */
                       GEOMETRE ; ANALYSE (𝔉(noeud)) ;

                 sinon m:= noeud ; noeud := 𝔊(noeud) ;

                       tant que noeud ∈ N faire
                       /* cas d'un nom de classe de forme non primitive */
                            ELECTEUR (2,Z,noeud) ; noeud := 𝔊(m) ;
                       fin tant que

                       si EXPLORATEUR (noeud) alors MARQUER (noeud)
                                              sinon SUPERVISEUR fsi

                 fsi

            fsi

        fin tant que

        A := B ;

    fin tant que

fin procèdure
```

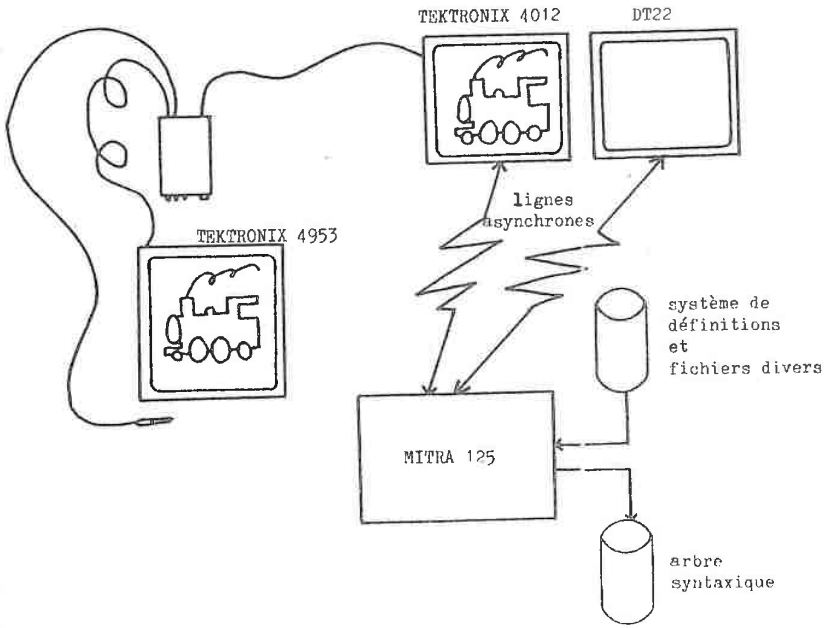
CHAPITRE IV

REALISATION PRATIQUE

Pour mettre en oeuvre notre algorithme d'analyse syntaxique, nous disposons d'une tablette graphique TEKTRONIX 4953 à principe magnètique reliée à un écran de visualisation TEKTRONIX 4012 lui-même connecté à un MITRA 125 par une ligne asynchrone à 9600 bauds. La partie conversationnelle est assurée par un écran alphanumèrique CII DT22.

Le LINGUISTE n'entre pas dans cette configuration car il a été mis au point et programmé à partir d'une console fonctionnant en temps partagé et reliée à l'IRIS 80 de l'Institut Universitaire de Calcul Automatique.

Le système de description ainsi que tous les fichiers dérivés (FCONTX, FINIT...) et l'arbre syntaxique se trouvent sur disque magnètique.



4-1 Restrictions de programmation

Le MITRA 125 ayant une capacité mèmoire réduite, il n'a pas été possible de programmer l'EXPLORATEUR avec tous les cas décrits dans le chapitre 3-2-2: nous nous sommes restreints à des figures uniquement composées de segments de droite, cet aspect n'étant pas fondamental dans le processus d'analyse syntaxique.

De plus, le LINGUISTE faisant l'objet d'une mise au point séparée, ses résultats ont été totalement simulés. Nous n'avons donc effectué les tests que sur un seul exemple car le calcul des informations produites par le LINGUISTE et leur saisie demandent une somme de travail très importante d'où il est pratiquement impossible d'éliminer les risques d'erreurs.

Nous avons essayé de choisir un système de description générant un dessin ni trop simple ni trop compliqué et contenant toutes les difficultés propres à une analyse de figure complexe, à savoir des règles récursives et des opérateurs (pseudo-)topographiques en nombre important. La forme générée est une maison, dont une représentation possible figure ci-dessous. La grammaire compte 19 règles. Elle est donnée page suivante.

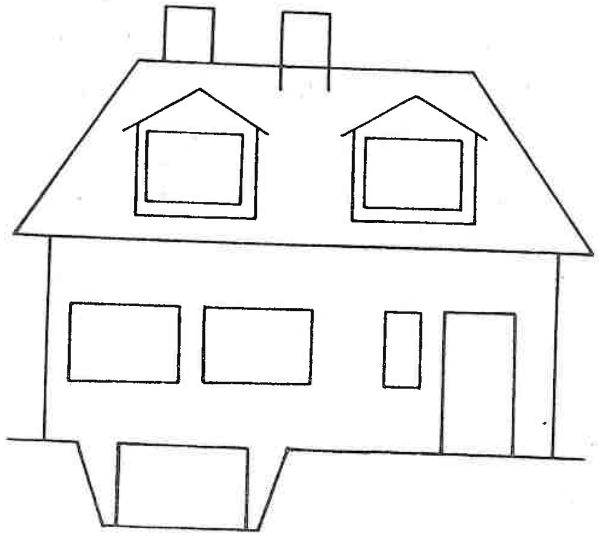


figure 4.1

MAISON ⟶ POSE SOUS (FACADE,TOIT) /1/

TOIT ⟶ INTERIEUR (SCHIEN,PANS) /2/

SCHIEN ⟶ CHIEN /3/ COTE DROIT (CHIEN,SCHIEN) /4/

CHIEN ⟶ INTERIEUR (FENETRE,AVANCEE) /5/

FENETRE ⟶ h F U /6/

U ⟶ -v + h + v /7/

AVANCEE ⟶ POSE SOUS (U,AUVENT) /8/

AUVENT ⟶ od + (-og) /9/

PANS ⟶ (od + FAITE + (-og)) F h /10/

FAITE ⟶ POSE SUR (CHEMINEE,FAITE1) /11/

FAITE1 ⟶ POSE DEVANT (CHEMINEE,h) /12/

CHEMINEE ⟶ v + h + (-v) /13/

FACADE ⟶ INTERIEUR (SFENETRE,MURS) /14/

SFENETRE ⟶ FENETRE /15/ COTE GAUCHE (FENETRE,SFENETRE) /16/

MURS ⟶ -v + h + (-og) + ENTREE + od + ENTREE + v /17/

ENTREE ⟶ POSE SUR (PORTE,h) /18/

PORTE ⟶ v + h + (-v) /19/

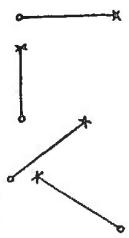
Les classes de primitives sont: h,v,od,og. Elles sont définies de la manière suivante:

h: T (0,5)

v: T (90,5)

od: T (45,40)

og: T (135,40)



Les expressions soulignées représentent les relations topographiques. Les noms de classes de formes non primitives sont écrits en majuscules.

