CENTRE DE RECHERCHE EN INFORMATIQUE DE NANCY

chateau du montet
54500 vandœuvre les nancy

DESCRIPTIONS STRUCTUREES ET ANALYSE DE FORMES COMPLEXES *APPLICATION A LA RECONNAISSANCE DE DESSINS*

ROGER MOHR

- 79 T 037 -

université de nancy 1
université de nancy 2
institut national polytechnique de lorraine

CENTRE DE RECHERCHE EN INFORMATIQUE DE NANCY

chateau du montet
54500 vandœuvre les nancy

DESCRIPTIONS STRUCTURÉES ET ANALYSE DE FORMES COMPLEXES *APPLICATION A LA RECONNAISSANCE DE DESSINS*

ROGER MOHR

- 79 T 037 -

université de nancy 1
université de nancy 2
institut national polytechnique de lorraine

DESCRIPTIONS STRUCTURÉES ET ANALYSE DE FORMES COMPLEXES *APPLICATION A LA RECONNAISSANCE DE DESSINS*

ROGER MOHR

- 79 T 037 -

UNIVERSITÉ DE NANCY I
U.E.R. DE SCIENCES MATHÉMATIQUES

CENTRE DE RECHERCHE EN
INFORMATIQUE DE NANCY

DESCRIPTIONS STRUCTUREES ET ANALYSE DE FORMES COMPLEXES

APPLICATION A LA RECONNAISSANCE DE DESSINS.

THESE

PRÉSENTÉE POUR L'OBTENTION DU
GRADE DE DOCTEUR ES SCIENCES
EN MATHÉMATIQUES APPLIQUÉES

SOUTENUE LE 13 JUIN 1979
PAR

ROGER MOHR

DEVANT LE JURY

RAPPORTEURS	C. PAIR	PRÉSIDENT
	J.P. CRESTIN	
	J.P. HATON	
EXAMINATEURS	R. BAJCSY	
	S. CASTAN	
	O. FAUGERAS	
	M. GRIFFITHS	

Je veux remercier ici

Claude PAIR qui m'a converti à la recherche informatique. C'est lui, par le sujet qu'il m'avait proposé en D.E.A., qui a été à l'origine de ce travail. Amical, discret et efficace, il a toujours prêté une oreille attentive à mes propos et son souci de rigueur m'a constamment incité à approfondir ma réflexion.

Ruzena BAJCSY qui a bien voulu s'intéresser avec compétence à ce travail. J'espère que nos premiers échanges se concrétiseront dans l'avenir par des liens scientifiques plus étroits.

Serge CASTAN qui m'a enseigné les techniques du traitement d'image et qui me fait aujourd'hui l'honneur de participer à ce jury.

Jean-Pierre CRESTIN, spécialiste d'informatique graphique et ayant de profondes connaissances en informatique théorique, il est particulièrement qualifié pour juger ce travail. Je lui suis reconnaissant d'avoir eu la patience de lire cette thèse dans sa version non corrigée et d'avoir accepté la tâche ingrate de rapporteur.

Olivier FAUGERAS pour l'intérêt amical qu'il a toujours porté à tous nos travaux ; c'est un plaisir pour moi que de le voir siéger aujourd'hui dans ce jury.

Michaël GRIFFITHS pour les discussions intéressantes que nous avons eu et qui m'ont aidé à préciser mes idées. Je lui suis aussi reconnaissant d'avoir eu l'infinie patience de lire mon manuscrit et pour le nombre impressionnant de remarques et critiques qu'il en a tirées.

Jean-Paul HATON ; il m'est impossible de résumer en quelques lignes cinq années de collaboration amicale ; je me contenterai donc de dire que sans ses encouragements, ce travail n'aurait pas abouti aujourd'hui et je lui suis reconnaissant pour son soutien constant.

Si une thèse est essentiellement un travail personnel, son aboutissement n'est possible que grâce à l'environnement scientifique et humain. Pour moi ce cadre a été le CRIN et l'IUCAL où je n'ai pas seulement cotoyé des chercheurs mais aussi des amis ; je voudrais en particulier remercier ici les membres de l'équipe Reconnaissance des Formes et ceux de l'équipe Castor et tout particulièrement G. MASINI et J.L. REMY avec lesquels j'ai eu de nombreux échanges fructueux.

Martine TESOLIN s'est chargée avec patience de la frappe et de la correction du manuscrit et M. DEBERT de son tirage ; je ne peux que les remercier et les féliciter pour le résultat.

Je voudrais également remercier Claudine pour toutes les fois où elle m'a distrait de ma tâche ; avec elle j'ai pu partager des moments de détente favorable à ma réflexion.

Je remercie la compagnie IBM France pour l'aide qu'elle a bien voulu nous accorder.

Ce travail a été réalisé partiellement dans le cadre du contrat SESORI 76-039.

Roger MOHR

TABLE DES MATIERES

CHAPITRE 2 : DESCRIPTION ET ANALYSE DE DESSINS DANS LE SYSTEME MIRABELLE

CHAPITRE 4 : MODÈLES DE DESCRIPTION POUR LA RECONNAISSANCE DES FORMES

CHAPITRE 1

INTRODUCTION ET RAPPELS

CHAPITRE 1

INTRODUCTION ET RAPPELS

1.- PRESENTATION DU SUJET ET MOTIVATION.

Un objet complexe ne peut pas être considéré comme une réunion d'atomes indépendants, sauf dans les cas extrêmes, il comporte en effet une *organisation structurée* faisant apparaître des parties reliées entre elles par des propriétés. Ce point de vue amène naturellemnt à poser le problème de la reconnaissance des formes comme l'identification de cette structure et non plus comme une simple classification ; par identification de la structure nous entendons construction de l'organisation liant composants et sous-composants, découverte des relations intervenant entre composants , identification de chaque composant.

Cette démarche est souvent présentée comme antagoniste de celle qui consiste à extraire de la forme des paramètres puis d'opérer son classement dans l'espace de décision. Plutôt que de s'opposer , ces démarches se complètent ce seront en effet des techniques classiques et bien rodées de classification qui permettront d'identifier les composants élémentaires de notre objet complexe. Ainsi, les techniques de classification sont adaptées à la reconnaissance des caractères de l'alphabet, mais ne peuvent résoudre le problème de reconnaissance d'une phrase écrite ; elles permettent sur une image aérienne multispectrale de distinguer les cultures, des routes, des forêts, mais leur nature les rend incapables d'interpréter une telle image en décrivant par exemple un réseau routier.

Affecter une classe à une forme - par exemple la classe A à un caractère-

est déjà un premier travail d'"interprétation sémantique" . Si cette forme a une structure que l'on a pu construire, si les composants de cette structure sont identifiés, le travail d'interprétation est bien plus fin et permet de répondre à des questions telles que :

"combien de fenêtres a cette maison ?" ou
"Quel est le rapport surface de façade sur surface des ouvertures ?"

L'approche structurelle permettra donc de mieux saisir l'information contenue dans une forme.

La perception humaine dépend de l'univers dans lequel est placé l'individu ; de plus la perception d'une forme particulière de cet univers dépend du contexte dans lequel elle apparaît : la forme de la figure 1.1. est perçue comme 13 ou comme B selon que l'on se place dans l'univers numérique ou alphabétique ; le caractère mal écrit de la figure 1.2. sera reconnu différemment selon que la lecture se fera verticalement ou horizontalemnet, le contexte est différent dans ces deux cas.

figure 1.1.

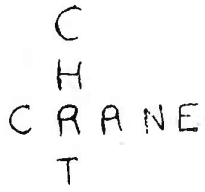


figure 1.2.

Pour la perception automatique, l'univers de travail est défini le plus souvent, implicitement par restriction au sujet traité. L'introduction du contexte a été prise en compte dans l'approche statistique soit par les probabilités conditionnelles :

- une partie X est identifiée, une forme U est à reconnaître, on peut alors essayer d'estimer p (U = Y | X) la probabilité que U soit de la classe Y quand X est présent - , soit en exprimant la dépendance par des matrices de covariance . L'approche structurelle offre directement un cadre naturel à l'utilisation du contexte : la structure sur laquelle on travaille exprime la dépendance de chaque constituant avec les autres ainsi que la nature de cette dépendance : le type de relations possibles.

Ces considérations ont naturellement amené les chercheurs à s'intéresser à cette approche et à tenter de résoudre les problèmes qu'elle pose. Ces problèmes se résument essentiellement dans les deux questions suivantes :

- quelles sont les structures possibles pour la classe des formes de l'univers considéré , c'est à dire quelles décompositions en sous-formes pourrons-nous avoir pour les formes à reconnaître, et comment ces sous-formes seront-elles reliées entre elles. Ce point est celui de la définition de l'univers et de la structure de ses objets.

- Comment , à partir d'une forme, reconstituer sa (ses) structure (s) . C'est le problème de l'analyse.

Le premier point concerne aussi la définition du résultat de l'algorithme d'analyse. Cette définition peut être implicite et apparaîtra alors dans la structure de l'algorithme. Ainsi par exemple lorsque Schoevaert-Brossault analyse la morphologie des spermatozoïdes [SCH 78] , son algorithme s'appuie sur la structure du spermatozoïde décomposé en tête et reste cyctoplasmique, portion proximale du flagelle , ainsi que sur la relation entre ces composants par exemple l'angle entre l'axe de la portion proximale du flagelle et celui de la tête. Mais de même que les premiers analyseurs syntaxiques qui intégraient la description syntaxique dans leurs algorithmes ont été suivis d'analyseurs généraux, de même assiste-t-on à la mise au point de techniques d'analyses générales permettant de travailler sur une structure d'objets pluri-dimensionnels ; c'est le cas de ce qui est présenté dans cette thèse, mais aussi d'autres travaux qui seront étudiés au chapitre 4.

2.- PRÉSENTATION DE CE TRAVAIL.

Cette thèse reprend et complète un travail continu qui a été entrepris depuis 1975 dans l'équipe de traitement du signal et de reconnaissance des formes du CRIN. Elle est le prolongement naturel de l'étude théorique qui avait été faite sur la description par des systèmes algébriques d'ensembles de formes et la conclusion de [MOH 75] contenait le germe de tous les résult publiés ici.

Ce travail a été soutenu par le contrat SESORI n° 76-039 sans lequel l'expérimentation de nos idées n'aurait pas été possible. Une partie des résultats et des idées ont déjà donné lieu à publications : [MOH 76] , [HAT 7 [MAS 78] , [MOH 78] . Les chapitres 3 et 4 présentent cependant des idées originales, ou des résultats plus fins que ceux qui avaient été présentés auparavant.

Le chapitre 2 présente le modèle formel qui sert de cadre au système MIRABELLE ; au paragraphe 2 nous y définissons un algorithme original d'analy syntaxique qui se généralise facilement à l'analyse de formes structurées. Pour améliorer son temps d'exécution, cet analyseur résoud ses choix en testant son environnement; le résultat de ces tests sont interprétés à l'aide de tables. La construction automatique de ces tables à partir de la description formelle de la forme se fait par le "compilateur" du système MIRABELLE. La définition des différents renseignements à mettre en table, leurs calculs effectifs font l'objet du chapitre 3. Les techniques utilisées s'inspirent largement de celles de la construction de compilateurs de compilateurs (GRI et le lecteur familier de ces travaux saisira aisément la généralisation qui en est faite.

Le chapitre 4 présente une étude critique de la plupart des travaux que l'on peut trouver dans ce domaine. Nous avons placé cette étude bibliographique après la présentation du système MIRABELLE : ceci nous permettra de discuter des avantages et faiblesses de notre système et par ailleurs de voir quelles idées de notre approche sont généralisables. A la lueur des différentes études citées, nous essayerons alors de dégager les principaux concepts intéressants dans le modèle structurel : les différentes façons de

structurer l'univers, les différents types de relations assemblant les composants, les stratégies d'analyse possibles.

Nous avons essayé de faire une présentation relativement concise ; ceci nous a amenés à supprimer des démonstrations simples et celles que le lecteur pourra trouver dans les références indiquées ; dans le même esprit la description de la réalisation effective de l'analyseur du système MIRABELLE n'a pas été incluse : elle a fait l'objet de la thèse de 3è cycle de G. Masini ([MAS 78]).

Enfin, dans un souci de clareté, les démonstrations non essentielles à la compréhension du texte ont été repoussées dans les annexes.

3.- NOTATIONS ET RAPPELS.

3.1.- SYSTÈME D'ÉQUATIONS À POINT FIXE.

Nous énonçons le théorème du point fixe que nous appliquerons ensuite dans le cadre simple des systèmes d'équations ensemblistes qui nous serviront aux chapitres suivants à décrire des ensembles définis par des équations récursives. Le vrai domaine d'application de ce théorème est la théorie des programmes (cf [LIV 78] par exemple), mais nous en ferons un usage plus simple ici.

3.1.1.- THÉORÈME DU POINT FIXE.

Définitions :

Un ensemble ordonné $(E, \leq)$ est *inductif* s'il admet un *plus petit élément* que nous noterons $\perp$ et si toute suite croissante $x_0 \ldots x_i \ldots$ d'éléments de E admet une *borne supérieure* notée $\sup_i (x_i)$

Une application f de E dans E est dite *continue* si pour toute suite croissant (x_i) la suite $f(x_i)$ admet une borne supérieure vérifiant

$$\sup_i(f(x_i)) = f(\sup_i(x_i)) \qquad \text{(passage à la limite)}$$

Toute fonction continue est nécessairement croissante :

si $a \leqslant b$ $\sup(a, b) = b$ or $f(\sup(a, b)) = \sup(f(a), f(b))$

donc $\sup(f(a), f(b)) = f(b)$.

Théorème du point fixe :

Soit $(E, \leqslant)$ un ensemble inductif ; soit f une applicati continue de E dans E ; l'équation $x = f(x)$ admet une plus petite solution $\mu(f)$ appeléeplus petit point fixe de

Démonstration :

Considérons la suite (x_i) définie par :

$x_0 = \perp$, $x_{n+1} = f(x_n)$

comme f est croissante, que $x_0 < x_1$, $x_i \leqslant x_{i+1}$;

notons $\mu(f) = \sup_i (x_i)$;

$f(\mu(f)) = \sup_i (f(x_i)) = f(\sup_i (x_i))$ par continuité ;

$\mu(f)$ est donc un point fixe de f

Pour tout autre point fixe z , vérifie :

$\perp \leqslant z$ et donc $x = f(\perp) \leqslant z$,... $x_n \leqslant f(z)$

et donc $\mu(f) \leqslant z$.

$\mu(f)$ est donc le plus petit point fixe de f .

Cette démonstration nous donne donc une méthode de construction par approximations successives du plus petit point fixe.

3.1.2.- Systèmes d'équation.

Quelques résultats d'abord pour justifier l'application du théorème du point fixe.

Proposition :

Soit E un ensemble, $\mathfrak{P}(E)$ l'ensemble de ses parties ; $(\mathfrak{P}(E))^n$ muni de l'inclusion est un ensemble inductif.

Démonstration :

La borne supérieure est obtenue par union et le plus petit élément est :

$$\perp = (\emptyset, \ldots, \emptyset)$$

Proposition :

soit f et g deux fonctions de E dans E ;
les applications de $\mathfrak{P}(E)$ dans $\mathfrak{P}(E)$

$$\overline{f} : A \longrightarrow \{f(x) \mid x \in A\}$$
$$\overline{f} \cup \overline{g} : A \longrightarrow \{f(x) \mid x \in A\} \cup \{g(x) \mid x \in A\}$$

sont continues (au sens des bornes supérieures).

Démonstration :

on a ici $\sup(A_i) = \bigcup_i (A_i)$ le résultat est alors la conséquence de : $\overline{h}(\bigcup_i A_i) = \bigcup_i \overline{h}(A_i)$.

Définition :

Soit E un ensemble et f_{ij} $i = 1, \ldots, n$ $j = 1, \ldots, n_i$ des fonctions de E sur E d'arité a_{ij} . On appelle *système d'équations à point fixe* sur $\mathfrak{P}(E)$ tout système d'équations de la forme :

$$(S) \qquad A_i = \bigcup_{j=1}^{n_i} \overline{f_{ij}}\,(B_1^{ij}, \ldots, B_{a_{ij}}^{ij}) \qquad i = 1, \ldots, n$$

avec B_k^{ij} étant soit un A_i, soit une partie de E.

On associe à (S) une application Φ_s de $(\mathcal{P}(E))^n$ dans lui-même définie par :

$$\Phi_s(E_1, \ldots, E_n) = (\Phi_1(E_1, \ldots, E_n), \ldots, \Phi_n(E_1, \ldots, E_n)$$

avec

$$\Phi_i(E_1, \ldots, E_n) = \bigcup_{j=1}^{n_i} \overline{f_{ij}}\,(C_1^{ij}, \ldots, C_{a_{ij}}^{ij})$$

avec

$$C_k^{ij} = E_p \quad \text{si} \quad B_k^{ij} = A_p$$

$$C_k^{ij} = B_k^{ij} \quad \text{si} \quad B_k^{ij} \in \mathcal{P}(E)$$

Une *solution* de (S) est un n-uple $(E_1, \ldots, E_n)$ vérifiant :

$$(E_1, \ldots, E_n) = \Phi_s(E_1, \ldots, E_n)$$

Proposition :

Tout système d'équations à point fixe admet une plus petite solution.

Démonstration :

Les propositions précédentes nous garantissent que nous sommes dans les hypothèses d'application du théorème du point fixe.