4-2 Résultats

4-2-1 Premières conclusions

Nous avons d'abord programmé une première version de l'ANALYSEUR tel qu'il est décrit au chapitre III, moins les restrictions du paragraphe précédent. Nous nous sommes alors aperçus qu'il était impossible d'analyser correctement les parties du dessin décrites par des règles récursives (numéros 4 et 16).

En considèrant la suite de fenêtres de la figure 4.1, l'ANALYSEUR prenait en compte soit une fenêtre uniquement, soit deux fenêtres et une seule fois les trois fenêtres, suivant le point de départ de l'analyse. Cela s'explique de la manière suivante: numèrotons les fenêtres (fig. 4.2) et considèrons que la fenêtre n° 3 est reconnue. L'ANALYSEUR peut essayer

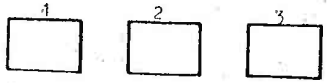
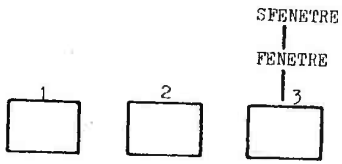


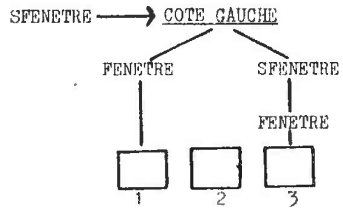
figure 4.2

d'appliquer les règles 15 ou 16. Aucune fenêtre ne se trouvant à droite de la troisième, la règle 16 est rejetée. L'arbre syntaxique obtenu est pour l'instant:



Le problème du choix se pose à nouveau, mais cette fois avec les règles 14 et 16. Comme elles contiennent toutes deux des opèrateurs topographiques, le calcul du score de leurs contextes va donner dans les deux cas 100 et l'ANALYSEUR va choisir indifféremment l'une ou l'autre. Si la règle 14 est choisie, la suite de fenêtres est assimilée à la seule fenêtre n° 3, première possibilité d'erreur. Supposons que ce ne soit pas le cas: la règle 16 comporte un opèrateur topographique, il faut une primitive de

départ pour poursuivre. Si le GEOGRAPHE donne une primitive appartenant à la fenêtre n° 1 et que l'analyse se dèroule correctement, on obtient l'arbre syntaxique suivant:



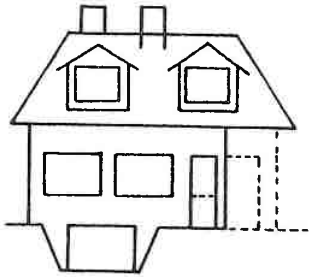
Mais alors la fenêtre n° 2 ne pourra plus jamais être prise en compte, car elle ne figure plus à gauche de la suite. Et quand bien même le GEOGRAPHE aurait donné une primitive de départ appartenant à la fenêtre n° 2, la suite entière ne serait analysée qu'à condition de ne pas retomber dans le premier type d'erreur décrit. Il en serait de même quel que soit la première fenêtre reconnue: à moins de procèder à la suite de choix adéquate parmi un nombre de choix équivalents, l'analyse n'est jamais complète ni correcte.

Pour supprimer le premier risque d'erreur, nous avons introduit une heuristique stipulant que, lors des phases ascendantes ou descendantes, les règles récursives doivent être essayées en priorité. Mais cela ne suffit pas: il faut introduire une seconde heuristique portant sur la recherche dans le plan des formes composant une suite récursive. A partir d'un élément de cette suite (une fenêtre par exemple), pour une description faisant intervenir l'opèrateur COTE GAUCHE, l'élément suivant doit être recherché "le plus à gauche possible" du premier. Plus généralement c'est celui qui se trouve à plus courte distance dans la direction définie par l'opèrateur topographique utilisé. Les seuls cas où cette seconde heuristique serait inutile sont les suites réduites à un ou deux éléments.

Nous avons également fait une autre constatation: par un jeu de corrections automatiques l'EXPLORATEUR peut déformer une figure parfaite qui est néanmoins acceptée par l'ANALYSEUR car conforme à la description donnée. Il est arrivé que l'analyse de la figure 4.1 ait donné la figure 4.3 (page suivante): la fenêtre située à l'extrême droite a été assimilée

à une porte par prolongation des deux côtés et le montant gauche de la porte est devenu partie du mur. Ceci montre bien les limites de notre ANALYSEUR: il reconnait un ensemble structuré correspondant à la description donnée selon un coefficient de probabilité. Les possibilités d'erreurs ne sont jamais nulles.

figure 4.3



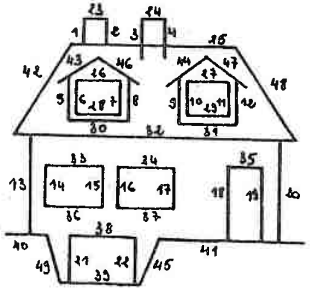
4-2-2 Un analyseur de dessins simplifié

Compte tenu des remarques précédentes, nous avons décidé de programmer une version simplifiée du système n'incluant pas les corrections mais contenant l'heuristique sur les règles récursives. Les principes de l'ANALYSEUR restent inchangés, mais sa puissance de correction a disparu: seules subsistent la recherche circulaire et la segmentation (cf § 3-2-2).

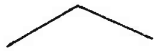
a - un exemple de déroulement de l'analyse

Nous allons expliciter brièvement un exemple d'analyse possible, tel qu'il s'est déroulé en réalité sur la figure 4.4. Les numéros portés sur le dessin servent à repérer les différentes primitives.

figure 4.4

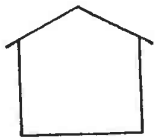


1ère étape:
- choix d'une primitive de départ pour l'analyse globale. Les segments obliques sont les plus caractèristiques, la primitive n° 46 est retenue.
- Etude du contexte: seule la règle n° 9 convient.
- Analyse: les primitives 46 et 48 forment un AUVENT.



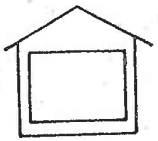
AUVENT

2ème étape:
- Etude du contexte: là encore seule la règle 8 convient.
- Elle contient un opèrateur topographique: intervention du GEOMETRE et choix d'une primitive de départ (n° 5).
- Analyse: les primitives 5, 30 et 8 forment un U qui, assemblé avec la partie précédente, donne une AVANCEE.



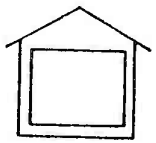
AVANCEE

3ème étape:
- Etude du contexte: toujours une seule règle possible (n° 5).
- La primitive 6 est choisie comme départ pour l'analyse descendante de la FENETRE.
- Analyse: les primitives 6,28,7 forment un U qui, complèté avec la primitive 26, donne une FENETRE.
  Le tout constitue un CHIEN.



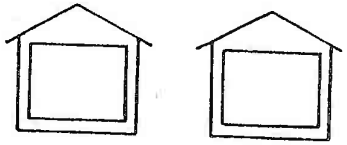
CHIEN

4ème étape:
- Etude du contexte: cette fois on peut appliquer les règles 3 et 4. Cette dernière étant récursive, elle est choisie en priorité.
- Analyse: aucun autre CHIEN n'est détecté sur le côté gauche de celui qui a été reconnu. La règle 4 ne convient pas.
- Le SUPERVISEUR effectue un retour-arrière et la règle 3 est appliquée. On obtient donc:



SCHIEN

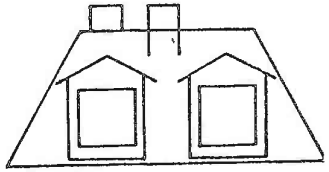
5ème étape:
- Etude du contexte: le choix se pose entre les règles 2 et 4. Priorité est donnée à la règle 4 récursive.
- Sur le côté droit de la partie reconnue, le GEOGRAPHE cherche la primitive caractèristique d'un CHIEN: il trouve la primitive 14.
- L'analyse du CHIEN se déroule de la même façon que précédemment, avec la primitive 14 comme départ. Les deux CHIENS reconnus forment une nouvelle SCHIEN.



SCHIEN

6ème étape:
- Etude du contexte: le choix est identique à la cinquième étape et la règle 4 récursive est toujours prioritaire.
- Analyse: iln'y a plus de CHIEN sur le côté droit de SCHIEN. La règle 4 ne convient pas.
- Le SUPERVISEUR effectue le retour-arrière pour appliquer la règle 2.
- Analyse: le GEOGRAPHE choisit la primitive 42 comme départ de l'analyse des PANS.

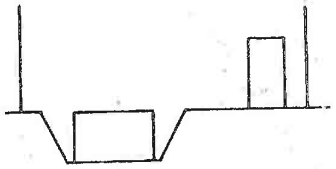
7ème étape:
- Etude du contexte: seule la règle 10 est possible.
- Analyse: les règles 11 et 12 sont utilisées pour l'analyse descendante du FAITE (avec deux interventions du GEOMETRE pour les opèrateurs POSE SUR et POSE DEVANT) qui est donc composé dans l'ordre, des primitives 25 et 1,23,2 et 3,24,4. Les primitives 48,32 et SCHIEN complètent le tout pour former le TOIT.



TOIT

8ème étape:
- Etude du contexte: il y a une unique possibilité, la règle n° 1.
- La primitive de départ pour l'analyse de FACADE choisie par le GEOMETRE porte le numèro 45 (les obliques étant les plus caractèristiques).

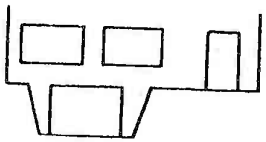
9ème étape:
- Etude du contexte (de la primitive 45): la règle 17 est choisie.
- Analyse: l'analyse descendante des MURS s'effectue avec deux appels au GEOMETRE pour les ENTREES.



MURS

10ème étape:
- Etude du contexte: un seul choix, la règle 14.
- Le GEOMETRE choisit la primitive 13 comme départ pour l'analyse de FENETRE.

- Analyse: elle se déroule de la même façon que celle de SCHIEN mais avec un seul retour-arrière car cette fois l'opérateur de description est COTE GAUCHE et l'analyse a débuté avec la FENETRE la plus à gauche.
  La FACADE entière est reconnue.



FACADE

11ème étape: - FACADE assemblée avec TOIT donne MAISON, d'après la règle n° 1 qui avait été choisie à la huitième étape. L'analyse est terminée.

Nous n'avons décrit ici qu'un déroulement possible de l'analyse. Il comporte tout de même trois retour-arrière alors qu'il ne peut en compter qu'au minimum deux, à condition de commencer l'analyse des suites récursives par le bon élément, l'extrême droit pour SCHIEN, l'extrême gauche pour SFENETRE. C'est un bon exemple d'analyse type.

b - résultats

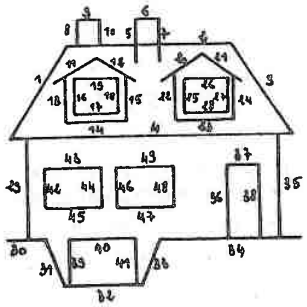
Les programmes sont écrits en FORTRAN. Le code généré n'est pas optimisé. Le système entier occupe environ 500 K octets répartis comme suit:

- 40 K mots (un mot MITRA 125 = 2 octets) pour le code en mèmoire principale
- 420 K octets pour les divers fichiers (FCONTX,FINIT...) en mèmoire secondaire

La configuration permet de traiter un dessin comptant jusqu'à 100 primitives, avec un système de description comprenant au maximum 50 règles de description utilisant au total 50 symboles distincts (non-terminaux et classes de primitives).

Jusqu'à prèsent, les tests ont été effectués avec la figure 4.4 pour quatre rangements diffèrents des primitives dans le tableau PRIM qui regroupe les listes des BLOCS de chaque classe:

- jeu A: les BLOCS sont rangés dans l'ordre de la numèrotation donnée sur la figure 4.4. Par exemple, le BLOC correspondant à la primitive n° 10 intervient en dixième position dans la liste des BLOCS de la classe des horizontales; le BLOC de la primitive 42 est en tête de la liste de la classe des obliques droites etc...
- jeu B: les BLOCS sont rangés dans l'ordre obtenu par le tracé réel (ce dernier a été déterminé d'après les schèmas exécutés par vingt personnes diffèrentes). Il est reproduit ci-dessous.



- jeu C: rangement provoquant toujours le meilleur choix (minimum de retour-arrière) avec les n° 43 et 46 comme primitives de départ (notations de la figure 4.3).
- jeu D: rangement provoquant le maximum de retour-arrière pour les mêmes points de départ.

Pour chaque jeu, les primitives caractèristiques les plus importantes du dessin (les obliques) ont été choisies comme primitives de départ d'une analyse. Les résultats sont regroupés dans le tableau de la page suivante. Les temps portés sont arrondis aux cinq secondes supèrieures.

primitive de départ* \ jeu	A	B	C	D
42 ou 48	1'30	1'20	1'25	2'40
43 ou 46	1'30	1'20	1'20	3'15
44 ou 47	1'30	1'20	1'20	3'10
45 ou 49	1'35	1'30	1'40	2'50

* notation de la figure 4.4

On constate que pour les trois premiers jeux les temps sont répartis dans la fourchette 1'20-1'40: les quelques erreurs faites par l'analyse se situent en début et ont peu d'influence sur le temps global. Par contre pour le jeu D, les retour-arrière sont échelonnés tout au long de l'analyse. Plus un retour-arrière se situe loin dans l'analyse, plus sa durée est grande (les informations à restaurer sont plus importantes), ce qui donne une moyenne de 3'00.