Exemple :

Sur $\mathcal{P}(\mathbb{N})$ considérons le système d'équations :

$$\begin{cases} A_1 = A_2 + A_2 \\ A_2 = 2 * A_2 \cup \{1\} \end{cases}$$

Il admet comme solution (E_1, E_2) avec

$$E_1 = \{x + y \mid x, y \in E_2\}$$

et

$$E_2 = \{2^n \mid n \geqslant 0\} \quad \text{ou} \quad E_2 = \{2^n \mid n \geqslant 0\} \cup \{0\}$$

Remarquons que la technique d'approximations successives qui nous donne la plus petite solution nous conduit à considérer :

$$x_0 = (\emptyset, \emptyset)$$

$$x_1 = (\emptyset, \{1\}) \quad = \Phi_s(x_0)$$

$$x_2 = (\{2\}, \{1, 2\}) = \Phi_s(x_1)$$

$$x_3 = (\{2, 3, 4\}, \{1, 2, 4\}) = \Phi_s(x_2)$$

et plus généralement

$$x_n = (R_n, P_n) \quad \text{avec} \quad P_n = \{2^k \mid k = 1, \ldots, n-1\}$$

$$R_n = \{x + y \mid x, y \in P_{n-1}\}$$

3.1.3.- Calcul du plus petit point fixe.

Nous nous plaçons dans l'hypothèse où le système d'équations à point fixe a une *plus petite solution finie*.

Dans ce cas, la méthode des approximations successives est une technique de calcul effective de cette solution et l'algorithme de calcul est :

initialisation de X_o à $(\emptyset, \ldots, \emptyset)$

pas 1 : calcul de $X_{k+1} = \Phi_s(X_k)$

pas 2 : tester si $X_{k+1} = X_k$; si oui alors c'est terminé
sinon recommencer au pas 1 avec $k + 1$

Optimisation du calcul :

Les (X_k) forment une suite croissante et il est donc inutile de recalculer pour X_{k+1} les éléments déjà calculés pour X_k.

Nous calculons donc à chaque étape Y_{k+1} qui contiendra tous les éléments de X_{k+1} n'appartenant pas à X_k .

L'algorithme devient alors :

initialisation : $X_o = (\emptyset, \ldots, \emptyset)$

$Y_1 = \Phi_s(X_o)$, $X_1 = X_o \cup Y_1$ (ici on désigne encore par $\cup$ l'union composante par composante)

pas 1 : calcul de Y_{k+1} :

$$Y_{k+1,i} = \{ x \mid x \notin X_{k,i}, \exists y \in X_k \quad x \in \Phi_{(i)}(y) \}$$

$$X_{k+1} = Y_{k+1} \cup X_k$$

pas 2 : si $Y_{k+1} = \emptyset$ alors c'est terminé

si $Y_{k+1} \neq \emptyset$ alors recommencer au pas 1 avec $k + 1$

Le calcul Y_{k+1} peut alors se faire sur les considérations suivantes :

si $y \in X_{k-1}$ alors $\Phi_s(y) \notin Y_{k+1}$ puisque $\Phi_s(y) \in X_k$

le calcul de Y_{k+1} se fera donc en n'utilisant que les éléments du complément de X_{k-1} dans X_k (c'est à dire ayant au moins une composante appartenant à un des Y_k) .

Exemple :

Reprenons le système donné §.1.3.2. :

$$\begin{cases} A_1 = A_2 + A_2 \\ A_2 = 2 * A_2 \cup \{1\} \end{cases}$$

$X_o = (\emptyset, \emptyset)$

$X_1 = (\emptyset, \{1\})$ $Y_1 = (\emptyset, \{1\})$

$X_2 = (\{2\}, \{1, 2\})$, $Y_2 = (\{2\}, \{2\})$

$X_3 = (\{2, 3, 4\}; \{1, 2, 4\})$ $Y_3 = (\{3, 4\}, \{4\})$

calcul de X_4 : calcul de la première composante : $A_2 + A_2$; on ne considérera donc que $\{1, 2, 4\} + \{4\} = \{5, 6, 8\}$

calcul de la seconde composante : $2 . A_2 \cup \{1\}$

on ne considère donc que

$$2 * \{4\} = \{8\}$$

d'où

$X_4 = (\{2, 3, 4, 5, 6, 8\}, \{1, 2, 4, 8\})$, $Y_4 = (\{5, 6, 8\}, \{8\})$.

L'annexe 1 étudie le gain obtenu par cette optimisation : il est proportionnel au nombre d'itérations nécessaires pour le calcul par approximations successives.

3.1.4.- LE PLUS GRAND POINT FIXE.

Nous aurons besoin au chapitre 3 du plus grand point fixe d'un système d'équations. Le théorème ci-dessous indique des conditions pour son existence et la démonstration nous donne la méthode de calcul.

Théorème :

Soit (S) un système d'équations ensemblistes sur $\mathfrak{P}(E)$ et $(f_i)_{i \in I}$ les fonctions qu'il fait intervenir. Si pour chaque $f_i : (E)^p \longrightarrow E$ la fonction image $\bar{f}_i : (\mathfrak{P}(E))^p \longrightarrow \mathfrak{P}(E)$ vérifie :

pour toute suite décroissante $((A_{k_j})j = 1, \ldots, p)_{k \in \mathbb{N}}$

$$f_i(\bigcap_k A_{k_j}, j = 1, \ldots p) = \bigcap_k f_i(A_{k_j}, j = 1, \ldots p)$$

alors (S) admet un plus petit point fixe.

Démonstration :

Soit P l'ensemble des solutions de (S) posons $X_0 = (E, \ldots, E)$

$\forall Y \in P \quad X_0 \supset Y$

Comme Φ_S est croissante

$X_0 \supset \Phi_S(X_0) \supset \Phi_S(Y) = Y$

et la suite $X_i = \Phi_S^i(X_0)$ est donc une suite décroissante contenant toutes les solutions de P ; les hypothèses nous permettent alors de passer à la limite et en posant $X = \bigcap_i \Phi_S^i(X_0)$ on aura

$$\Phi_{(S)}(X) = \bigcap_i \Phi_S^{i+1}(X) = X$$

et $\forall Y \in P \qquad X \supset Y$

X est donc le plus grand point fixe de (S)

Remarques :

Les hypothèses du théorème sont en particulier vraies chaque fois que E est fini, car dans ce cas toute suite décroissante est stationnaire.

Si le système d'équations comporte des opérations d'intersections, le plus grand point fixe est encore défini car l'intersection permet aussi de passer à la limite inférieure pour des suites décroissantes.

3.2.- Relations.

Soit E et F deux ensembles, une relation R de E dans F est un sous-ensemble de $E \times F$; si $(x, y) \in R$ on notera $x R y$.

L'image de x par R est l'ensemble R(x) défini par

$$R(x) = \{y \in F \mid x R y\} \quad .$$

Soit R une relation de E dans F et S une relation de F dans G on notera $R \circ S$ (attention à l'ordre), la relation définie par :

$$\forall x, z \quad (x R \circ S z \iff \exists y \in F \ (x R y \ \underline{et} \ y S z))$$

et R^{-1} désigne la relation de F dans E définie par :

$$\forall x, y \quad (y R^{-1} x \iff x R y) \quad .$$

R^{-1} est l'inverse de la relation R .

Soit R une relation de E dans E et Id la relation identité ; on appellera *fermeture transitive de* R la relation R^* définie comme plus petit point fixe de l'équation : $R^* = Id \cup R \circ R^*$.

On démontre sans difficulté que R^* est la plus petite relation réflexive et transitive contenant R .

On appellera *fermeture transitive stricte de* R^+ la plus petite solution du système d'équation :

$$R^+ = R \cup R \circ R^+$$

R^+ est la plus petite relation transitive contenant R et $R^+ \cup Id = R^*$.

3.3.- Grammaire à contexte libre, arbre syntaxique.

Le but de ce paragraphe est d'indiquer les notations utilisées. Le lecteur qui souhaite lire une présentation simple de la théorie des langages dans une perspective de reconnaissance des formes, pourra lire le premier chapitre de [FU 74] .

3.3.1.- Grammaire à contexte libre et langages algébriques.

Soit V un ensemble fini, nous noterons V^* l'ensemble des mots sur V , c.a.d. l'ensemble des suites finies d'éléments de V , soit $\alpha = a_1 \ldots a_n$ et $\beta = b_1 \ldots b_p$ deux mots sur V on notera $\alpha\beta$ la

concaténation de α et β :

$$\alpha\beta = a_1 \dots a_n \; b_1 \dots b_p$$

V^* muni de la concaténation est le *monoïde libre* engendré par V .

Nous noterons $\wedge$ le *mot vide*, élément neutre de la concaténation.

Une *grammaire à contexte libre* est un quadruplet $G = (N, T, \rightarrow, X)$ tel que :

- N est un ensemble fini appelé *vocabulaire non-terminal*
- T est un ensemble fini disjoint de N appelé *vocabulaire terminal*.
- $\rightarrow$ est une relation finie entre N et $(N \cup T)^*$ qui est l'ensemble des *productions* de G .
- X est un élément de N : l'*axiome* de la grammaire.

On notera $V = N \cup T$.

On associe à G une relation $\xrightarrow{*}$ sur $(N \cup T)^*$ appelée *dérivation* et définie par

$\xrightarrow{*}$ est la fermeture transitive de $\xrightarrow{'}$

$\xrightarrow{'}$ est la plus petite relation contenant $\rightarrow$ et stable par concaténation, c.a.d.

$$\forall \alpha, \beta \in V^* \quad A \rightarrow \lambda \quad \Rightarrow \quad \alpha A \beta \xrightarrow{'} \alpha \lambda \beta$$

Par abus de notations nous confondrons $\longrightarrow$ et $\xrightarrow{'}$.

Le *langage* L *engendré par* G est défini par :

$$L = \{\alpha \in T^* \mid X \xrightarrow{*} \alpha\}$$

Pour chaque $A \in N$ on pose :

$$L(A) = \{\alpha \in T^* \mid A \xrightarrow{*} \alpha\}$$

On peut de façon canonique associer à une grammaire à contexte libre G un *système d'équations* (S) défini par :

$$(S) : A = \bigcup_{A \rightarrow \lambda} \lambda \qquad \forall A \in N$$

Ce système admet une plus petite solution sur $\mathcal{P}(T^*)$ qui est $(L(A))_{A \in N}$: il a donc equivalence des deux definitions :

Exemple :

G : $X \longrightarrow a\,X\,b$ $\qquad T = \{a, b, c\}$

$X \longrightarrow A$ $\qquad N = \{X, A\}$

$A \longrightarrow cA$

$A \longrightarrow c$

(S) : $X = \{a\}\,X\,\{b\} \cup A$

$A = \{c\}\,A \cup \{c\}$

On montre par récurrence que

$$\forall m > 0 \quad A \xrightarrow{*} c^m$$

et $\quad A \xrightarrow{*} \alpha \Rightarrow \exists m > 0 \quad \alpha = c^m$.

et de là on en déduit

$$\forall m > 0 \;,\; \forall n \geqslant 0 \quad X \xrightarrow{*} a^n c^m b^n$$

et $\quad X \xrightarrow{*} \alpha \Rightarrow \exists n \geqslant 0 \;,\; m \geqslant 0 \quad \alpha = a^n c^m b^n$

Le raisonnement est cependant plus rapide sur le système d'équation : il suffit de montrer que :

$(\{a^n c^m b^n \mid n \geqslant 0 \,,\, m > 0\} \,,\, \{c^m \mid m > 0\})$ est la solution du système, et soit d'utiliser alors un théorème montrant que ce système n'admet qu'une solution, soit de montrer que les ensembles ci-dessus sont contenus dans toute solution.

3.3.2.- DESCRIPTION SYNTAXIQUE.

Les *ramifications sur* V [PAI 68] sont des suites finies d'arbres dont les noeuds sont étiquetés par des éléments de V et dont les fils de chaque noeud sont ordonnés. Il est commode de munir les ramifications de deux lois de compositions, l'une interne , notée + qui correspond à la concaténation, l'autre externe à opérateur dans V notée $\times$ qui est l'enracinement. La

figure 1.3. illustre comment opèrent + et x .

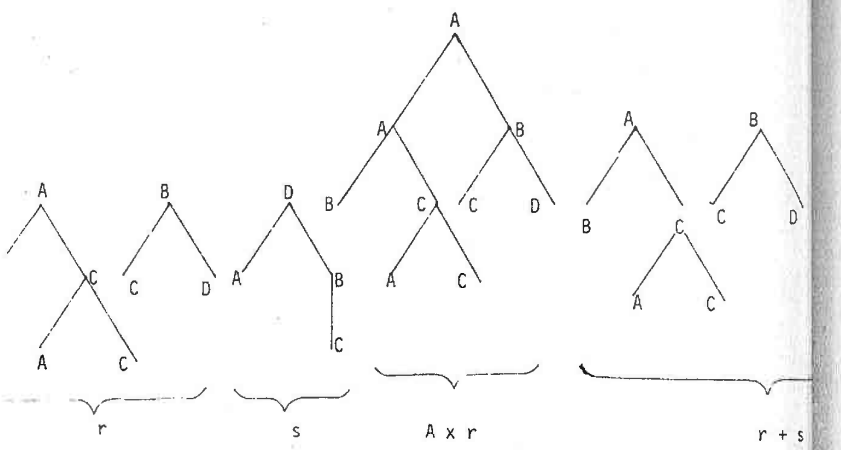


figure 1.3.

On note $\hat{V}$ l'ensemble des ramifications sur V

La ramification vide sera notée $\wedge$.

Toute ramification r non vide se décompose de façon unique en $r = (a \times s) + t$.

On peut alors définir par récurrence les fonctions suivantes :

ρ : (mot des racines) $V \longrightarrow V^*$

$\rho(\wedge) = \wedge$

$\rho(a \times r + t) = a\rho(t)$

pour la ramification r de la figure 1.3. $\rho(r) = AB$.

φ_T : (mot des feuilles dans T) $V \longrightarrow V^*$

$\varphi_T(\wedge) = \wedge$

$\varphi_T(a \times r + t) =$ si $r = \wedge$ et $a \in T$ alors $a\ \varphi_T(t)$

sinon $\varphi_T(r)\ \varphi_T(t)$

Pour la ramification de la figure 1.3. $\varphi_V(r) = B\ A\ C\ C\ D$.

Par abus de langage nous appellerons arbre une ramification n'ayant qu'une seule racine (la longueur du mot des racines est 1).

Soit $G = (N, T, \rightarrow, X)$ une grammaire ; $V = N \cup T$; soit $\alpha \in L(X)$ un mot défini par G . On appelle arbre syntaxique de α et associé à G toute ramification r de V vérifiant :

. $\varphi_T(r) = \alpha$

. pour tout noeud étiqueté par A vérifie :

$A \in N$ et alors les fils de ce noeuds forment un mot α tel $A \rightarrow \alpha$

$A \in T$ et A est une feuille.

Sauf mention contraire on supposera aussi que $\rho(r) = X$; il nous arrivera cependant parfois de préciser : arbre syntaxique de racine A .

Un arbre syntaxique associé à la grammaire donnée dans l'exemple précédent est dessiné figure 1.4. ; il donne la structure du mot aacccbb relativement à cette grammaire.

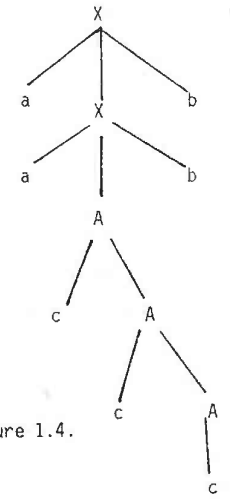


figure 1.4.

Le but de *l'analyse syntaxique* est de fournir la structure d'un mot conformément à une grammaire donnée ; cette structure est fournie par l'arbre syntaxique du mot. Si plusieurs arbres syntaxiques peuvent être associés à un mot , la grammaire est dite *ambigue*.

3.4.- DESCRIPTION DES ALGORITHMES.

Pour décrire nos algorithmes nous utiliserons un langage *peu strict* et inspiré de la notation l'Algol 60. Les types manipulés seront très divers : entiers, arbres, ensembles, ... Les objets pourront être indicés.

Instructions élémentaires :

Affectation : le symbole d'affectation sera ←

Exemple : A ← {x | x ∈ E et p(x)}

B ← racine (r) (ici r est une ramification et B un idendificateur repérant un mot)

(X, Y) ← (X + Y , X - Y) : affectation multiple.

Appel de procédure :

Exemple : imprimer (x, y, x + y)

Choix d'un élément : c'est une des instructions permettant d'introduire l'indéterminisme dans nos algorithmes.

Exemple : choix de x ∈ E tel que p(x) .

Instructions composées :

conditionnelle : si < expression booléenne > alors < liste d'instructio
sinon < liste d'instructio

ou cas < expression booléenne > alors < liste d'instructio
.....
< expression booléenne > alors < liste d'instructio
sinon < liste d'instructio

fin cas

la sémantique de ces instructions est immédiate.

itération : tant que < exp. booléenne > faire < liste d'instructions>
fait

ou

pour chaque x ∈ E faire < liste d'instructions>
fait

la sémantique de ces instructions mériterait d'être précisée pour une utilisation générale précise (modification de E en cours d'éxécution par exemple) mais pour l'usage qui en est fait ici ce n'est pas nécessaire.

choix : choix entre - < liste d'instructions>
.....
- < liste d'instructions> fin choix

Cette instruction, de signification intuitive évident, est la seconde façon d'introduire l'indéterminisme.

Procédure.

Il existe une procédure particulière - erreur - qui lorsqu'elle est appelée provoque l'arrêt du déroulement de l'algorithme. Comme dans tout algorithme indéterministe , cette procédure ne doit être activée que si aucun des choix précédents ne peut éviter cet incident.

déclaration de procédure :

elle suit la syntaxe suivante :

procédure < nom de la procédure > (< liste des paramètres d'entrée >)
résultat (< listes des types des résultats >)
< liste d'instructions du corps de procédure >
fin procédure

une instruction particulière de la procédure prépare des résultats ; sa syntaxe est : résultat : < liste d'expression >

Exemple : procédure xyz (r, s, a)

résultat (arbre)

si racine de (r) = racine de (s) alors résultat : r + s

sinon résultat : a x(r+s)

fin procédure.

Les appels de procédures pourront bien sûr être récursifs.

Commentaires :

Ils pourront être placés dans tout le texte et seront encadrés des symboles % .

Exemple :

```
% version indéterministe de la recherche d'un circuit Hamiltonien %
lire (E, Γ) ; % E est un ensemble, et Γ la relation du graphe %
n ← card(E) ;
choix de x1 ∈ E ; E ← E - {x1}
i ← 2 ;
tant que i < n faire
    % ici on a déjà construit une chaine depuis x1 jusqu'à x(i-1) ;
    E contient les points qui ne sont pas encore dans cette chaine %
    choix de xi ∈ E ;
    si non (x(i-1) Γ xi) alors erreur fsi
    i ← i + 1
fait
```



```
% on vérifie maintenant que le chemin est bien un circuit %
si non (xn Γ x1) alors erreur
       sinon imprimer (x1, ..., xn) fsi
```

On remarquera que l'indéterminisme permet de construire un algorithme s'éxécutant en un temps linéaire en n alors qu'il n'existe aucune version déterministe s'éxécutant en un temps polynomial en n .

CHAPITRE 2

DESCRIPTION ET ANALYSE DE DESSINS DANS LE SYSTEME MIRABELLE

CHAPITRE 2

DESCRIPTION ET ANALYSE DE DESSINS DANS LE SYSTÈME MIRABELLE

1- MODÈLE POUR LA DESCRIPTION DE DESSINS.

Les formes complexes vont être décrites comme un assemblage de sous-formes, avec au départ des formes élémentaires ou *primitives*. Nous allons d'abord définir les formes et les opérations d'assemblages nous permettant de construire des formes plus complexes, enfin les systèmes qui permettent de décrire des formes structurées.

1.1.- DÉFINITION DES FORMES.

1.1.1.- DÉFINITION.

Une *forme* est la classe des différentes *représentations* d'un concept : ainsi les différents dessins d'un arc de cercle constituent des représentations différentes d'une même forme : l'arc de cercle. Dans le cas particulier des dessins, on considérera que le translaté d'un dessin est encore un dessin représentant la même forme. De même la représentation d'un caractère A, même légèrement altéré par une déformation sera encore considérée comme une représentation de la forme A. Nous ne définirons pas ici ce qu'est une déformation (on pourra consulter par exemple [PAV 69]) car ce ne sera pas

F

figure 1.1. : trois représentations d'une même forme.

utile pour ce travail : les déformations sont prises en compte directement par l'algorithme de segmentation que nous utilisons ([BEL 78]) .

Nous supposerons donc dans ce chapitre et dans le suivant que toutes formes (c.a.d. classes de représentations) restent *invariante par translations dans le plan*. On pourrait sans aucune difficulté reprendre la même étude en considérant que les formes restent invariantes lorsqu'on fait opérer sur elles un groupe de transformations G ; cette généralisation et sans intérêt pour notre étude concrète ; elle permettrait , si l'on prend par exemple le groupe des déplacements du plan, de rendre compte des assemblages par raccordement de tangentes ([MOH 71]) .

On associe à chaque représentation de formes des *attributs* qui seront ici tous numériques :

- deux points particuliers de la représentation que nous appelerons *origine* et *extrémité* et qui seront schématisés respectivement par 0 et x sur les figures. Ces deux points n'ont aucune signification concrète pour la
- quatre valeurs correspondant aux valeurs extrêmes en abscisses et ordonnées de la représentation ; elles seront notées m_x et M_x , m_y et M_y .

(cf. figure 2.2.)

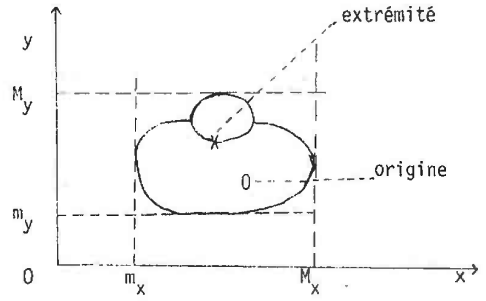


figure 2.2.

Formellement dans notre cas, une représentation r est un triplet (S, o, e où S est une partie du plan (le support visible de la forme), o , et e sont deux points du plan : l'origine et l'extrémité. La plus petite forme contenant la représentation r est $\{ \vec{t}(r) \mid \vec{t} \in$ translation du plan$\}$.

On définit pour la représentation r

$$m_x(r) = \inf(x) \quad (x, y) \in S$$

et similairement $m_y(r)$, $M_x(r)$ et $M_y(r)$

1.1.2.- Formes primitives dans le système MIRABELLE.

Dans notre système de description et d'analyse de dessins , nous nous sommes limités aux primitives suivantes :

classe I : segments de droites faisant un angle donné avec l'horizontale, quelque soit la longueur. L'origine et l'extrémité sont indiquées à la figure 2.2..

classe C : arcs de cercle faisant un angle donné avec l'horizontale comme indiqué à la figure 2.2.. Là encore les notions de courbure et de longueurs n'interviennent pas dans la définition.

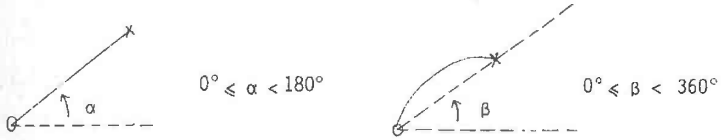


figure 2.2.

La définition des formes primitives se fait en indiquant : le nom de la forme primitive, l'angle α que doit faire cette primitive avec l'horizontale, la tolérance δ admise sur cette angle :

exemple : VERTICALE : T (90, 10)

tout segment de droite faisant un angle α , 80° < α < 100° appartiendra donc à la classe VERTICALE.

Remarque :

il est alors facile de définir des classes de primitives comme OBL1 classe des segments de droite oblique décrite par : OBL1 : T (45, 30).

1.2.- Assemblage de formes.

1.2.1. Assemblage par coïncidence.

On peut composer une nouvelle forme à partir de deux formes F_1 et F_2 en choisissant dans F_1 et F_2 les couples de représentations

dont des poles choisis coïncident.

On peut ainsi définir :

$F_1 + F_2 = \{(S, o, e) \mid \exists (S_i, o_i, e_i) \in F_i \ (i = 1, 2)$,

$o_2 = e_1$, $S = S_1 \cup S_2$, $o = o_1$, $e = e_2$ }

(concaténation bout à bout)

$F_1 \circ F_2 = \{(S, o, e) \mid \exists (S_i, o_i, e_i) \in F_i \ (i = 1, 2)$,

$o_1 = o_2$, $S = S_1 \cup S_2$, $o = o_1$, $e = e_2$ }

(mise en commun des origines)

$F_1 \times F_2 = \{ (S, o, e) \mid \exists (S_i, o_i, e_i) \in F_i \ (i = 1, 2)$

$e_1 = e_2$, $S = S_1 \cup S_2$, $o = o_1$, $e = e_2$ }

(mise en commun des extrémités)

$F_1 \underline{f} F_2 = \{(S, o, e) \mid \exists (S_i, o_i, e_i) \in F_i \ (i = 1, 2)$

$e_1 = e_2$, $o_1 = o_2$, $S = S_1 \cup S_2$, $o = o_1$, $e = e_1$)

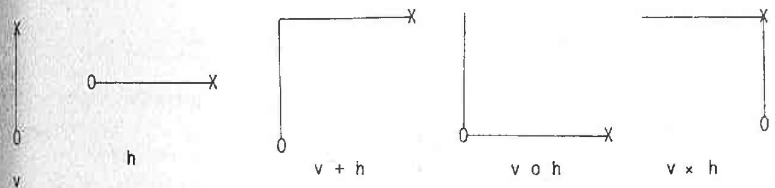
(mise en commun des origines et des extrémités).

Enfin par commodité d'utilisation introduisons le -unaire

$-F = \{(S, o, e) \mid (S, e, o) \in F \}$

(permutation des origines et extrémités).

La figure 2.3. illustre ces opérations. On remarquera que si les opérations +, o, x, - sont toujours définies, (car les classes sont invariantes par translation) le résultat de l'opération $\underline{f}$ peut au contraire être la *forme vide* ou *indéfinie* que nous noterons ω .



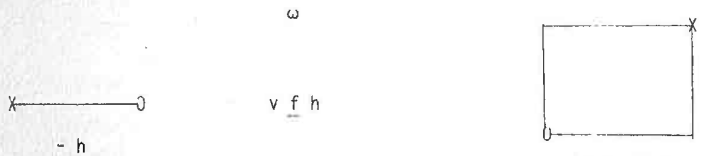


fig. 2.3.

Propriétés de ces opérations

. On démontre facilement que +, o, x et $\underline{f}$ sont associatives.

. De plus, elles permettent de construire toutes les "formes possibles" comme l'indique la proposition suivante :

Proposition :

Soit (S_i, o_i, e_i) $(i = 1, n)$ des représentations des formes F_i , soit $S = \bigcup_{i=1}^{n} S_i$, soit $G = (\{o_i\} \cup \{e_i\}, \Gamma)$ le graphe défini par $o_i \ \Gamma \ e_i$. Si G est connexe, il existe une forme F telle qu'il existe o_i et e_j telles que $(S, o_i, e_j) \in F$, F est obtenue à partir des F_i en utilisant les opérations $\{+, -\}$ ou les opérations $\{+, o, x\}$.

La démonstration se fait sans difficulté majeure par récurrence sur n (cf. [MOH 71]).

La proposition indique que $\{+, -\}$ et $\{+, o, x\}$ sont des ensembles minimaux. Pour faciliter les constructions et permettre la construction d'ensembles algébriques de formes plus larges (cf. § 1.3) , on gardera toutes les opérations introduites précédemment.

1.2.2.- CONCATÉNATIONS TOPOGRAPHIQUES

Les opérations de concaténations définies en 1.2.1. permettent des assemblages précis qui ne suffisent pas à rendre compte d'une classe suffisamment large de formes.

En particulier, on ne peut pas rendre compte du fait qu'une forme est au-dessus ou à l'intérieur d'une autre.

Soit R une *relation* portant sur des représentations, on peut alors définir :

$$R(F_1, F_2) = \{(S, o, e) \quad \exists r_i = (S_i, o_i, e_i) \in F_i (i = 1, 2) \wedge S = S_1 \cup S_2$$
$$R(r_1, r_2) \wedge o = o_1 \wedge e = e_1 \}$$

Pour simplifier les calculs lors de l'utilisation de ces relations nous nous sommes limités aux relations faisant intervenir les coordonnées extrêmes m_x , M_x, m_y, M_y associées aux représentations.

Une représentation donnée divise le plan en 9 régions et les différentes relations que nous avons définies , précisent les régions en relations (cf. figure 2.4.) .

Par exemple , INT (intérieur) correspond à la région 5 . On remarquera que l'oiseau est "intérieur" au chat sans que celui-ci l'ait mangé .

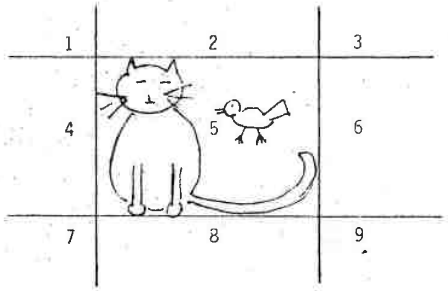


Figure 2.4.

Dans le système MIRABELLE , nous avons retenu les relations topographiques suivantes :

PSUR (posé sur)

PSUR (A, B) $\Leftrightarrow$ $m_y(A) = M_y(B)$ *et* $M_x(A) < M_x(B)$ *et* $m_x(A) > m_x(B)$

PSOUS (posé sous)

PSOUS (A, B) $\Leftrightarrow$ $m_y(B) = M_x(A)$ *et* $M_x(A) < M_x(B)$ *et* $m_x(A) > m_x(B)$

PDEVANT (posé devant)

PDEVANT (A, B) $\Leftrightarrow$ $m_y(A) > m_y(B)$ *et* $M_x(A) < M_x(B)$ *et* $m_x(A) > m_x(B)$

PCTRED (posé contre droite)

PCTRED (A, B) $\Leftrightarrow$ $m_x(A) = M_y(B)$ *et* $m_y(A) > m_y(B)$ *et* $M_y(A) < M_y(B)$

PCTREG (posé contre gauche)

PCTREG (A, B) $\Leftrightarrow$ $M_x(A) = m_x(B)$ *et* $m_y(A) > m_y(B)$ *et* $M_y(A) < M_y(B)$

SUR

SUR (A, B) $\Leftrightarrow$ $m_y(A) > M_y(B)$

SOUS

SOUS (A, B) $\Leftrightarrow$ $M_y(A) < m_y(B)$

INT (intérieur)

INT (A, B) $\Leftrightarrow$ $M_y(A) < M_y(B)$ *et* $m_y(A) > m_y(B)$ *et* $M_x(A) < M_x(B)$ *et* $m_x(A) > m_x(B)$

EXT (extérieur)

EXT (A, B) $\Leftrightarrow$ INT (B, A)

AGAUCHE

AGAUCHE (A, B) $\Leftrightarrow$ $M_x(A) < m_x(B)$

ADROITE

ADROITE (A, B) $\Leftrightarrow$ AGAUCHE (B, A)

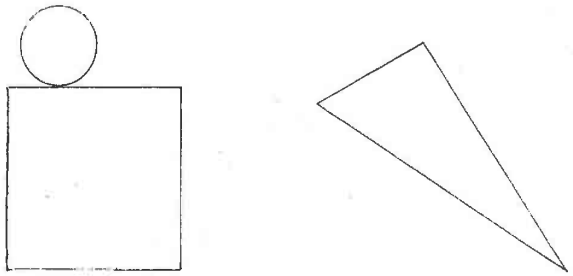
CDROIT (contre le côté droit)

CDROIT (A, B) $\Leftrightarrow$ $M_x(B) = m_x(A) \wedge M_y(B) > M_y(A) \wedge m_y(B) < m_y(A)$

CGAUCHE (contre le côté gauche)

CGAUCHE (A, B) $\Leftrightarrow M_x(A) = m_x(B) \wedge M_y(A) < M_y(B) \wedge m_y(B) < m_y(A)$

La figure suivante illustre AGAUCHE (PSUR (cercle, carrê), triangle).



1.3.- Système de descriptions.

Considérons le système d'équations ensemblistes suivant , portant sur des formes :

montagne = mont $\cup$ montagne + mont

mont = h $\underline{f}$ (ob_1 + ob_2)

Supposons que h, ob_1 et ob_2 désignent les formes dont des représentations sont données figure 2.5. et que les tailles des représentations peuvent être quelconques dans chaque forme. La figure 2.6. donne alors une représentation de mont, la figure 2.7. une représentation de montagne.

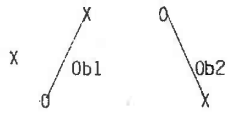


figure 2.5.

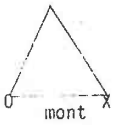


figure 2.6.

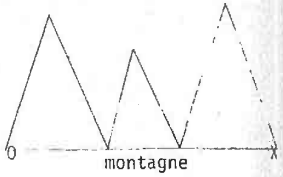


figure 2.7.

La première équation indique que l'ensemble montagne contient mont, mais contient aussi mont + montagne.

On se convainc aisément que la *plus petite solution* de cette équation récursive est mont, mont + mont, (mont + mont) + mont, ...

Ce système d'équations a donc non seulement défini la forme montagne, mais de plus , indique que chaque représentation de cette forme se décompose en ((mont +...)...)+ mont et ainsi fournit une *structure* de cette représentation : l'arbre syntaxique associé à la forme.

Souvent, lorsque nous voudrons distinguer chacune des décompositions possibles d'une forme, nous écrirons le système sous forme de *règles* (par analogies avec les notations de la théorie des langages):

montagne → mont
montagne → montagne + mont
mont → h $\underline{f}$ (ob_1 + ob_2)

1.3.1. Système d'équation.

Un *système d'équations* S d'inconnues $\{A_1, \ldots, A_n\}$ = N , utilisant les primitives $a_1, \ldots, a_p$ est un ensemble de n équations :

$$A_i = \bigcup_{j=1}^{p_i} \alpha_{ij} (A_1, \ldots, A_n, a_1, \ldots, a_p) \quad (i = 1, \ldots, n)$$

où α_{ij} est une formule utilisant les relations et les opérations de concaténation portant sur des formes primitives et/ou des inconnues.

Comme nous l'avons vu chapitre 1 , § 1.3.1., un tel système d'équations S admet toujours une *solution minimale* ($S(A_1)$, ... , $S(A_n)$) sur le treillis des formes muni de l'inclusion (ce résultat est le cas particulier d'un résultat plus général qui est démontré dans [MOH 75]) ; $S(A_i)$ est la forme décrite par l'inconnue A_i ; $S(A_1)$ est particularisée : c'est la forme décrite par S .

Exemple de description :

Ici les primitives sont les classes suivantes, invariantes par homothétie (et par translation par voie de conséquences) :

h : segments horizontaux 

v : segments verticaux

d : "obliques droites" 

g : "obliques gauches" 

a : arcs de cercles verticaux concaves à droite 

b : arcs de cercles verticaux concaves à gauche 

e : arcs de cercles horizontaux de concavité supérieure 

f : arcs de cercles horizontaux de concavité inférieure 

k : arcs de cercles obliques de concavité inférieure 

On peut aboutir à la description suivante de voiliers (figure 2.8.)

scène = PSUR (voilier, eau)

eau = e + eau U e

voilier = PSUR (voiles, coque)

coque = g + h + -d

voiles = base + (mât f grande voile) U base + (mât f grande voile x foc)

base = v

mât = v

grande voile = (h x cordage + d) f a

foc = cordage + (((f x cordage) + k) f b)

cordage = v U d U g

où PSUR est la relation topologique spécifiée précédemment

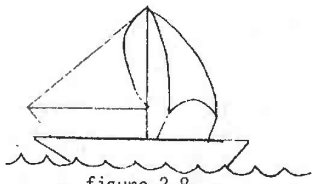


figure 2.8.

1.3.2. Sous formes d'une forme.

Soit un système S et soit N l'ensemble de ses inconnues, T l'ensemble de ses formes primitives.