Bien entendu, l'ordre de rangement des primitives n'est pas le seul facteur de variation des durées d'exécution. Dans la même optique (mais à un degré moindre, car leur consultation est moins fréquente), il y a l'ordre de rangement des contextes et des initiales: à score égal, l'ELECTEUR choisit la possibilité de tête, respectant le classement fourni par le LINGUISTE qui dépend de l'ordre d'écriture des règles de description. Il faut également compter avec la façon d'écrire la grammaire c'est-à-dire la façon d'agencer les différentes sous-formes qui composent le dessin: plus elle est proche du tracé manuel, plus l'analyse est rapide. Par exemple, pour décrire un chien assis il y a ces deux possibilités parmi d'autres:

CHIEN ⟶ INTERIEUR (FENETRE,AVANCEE)
(1) AVANCEE ⟶ POSE SOUS (U,AUVENT)
AUVENT ⟶ etc...

CHIEN ⟶ POSE SOUS (AVANCEE,AUVENT)
(2) AVANCEE ⟶ INTERIEUR (FENETRE,U)
AUVENT ⟶ etc...

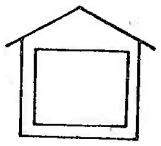


figure 4.5

La première solution est la plus naturelle, car pour tracer la figure 4.5, on dessine en général d'abord le cadre puis on place la fenêtre à l'intérieur. Avec une telle description, les règles et les primitives adéquates se présentent dans l'ordre des choix de L'ELECTEUR, ce qui atténue fortement les risques de mauvais aiguillages. Ainsi les temps correspondants aux primitives de départ situées dans le haut du dessin (42,43,44,46,47,48) par où l'opérateur commence habituellement le tracé, sont-ils plus courts que ceux donnés pour les primitives du bas (45,49), le système de description procèdant également du haut vers le bas.

4-2-3 Quelques remarques sur la première version

Avant d'être abandonnée, la première version de l'ANALYSEUR comportant les corrections automatiques a été testée avec le jeu A.Les temps obtenus avec le dessin non bruité sont légèrement majorés (entre 5 et 10 secondes) vu la complexité accrue de l'EXPLORATEUR.

La durée de la reconnaissance avec un dessin parasité dépend de trois facteurs:

- le nombre des parasites évidemment
- leurs places sur le dessin c'est-à-dire l'influence qu'ils auront sur les choix de l'ELECTEUR par distorsion des initiales et contextes.
- leurs places relatives: les corrections sont plus faciles à mettre en place si les parasites sont dispersés au lieu d'être concentrés sur un même endroit de la figure à analyser.

Avec un seul parasite, les temps observés sont inchangés pour les cas les plus favorables. Sinon,il y a majoration de 10 secondes à une minute pour les cas extrêmes (par exemple, primitive de départ n° 45 ou 49, la primitive 39 étant absente).

Avec deux parasites et si les primitives bruitées sont bien placées, la durée de l'analyse ne varie pas par rapport à une figure parfaite, à cinq secondes près. Sinon, elle peut prendre de deux à cinq minutes.

Avec trois parasites, pour les cas les plus favorables, les temps sont toujours pratiquement les mêmes. Pour les autres possibilités, il faut compter au moins quatre minutes et parfois même jusqu'à dix minutes et au-delà, sans mentionner les cas où le résultat est aberrant (cf § 4-2-1).

Toutes les configurations n'ont pas été testées exhaustivement. Les résultats donnés découlent de l'observation d'une quarantaine de cas. On peut tout de même conclure que l'ANALYSEUR reste efficient car l'émission d'un seul parasite reste exceptionnelle dans la réalité.

CHAPITRE V

BILAN ET PERSPECTIVE

5-1 Critique du système d'analyse

Nous ne reviendrons pas dans ce chapitre sur les propriétés inhèrentes au modèle syntaxique: simplicité, naturel, impossibilité de décrire tous les cas de figures etc... Tout cela a déjà été dit au chapitre I. Nous allons essayer de la manière la plus objective possible de faire la part des qualités et des défauts du processus d'analyse sans avoir la prétention d'en dresser la liste exhaustive.

5-1-1 Avantages

Tel qu'il est décrit prèsentement, notre système possède cet avantage unique: il n'a besoin que d'un nombre très restreint d'heuristiques pour fonctionner. Toutes les informations de base nécessaires au déroulement de l'analyse sont calculées à partir de la grammaire et enrichies au fur et à mesure de l'analyse par le système lui-même. Il ne faut aucune connaissance a priori sur la forme, la taille, le nombre d'éléments et la construction du dessin.

Les heuristiques sont utilisées uniquement pour résoudre le problème des règles récursives et pour permettre d'analyser et de corriger éventuellement des formes erronées. La correction reste l'un des aspects les plus difficiles de l'intelligence artificielle et sur ce point, on est loin d'atteindre les performances humaines. D'ailleurs ce problème n'a été abordé ni dans ESP^3 (18) ni dans HPL (19).

L'atout majeur de notre phase d'analyse est sans aucun doute sa rapidité (voir chapitre précédent). Les comparaisons avec d'autres systèmes existants sont difficiles car bien souvent les principes et les buts recherchés sont différents. Par exemple, pour ESP^3 et HPL, il s'agit de déterminer la ou les occurrences d'une figure donnée parmi un ensemble de figures. On peut considèrer que notre ANALYSEUR effectue le même travail si on ne tient pas compte du test final portant sur la quantité de primitives utilisées. Sur gros ordinateur, les temps d'exécution pour ESP^3 s'échelonnent de 14 secondes pour des figures élémentaires (un "six" et un "s") à 12 minutes pour une figure complexe comptant 26 segments de droite. Pour HPL, il faut déjà 30 secondes pour le test de l'aiguille dans la meule de foin: trouver l'aiguille en question (un segment de droite) parmi un ensemble de figures

comprenant au total 22 primitives. Et, en plus, ces deux systèmes ne produisent pas de description de la scène analysée. On est loin de la minute et demie nécessaire à notre ANALYSEUR pour reconnaitre un dessin de 50 primitives. Sa rapidité est due en majeure partie au prétraitement du système de description effectué par le LINGUISTE. Les renseignements obtenus - initiales, contextes, primitives caractèristiques - facilitent et accèlèrent notablement l'analyse, ce qui prouve que la nature essentiellement déclarative de notre modèle est un bon choix (par opposition à des approches procèdurales comme ESP^3 et HPL). Il ne faut cependant pas oublier que le découpage en primitives n'est pas compris dans les temps donnés et que les figures de test étaient parfaites. Un parasite placé à un endroit judicieux augmente beaucoup le temps de calcul.

Néanmoins, notre système est capable de traiter des formes erronnées, voir § 4-2-3. Si le dispositif est essentiel pour des systèmes travaillant à partir d'épreuves photographiques qui comportent toujours plus ou moins de parasites, soit à cause du capteur, soit à cause d'une mauvaise prise de vue (cf par exemple (16)), il est rare qu'un système de reconnaissance de dessins - i.e. de figures uniquement composées de traits - en soit équipé, voir (18),(19),(20)... Le procédé prend tout son intérêt lorsqu'il y a émission de parasites sur la ligne asynchrone ou plus simplement maladresse de l'opèrateur au cours du tracé. Dans ce cas, objectera-t-on, il serait plus simple de tout recommencer. Bien sûr, mais à condition de s'en apercevoir à temps et notre outil n'y gagnera pas en puissance.

La troisième particularité importante est l'arbre syntaxique produit en fin d'analyse. Le système ne se contente pas d'affirmer ou d'infirmer l'appartenance d'une figure à une classe donnée comme (21) ou de la détecter dans un "paysage" comme (18)(19), mais il en génère une description prècise et structurée, utilisable pour n'importe quel traitement postèrieur. Par exemple localisation, modification ou demande de caractèristiques de certaines parties de la figure analysée. C'est un point très important pour des applications pratiques éventuelles.

Enfin l'ANALYSEUR peut apprèhender le dessin par l'une quelconque de ses parties. Il n'est point besoin de fournir d'indications spèciales sur la structure ou la forme générale du dessin pour pouvoir l'analyser. De par son principe même, la reconnaissance en est totalement indépendante et,

a priori, il n'y a pas de limites à la complexité de la figure à analyser, quelqu'en soit la nature, à condition bien sûr de pouvoir fournir un système de description adapté au modèle choisi.

5-1-2 Inconvénients et amèliorations possibles.

a - les relations topographiques

Il existe des cas que l'ANALYSEUR, tel qu'il est conçu actuellement, ne sait pas résoudre ou plus exactement pour lesquels il donne une description qui ne concorde pas avec la réalité. Deux cas ont été montrés dans le chapitre précédent: ils résultent d'un mauvais choix à la rencontre d'une relation topographique combiné dans le deuxième exemple avec des corrections mal placées.

Le faite du toit est décrit dans la grammaire comme deux cheminées posées sur un segment horizontal (figure 5.1). L'une des cheminées est raccordée au toit par l'intermèdiaire de l'opèrateur pseudo-topographique POSE SUR, l'autre par l'intermèdiaire de POSE DEVANT, pour distinguer leurs positions relatives dans l'espace. Les définitions de ces opèrateurs étant très proches, il peut y avoir inversion: la cheminée se trouvant en avant (B) étant reconnue comme se trouvant en arrière (A) et vice-versa.

Dans l'anomalie décrite au paragraphe 4-2-1, une fenêtre a été assimilée à une porte par prolongation de ses deux montants. Ultèrieurement, le montant gauche de la porte est devenu un mur, les segments restants étant considérés comme parasites (figure 5.2).

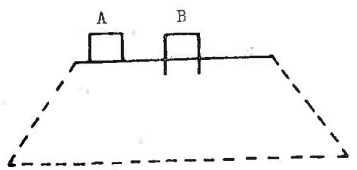


figure 5.1

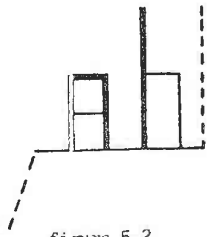


figure 5.2

Dans les deux cas, l'ELECTEUR a choisi comme primitive de départ pour l'analyse de la sous-forme considèrée un segment appartenant à une sous-forme similaire. La suite de l'analyse (avec ou sans corrections) n'ayant pas infirmé ce choix, la sous-forme adéquate n'est plus jamais prise en compte dans sa vraie fonction: il y a échange de fonction (figure 5.1) ou assimilation à des parasites (figure 5.2). La stratègie d'analyse doit être révisée. Il faut introduire un outil permettant d'avoir une vision globale de certaines portions du plan: pour l'analyse d'une sous-forme, chercher toutes les figures correspondant à la description et présentes dans la zone de travail, puis sélectionner la meilleure (celle dont le score est le plus élevé). Le recours à un tel mécanisme s'il est systèmatique sera coûteux en temps de calcul. Il faudra l'utiliser pour des figures assez simples qui apparaissent ou risquent d'apparaitre souvent sur le dessin, soit isolèment, soit intègrées au tracé: PORTE ou CHEMINEE dont les configurations se retrouvent dans FENETRE par opposition à CHIEN qui est une sous-forme plus complexe.

b - le problème de l'échelle

Nous avons déjà parlé des inconvénients de l'absence d'échelle dans le chapitre I. L'analyse n'en est que plus rapide mais elle peut déboucher sur la reconnaissance de figures complètement déformées (figure 5.3) ou de conception invraisemblable (figure 5.4) donc à rejeter.

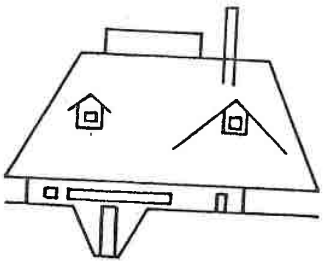


figure 5.3

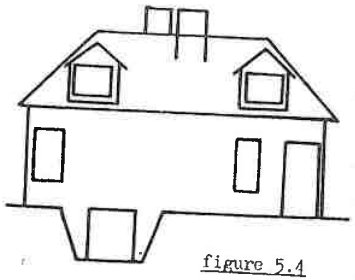
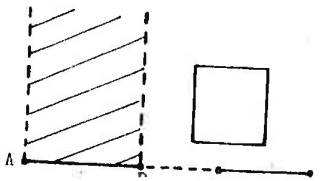


figure 5.4

figure 5.5



Cependant pour certains cas, la possession de rapports de grandeur peut s'avèrer prècieuse: la prolongation par tronçons deviendrait inutile puisque l'on connaitrait la longueur exacte de chaque primitive. De même, la segmentation et la recherche en faisceau seraient facilitées par des mesures sur les différentes primitives sélectionnées. Enfin, des cas comme la figure 5.5 seraient facilement résolus: supposons qu'une partie du dessin à analyser soit un "CARRE au-dessus d'une horizontale", et qu'un parasite ait divisé le segment en deux tronçons AB et CD. A partir du segment AB, on ne peut trouver le CARRE (la recherche s'effectuent dans la zone hachurée) et comme dans notre système il n'est pas prèvu de prolonger AB dans cette configuration, la seule manière d'aboutir demande de commencer l'analyse par le CARRE. Avec une échelle, le segment AB est tout de suite ramené à sa juste longueur AD, la suite de l'analyse ne posant aucune difficulté.

c - le découpage en primitives

Le même genre de problème qu'au paragraphe précédent se présente avec la répartition entre segments de droite et arcs de cercle. S'il y a une seule erreur et même si l'ECHANTILLONNEUR a dû choisir entre deux possibilités de score égal, le dessin ne sera jamais accepté par le système actuel, sans aucun moyen d'y remèdier. Un processus interactif ECHANTILLONNEUR-ANALYSEUR permettant de revoir le découpage avec de nouvelles contraintes (comme par exemple: prèsence obligatoire d'un segment de droite à un endroit prècis) qui pourraient encore être affinées par une échelle semble donc nécessaire.