Soit f une représentation d'une forme de S(A) : définissons ce qu'est un *arbre syntaxique* associé à f :

. si f est une forme primitive a , l'arbre syntaxique associé à f est l'arbre à un noeud (f, a)

. si f est la représentation d'une forme non primitive $f \in S(A)$ et il existe au moins un terme de l'équation définissant l'inconnue A tel que

$$f \in w(S(A_1), \ldots, S(A_i))$$

il existe donc $f_1 \in A_1, \ldots, f_k \in A_k$ tels que $f = w(f_1, \ldots, f_k)$. soient alors $t_1, \ldots, t_k$ les arbres syntaxiques associés à $f_1, \ldots, f_k$ alors

$(f, A) \times (t_1 + \ldots + t_k)$ est un arbre syntaxique associé à f

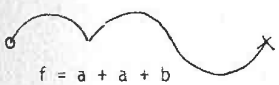
(ici + et x sont les opérations de ramification (cf. chapitre 1, §.3.3.)).

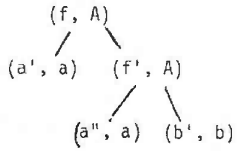
Exemple :

soit le système

A = a + A U b où a et b sont les primitives suivantes :

L'arbre syntaxique (il est unique ici) associe à la forme a + a + b est donné ci-dessous, on remarquera que l'on a distingué les deux représentations de a présentes :





On définit alors comme *sous formes* d'une représentation f l'ensemble les formes apparaissant dans au moins un arbre syntaxique de f .

Si une forme f admet plus d'un arbre syntaxique le système sera dit *ambigu*.
On supposera le plus souvent que les systèmes ne sont pas ambigus : à chaque représentation d'une forme sera donc associé au plus un arbre syntaxique.

1.3.3. DÉRIVATION.

Dans certaines démonstrations où l'on raisonne cas par cas, on préfère utiliser les dérivations sur les formules plutôt que sur le système d'équations ce dernier permettant par ailleurs, un raisonnement plus global. C'est pourquoi nous définissons ici les dérivations.

Soit $G = (N, T, \rightarrow, A_1)$ une grammaire à contexte libre,

avec $N = \{ A_1, \ldots, A_n \}$ le vocabulaire non-terminal

$T = \{a_1, \ldots, a_p\} \cup \{+, o, x, -, f, (,)\}$

$\cup \{ POSESUR, \ldots, INT \}$ le vocabulaire terminal et vérifiant, pour toute règle $A \rightarrow \mu$, que μ est une formule *complètement parenthésée*.

On peut associer de façon bijective à G le système d'équations S :

$$(S) : A_i = \bigcup_{A_i \rightarrow \mu} \mu$$

On peut alors définir

$$\mathcal{L}(A_i) = \{ \alpha \mid A_i \overset{*}{\rightarrow} \alpha \}$$

le langage des formules engendrées par A_i .

Posons alors

$\mathcal{S}(A_i) = I(\mathcal{L}(A_i))$ où I est l'interprétation qui à chaque formule associe la forme correspondante (éventuellement indéfinie).

Le résultat suivant montre l'équivalence des deux approches :

Théorème :

Soit S un système à point fixe d'inconnues $A_1, \ldots, A_n$; G la grammaire associée ; avec les notations précédentes

$(\mathcal{S}(A_i))_{i=1,\ldots,n}$ est la solution minimale de S

La démonstration relativement technique pourra être trouvée dans [MOH 75].

La bijection entre système à point fixe et grammaire étant évidente, nous utiliserons selon les paragraphes, tantôt une terminologie, tantôt l'autre.

Exemple :

Soit le système suivant :

scène = INTERIEUR (rectangle, rectangle)
Rectangle = (hauteur + largeur) f (largeur + hauteur)

hauteur = v U v + hauteur

largeur = h U h + largeur

où v et h sont des segments verticaux et horizontaux de longueur unité définie à une translation près.

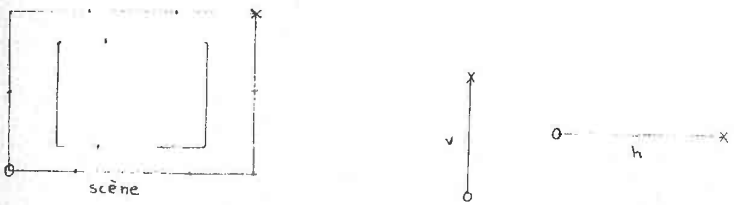


figure 2.9

Avec les règles de réécriture associées à un tel système , nous pouvons avoir la dérivation suivante (en supprimant les parenthèses inutiles)

scène → INTERIEUR (rectangle, rectangle)

→ INTERIEUR((hauteur + largeur) f (largeur + hauteur) , rectang

.....

→ INTERIEUR ((v + v + h + h + h)f(h + h + h + v + v),

(v + h + h) f (h + h + v))

L'interprétation de cette dernière formule donnant une forme de ‹scène› dont une représentation est donnée à la figure 2.9..

Remarque :

Si l'on engendre des figures à partir des formules obtenues par dérivation on risque d'obtenir des formes indéfinies :

ainsi par dérivation on obtient :

rectangle $\overset{*}{\rightarrow}$ (h + v + v) f (v + h + h + h)

or l'interprétation de cette dernière formule conduit à la forme indéfinie car la double coïncidence des poles est impossible ; de même :

scène $\overset{*}{\rightarrow}$ INTERIEUR ((h + v) f (v + h),(h + v + v) f (v + v + h)) .

L'interprétation de chacune des composantes rectangles donne effectivement une forme, mais la taille du premier rectangle étant inférieure à celle du second, la relation INTERIEUR ne peut jamais être vérifiée ; donc l'interprétation de cette formule conduit aussi à la forme indéfinie.

Ce problème posé par les formules dont l'interprétation est indéfinie, montre bien que les systèmes sont plutôt des *systèmes de description* que des *systèmes de génération*. La réunion de ces deux aspects dans les grammaires de Chomsky n'est donc plus réalisée ici ; ce n'est pas gênant pour cette étude où nous ne nous intéressons qu'à la reconnaissance de formes que nous observons réellement et qui sont par conséquent l'interprétation d'une formule.

1.3.4.- SYSTÈME RÉDUIT.

La notion de réduction est celle que l'on trouve en théorie des langages ; elle permet d'éliminer dans un système les équations inutiles.

Soit F(fils) le graphe sur N U T défini par A F B si et seulement si B apparaît dans le second membre d'une équation de premier membre A .

Un système est *réduit supérieurement* si pour tout élément B de N U T , il existe un chemin de A_1 (l'axiome) à B . On dit aussi que B est accessible.

Un système est *réduit inférieurement* si pour toute inconnue A $S(A) \neq \emptyset$. On dit aussi que A est productif.

Un système est *réduit* s'il est à la fois réduit inférieurement et supérieurement.

Proposition :

tout système admet un système réduit équivalent.

Démonstration :

il suffit d'abord d'opérer une réduction inférieure en supprimant toutes les équations faisant intervenir les inconnues A telles que $S(A) = \emptyset$; le nouveau système est évidemment équivalent au système initial. On procède alors à une réduction supérieure en supprimant toutes les inconnues et primitives non accessibles : le système reste réduit inférieurement et est encore équivalent au système initial.

Exemple :

$$\begin{cases} A = a \cup a + B + C \\ B = b \cup b + B \\ C = c + C \end{cases}$$

On a $S(C) = \emptyset$ et donc on aboutit au système équivalent

$$A = a$$
$$B = b \cup b + B$$

b et B sont inaccessibles ; on aboutit alors au système réduit :

$$A = a \quad .$$

2- UN ALGORITHME D'ANALYSE SYNTAXIQUE.

Les algorithmes d'analyse syntaxique procèdent généralement de la gauche vers la droite ; l'intérêt évident est de pouvoir les dérouler au fur et à mesure de la lecture de la phrase à analyser. Pour des formes linéaires comme le discours , ces méthodes présentent encore un intérêt certain, mais pour des formes bidimensionnelles où la notion de début et de fin n'a plus de sens, ces techniques d'analyse deviennent caduques. L'algorithme fondamental que nous exposons dans ce chapitre permet de démarrer l'analyse en n'importe quel point de la phrase et de propager cette analyse ensuite à partir de ce point. Même dans le cas de la parole, une telle technique est intéressante car elle permet de démarrer l'analyse à partir d'un point intéressant : mot identifié avec certitude ou sémantiquement important ([MAR 79]).

2.1.- PRINCIPE.

Soit $G = (N, T, \longrightarrow, X)$ une grammaire à contexte libre. Posons $V = N \cup T$

On supposera qu'il n'existe pas de règles de production du type :

$$A \longrightarrow \Lambda$$

où Λ désigne le mot vide.

Cette restriction qui n'est pas obligatoire, simplifie l'étude que nous allons faire. Par ailleurs, on peut toujours se ramener à ce cas.

Le déroulement de l'algorithme est alors le suivant :

Soit a_k un élément de la chaîne à analyser.

L'analyseur commence par choisir une règle $A \rightarrow \lambda a_k \lambda'$ (λ , $\lambda' \in V^*$).

On obtient alors un arbre syntaxique de racine A et de feuilles $\lambda a_k \lambda'$ que l'on complète vers le bas par deux analyses descendantes, sauf si bien sûr λ ou λ' appartiennent à T^* .

Une analyse procédant de la gauche vers la droite complètera λ' , la seconde de la droite vers la gauche complètera l'arbre sous λ (cf. figure 2.10).

Ensuite, le processus est itéré : on recherche une règle $B \longrightarrow \mu A \mu'$ $(\mu , \mu' \in V^*)$ on complète l'arbre syntaxique

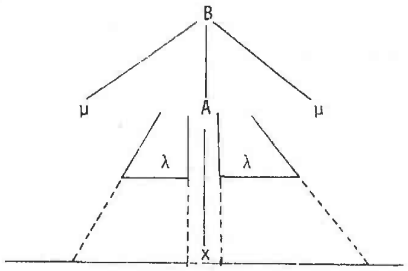


Figure 2.10.

sous μ et μ' par une analyse descendante et ainsi de suite jusqu'à la fin de l'analyse.

Plus formellement on obtient la procédure d'analyse suivante :

soit $a_1 \ldots a_n$ la chaîne à analyser, soit t l'arbre déjà construit et soit i (resp. j) l'indice du premier terminal à gauche (resp. à droite) non encore analysé (cf. figure 2.11.) .

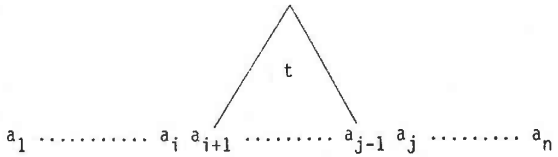


figure 2.11.

Si l'analyse a débuté avec le terminal a_k , l'appel de la procédure d'analyse est le suivant :

$i \leftarrow k - 1$; $j \leftarrow k + 1$;

arbre syntaxique ⟵ REMONTE (a_k)

Et la procédure REMONTE s'écrit ainsi :

(i, j, n sont des variables globales)

procédure REMONTE (t) résultat (arbre)

% t est un arbre syntaxique partiel et complet vers le bas correspondant à la chaîne $a_{i+1} \ldots a_{j-1}$; le résultat de Remonte est l'arbre syntaxique complet de la phrase % .

B = racine de t :

si B = X et i = 0 et j = n + 1 alors % fin de l'analyse %

résultat : t

sinon choisir une règle $A \longrightarrow C_1 \ldots C_k \; B \; D_1 \ldots D_q$;

% on complète maintenant par deux analyses descendantes l'arbre sous $C_1 \ldots C_k$ et sous $D_1 \ldots D_q$ %

pour m de k à 1 faire $r_m \leftarrow$ AGAUCHE (C_m) ;

pour m de 1 à q faire $s_m \leftarrow$ ADROITE (D_m) ;

résultat : REMONTE $(A \times (r_1 + \ldots + r_k + t + s_1 + \ldots + s_q))$.

fsi

fin procédure

La procédure ADROITE (resp. AGAUCHE) est une procédure d'analyse syntaxique descendante classique procédant de gauche à droite (resp. de droite à gauche) qui retourne comme résultat un arbre.

Par exemple pour AGAUCHE :

procédure AGAUCHE (C) résultat (arbre)

si $C \in N$ alors choisir une règle $C \rightarrow B_1 \ldots B_q$;

pour m de q à 1 faire $t_m \leftarrow$ AGAUCHE (B_m) ;

résultat $(C \times (t_1 + \ldots + t_q))$

sinon % $C \in T$ %

si $C \neq a_i$; alors erreur

sinon $i \leftarrow i - 1$; résultat (a_{i+1})

fsi

fsi

Dans cette dernière procédure, il faut comprendre "erreur" au sens de la primitive FAIL dans SNOBOL 4 : l'erreur peut être due soit à un mauvais choix soit à une erreur dans la chaîne à analyser ; on essaiera donc tous les choix par une technique de retour arrière avant de conclure à une erreur dans la chaîne.

2.2.- MISE EN OEUVRE DE CETTE ALGORITHME.

2.2.1.- L'ANALYSE DESCENDANTE.

Les procédures ADROITE et AGAUCHE sont des procédures d'analyse syntaxique descendante procédant respectivement de la gauche vers la droite et de la droite vers la gauche. Les techniques d'analyse descendante sont bien connues et nous n'étudierons pas ces algorithmes ici (voir par exemple [AHO 73]) .

Remarquons que si nous voulons que ces deux procédures soient déterministes avec lecture de k caractères à l'avance G devra être LL(k) (notation de Knuth [KNU 65]) , ainsi que $\widetilde{G}$, sa grammaire réfléchie. Mais s'il est en général facile de construire une grammaire qui soit LL(1) ce n'est plus le cas si l'on veut que simultanément $\widetilde{G}$ le soit aussi ; en particulier, la première condition interdit la récursivité à gauche et la seconde la récursivité à droite. On préfèrera donc construire une grammaire G LL(k) et une grammaire G' LL(k) ayant les mêmes classes syntaxiques que G et définissant le langage réfléchi de G .

Exemple :

G : $X \to a\ X \mid A$
$A \to B \mid cC$
$B \to bcc$
$C \to ac \mid Ac$

$\widetilde{G}$: $X \overset{\sim}{\to} A \mid Xa$
$A \overset{\sim}{\to} B \mid Cc$
$B \overset{\sim}{\to} ccb$
$C \overset{\sim}{\to} ca \mid cA$

on peut choisir par exemple pour G'

G' : $X \to A\ D$
$D \to \Lambda \mid aA'$
$A \to c\ E$
$E \to c\ F \mid ac$
$F \to E\ c \mid b$
$B \to ccb$
$C \to c\ G$
$G \to a \mid A$

G' est LL(1) alors que $\widetilde{G}$ n'est LL(k) pour aucun k .

On remarquera que G' n'est pas réduite (le non-terminal B n'est pas accessible depuis l'axiome) mais les règles définissant B et C sont nécessaires pour définir les mêmes catégories syntaxiques que G .

2.2.2.- L'ANALYSE ASCENDANTE ET LE CONTEXTE.

La procédure REMONTE choisit une règle $A \to \lambda B \lambda'$. Le choix de cette règle doit permettre de poursuivre l'analyse avec λ et λ' . Les caractères à droite de la sous-chaine analysée $a_j, a_{j+1} \ldots$ doivent donc permettre l'analyse à partir de λ' et ceux situés à gauche $(..a_{i-1}, a_i)$ l'analyse à partir de λ .

Un test sur ce *contexte* droit et gauche va donc permettre de limiter le nombre de choix possibles.

Définissons ce contexte : (ces notations sont tirées de [PAI 73])

Soit $\mathfrak{I}$ ($\mathfrak{I}$ comme initiale) la relation définie sur le vocabulaire V de la grammaire :

$A\ \mathfrak{I}\ B \iff \exists \mu \in V^*\ \ A \to B\mu$

Soit $\mathfrak{I}^*$ la fermeture transitive de $\mathfrak{I}$; il est évident que :

$A\ \mathfrak{I}^* B \iff \exists \mu \in V^*\ \ A \overset{*}{\to} B\mu$.

De même définissons $\mathfrak{F}$ ($\mathfrak{F}$ comme finale) :

$A\ \mathfrak{F}\ B \iff \exists \mu \in V^*\ \ A \to \mu B$.

Soit $\mathfrak{F}^*$ la fermeture transitive de $\mathfrak{F}$; il est évident que :

$A \ \mathfrak{F}^* B \iff \exists \mu \in V^* \quad A \overset{*}{\to} \mu B$

Soit enfin $\mathcal{J}$ (successeur) défini sur $V \times V$ par $A \ \mathcal{J} \ B \iff$ $\exists \mu, \mu' \in V^*$, $\exists C \in N \quad C \to \mu AB \mu'$

On peut alors définir la relation $\mathcal{V}$ (voisin) par $\mathcal{V} = \mathfrak{F}^{*-1}_{o} \ \mathcal{J}_o \mathfrak{J}^*$

Propriété :

$A \ \mathcal{V} \ B \iff \exists C \ \exists \mu, \mu' \in V^* \quad C \overset{*}{\to} \mu AB \mu'$

Démonstration :

partie directe

$$A \ \mathcal{V} \ B \iff A \mathfrak{F}^{*-1}_{o} \ \mathcal{J}_o \ \mathfrak{J}^* B$$

$$\iff \exists U, V \quad U \mathfrak{F}^* A, \quad U \mathcal{J} V, \quad V \mathfrak{J}^* B$$

$$\iff \exists \lambda, \lambda', \mu, \mu \in V^* \quad \exists C \in M$$

$$C \to \lambda U V \lambda', \quad U \overset{*}{\to} \mu A, \quad V \overset{*}{\to} B \mu'$$

$$\Rightarrow \exists \mu \mu', \exists C \in M \quad C \overset{*}{\to} \mu AB \mu'$$

La réciproque se démontre par récurrence sur l'entier n tel que : $C \overset{n}{\to} \mu AB \mu'$.