d - les règles récursives

Le cas des règles de description récursives est plus délicat: nous avons vu dans le chapitre précédent que l'adjonction d'une heuristique sur leur priorité lors des phases de choix permet de traiter correctement toute suite récursive réduite à deux éléments. Pour des suites plus importantes, il faudrait introduire une autre heuristique portant sur les positions relatives dans le plan de leurs éléments: l'élément correct complètant un sous-ensemble quelconque de la suite est celui qui se trouve à la plus courte distance euclidienne du sous-ensemble. Il semble que cette notion se généralise assez bien avec le système topographique choisi dans notre modèle. Pour un système plus sophistiqué, la question reste entière.

e - les corrections

A partir du seul système de description, l'ANALYSEUR ne peut rivaliser avec un humain sur le plan de la compréhension du tracé (toutefois les résultats obtenus sont loins d'être décourageants). Ainsi n'est-il pas capable de toujours effectuer les meilleures corrections sur un dessin bruité ou non: voir les figures 4.3 et 5.2. Nous avons exposé précédemment un moyen d'y parvenir dans le cas des relations topographiques: attribuer des scores partiels à chaque sous-forme détectée et sélectionner le score le plus élevé. En généralisant le procédé à n'importe quelle partie du dessin (qu'elle constitue ou non une sous-forme) sujette à des corrections, on réduit considèrablement les risques d'erreurs par comparaison des scores des différentes combinaisons de corrections possibles.

Bien entendu, l'utilisation d'un tel procédé se fera au détriment de la rapidité d'exécution. Il en sera de même avec toutes les amèliorations décrites auparavant: elles risquent d'augmenter dans des proportions notables les temps donnés dans le chapitre IV.

f - les retour-arrière

Pour diminuer la durée de l'analyse, on peut intervenir à deux niveaux: au niveau programmation, en optimisant l'écriture des programmes et en adoptant des algorithmes de gestion des ressources plus performants, et au niveau conceptuel en revoyant les stratègies les moins sophistiquées. Nous pensons surtout au processus de retour-arrière qui totalise un temps de calcul assez important du fait du volume des informations à restaurer et de la durée des échanges mèmoire principale-mèmoire secondaire.

Pour le moment (cf § 3-2-4 e), en cas d'échec, il y a retour systèmatique au NIVEAU de choix précédent, exception faite si son score est très faible par rapport à celui qui le précède. La cause de l'échec provenant rarement du NIVEAU immèdiatement infèrieur, il faut procèder à une cascade de retour-arrière inutiles avant de retrouver la bonne voie. Par exemple, supposons qu'au cours de l'analyse de CHIEN (grammaire du chapitre IV), l'AUVENT soit reconnu (en pointillés sur la figure 5.6). La règle 8 nous apprend

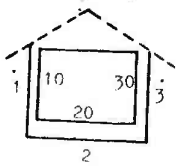


figure 5.6

qu'il faut maintenant chercher une sous-forme U qui apparait en deux endroits sur le schèma: primitives 1,2,3 ou 10,20,30. Si la primitive 10 est choisie comme départ, l'ANALYSEUR va emmagasiner quatre NIVEAUX de choix (choix de la primitive de départ, choix d'une règle dans la phase ascendante et choix de l'EXPLORATEUR pour les primitives 20 et 30) avant d'échouer, et donc générer trois retour-arrière inopèrants.

Dans ce cas-ci, il aurait fallu revenir directement sur le choix de la primitive de départ, ce qui n'est possible qu'en ayant connaissance d'une part de la cause de l'échec, d'autre part de la nature de la suite des choix qui l'a provoqué. Il faut en outre disposer d'une fonction capable d'analyser ces deux facteurs et d'en déduire les points de retour de plus forte probabilité. Sans aller jusque là, on peut intègrer au système d'analyse un processus permettant le traitement de quelques combinaisons privilègiées (celles qui risquent de se reproduire frèquemment): revenir au dernier choix de la primitive de départ lorsque deux sous-formes ne vérifient pas la relation topographique qui les relie (exemple de la figure 5.6), privilègier le choix des règles de description au détriment des choix de l'EXPLORATEUR lorsque le premier offre plusieurs scores élevés (puisqu'en principe les parasites sont rares) etc...

Il serait également bon de pouvoir conserver les morceaux d'arbre syntaxique correspondant à des parties du dessin ayant des scores parfaits. Lors d'une dégradation trop rapide du score global de reconnaissance, cela permettrait de recommencer l'analyse avec de nouvelles bases en évitant des opèrations inutiles par adjonctions des sous-arborescences mèmorisées à l'arbre principal au fur et à mesure du processus de reconnaissance.

Nous pensons avoir examiné les principaux points délicats du système en essayant de proposer des solutions qui, si elles ne sont pas toujours les meilleures, ont le mèrite de la clarté et de la généralité.

5-1-3 Poursuite des recherches

Le système d'analyse défini actuellement est loin d'être parfait, comme nous venons de le voir. Il fonctionne correctement sur des cas simples mais sa puissance est limitée tant au point de vue taille qu'au point de vue complexité des dessins à traiter.

La première question n'est redevable qu'au matériel employé et sort donc du cadre de cette étude.

Le deuxième point dépend d'abord du modèle choisi qu'il faudra rendre plus efficient avec:

. Introduction d'attributs (au sens de KNUTH et LOHRO (22)) autrement dit de paramètres définis en fonction de ceux du contexte ou de la sous-forme considérés: par exemple longueur d'un segment, angle formé par deux segments, direction d'un alignement de points...

. Extension à un modèle à plusieurs pôles (sans se limiter à l'origine et à l'extrémité) peut-être inspiré par celui de NARASHIMAN (23), pour accroitre la variété des formes admises par le système et ramasser l'écriture de leur description.

. Adjonction éventuelle de nouveaux types de primitives tel le point, avec de nouvelles caractèristiques: épaisseur et, pourquoi pas, couleur du tracé.

. Affinement des relations topographiques avec possibilité de définir dans le plan des zones plus réduites, sans se cantonner à la forme rectangulaire actuelle (également par un système d'attributs).

. Révision de l'utilisation des règles de description récursives: simplifier le schèma d'écriture classique et ne plus se borner à des relations linèaires (essentiellement juxtapositions) entre les éléments de la suite récursive.

. Possibilité de décrire des objets en trois dimensions, ce qui risque d'être essentiel pour une application pratique. Il faudra passer à reprèsentation spatiale à trois coordonnées et y adapter le système d'analyse entier. Mais peut-on encore parler de dessin à ce niveau?

Avant d'en arriver là, il reste certaines questions importantes à résoudre pour parvenir à un système élaboré capable d'ingèrer ces modifications:

. Mettre en oeuvre et tester le traitement des primitives de la classe "arc de cercle" et adapter le fonctionnement de l'EXPLORATEUR aux nouvelles exigences: diversité des types et des caractèristiques des primitives, notion d'attribut...

. Créer un interface ECHANTILLONNEUR-ANALYSEUR susceptible de corriger des erreurs éventuelles du découpage en primitives et de résoudre les cas ambigus par les précisions fournies par les règles de description utilisées au fur et à mesure du processus d'analyse.

. Adjoindre des heuristiques plus nombreuses et/ou plus fines pour parfaire le traitement des règles récursives, des retour-arrière et gagner en prècision dans le choix des règles, donc accroitre la rapidité du système et élargir son champ d'application.

. Définir la structure de l'INTERPRETEUR, ses différents modes d'actions et les outils correspondants: modification de parties du dessin, changement d'échelle ou de perspective etc... Inventer un langage d'utilisation spècifique en fonction des domaines d'application possibles et de l'usage qui y sera fait.

C'est un programme suffisamment riche et passionnant pour une seconde étape vers un système perfectionné à usage courant.

5-2 Perspective

La reconnaissance des formes est appelée à un grand avenir dans notre société où l'automatisation et le recours à l'audio-visuel se généralisent. Il s'agit de simplifier l'interface homme-machine - remplacer le support imprimé (programmes,données) par la parole ou l'écriture manuelle, le document numèrisé par son image - ou bien automatiser les traitements acoustiques et visuels fastidieux ne nécessitant aucune décision humaine importante.

Nous ne parlerons pas de la reconnaissance de la parole bien qu'elle soit directement apparentée aux recherches sur l'interprètation d'images: le système MYRTILLE (24) développé dans notre laboratoire utilise le même algorithme de base que notre ANALYSEUR. On pourra aussi se reporter aux systèmes réalisés aux USA (25).

On peut imaginer des applications dans toutes les branches scientifiques: observations d'échantillons pathologiques en cytologie, comparaison de l'origine de lames minces en pétrographie, observations d'aérodromes par satellite en génie militaire, examen microscopique de structures d'alliages en métallurgie. Les systèmes opèrationnels sont déjà nombreux: en mèdecine particulièrement, on peut citer le système permettant d'isoler les cellules sur une prèparation (26) ou de déterminer leur type (30), analyser et classer les chromosomes (27)(28). Dans d'autres domaines, au Centre Technique des Forêts Tropicales, il existe un système qui permet d'étudier la structure des bois donnant un rétrècissement isotrope au sèchage (29). Il y a également l'examen de clichés astronomiques (31), le contrôle de circuits électroniques (32), l'analyse de dessins industriels (20) etc... sans compter les programmes de reconnaissance de caractères manuscrits (cf (11) (33)(34) ou le travail de A. BELAID à Nancy (12) avec une approche voisine de la nôtre). Bien entendu ce n'est là qu'un faible aperçu.

Où va s'insèrer notre système parmi tous ceux-ci? En premier lieu, nous avions pensé à l'architecture: l'opèrateur dessine à main levée sur la tablette graphique son modèle de bâtiment ou son plan. Après le passage de l'ANALYSEUR et le profilage des primitives tordues, il peut demander la modification de proportions, d'agencements, essayer diffèrents types de fenêtres, portes, cheminées... rangées dans une base de données annexe. Ou encore, après avoir dessiné les vues de profil et de face, visualiser la vue de trois-quarts. Mais il semble que ce soit plutôt l'aspoct conception assistée par ordinateur qui intèresse les architectes (35): système RES de génération de structures spatiales; CARLA, système d'analyse régionale pour trouver des plans d'amènagement optimum; système de développement de constructions spatiales portantes en barres (qui a servi, entre autres, à l'élaboration de la toiture du stade olympique de MUnich) etc...

Cependant les deux orientations ne sont pas forcèment incompatibles - pour la conception assistée par ordinateur (en abrégé CAO) voir (36). Notre système est suffisamment général pour pouvoir s'adapter à l'une comme à l'autre. En généralisant son champ d'activité (dessins, photographies, images télévisées...), en établissant l'interchangeabilité de l'INTERPRETEUR (spècifique à chaque utilisation) et des structures de l'ECHANTILLONNEUR (tablette graphique, écran graphique, acquisition de photographies...), l'ANALYSEUR proprement dit devient un outil pour la plupart des systèmes nommés précédemment, en particulier (20)(27)(28)(32). En CAO, il trouve aussi son utilisation comme instrument ou comme complèment: vérification de la validité des structures générées, transport de la description obtenue et ainsi de suite.

A cause de sa grande rapidité, de sa capacité et de sa simplicité d'utilisation, nous pensons que notre système devrait être promis à un bel avenir.

CONCLUSION

L'image reste un moyen agrèable de communication car elle permet de traduire une information complexe et prècise sous une forme condensée. Elle peut être efficacement utilisée pour faciliter le dialogue entre l'homme et l'ordinateur.

Le système de reconnaissance et d'interprétation de dessins développé dans cette étude se rèclame de cette orientation. Il repose sur la généralisation des langages à contextes libres qui nous offrent une grande souplesse d'adaptation par leur richesse et leur simplicité. Un analyseur syntaxique permet d'extraire la structure et la sèmantique d'une figure à partir de sa seule description.

Nos efforts ont surtout porté sur la mise en oeuvre de ce système. Cela nous a permis de mieux cerner les différents problèmes posés par la reconnaissance de dessins et de prouver l'efficacité de notre méthode, ce que n'aurait pas permis un exposé théorique. Nous avons montré dans le chapitre V ses qualités évidentes de rapidité (temps de réponse de l'ordre de une minute et demie sur un petit calculateur), d'adaptabilité et sa facilité d'utilisation qui éclipsent des défauts en grande partie dus à une simplicité voulue du modèle.

Cette première réalisation ne doit être qu'un tremplin vers l'élaboration d'un outil plus complet et plus général. Nous espèrons y avoir efficacement contribué et pouvoir encore continuer dans ce sens.