Si nous reprenons la grammaire G de § 2.2.1, nous avons par exemple :

$c \mathcal{J} C \ \mathfrak{J} A \ \mathfrak{J} B \ \mathfrak{J} b \quad$ donc $c \ \mathcal{V} \ b$

(il faut noter que $\mathfrak{F}^*$ contient $\mathfrak{F}^o$: l'égalité).

On peut alors définir $\mathcal{C}_r$: contexte pour la règle r .

Pour simplifier la définition de cette relation nous supposerons que chaque terme du vocabulaire V n'apparaît au plus qu'une seule fois dans le second membre de la règle. Nous distinguerons selon la forme de la règle r les différents cas :

la règle r est de la forme $A \to \mu CBD \mu' \ /r/ \ (\mu, \mu' \in V^*)$

$$\mathcal{C}_r(B) = (\mathfrak{F}^{*-1}(C) \times \mathfrak{J}^{*-1}(D)) \cap (T \times T)$$

* la règle r est de la forme $A \to \mu CB \quad (\mu \in V^*)$

$$\mathcal{C}_r(B) = (\mathfrak{F}^{*-1}(C) \times \mathcal{V}(A)) \cap (T \times T)$$

* la règle r est de la forme $A \to BD\mu \quad (\mu \in V^*)$

$$\mathcal{C}_r(B) = (\mathcal{V}^{-1}(A) \times \mathfrak{J}^*(D)) \cap (T \times T)$$

* la règle r est de la forme $A \to B$

$$\mathcal{C}_r(B) = \bigcup_s \mathcal{C}_s(A)$$

On conviendra que si B n'apparaît pas dans le second nombre de la règle r : $\mathcal{C}_r(B) = \emptyset$.

Par convention, on introduit deux délimiteurs de la phrase à reconnaître : $\vdash$ et $\dashv$ et l'on conviendra qu'il existe une règle factice

$X' \to \ \vdash X \dashv \ /0/$

de numéro 0 . Par conséquent

$$\mathcal{C}_0(X) = \{(\vdash, \dashv)\}$$

Dans la procédure REMONTE , le choix de la règle $A \to \backslash \ B \lambda'/r/$ ne sera fait que si

$$(a_i, a_j) \in \mathcal{C}_r(B) \ .$$

Ce point est justifié par la proposition suivante qui précise que l'utilisation de la règle r impose que (a_i, a_j) appartiennent en contexte.

Proposition :

Soit la règle $r \quad A \to \mu \ B \mu'$;

$(a_i, a_j) \in \mathcal{C}_r(B)$ si et seulement si

$\exists \lambda, \lambda' , \gamma, \gamma' \in V^*$

$(X \overset{*}{\to} \lambda A \lambda' \to \lambda \mu B \mu' \lambda'$

et $\lambda \mu \overset{*}{\to} \gamma a_i$ et $\mu' \lambda' \overset{*}{\to} a_j \gamma')$

Démonstration :

il faut distinguer les quatre formes possibles de la règle r :

nous ne ferons la démonstration que pour le cas où la règle est, par exemple, de la forme $A \longrightarrow \alpha\ CB$.

Dans ce cas :

$$a_j \in \mathcal{V}(A) \quad , a_i \in \mathcal{F}^{-1}(C)$$

$$\Leftrightarrow \quad \exists \beta, \beta' \quad X \xrightarrow{*} \beta A a_j \beta'$$
$$\exists \delta \qquad C \xrightarrow{*} \delta a_i$$

$$\Leftrightarrow \quad X \xrightarrow{*} \beta A a_j \beta' \longrightarrow \beta \alpha CB a_j \beta'$$
$$\beta \alpha C \xrightarrow{*} \gamma a_i$$
$$a_j \beta \xrightarrow{*} a_j \beta$$

La démonstration est similaire dans les autres cas.

Remarque :

Cette définition est plus fine que celle qui avait été donnée dans [HAT 77] et qui ne vérifiait pas la proposition précédente ; considérons par exemple, la grammaire suivante

$X \longrightarrow a\ A$ /1/ $\quad A\ d$ /2/

$A \longrightarrow B$ /3/ $\quad a\ B\ b$ /4/

$B \longrightarrow b\ B\ a$ /5/ $\quad ba$ /6/

$\mathcal{C}_1(A) = \{(a, \dashv)\}$, $\quad \mathcal{C}_2(A) = \{(\vdash, d)\}$

$\mathcal{C}_3(B) = \{(a, \dashv)\ , (\vdash, d)\}$ $\quad \mathcal{C}_4(B) = \{(a, b)\}$

$\mathcal{C}_5(B) = \{(b, a)\}$

On remarque donc que lors de l'éxécution de REMONTE, si B est la racine du sous-arbre déjà analysé, le choix entre les règles 3, 4 ou 5 peut se faire de façon déterministe en testant les contextes puisque

$$\mathcal{C}_j(B) \cap \mathcal{C}_i(B) = \emptyset \quad (i \neq j)$$

Par contre, dans la définition du contexte proposée en [HAT 77] , en utilisant les notations ci-dessus, on aurait eu :

$\mathcal{C}_3(B) = \{(\vdash, \dashv)\ , (a, b)\ , (\vdash, b), (a, \dashv)\}$

$\mathcal{C}_4(B) = \{(a, b)\}$

Donc, dans le contexte (a, b) la procédure REMONTE n'aurait pas de critère pour choisir entre la règle 3 et la règle 4.

2.3.- VERSION ASCENDANTE DE L'ALGORITHME.

On peut aussi, plutôt que d'analyser alternativement en phase ascendante puis descendante, n'opérer que en phase ascendante. L'algorithme devient avec les notations utilisées § 2.1. :

choix d'un caractère a_k dans la chaîne ; $t \leftarrow a_k$.

$i \leftarrow k - 1$; $j \leftarrow k + 1$:

$\rho \leftarrow$ racine de t

tant que $\rho \neq X$ et $i \neq 0$ et $j \neq n + 1$ faire

choix entre

- % lecture à droite % $t \leftarrow t + a_j$; $j \leftarrow j + 1$
- % lecture à gauche + $t \leftarrow a_i + t$; $i \leftarrow i - 1$
- % réduction en appliquant une règle sur une partie de la racine de t %
  choix de r_1, r_2, r_3 tels que $r_1 + r_2 + r_3 = t$;
  choix d'une règle $A \rightarrow$ racine de r_2 ; $t \leftarrow r_1 + A \times r_2 + r_3$;

ρ ← racine de t

fait

Cet algorithme comporte beaucoup de choix et va donc être d'une utilisation particulièrement souple. Il a été implanté par Mari [MAR 79] , pour l'analyse du discours continu avec la restriction suivante : au second choix , r_1 et r_3 sont vides.

Les techniques de codage du mot des racines de la ramification t par un automate linéaire ne permettent malheureusement pas l'adaptation de cet algorithme à des formes bidimensionnelles, aussi, nous ne détaillerons pas ici la mise en oeuvre de cet algorithme.

3- GÉNÉRALISATION À L'ANALYSE DE DESSINS.

Nous reprenons ici la procédure donnée §.2. pour l'analyse des chaînes en la transposant pour l'analyse des dessins décrits dans le système MIRABELLE. Pour simplifier l'exposé , nous nous limiterons au cas où toutes les règles sont de la forme :

A ⟶ B op C

ou

A ⟶ RELATION (B, C) B, C ∈ V **U** - V

On peut toujours se ramener à ce cas, mais cette restriction n'est pas imposée dans notre système.

Les procédures vont travailler à l'intérieur de zones du plan dans lesquelles, elles vont avoir à opérer la reconnaissance d'une sous-forme donnée ; ceci explique l'introduction des paramètres supplémentaires qui n'apparaissaient pas dans l'algorithme donné §2.1..

La description de l'algorithme est générale, car nous voulons simplement poser les problèmes ; une étude décrivant la réalisation effective de cet analyseur a été faite par Masini [MAS 78] .

3.1.- PHASE ASCENDANTE.

```
procédure REMONTE (% zone d'analyse % Z ,
                   % arbre syntaxique déjà construit % t ,
                   % origine et extrémité de la forme correspondant
                     à t : % O, E ,
                   % sous forme à reconnaître % But)
        résultat (% il s'agit d'un triplet :
                     un arbre syntaxique de racine But % arbre ,
                   % l'origine et l'extrémité de la forme correspondante %
                     point du plan ,
                     point du plan )
% toutes les variables sont locales aux procédures, leurs types sont implicites ; en particulier t' et t" sont des arbres syntaxiques servant à garder des analyses partielles %
soit B la racine de t ;
si B = But alors % analyse terminée % résultat : (t, O, E)

sinon choix d'une règle A ⟶ λB λ' ;

 cas   la règle est A → B + C alors
                        (t", E') ← ADROITE(Z, C, E)
                        t' ← t ; O' ← O ;
                        % analyse de la forme C en partant de
                        l'extrémité E de B ; l'extrémité E'
                        de la forme C devient l'extrémité de
                        la forme B + C %
       la règle est A → B∘C  alors (t", E') ← ADROITE (Z, C, O) ;
                                   t' ← t ; O' ← O ;
                        % analyse de C en partant de l'origine
                        de B cette fois %
       la règle est A → B × C  alors (t", O") ← AGAUCHE (Z, C, E) ;
                                     t' ← t ; O' ← E ; E' ← E ;
                        % analyse de C en partant de l'extrémité E de B qui correspond à l'extrémité de C ; l'origine O' et l'extrémité E' du résultat restent celles de B %
```

la règle est $A \rightarrow C + B$ alors (t',O') ← AGAUCHE(Z, C, O)
t" ← t ; E' ← E
% cas symétrique de la règle $A \rightarrow B + C$: cette fois C est analysé en partant de son extrémité %

la règle est $A \rightarrow C_{o} B$ alors (t', E") ← ADROITE (Z, C, O)
t" ← t ; O' ← O ; E' ← E ;

la règle est $A \rightarrow C \times B$ alors (t', O') ← AGAUCHE (Z,C,E)
E' ← E ; t" ← t

la règle est $A \rightarrow B \underline{f} C$ alors choix entre
. (t", E') ← ADROITE (Z, C, O) ;
si E' ≠ E alors erreur fsi
. (t", O') ← AGAUCHE (Z, C, E) ;
si O' ≠ O alors erreur fsi
fin choix ;
E' ← E ; O' ← O ; t' ← t ;
% les formes issues de B et C ayant leurs origines qui coïncident ainsi que leurs extrémités, l'analyse peut partir soit de l'origine O de B, soit de son extrémité E %

la règle est $A \rightarrow REL(B,C)$ alors soit Z' la sous zone de Z désignée par REL ;
choix de p primitive de C dans Z'
soit O" et E" l'origine et l'extrémité de p ;
(t", U, V) ← REMONTE (Z', p, O", E", C) ;
t ← t' ; O' ← O ; E' ← E

la règle est $A \rightarrow REL(C,B)$ alors soit Z' la sous zone de Z désignée par REL^{-1} ;
choix de p primitive de C dans Z' ;
soit O" et E" l'origine et l'extrémité de p ;
(t', O', E') ← REMONTE (Z', p, O", E", C) ;

t" ← t
% dans ces deux derniers cas la reconnaissance de la sous-forme se fait dans la zone où elle doit être localisée et en redémarrant l'analyse par un choix initial de primitive %

fin cas

résultat : REMONTE (Z, A × (t' + t") , O', E', But)
fsi

fin procédure.

L'appel initial aboutissant à l'analyse complète étant :

Z ← tout le plan ; choix d'une primitive p de départ, d'origine O' et d'extrémité E' ;

(t, O, E) ← REMONTE (Z, p, O', E', A_1) ;

si toutes les primitives n'ont pas été utilisées alors % l'analyse n'a pas pris en compte tout le dessin %
erreur fsi

On remarquera le parallèle avec la procédure REMONTE décrite §.2.1., ADROITE et AGAUCHE faisant maintenant respectivement une analyse de l'origine vers l'extrémité et de l'extrémité vers l'origine. Ce parallèle est cependant remis en cause par l'utilisation des règles contenant des relations : lorsque dans l'analyse descendante une sous-forme de la classe B est reconnue et que l'on recherche une sous-forme de la classe C telle que Rel(B, C) , l'analyse de C se fera par une nouvelle phase ascendante : dans MIRABELLE , les relations portent sur des zones ; donc, ayant localisé B , on peut en déduire la zone Z' où doit se trouver C ; une nouvelle analyse redémarre donc dans Z' avec pour objectif d'identifier une sous-forme de la classe C . Lorsque les relations ne spécifient pas seulement des zones mais aussi des contacts (comme PSUR, PCTRED, ...) alors après l'identification de la sous-forme

de la classe C , il faut en plus tester si ce contact est établi. Ce dernier point n'apparaît pas dans l'algorithme ci-dessus pour ne pas alourdir la présentation ; cela ne change rien fondamentalement.

3.2.- Phase descendante.

Les procédures ADROITE et AGAUCHE étant symétriques, nous ne détaillerons que AGAUCHE ; partant du point E qui doit être l'extrémité d'une forme A à analyser, il retourne l'arbre associé à A après analyse, ainsi que l'origine de la représentation de A analysée.

```
procédure   AGAUCHE (% zone de travail % Z ,
                     % classe de la sous-forme à analyser % A ,
                     % localisation de son extrémité % E )
            résultat
                     (% le résultat est un couple :
                          arbre syntaxique associé à A % arbre,
                     % origine de la forme A % point du plan ).

si A est une forme primitive alors
        si une forme de la classe A est présente dans Z avec son
         extrémité en E        alors soit O son origine ;
                                     résultat : (A, O)
                               sinon erreur
        fsi
sinon % A est une forme non primitive %
        choix d'une règle A → λ ;
        cas la règle est A ⟶ B + C
                               alors (t", O) ← AGAUCHE (Z, C, E) ;
                                     (t', O) ← AGAUCHE (Z, B, O") ;
           la règle est A → BoC
                               alors (t", O) ← AGAUCHE (Z, C, E) ;
                                     (t', E") ← ADROITE (Z, B, O) ;
           la règle est A → B × C
                               alors (t", O") ← AGAUCHE (Z, C, E) ;
                                     (t', O)  ← AGAUCHE (Z, B, E) ;
```



```
           la règle est A → B f C
                               alors % ici trois possibilités s'ouvrent
                                     à nouveau : %
                                     choix entre :
                                     . (t", O) ← AGAUCHE (Z, C, E) ;
                                       (t', O) ← AGAUCHE (Z, B, E) ;
                                      si O ≠ O' alors erreur fsi
                                     % on analyse B et C à partir de E
                                     et on vérifie la coïncidence de leurs
                                     origines %
                                     . (t", O) ← AGAUCHE (Z, C, E) ;
                                       (t', E') ← ADROITE (Z, B, O) ;
                                      si E ≠ E' alors erreur fsi
                                     % cette fois après l'analyse de C
                                     on repart de l'origine pour analyser
                                     B et on vérifie la coïncidence de
                                     leurs extrémités %
                                     . (t', O) ← AGAUCHE (Z, B, E) ;
                                       (t", E') ← ADROITE (Z, C, O) ;
                                      si E ≠ E' alors erreur fsi
                                     % cas symétrique du cas précédent %
           la règle est A → REL(B, C) alors
                                     (t', O) ← AGAUCHE (Z, B, E) ;
                                     soit Z' la sous-zone de Z délimitée
                                     par REL ; choix de p primitive de C
                                     ayant une occurrence dans Z' ; soient
                                     O' et E' leurs origines et extrémités ;
                                     (t", O", E") ← REMONTE (Z',p, O', E', C)
        fin cas
        résultat : (A × (t' + t") , O)
fsi
fin procédure.
```

Note :

l'arbre syntaxique ainsi obtenu n'est pas utilisable ; il n'indique pour chaque forme que ses sous-formes mais non de quelle façon elles sont assemblées :

par exemple, l'utilisation de la règle $A \rightarrow A + C$
ou $A \rightarrow B \circ C$

donnerait donc le même arbre. De fait, on indique à chaque noeud de l'arbre les numéros de règles utilisés ainsi que les attributs numériques associés à la sous-forme, ceci afin de permettre les traitements qui peuvent suivre la reconnaissance, par exemple en vue de l'interprétation de la forme reconnue [HAT 77] .

3.3.- Exemple d'analyse.

Plutôt qu'une explication de l'algorithme précédent, cet exemple est destiné à introduire le chapitre suivant en indiquant quels renseignements tirés de la description seront utiles pour l'analyseur.

Considérons la description suivante où les primitives sont

ert :T(90,5), hor : T(0, 5) , ob 45 : T(45, 20), ob 135 : T(135, 20) ,

arche : C(0, 10) .

Il s'agit d'un système décrivant un donjon :

DONJON = sommet f corps /1/
corps = INT (étages, murs) /2/
murs = U /3/ - vert + hor + porte + hor + vert /4/
sommet = hor f (ob 45 + - ob 135) /5/
créneaux = ob 45 + - ob 135 + U + créneaux /7/
ob 45 + - ob 135 /8/
U = - vert + hor + vert /9/

porte = hor f (vert + arche + - vert) /10/
fenêtre = U f hor /11/
U f (ob 45 + - ob 135) /12/
U f arche /13/
étages = fenêtres/14/
SUR **(fenêtres,** étages) /15/
fenêtres = fenêtre /16/
ADROITE (fenêtre, fenêtres) /17/

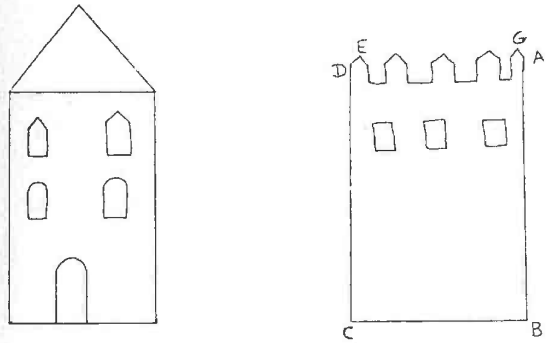


figure 2.12. : exemples de donjons.