BIBLIOGRAPHIE

1 C. PAIR
Sur les notions algèbriques liées à l'analyse syntaxique
RAIRO R-3, pp 3-29 (1970)

2 A. QUERE
Etude des ramifications et des bilangages
Thèse de spécialité, faculté des Sciences, NANCY I (1969)

3 J.P. HATON, R. MOHR
A parsing algorithm for imperfect patterns and its application
3d IJCPR, San Diego, California (1976)

4 D. E. KNUTH
On the translation of languages from left to right
Information and control, 8:6, pp 607-639 (1965)

5 R. MOHR
Généralisation de la notion de langage à contexte libre
RAIRO R-2, pp 55-88 (1975)

6 R. MOHR
Modèle algèbrique pour l'analyse syntaxique de figures
Thèse de spécialité, faculté des Sciences, NANCY I (1973)

7 J.P. HATON, R. MOHR
Deuxième rapport intermèdiaire du contrat SESORI 76-039
CRIN, Université de NANCY I (1977)

8 J.P. HATON, R. MOHR
Rapport final du contrat SESORI 76-039. Utilisation des méthodes syntaxiques en reconnaissance des formes
CRIN, Université de NANCY I (1977)

9 C. PAIR
Analyse syntaxique
Ecole d'été EDF-CEA-IRIA (1973)

10 J.C. DERNIAME, C. PAIR
Problèmes de cheminement dans les graphes
Dunod (1973)

11 M. BERTHOD
Une méthode syntaxique de reconnaissance des caractères manuscrits en temps réel avec un apprentissage continu
Thèse de spécialité, Université de PARIS VI (1974)

12 A. BELAID
Segmentation de tracés en vue de leur analyse. Application à la reconnaissance structurelle des caractères manuscrits
Congrès AFCET (novembre 1978) (à paraitre)

13 M.G. THOMASON, R.C. GONZALEZ
Syntactic recognition of imperfectly specified patterns
IEEE transactions on computers, pp 93-95, (January 1975)

14 W.B. SMITH
Error detection in formal languages
Journal of computers and systems sciences 4, pp 385-405 (1970)

15 K.S. FU
Syntactic methods in pattern recognition
Academic Press, NEW-YORK (1974)

16 Y. SHIRAI
A context sensitive line finder for recognition of polyhedra
Artificial intelligence 4, pp 95-119 (1973)

17 A.V. AHO, J.D. ULLMAN
The theory of parsing, translation and compiling
Prentice-Hall, ENGLEWOOD CLIFFS, New Jersey (1973)

18 L.G. SHAPIRO, R.J. BARON
ESP^3: a language for pattern description and a system for pattern recognition
IEEE transactions on software engineering, vol. SE-3, n° 2, pp 169-183 (March 1977)

19 H. WILLIAMS
A net-structure learning system for pattern description
Pattern recognition, Pergamon Press, vol. 8, pp 261-271 (1976)

20 V. GALLO
A program for geometrical pattern recognition based on the linguistic method of the description and analysis of geometrical structures
Advance papers of the 4th IJCAI, TBILISI, vol. 2, pp742-745 (1975)

21 R. NEVIATA, T.O. BINFORD
Description and recognition of curved objects
Artificial intelligence 8, pp 77-98 (1977)

22 D.E. KNUTH
Semantics of context-free languages
J. Math. Syst. Theory 2, pp 127-146 (1968)

23 R. NARASHIMAN
Syntax directed interpretation of classes of pictures
Communication ACM, 9,3, pp 166-173

24 J.P. HATON, J.F. MARI, J.M. PIERREL
Myrtille: un système de compréhension du discours parlé
Congrès AFCET-IRIA, tome I, pp 176-185 (1978)

25 Speech recognition
D.R. REDDY editor, Academic Press (1976)

26 F. MEYER
cité dans LA RECHERCHE n° 87, pp 247-256 (mars 1978)

27 R.S. LEDLEY
High-speed automatic analysis of biomedical pictures
Sciences 146, pp 216-223 (1964)

28 J. VAN DAELE, A. OOSTERLINCK, H. VAN DEN BERGHE
Système automatisé de traitement et reconnaissance d'images appliqué à la classification des chromosomes humains
Congrès AFCET-IRIA, tome II, pp 841-847 (1978)

29 A. MAIRAUX, O. PERRAY, J. SERRA
cités dans LA RECHERCHE n° 87, pp 247-256 (mars 1978)

30 A. CLAINCHARD, H. MAITRE, B. BOUTROIS, D. ROCHE, J. FLEURET
Etude et réalisation d'une mèthode automatique de reconnaissance de types cellulaires dans les tissus hématopoiétiques
Congrès AFCET-IRIA, tome II, pp 855-862 (1978)

31 A. BIJAOUI, G. LAGO, J. MARCHAL, Ch. OUNNAS
Le traitement automatique des clichés astronomiques
Congrès AFCET-IRIA, tome II, pp 848-854 (1978)

32 R.T. CHIEN, W. SNYDER
Visual understanding of hybrid circuits via procedural models
Advance papers of the 4th IJCAI, TBILISI, USSR, vol. 2 (1975)

33 S. HANAKI, T. TEMMA, H. YOSHIDA
An on-line character recognition aimed at a substitution for a billing machine keyboard
Pattern recognition, Pergamon Press, vol. 8, pp 63-71 (1976)

34 G.M. MILLER
On line recognition of hand generated symbols
F.J.C.C. pp 399-412 (1969)

35 Architekt und Computer / ein Mensch-Maschine System
Goethe Institut, München (1977)

36 Bulletin de liaison de l'IRIA n° 35, pp 2-19 (avril 1977)

37 R.S. LEDLEY, L.S. ROTOLO, T.J. GOLAB, J.D. JACOBSEN, M.D. GINSBERG, J.B. WILSON
FIDAC: Film Input to Digital Automatic Computers and associated syntax directed pattern recognition programming system
Optical and Electro-optical information processing, CAMBRIDGE, MA, MIT Press, pp 591-613 (1965)

38 S.L. HOROWITZ
A syntactic algorithm for peak detection in waveforms with applications to cardiography
Communication ACM, n° 5, vol. 18 (1975)

39 J.T. TOU, R.C. GONZALES
Recognition of handwritten characters by topological feature extraction and multilevel categorization
IEEE transactions on computers (July 1972)

40 A.C. SHAW
A formal picture description scheme as a basis for picture processing systems
Information and control 14, pp 9-52 (1969)

41 T. VAMOS, Z. VASSY
Indutrial pattern recognition experiment - a syntax aided approach
1st IJCPR, WASHINGTON (1973)

42 R. KIRSCH
Computer interpretation of english text and picture patterns
IEEE transactions on electronic computers, EC-13, pp 363-376 (1964)

43 OTA
Mosaic grammars
University of Pennsylvania, Moore SCHOOL. report n° 73-10

44 ROSENFELD
Isotonic grammars, parallel grammars and picture grammars
Machine intelligence, vol. 7, pp 281-294 (1971)

45 M.F. DACEY
The syntax of a triangle and some other figures
Pattern recognition, vol. 2, pp 11-31 (1970)

46 M.F. DACEY
POLY: a two dimensional language for a class of polygons
Pattern recognition, vol. 3, pp 197-208 (1971)

47 A.B.S. HUSSAIN, R.W. DONALDSON
Suboptimal sequential decision schemes with on-line feature ordering
IEEE transactions on computers (July 1972)

48 J.D. PATTERSON
The linear mean square estimation technique of recognition
1st IJCPR, pp 41-49 (1973)

49 Y.E. CHO
The generating properties of context-free picture grammars
2nd IJCPR, COPENHAGUE, pp 90-94 (1974)

NOM DE L'ETUDIANT : Monsieur MASINI Gérald

NATURE DE LA THESE : DOCTORAT DE 3e CYCLE en Informatique

VU, APPROUVE

et PERMIS D'IMPRIMER

NANCY LE 14 SEP 1978

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

Le Président

M. BOULANGE