L'analyse débute par le choix d'une primitive de départ p ; l'appel de REMONTE (Z, p, o, e, donjon) conduit alors à choisir une règle où apparaît p .

. supposons d'abord que p soit une horizontale hor ; hor apparaît dans les règles 4, 5, 9, 10 et 11 .

La règle 11 spécifie qu'à l'origine de hor se trouve l'origine de U , qui est l'origine de -vert, c'est-à-dire l'extrémité d'une verticale ;

de même, on doit trouver l'extrémité d'une verticale à l'extrémité de hor .

La règle 9 spécifie que l'origine de hor et son extrémité coïncident avec l'origine d'une verticale.

la règle 10 spécifie encore que l'origine de hor et son extrémité coïncident avec celle d'une verticale, mais comme de plus, l'origine et l'extrémité de hor sont origine et extrémité de "porte", la règle 4 indique qu'on doit en plus observer l'extrémité d'une horizontale à l'origine de hor , et l'origine d'une autre horizontale à son extrémité.

En étudiant encore la règle 5 et la règle 4 , on voit que le *contexte* (c. a. d. les primitives apparaissant au point origine et extrémité de hor) permettent de lever le choix dans ce cas.

. Supposons que nous ayons pris pour p une verticale vert : un test sur le contexte ne nous permettra pas toujours de lever le choix entre la règle 4 et la règle 5 : dans les deux cas, on pourra observer une oblique à l'extrémité de vert et une horizontale à son origine.

Supposons que l'horizontale CB (figure 2.12.) ait été choisie comme point de départ ; la présence de DC et AB conduit donc REMONTE à choisir la règle 9 :

$$U = - \text{ vert} + \underline{\text{hor}} + \text{vert}$$

par une analyse descendante les deux verticales sont alors analysées et nous obtenons la sous-forme U . Un nouvel appel de REMONTE nous conduit alors à choisir entre les règles 3, 7, 11, 12, 13 . Les présences de DE en D et celle de GA excluent l'utilisation des règles 7, 11 et 13. Si la règle 12 est choisie, la non coïncidence de E et C nous fera revenir sur ce choix. Supposons donc la règle 3 choisie. Nous avons alors analysé "murs".

Un nouvel appel de REMONTE va alors appliquer la seule règle possible : 2 ; une nouvelle analyse ascendante va reprendre avec , pour objectif l'analyse "d'étages" : il se pose à nouveau le choix d'une primitive de départ.

Une fois "corps" analysé REMONTE choisit la règle 1 , seule règle possible. Une analyse descendante est alors effectuée pour reconnaître "sommet" ; supposons que ADROITE ait été lancée à partir du point D ; le premier choix qu'aura à faire cette procédure sera de choisir entre la règle 5 et la règle 6. Or, la seule présence d'une origine de la primitive Ob 45 en D exclut la règle 5 , l'analyse descendante de créneaux va donc être exécutée.

Nous voyons donc que nous pouvons tirer de la description des renseignements nous permettant de guider les choix de l'analyse, c'est ce qui va faire l'objet du chapitre 3.

CHAPITRE 3

COMPILATION D'UNE

DESCRIPTION DE DESSINS

CHAPITRE 3

COMPILATION D'UNE DESCRIPTION DE DESSINS

Compilation : du latin cum : avec (dans le sens de réunion)
et pilare : piller

a) ouvrage qui est essentiellement un recueil d'extraits d'un ou plusieurs auteurs

b) péjoratif : plagiat. (Robert).

Nous avons appelé compilateur notre programme qui, à partir d'une description formelle d'une classe de dessins, construit l'ensemble des tables permettant le travail de l'analyseur. Nous allons dans ce chapitre chercher les renseignements qui peuvent être tirés de la description et qui permettront à l'algorithme décrit au chapitre précédent §.3. de se dérouler de façon non combinatoire, c.a.d. lui permettront par des tests de limiter les choix possibles. Ces renseignements sont essentiellement :

- les formes primitives pouvant apparaître à l'origine ou à l'extrémité d'une forme (initiales)
- les formes primitives pouvant apparaître dans l'environnement d'une forme" : les contextes.
- les formes permettant au mieux de démarrer l'analyse : les formes caractéristiques.

Nous avons gardé le nom de compilateur pour ce programme pour plusieurs raisons :

- sur le plan technique, sa réalisation emprunte beaucoup à celle des compilateurs: analyse syntaxique, manipulation d'expressions, gestion de tables de symboles, ...

- son travail se rapproche plus du sens du mot compilation que celui que fait le compilateur d'un langage de programmation

- enfin comme un compilateur nous travaillons "avec" des "piles" !

Notations.

Par analogie avec la notation utilisée en théorie des langages (cf. chapitre 1 , §.3.3.) nous noterons T l'ensemble des noms de formes primitives, N l'ensemble de noms de formes non-primitives de notre système de description (S) (c.a.d. les inconnues du système d'équations, cf. chapitre 2 §.1.3.) .

$A_1 \in N$ désignera la première inconnue du système (c.a.d. l'axiome de la grammaire) .

$$V = N \cup T \quad .$$

Nous noterons $-X \doteq \{-x \mid x \in X\}$ lorsque $X \subset V$.

Pour tout $A \in N$ S(A) désignera la solution du système d'équations qui est associée à l'inconnue A .
Plutôt que de considérer la description comme un système d'équations on la considère ici comme un ensemble de règles de la forme $A \to \lambda/r/$ ou r désigne le numéro de la règle ; on notera $\mathcal{R}$ l'ensemble des règles.

1.- LES INITIALES.

1.1.- DÉFINITION ET UTILISATION.

Dans notre modèle , les formes sont munies comme on l'a vu, de deux pôles de concaténation , l'origine et l'extrémité ; nous allons généraliser

ici les relations initiales ($\mathcal{J}^*$) et finales ($\mathcal{S}^*$) définies au chapitre 2 §.2.2.2., par "présent à l'origine" et "présent à l'extrémité". Une différence surgit alors : il peut y avoir plusieurs primitives présentes à l'origine d'une forme, et les relations porteront donc maintenant sur *l'ensemble des parties des primitives*. Ainsi si la forme A admet à son origine l'origine des primitives a et b , nous noterons $A \mathcal{J}^* \{a, b\}$. De plus, si deux représentations d'une même primitive ont la même origine, faut-il les distinguer ou les confondre ? Ce choix dépend essentiellement du problème à traiter :

- si une classe de primitives contient des obliques, on peut imaginer plusieurs occurrences d'obliques issues du même point et il faudrait donc distinguer les différentes occurrences.

- si deux segments de droites verticaux se superposent , on peut penser que l'on n'observera qu'un seul segment sur le dessin : on doit donc les confondre.

Nous nous restreindrons ici au cas le plus simple où elles seront *confondues* (cf. [HAT 77] où l'autre cas est traité sans difficulté majeure : il suffit de remplacer dans ce qui suit l'union par l'union disjointe).

1.1.1.- DÉFINITION.

Soit φ une formule portant sur des formes. On définit $\mathcal{O}_r$ comme étant l'ensemble des pôles des formes de φ coïncidant avec l'origine de la forme spécifiée par φ . Ainsi si $\varphi = \ldots A \ldots$, alors si l'origine de A coïncide avec celle de φ $A \in \mathcal{O}_r(\varphi)$; si par contre l'extrémité de A coïncidait avec celle de φ alors nous aurions, par convention de notation $-A \in \mathcal{O}_r(\varphi)$.

$$\mathcal{O}_r(\varphi) = \underline{\text{cas}}\ \varphi \text{ est réduit à } A \in V \ \underline{\text{alors}}\ \{A\}$$
$$\varphi = -\mu \ \underline{\text{alors}}\ \mathcal{O}_{\bar{x}}(\mu)$$
$$\varphi = \mu \circ \mu' \ \underline{\text{alors}}\ \mathcal{O}_r(\mu) \cup \mathcal{O}_r(\mu')$$

$\varphi = \mu + \mu'$ alors $\theta_r(\mu)$

$\varphi = \mu \times \mu'$ alors $\theta_r(\mu)$

$\varphi = \mu \underline{f} \mu'$ alors $\theta_\alpha(\mu) \cup \theta_r(\mu')$

$\varphi = Rel(\mu,\mu')$ alors $\theta_\alpha(\mu)$

fin cas

de même on définit ζ_α comme l'ensemble des pôles de formes présentes à l'extrémité :

$\zeta_\alpha(\varphi)$ = cas

$\varphi = A \in V$ alors $\{- A\}$

$\varphi = - \mu$ alors $\theta_r(\mu)$

$\varphi = \mu + \mu'$ alors $\zeta_\alpha(\mu')$

$\varphi = \mu \circ \mu'$ alors $\zeta_\alpha(\mu')$

$\varphi = \mu \times \mu'$ alors $\zeta_\alpha(\mu') \cup \zeta_\alpha(\mu)$

$\varphi = \mu \underline{f} \mu'$ alors $\zeta_\alpha(\mu) \cup \zeta_\alpha(\mu')$

$\varphi = REL(\mu, \mu')$ alors $\zeta_\alpha(\mu)$

fin cas

Exemple :

si $\mu = (A + B) \circ (-C + d)$

$\theta_r(\mu) = \{A, - C\}$

$\zeta_\alpha(\mu) = \{-d\}$

$\mathfrak{J}$ est alors défini par

$A \mathfrak{J} 0 \iff A \in N \wedge \exists \mu \ (A \to \mu \wedge \theta_r(\mu) = 0)$

$- A \mathfrak{J} E \iff A \in N \wedge \exists \mu \ (A \to \mu \wedge \zeta_\alpha(\mu) = E)$

exemple :

si $A \longrightarrow - c \underline{f} d$

$A \mathfrak{J} \{- c, d\}$, $- A \mathfrak{J} \{c, - d\}$

si $A \longrightarrow a \circ a$

$A \mathfrak{J} \{a\}$, $- A \mathfrak{J} \{a\}$

ce qui est en accord avec la convention prise plus haut de confondre les présentations d'une même primitive ayant même origine ou même extrémité.

Considérons maintenant $\mathfrak{J}$ défini non plus sur N mais sur $\mathcal{P}(V \cup -V)$ - ensemble des parties de $V \cup -V$ - par l'extension suivante :

$$\forall \alpha \ \forall B \ \forall \beta_B (B \in \alpha \wedge B \mathfrak{J} \beta_B) \Rightarrow \alpha \mathfrak{J} (\bigcup_{B \in \alpha} \beta_B)$$

on définit alors $\mathfrak{J}^*$ comme la fermeture reflexive et transitive de $\mathfrak{J}$.

Exemple :

si nous avons : $A \to CoB$

$B \to a \underline{f} b$

$C \to - (a \times - b)$

nous aurons $A \mathfrak{J} \{C, B\}$

$B \mathfrak{J} \{a, b\}$

$C \mathfrak{J} \{-a, b\}$

et donc $A \mathfrak{J}^* \{a, -a, b\}$

Exemple de calcul opéré par MIRABELLE.

(par initiale on comprend ici $\mathfrak{J}^*(A)$ et par finale $\mathfrak{J}^*(-A)$) .

L'opérateur . remplace l'opérateur o , et * l'opérateur x , ces caractères n'existant pas sur cette console.

```
 REGLES DELA GRAMMAIRE:
?*****************************************
?*                                       *
?*  EXEMPLE DE CALCUL D'INITIALES (I*)   *
?*                                       *
?*****************************************
?*
?  AA :  BB . CC    1
?  AA :  A + AA
?  BB : B
?  CC :  C
? CC :  C * C
? .
 VOULEZ-VOUS LA CODIF DES REGLES?
?N
  VOULEZ VOUS LES FILS DE CHAQUE N.T. ?
?N
 LISTE DES INITIALES ?
?O

AA
 :  A
 :  B C

BB
 :  B

CC
 :  C
 LISTES DES FINALES?
?OUI

AA
 :  -C

BB
 :  -B

CC
 :  -C
```

En effet

AA $\mathfrak{J}$ {BB, CC} $\mathfrak{J}$*{B, C}

AA $\mathfrak{J}$ {A} donc AA $\mathfrak{J}$*{A}

de même

- AA $\mathfrak{J}$ {- CC} $\mathfrak{J}$*{- C}

On remarquera que la dernière règle CC → C × C indique que deux représentations de C coïncident par leurs extrémités ; nous avons convenu de les confondre ; dans le cas contraire, nous aurions en -CC $\mathfrak{J}$ {-C', -C"} en distinguant les deux occurrences.

1.1.2.- UTILISATION DES INITIALES.

$\mathfrak{J}$* (A) généralise la notion d'initiale (forme présente à l'origine) et $\mathfrak{J}$*(- A) généralise les finales (formes présentes à l'extrémité) ; la définition que nous avons donnée se justifie par le théorème suivant valable pour les hypothèses énoncées après.

THEOREME :

Pour un système S d'inconnues N et portant sur les primitives T, on a

$(\forall A \in N \cup -N)\ (\forall \{a_1, \ldots, a_n\} \in \mathfrak{P}(T \cup -T))$

$A\ \mathfrak{J}^* \{a_1, \ldots, a_n\} \iff \exists f \in S(A)$ tel que à l'origine de f apparaissent les origines de $a_1, \ldots, a_n$ et elles seules.

Démonstration :

(voir annexe 2)

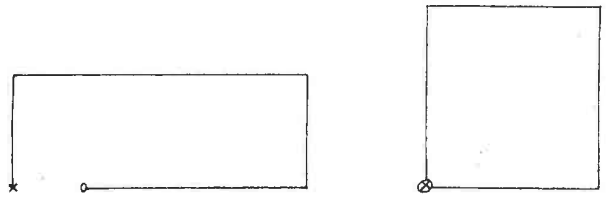
Hypothèse H1 :

Pour toute forme décrite par le système, il n'y a de coïncidences de pôles de primitives que celles explicitement spécifiées par la description.

Cette hypothèse paraît raisonnable mais elle peut être mise en défaut accidentellement :
considérons le système réduit à la seule règle

(S) A → a + b + -a + -b

où a et b sont des segments de longueur quelconque, a horizontal, b vertical.
La classe de figures de S(A) est celle de la figure 3.1 a mais le rectangle (figure 3.1. b) n'est pas exclu : or A $\mathfrak{J}^*$ {a} , -A $\mathfrak{J}^*$ {b} mais nous n'avons pas A $\mathfrak{J}^*$ {a, b} -A $\mathfrak{J}^*$ {a, b} bien que en 3.1.b nous ayons l'origine de a et de b présent à l'origine de A (confondue avec l'extrémité).



- a -

- b -

figure 3.1.

Hypothèse H2 :

Toute formule admet une interprétation.

Comme nous l'avons vu au chapitre 2 , §.1.3.2. , cette hypothèse est fausse en général. Cependant, si cette hypothèse est fausse , elle ne met en cause que l'implication directe du théorème (voir démonstration).

Par conséquent, si l'on observe une forme f de S(A) elle vérifiera encore A $\mathfrak{J}^*$ $\{a_1,\ldots, a_n\}$ si $a_1,\ldots, a_n$ sont les primitives présentes à l'origine de f .

Revenons alors à la procédure d'analyse descendante ADROITE (le cas est symétrique pour AGAUCHE) :

- étant en un point O qui est l'origine d'une forme de la classe A (A ∈ N) il faut choisir parmi les règles $A \to \lambda$ celle qui décrit la sous forme présente.

Le théorème précédent (même en ne gardant que l'implication réciproque) nous indique que l'on ne peut choisir la règle $A \to \lambda$ que si les primitives devant apparaître à l'origine de la forme décrite par λ sont présentes en O ; plus précisément

- soit P l'ensemble des primitives dont une représentation a son origine en O (si p a son extrémité en O , -p ∈ P) , on ne peut choisir la règle $A \to \mu$ que si

$$\exists \alpha \in \mathfrak{J}^*(\mathcal{Br}(\mu)) \quad \alpha \subset P \quad .$$

La définition des différents $\mathfrak{J}^*(\mathcal{Br}(\lambda))$ peut alors nous aider dans le choix de la procédure ADROITE, celle des $\mathfrak{J}^*(\mathcal{Bx}(\lambda))$ dans la procédure AGAUCHE.

Remarque :

Il ne suffit pas que tous les seconds membres $\lambda_1, \ldots, \lambda_k$ d'une même inconnue vérifient

$$\mathfrak{J}^*(\mathcal{Br}(\lambda_i) \cap \mathfrak{J}^*(\mathcal{Br}(\lambda_j)) = \emptyset \quad (i \neq j)$$

pour que le choix soit déterministe ; il suffit qu'il existe un $\alpha \in \mathfrak{J}^*(\mathcal{Br}(\lambda_i))$ et un $\beta \in \mathfrak{J}^*(\mathcal{Br}(\lambda_j))$ avec $\alpha \subset \beta$ pour qu'en présence de β l'on ne puisse choisir entre les règles $A \to \lambda_i$ et $A \to \lambda_j$

Et même si cette dernière condition ne se présente pas , on peut avoir le problème suivant :

Observer un ensemble de primitives P tel que

(1) $\exists \alpha \in \mathfrak{J}^*(\mathcal{O}_R(\lambda_i)) \quad , \quad \exists \beta \in \mathfrak{J}^*(\mathcal{O}_R(\lambda_i))$

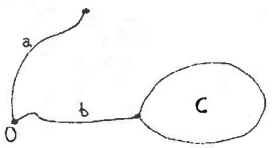
$\alpha \subset P \quad \wedge \quad \beta \subset P \quad \wedge \quad \alpha \cap \beta = \emptyset$

Ce peut être le cas avec la description suivante :

$X = A \circ B$
$A = \{a\} \cup \{b\}$
$B = a + C \cup b + C$
.....

On a $\mathfrak{J}^*\{A\} = (\{a\},\{b\})$; $\mathfrak{J}^*\{B\} = (\{b\},\{a\})$ $\mathfrak{J}^*(X) = \{\{a\}, \{b\},\{a,b\}\}$

La première équation spécifie qu'il peut y avoir à l'origine d'une forme de X , les primitives a et b . La condition (1) est donc remplie par une forme de la figure 3.2..



$X \longrightarrow AoB \xrightarrow{*} ao(b+C)$

figure 3.2.

Une condition de déterminisme pour l'analyse descendante utilisant le contexte sera donnée §.2.4..

1.2.- CALCUL EFFECTIF DES INITIALES.

A partir de $\mathfrak{J}$ le calcul de $\mathfrak{J}^*$ est une fermeture transitive sur l'ensemble des parties de $V \cup -V$. Si l'on a $card(V) = 40$ (cas tout à fait raisonnable) l'ensemble des parties de $V \cup -V$ possède 2^{80}

éléments ! Il est donc hors de question de travailler sur le graphe effectif et donc toute technique globale telle que l'algorithme de Warshall ou celui des matrices de bits avec opérations simultanées sur n bits est à exclure ; cette matrice de booléens représentant le graphe ne contiendrait d'ailleurs pratiquement que des 0 . Enfin, on ne cherche à construire que $\mathfrak{J}^*(A)$ $(A \in N \cup -N)$.

Nous avons donc choisi une méthode simple : la fermeture transitive par l'algorithme de la pile [DER 71] . Il a été demontré qu'en moyenne il est plus efficace que Warshall : $O(n^2 \log n)$ étapes [BLO 76] .
De plus, dans le cas réel de cette application le nombre d'initiales d'un élément A est très petit par rapport à $card(V \cup -V)$ et donc une réalisation ad hoc permet d'être bien en dessous du nombre moyen d'étapes.

Algorithme :

```
pour chaque élément A ∈ N ∪ -N faire
    % calcul de J*(A) %
        % initialisation %
        pour toute partie α de V ∪ -V mettre la marque de α à 0 ;
        initialiser la pile avec {A} ; marquer ({A}) ;

        % itération générale %
        tant que pile non vide faire
        % la pile contient toutes les parties accessibles dont les
        successeurs n'ont pas encore été calculés %
            dépiler (α) ;
            si α ⊂ T ∪ -T alors noter A J*α
                sinon α = {a1,..,an,A1,...,Am}  ai ∈ T ∪ -T, Aj ∈ N ∪ -N
(1)                 pour chaque (β1,...,βm) tel que Aj J βj faire
                        β ← {a1,...,an} ∪ β1 ∪...∪βm ;
                        si β n'est pas marqué alors empiler (β) ;
                            marquer (β) fsi
                    fait
            fsi
    fait
```

Le marquage ne peut se faire à l'aide d'une table de (2^{80}) bits et se fera donc par insertion dans une table en adressage dispersé : ne seront mis en table que les ensembles marqués.

Une optimisation évidente peut être faite à la ligne (1) : comme on ne cherche à construire $\mathfrak{J}^*$ que entre $N \cup -N$ et $\mathfrak{P}(T \cup -T)$ on peut directement insérer les β_j tels que $A_j \, \mathfrak{J}^* \beta_j$, $\beta_j \subset T \cup -T$, ceci si le calcul de $\mathfrak{J}^*(A_j)$ a déjà été fait.

Le temps d'exécution sur un exemple comme celui détaillé à la fin de ce chapitre (card (T) = 9 , card (N) = 12) est suffisamment rapide pour ne pas créer de temps d'attente visible lors d'une éxécution en temps partagé sur IRIS 80.

2.- LE CONTEXTE IMMÉDIAT.

Nous allons maintenant généraliser la notion de contexte que nous avions introduite au chapitre 2 §.2.1. en définissant les ensembles de primitives qui doivent coïncider avec une sous forme B pour pouvoir utiliser une règle r dans la procédure REMONTE.
Les définitions du contexte que nous allons donner vont nous permettre en particulier d'exprimer une condition de déterminisme pour les phases ascendantes et descendantes de l'algorithme.

2.1.- DÉFINITION.

Considérons d'abord un exemple : soit les règles suivantes dans notre système

$$A \longrightarrow a + B \quad /1/$$

$$B \longrightarrow c \; o \; d \quad /2/$$

La règle 2 indique qu'à l'origine de B on peut trouver l'origine d'une représentation de d ; si cette règle est utilisée pour définir la forme que l'on observe , les origines de d et c coïncident aussi avec celle de la sous-forme de nom B et si l'on se reporte à la règ e 1 , celle-ci peut être concaténée à l'extrémité d'une représentation de a .

Remarquons que si A est l'axiome (c.a.d. la 1ère inconnue) de notre système, rièn ne peut apparaître à l'extrémité de B , et donc de d .

Nous noterons ceci

$$\mathcal{C}(d) = (\{c, -a\} , \emptyset)$$

NOTE :

Comme il nous faudra dans toute règle $r : A \to \lambda$ distinguer les différentes occurrences de V dans λ nous supposerons pour simplifier l'exposé que tout élément de V n'admet au plus qu'une seule occurrence dans tout second membre λ . Ce point n'est pas essentiel et cette restriction n'est d'ailleurs pas imposée dans le système MIRABELLE où chaque occurrence est distinguée par sa position dans la règle.

La relation D_r

Elle généralise la relation $\mathfrak{J}$ du chapitre 2. Pour un nom de forme A apparaissant dans le second membre de la règle r, D_r définit les objets en coïncidence *directe* dans cette règle avec un pôle d'un élément A ; nous noterons $D_r(A)$ ceux qui sont en coïncidence avec l'origine de A , $D_r(-A)$ ceux qui sont en coïncidence avec l'extrémité.

Soit $C \longrightarrow \mu$ la règle de numéro r et dans laquelle apparaît A , on a :

$$A \; D_r \; \alpha$$

$$-A \; D_r \; \beta \qquad \text{avec :}$$

$\alpha = \{ a \in V \cup -V \mid$ l'origine de a coïncide avec celle de A dans la formule $\mu \}$

$\beta = \{ a \in V \cup -V \mid$ l'origine de a coïncide avec l'extrémité de A dans la formule $\mu \}$

Dans la suite, nous utiliserons le plus souvent cette notation en l'étendant de la façon suivante :

$$D_r(A, B) = (D_r(A), D_r(B))$$
$$D_r(A, -A) = (D_r(A), D_r(-A))$$
$$D_r(A, \emptyset) = (D_r(A), \emptyset)$$

Exemple :

$B \longrightarrow a + (b \circ c)$ /règle r/

$(a, -a)\ D_r\ (\emptyset, \{b, c\})$

$(b, -b)\ D_r\ (\{c, -a\}, \emptyset)$

$(c, -c)\ D_r\ (\{b, -a\}, \emptyset)$

$(\emptyset, -c)\ D_r\ (\emptyset, \emptyset)$

La relation P_r

Cette relation va nous permettre de "remonter" les règles pour déterminer le contexte introduit à un niveau supérieur. (P_r pour "père pour la règle r").

Si dans la règle $C \longrightarrow \varphi$ /r/ l'origine de A coïncide avec celle de la forme englobante C, toutes les formes ayant un pole en coïncidence avec l'origine de C auront par conséquent un pôle en coïncidence avec celle de A. P_r note simplement le fait qu'un pôle d'une sous-forme

coïncide avec un pôle d'une forme englobante, avec la convention de noter A pour l'origine de A, $-A$ pour son extrémité.

Soit donc C le premier membre de la règle $r : C \rightarrow \varphi$ /r/

Par définition, si A a une occurrence dans φ :

$A\ P_r\ C \Longleftrightarrow$ dans φ l'origine de A coïncide avec celle de φ (c.a.d. $A \in \mathcal{B}_2(\varphi)$)

$-A\ P_r\ C \Longleftrightarrow$ dans φ l'extrémité de A coïncide avec l'origine de φ (c.a.d. $-A \in \mathcal{E}_2(\varphi)$)

$A\ P_r\ -C \Longleftrightarrow$ dans φ l'origine de A coïncide avec l'extrémité de φ (c.a.d. $-A \in \mathcal{E}_1(\varphi)$)

$-A\ P_r\ -C \Longleftrightarrow$ dans φ l'extrémité de A coïncide avec celle de φ (c.a.d. $-A \in \mathcal{B}_1(\varphi)$).

Comme pour D_r on étend P_r en posant

$$P_r(\emptyset) = \emptyset$$
$$P_r(A, B) = (P_r(A), P_r(B)) \quad A, B \in (N \cup -N \cup \{\emptyset\})$$

Exemple :

soit $B \longrightarrow a + b + c$ /règle r/

$(a, -a)\ P_r\ (B, \emptyset)$

$(b, -b)\ P_r\ (\emptyset, \emptyset)$

$(c, -c)\ P_r\ (\emptyset, -B)$

$(\emptyset, -a)\ P_r\ (\emptyset, \emptyset)$

La relation $\mathcal{C}$ et le contexte

La relation $\mathcal{C}$ est définie sur $(V \cup -V \cup \{\emptyset\})^2 \times [\mathcal{P}(V \cup -V)]^2$;

elle ne sera en fait que significative dans les cas suivants :

$\mathcal{C}(A, -A)$ indiquera le couple d'ensembles des formes primitives pouvant apparaître simultanément et respectivement à l'origine et à l'extrémité d'une forme A ; $\mathcal{C}(-A, A)$ étant le couple symétrique ;

$\mathcal{C}(A, \emptyset)$, (resp. $\mathcal{C}(-A , \emptyset)$) est un couple $(\alpha , \emptyset)$ où α est un ensemble des formes primitives pouvant coïncider avec l'origine (resp. l'extrémité) d'une forme A ; $\mathcal{C}(\emptyset, A)$ est le couple symétrique de $\mathcal{C}(A, \emptyset)$.

$\mathcal{C}$ est défini par le système suivant :

(dans ce système on ne travaille que sur des couples et toute la notation s'étend naturellement aux couples de la façon suivante :

soit $f : E \to F$ on notera encore $f : E \times E \to F \times F$ défini par $f(x, y) = (f(x), f(y))$)

$\mathcal{C}(A, B) = \bigcup_{r \in \mathfrak{R}} \mathcal{C}_r(A, B)$ où $\mathfrak{R}$ est l'ensemble des numéros de règles

$\mathcal{C}(A, B) =$ si $(A, B)\ P_r(\emptyset, \emptyset)$ alors $\mathfrak{I}^*(D_r(A, B))$

sinon $\mathcal{C}(P_r(A, B)) \dot{\cup} \mathfrak{I}^*(D_r(A, B))$ fsi

où $E \dot{\cup} F = \{\alpha \cup \beta \mid \alpha \in E , \beta \in F\}$

De plus, on pose si X est l'axiome du système :

$(U, -U)\ \mathcal{C}(\emptyset, \emptyset)$
$(U, \emptyset)\ \mathcal{C}(\emptyset, \emptyset)$
$(\emptyset, U)\ \mathcal{C}(\emptyset, \emptyset)$ avec $U \in \{X, -X\}$

Cette dernière convention exprime qu'une forme de $S(X)$ peut avoir un contexte vide, et ce parce qu'elle peut ne pas être une sous forme d'une forme plus complexe.

Commentaire :

Soit $A \in V$; si $P_r(A, -A) = (\emptyset, \emptyset)$, dans la règle r la sous forme A n'est en contact qu'avec les sous-formes $D_r(A)$ et $D_r(-A)$, et donc à l'origine de A ne peuvent apparaître que les éléments de $\mathfrak{I}^*(D_r(A))$ à l'extrémité les éléments de $\mathfrak{I}^*(D_r(-A))$;

Supposons maintenant que $P_r(A, -A) = (B, \emptyset)$; l'origine de A coïncide dans la règle r avec l'origine de la forme englobante B , donc en plus des éléments de $\mathfrak{I}^*(D_r(A))$ apparaîtra, à l'origine de B, des éléments en contact avec l'origine de B :

$\mathcal{C}_r(B, \emptyset) = \{(\alpha , \emptyset) \mid \alpha$ est en contact avec l'origine de $B\}$

C'est ce qu'exprime $\mathcal{C}(P_r(A, -A)) \cup \mathfrak{I}^*(D_r(A, -A))$.

Exemple de détermination du contexte dans MIRABELLE

```
 REGLES DELA GRAMMAIRE:
?************************************
?*                                   *
?* EXEMPLE DE CALCUL DES CONTEXTES   *
?*                                   *
?************************************
?*
?  AA : BB @ CC    1
?  BB : -A         2
?  BB : -A ,|\\+

I
SYNTAXE D'EXPRESSION INCORRECTE              (1)
?  BB : -A.B + BB  3
?  CC : CC + D     4
?  CC :  C         5
?.
 VOULEZ-VOUS LA CODIF DES REGLES?
?N
  VOULEZ VOUS LES FILS DE CHAQUE N.T. ?
?N
 LISTE DES INITIALES ?
?N
 LISTES DES FINALES?
?N
 VOULES VOUS LES CONTXTES?
?O

     AA
CET IDENTIFICATEUR APPARAIT EN POSITION  ØDANS LA REGLE  Ø
CONTEXTES POSSIBLES :
 :
 :


     BB
CET IDENTIFICATEUR APPARAIT EN POSITION 21DANS LA REGLE  3
CONTEXTES POSSIBLES :
 :  -B
 :  -C

 :  -B
 :  -D
```

```
CET IDENTIFICATEUR APPARAIT EN POSITION 13DANS LA REGLE  1
CONTEXTES POSSIBLES :
 :  C
 :  -C

 :  C
 :  -D


     CC
CET IDENTIFICATEUR APPARAIT EN POSITION 13DANS LA REGLE  4
CONTEXTES POSSIBLES :
 :  B -A
 :  D

 :  -A
 :  D

CET IDENTIFICATEUR APPARAIT EN POSITION 15DANS LA REGLE  1
CONTEXTES POSSIBLES :
 :  B -A
 :  A

 :  -A
 :  A
.........

     A
CET IDENTIFICATEUR APPARAIT EN POSITION 17DANS LA REGLE  3
CONTEXTES POSSIBLES :
 :
 :  B C

 :
 :  B -B

CET IDENTIFICATEUR APPARAIT EN POSITION 13DANS LA REGLE  2
CONTEXTES POSSIBLES :
 :  -C
 :  -B

 :  -D
 :  -B

 :  -C
 :  C

 :  -D
 :  C


     B
CET IDENTIFICATEUR APPARAIT EN POSITION 19DANS LA REGLE  3
CONTEXTES POSSIBLES :
 :  C -A
 :  B -A

 :  C -A
 :  -A

 :  -A -B
 :  B -A

 :  -A -B
 :  -A
```



```
     C
CET IDENTIFICATEUR APPARAIT EN POSITION 11DANS LA REGLE  5
CONTEXTES POSSIBLES :
 :  B -A
 :  D

 :  -A
 :  D

 :  B -A
 :  A

 :  -A
 :  A


     D
CET IDENTIFICATEUR APPARAIT EN POSITION 15DANS LA REGLE  4
CONTEXTES POSSIBLES :
 :  -C
 :  A

 :  -D
 :  A

 :  -C
 :  D

 :  -D
 :  D

*STOP* Ø
```

. On remarquera l'existence d'une règle 0 factice qui sert à établir la convention (AA, AA) $\mathcal{C}$ (Ø, Ø) (AA est l'axiome)

. par exemple le contexte de (A, -A) dans la règle 3 :

$$BB \longrightarrow -AoB + BB \qquad /3/$$

$D_3(A) = \emptyset$

$D_3(-A) = B$; donc tout élément de $\mathcal{C}_3(A, -A)$ contiendra $(\emptyset, \{B\})$

de plus, l'extrémité de A coïncide avec celle du 1er membre BB :

$$(A, -A)\ P_r\ (\emptyset, BB)$$

$(BB, -BB)\ \mathcal{C}\ (\{-B\}, \{-C\}), (\{-B\}, \{-D\}), (\{C\}, \{-C\}), (\{C\}, \{-D\})$

et donc

$(\emptyset, BB)\ \mathcal{C}\ (\emptyset, \{-B\}), (\emptyset, \{C\})$

et donc

$(A, -A)\ \mathcal{C}_3\ (\emptyset, \{B, -B\}), (\emptyset, \{B, C\})$

2.2.- UTILISATION.

Sous les mêmes hypothèses H_1 et H_2 du théorème précédent nous avons :

Théorème :

Soit f une forme engendrée par un système S,
$f' \in S(A)$ sous forme de f d'origine O et d'extrémité E.
Soit α l'ensemble des primitives et opposés de primitives apparaissant en O et n'appartenant pas à f', β celles apparaissant en E et n'apparaissant pas en f',
alors

$(A, -A)\ \mathcal{C}\ (\alpha, \beta)$

La réciproque est fausse.

La démonstration ainsi qu'un contre exemple de la réciproque sont donnés dans l'annexe 3.

Si donc dans la procédure REMONTE on observe $(\alpha, \beta) \notin \mathcal{C}_r(A, -A)$ il est impossible que le choix de la règle r nous conduise au résultat.

Démonstration :

(voir annexe n° 3)

L'utilisation du contexte devient alors évidente : lorsque la procédure REMONTE doit faire le choix d'une règle r où apparaît B, ce choix est guidé par $\mathcal{C}_r(B, -B)$: soit α et β les primitives (ou opposés de primitives) dont l'origine coïncide respectivement avec l'origine et l'extrémité de la sous-forme B reconnue ; on ne pourra choisir que les règles r vérifiant :

$(B, -B)\ \mathcal{C}_r\ (\alpha, \beta)$

On peut à partir de là en tirer une condition de déterminisme de la procédure REMONTE : soit $\mathcal{R}_B$ les numéros de règles où B a une occurrence dans le second membre :

$$\forall B \quad \forall r_1, r_2 \in \mathcal{R}_B \quad r_1 \neq r_2 \Rightarrow \mathcal{C}_{r_1}(B, -B) \cap \mathcal{C}_{r_2}(B, -B) = \emptyset$$

Remarque :

L'utilisation des contextes dans REMONTE ne se fait donc qu'en utilisant les éléments de $\mathcal{C}(A, -A)$. Cette remarque jointe fait que

$$\forall A \in V \quad \mathcal{C}(A, \emptyset) = \{(\alpha, \emptyset) \mid \exists \beta \quad (A, -A)\ \mathcal{C}\ (\alpha, \beta)\}$$
$$\mathcal{C}(\emptyset, A) = \{(\emptyset, \beta) \mid \exists \alpha \quad (-A, A)\ \mathcal{C}\ (\alpha, \beta)\}$$
$$\mathcal{C}(-A, A) = \{(\alpha, \beta) \mid (\beta, \alpha)\ \mathcal{C}(A, -A)\}$$

montre que le calcul de $\mathcal{C}$ n'est donc utile que pour les couples $(A, -A)$.

2.3.- CALCUL EFFECTIF DE $\mathcal{C}$.

$\mathcal{C}$ est définie par un système à point fixe et son calcul peut donc être entrepris par la méthode des approximations successives. Le domaine de définition de $\mathcal{C}$ étant fini , le nombre d'itération de l'approximation successive sera donc fini .

Le calcul peut cependant être accéléré par la proposition suivante qui montre que $\mathcal{C}$ peut se décomposer en un produit de deux termes plus simples

Proposition :

soit $\mathcal{C}'$ définie par :

$\mathcal{C}'_r(A, B) =$ si $(A, B)\ P_r(\emptyset, \emptyset)$ alors $D_r(A, B)$

sinon $\mathcal{C}'(P_r(A, B)) \dot{\cup} D_r(A, B)$ fsi

$\mathcal{C}'(A, B) = \bigcup_r \mathcal{C}'_r(A, B)$

alors $\mathcal{C}' \circ \mathcal{J}^* = \mathcal{C}$

Démonstration :

1.- Montrons d'abord que $\mathcal{C}' \circ \mathcal{J}^*$ vérifie le système définissant $\mathcal{C}$.

Soit $\mathcal{Z}$ l'application qui à la relation R fait correspondre $\mathcal{Z}(R)$ définie par :

$\mathcal{Z}_r(R)(A, B) =$ si $(A, B)\ P_r(\emptyset, \emptyset)$ alors $\mathcal{J}^*(D_r(A, B))$

sinon $R(P_r(A, B)) \dot{\cup} \mathcal{J}^*(D_r(A, B))$ fsi

$\mathcal{C}(R) = \bigcup_r \mathcal{Z}_r(R)$

Montrons que si

$R \subset \mathcal{C}' \circ \mathcal{J}^*$ alors $\mathcal{C}(R) \subset \mathcal{C}' \circ \mathcal{J}^*$:

distinguons les cas :

- soit $(A, B)\ P_r(\emptyset, \emptyset)$; dans ce cas

  $(A, B)\ \mathcal{Z}_r(R)\ (\alpha, \beta) \iff (\alpha, \beta) \in \mathcal{J}^*(D_r(A, B))$

  or dans ce cas, par définition de $\mathcal{C}'$

  $(A, B)\ \mathcal{C}' \circ \mathcal{J}^*(\alpha, \beta)$

- soit $\neg(A, B) P_r(\emptyset, \emptyset)$; dans ce cas

  $(A, B)\mathcal{Z}_r(R)\ (\alpha, \beta) \iff$

  $\exists (\alpha_1, \beta_1)\ ,\ (\alpha_2, \beta_2)$ $\alpha = \alpha_1 \cup \alpha_2$

  $\beta = \beta_2 \cup \beta_2$

  et $(A, B)\ P_r \circ R\ (\alpha_1, \beta_1)$

  et $(A, B)\ D_r \circ \mathcal{J}^*(\alpha_2, \beta_2)$

dans ce cas

$(A, B)\ D_r\ (\gamma, \delta)\ \mathcal{J}^*\ (\alpha_2, \beta_2)$

$(A, B)\ P_r\ (U, V)\ R\ (\alpha_1, \beta_1)$

or $R \subset \mathcal{C}' \circ \mathcal{J}^*$

donc

$\exists (\mu, \nu)\ (A, B)\ P_r(U, V)\ \mathcal{C}'(\mu, \nu)\ \mathcal{J}^*(\alpha_1, \beta_1)$

donc

$(A, B)\ \mathcal{C}'\ (\gamma \cup \mu\ ,\ \delta \cup \mu)\ \mathcal{J}^*(\alpha\ ,\ \beta)$

donc

$\mathcal{C}(R) \subset \mathcal{C}' \circ \mathcal{J}^*$; comme la relation vide Ω vérifie $\Omega \subset \mathcal{C}' \circ \mathcal{J}^*$

$\mathcal{C} = \overset{\infty}{\underset{i=1}{\cup}} \mathcal{C}(\Omega)$ vérifie $\mathcal{C} \subset \mathcal{C}' \circ \mathcal{J}^*$

2- Réciproque :

La démonstration est identique en prenant cette fois pour $\mathcal{C}$ la fonctionnelle définissant $\mathcal{C}'$ et en montrant que

$$R \circ \mathcal{J}^* \subset \mathcal{C} \Rightarrow \mathcal{C}(R) \circ \mathcal{J}^* \subset \mathcal{C}$$

d'où il vient par le même raisonnement

$$\mathcal{C}' \circ \mathcal{J}^* \subset \mathcal{C}$$

Le calcul de $\mathcal{C}$ va donc se faire :

a) par le calcul de $\mathcal{C}'$.

b) puis par le produit $\mathcal{C}' \circ \mathcal{J}^*$ qu'il est inutile de détailler.

calcul de $\mathcal{C}'$:

```
% initialisation %
Pour chaque  A ∈ V  faire
    pour chaque règle  r  où  A  a une occurrence faire
    pour  X ∈ {A, -A}  faire
        si  P_r(X, ∅) = (∅, ∅)  alors  C_r(A, ∅) ← (D_r(A),  ∅)
        si  P_r(∅, -X) = (∅, ∅) alors  C_r(∅, -A) ← (∅, D_r(-A))
        si  P_r(X, -X) = (∅, ∅) alors  C_r(A, -A) ← (D_r(A), D_r(-A))
    fait  fait  fait
% itération calculant le point fixe %
 stabilisé ← faux  ;
tant que non stabilisé faire
    stabilisé ← vrai  ;
    pour chaque  A ∈ N U T  faire
        pour chaque règle  r  où apparaît  A  faire .
```



```
soit  (X, Y) = P_r(A, -A)  % Soit  -X = Y,  soit X = ∅  ou
                                          Y = ∅  %
si  X ≠ ∅  ou  Y ≠ ∅  alors pour chaque règle  s  où
                    apparaît  X  faire
        pour chaque (α, β) ∈ C_s(X, Y) faire
            γ ← α U D_r(A)
            δ ← β U D_r(-A)
            si  (γ, δ) ∉ C_r(A, -A)  alors
                stabilisé ← faux  ;
            C_r(A,-A) ← C_r(A,-A) U {(γ, δ)}
            fsi
        fait fait
    sinon % rien, le calcul a déjà été
        fait à l'initialisation  %
    fsi
fait fait fait
```

Note :

le calcul indiqué ne calcule que $\mathcal{C}_r(A, -A) \quad \forall A \in V$ (cf. remarque à la fin de §.2.2.).

Optimisation de l'algorithme.

L'optimisation générale des calculs par approximations successives (cf. chapitre 1,§.3.1.) nous conduit à ne prendre en considération que les éléments (α, β) calculés lors de l'itération précédente, les autres ayant déjà donné lieu au calcul du contexte. En utilisant cette optimisation le temps de calcul réduit à quelques secondes (moins de 5) pour l'exemple de la fin du chapitre (card (N) = 12 , card (T) = 8) traité sur IRIS 80.

2.4.- DÉTERMINISME POUR L'ANALYSE DESCENDANTE.

Considérons la procédure ADROITE : lorsqu'elle doit choisir entre les règles $A \rightarrow \lambda$ et $A \rightarrow \mu$, elle observe au point 0 considéré des primitives de $\mathfrak{J}^*(A)$, mais aussi des primitives de la première composante de $\mathcal{C}(A, -A)$.

Soit $\mathcal{C}_1(A)$ (resp. $\mathcal{C}_2(A)$) la première (la seconde) composante de $\mathcal{C}(A, -A)$

On observe donc en 0 un ensemble de primitives P tel que

$$\exists \alpha \in \mathfrak{J}^*(A) \ , \quad \exists \beta \in \mathcal{C}_1(A) \ , \ P = \alpha \cup \beta \ .$$

Le choix entre la règle $A \rightarrow \lambda$ et $A \rightarrow \mu$ pourra donc être résolu si :

$$\forall \alpha \in \mathfrak{J}^*(\mathcal{G}_r(\lambda)) \ , \quad \forall \beta \in \mathfrak{J}^*(\mathcal{G}_r(\mu))$$
$$\forall \gamma, \delta \in \mathcal{C}_1(A)$$
$$\alpha \cup \gamma \neq \beta \cup \delta$$

Une condition symétrique peut être établie pour AGAUCHE en remplaçant $\mathcal{G}_r$ par $\mathcal{G}_\ell$ et $\mathcal{C}_1$ par $\mathcal{C}_2$.

3.- LE CHOIX DU POINT DE DÉPART.

Le problème qui se pose est de choisir une primitive P servant de base de départ à la reconnaissance d'une forme de la classe A, qui doit être localisée dans une zone Z .

Pour espérer éviter les retours arrières dûs à de mauvais choix, on peut considérer les points suivants :

1) la primitive p doit être reconnue avec certitude ; une erreur de segmentation ou d'identification ne peut conduire qu'à une erreur.

2) p doit appartenir à la forme de nom A que l'on veut reconnaître

3) p devrait conduire à l'utilisation de règles où les choix se feront de la façon la plus déterministe possible

Le premier point concerne la détection des primitives et nous ne l'aborderons pas ici. Le second point indique que d'une certaine façon p doit être une sous forme *caractéristique* de A . Le troisième point va nous amener à considérer toutes les possibilités d'analyses nous permettant d'aboutir à A depuis p et de mesurer en un certain sens le degré d'indéterminisme qui pourra se produire.

3.1.- PRIMITIVES CARACTÉRISTIQUES, OBLIGATOIRES.

Nous allons ici définir les primitives qui caractérisent les classes de formes. Mais comme ces caractéristiques peuvent être absentes, nous serons aussi amenés à nous intéresser aux primitives apparaissant obligatoirement dans une forme.

3.1.1.- DÉFINITIONS.

Soit la relation F définie sur V par :

A F B $\iff$ B apparaît dans une règle de premier membre A .

Nous notons Γ la fermeture transitive de F . Par définition même :

$A \ \Gamma \ B \Rightarrow \exists f \in S(A) \ , \quad \exists f' \in S(B)$ f' est une sous-forme de f

Définition :

Soit $A, B \in V$; B est *caractéristique* de A si tout chemin dans le graphe F allant de A_1 (l'axiome) à B passe par A . On notera C(A) l'ensemble des formes caractéristiques de A .

Proposition :

Soit $g \in S(A_1)$ et p une primitive sous forme de g . Si $g \in C(A)$ alors il existe $f \in S(A)$ telle que f' est une sous-forme de g admettant p comme sous-forme.

Démonstration :

Il suffit de considérer l'arbre syntaxique associé à g : sa racine est A_1 ; tout chemin de A_1 à p passe par A puisque p est caractéristique de A , d'où le résultat.

Définition :

$\forall A \in V$ posons

$O(A) = \{ B \in V \mid \forall f \in S(A) , \exists f' \in S(B)$ f' est une sous-forme de $f \}$

$O(A)$ est l'ensemble des *formes obligatoires* de A .

Soit $\mathfrak{R}_A \subset \mathfrak{R}$ le sous-ensemble des règles de premier membre A et pour toute règle r , notons V_r l'ensemble des éléments de V apparaissant dans le second membre de la règle r .

Théorème :

$O(A)$ $(A \in V)$ est *la plus grande solution* du système suivant défini sur $\mathfrak{P}(V)$:

$$\forall A \in N \quad O'(A) = \bigcap_{r \in \mathfrak{R}_A} \left(\bigcup_{B \in V_r} O'(B) \right) \cup \{A\}$$

$$\forall a \in T \quad O'(a) = \{a\} .$$

Ce théorème est la conséquence immédiate des deux lemmes suivants :

Lemme 1 :

$O(A)$ contient toute solution $O'(A)$ du système :

Démonstration :

soit $f \in S(A)$; opérons par récurrence sur l'arbre syntaxique de f pour montrer que, pour tout B de $O'(A)$, il existe $f' \in S(B)$, f' sous forme de B ;

si l'arbre est de taille 1 $A \in T$ et le résultat est immédiat ;

supposons le résultat établi jusqu'au rang $n - 1$ et passons au rang n :

Supposons $B \neq A$, cas où le résultat est évident :

f se décompose selon une règle r : $A \to \lambda$; par définition

$B \in \left(\bigcup_{X \in V_r} O'(X) \right)$; soit $X \in V_r$ tel que $B \in O'(X)$;

il existe $f'' \in S(X)$ tel que f'' est une sous-forme de X ; par hypothèse de récurrence il existe $f' \in S(B)$ tel que f' est sous-forme de f'' et donc aussi de f . D'où le lemme 1.

Lemme 2 :

$O(A)$ $(A \in N)$ est une solution du système.

Ce second lemme s'établit sans difficulté en démontrant l'inclusion directe puis réciproque.

Exemple :

reprenons la description donnée au chapitre 2, §.1.3.1.

- e est une primitive caractéristique et obligatoire de "eau"
- h est une primitive obligatoire mais non caractéristique de "coque"
- k est une primitive caractéristique mais non obligatoire de "voiles" .

Remargue : une extension possible de la notion de caractéristique :

Si l'expérience montre que frêquemment $O(A) \cap T$ contient beaucoup d'éléments, il arrive malheureusement que $C(A) \cap T$ soit vide ; ce dernier cas se présente lorsque les primitives apparaissent dans de nombreuses sous-formes (comme c'est en particulier le cas des verticales et horizontales dans les dessins). Pour éviter cet inconvénient, on peut étendre la notion de caractéristique en ne considérant non plus la primitive isolée , mais la primitive (ou toute autre sous-forme d'ailleurs) dans son contexte au sens du §.2.. Par exemple, pour la description du chapitre 2 §.1.3.1. , h n'est pas caractéristique de "grand-voile" car h a une occurrence dans coque ; par contre h devient caractéristique de "grand-voile" dans un des contextes ({ -x, d} , {v, -v, a }) où x désigne v , d ou g ; h est caractéristique de "coque" dans le contexte ({g} , {-d}) .

3.1.2.- CALCUL EFFECTIF.

Le calcul de O :

nous sommes dans le cas favorable où le plus grand point fixe peut être calculé par approximation successive de la façon indiquée au chapitre 1 , §.3.1..

Le calcul de C :

plutôt que de déterminer tous les chemins de A_1 à chaque primitive p dans le graphe de F et d'en déduire C conformément à la définition, il est préférable de procéder de la façon suivante :

$C(A) = T - E(A)$

$E(A) = F_A^*(A_1)$

où F_A est la relation du graphe de F restreint à $V - \{A\}$ et F_A^* la fermeture transitive de F_A .

La correction de cette définition est une conséquence de la constatation suivante :

si le système est réduit supérieurement, pour $p \in T$ il existe un chemin dans F de A_1 à p ; or, s'il n'existe pas de chemin de A_1 à p dans F_A , tout chemin de A_1 à p passe par A .

3.2.- GARANTIR UN BON DÉBUT D'ANALYSE.

La mise en oeuvre de l'algorithme d' analyse a montré que l'analyse était très rapide si la forme de départ apparaissait dans peu de règles de description ; ceci est dû à une constatation évidente : le nombre de choix possibles est d'autant plus restreint que la forme et les formes englobantes apparaissent dans peu de règles. C'est ce point que nous allons préciser ici.

Pour chaque $A \in N \cup T$ nous dirons que A est déterministe ascendant si

$$\forall r_1 , r_2 \in \mathcal{R}_A \quad r_1 \neq r_2 \Rightarrow C_{r_1}(A, -A) \cap C_{r_2}(A, -A) = \emptyset$$

C'est-à-dire que le contexte permet de définir de façon déterministe le choix à faire dans la procédure REMONTE.

Soit alors

det (A, B) = inf(longueur (α)) α est un chemin du graphe F d'origine A et d'extrémité B vérifiant : pour chaque point C de α , C est déterministe ascendant.

det (A, B) est donc le minimum d'appels successifs à REMONTE pour lesquels les choix de règles se feront de façon déterministe à partir de l'appel de REMONTE (Zone, Arbre, O, E, A) avec racine (Arbre) = B .

On ne compte cependant pas dans les appels de REMONTE ceux effectués par les procédures d'anlyses descendantes lorsque sont choisies les règles utilisant des relations topographiques.

Remargue :

il aurait été possible d'introduire dans le critère det (A, B) des considérations de déterminismes sur l'analyse descendante ; c'est ce qui aurait été fait dans [HAT 77] d'une manière simplifiée.
Une définition plus complète (mais plus complexe aussi) peut être donnée en utilisant la condition de déterminisme donnée §.2.4. :
on évalue la taille du plus petit sous-arbre partiel qui peut être analysé de façon déterministe ascendante et descendante et det (A, B) sera alors cette valeur.

3.3.- CRITÈRES DE CHOIX D'UNE PRIMITIVE DE DÉPART .

Ayant à reconnaitre une représentation d'une forme A dans une zone Z il nous faut choisir pour point de départ de l'analyse une primitive p parmi un ensemble P_Z de primitives présentes dans cette zone :

Considérons les cas suivants :

. $p \in C(A)$

dans ce cas nous garantissons l'existence d'une représentation de A s'appuyant sur p ; mais il n'est pas sûr que cette représentation soit entièrement contenue dans Z .

. p vérifie det (A, p) maximum quand p parcourt $\Gamma(A) \cap P_Z$

dans ce cas, si l'analyse conduit effectivement à une analyse correcte,

pour au moins det (A, p) appels de REMONTE, le choix de la règle se fera de façon déterministe.

. $p \in O(A)$

on ne peut conclure dans ce cas : seule l'absence de p garantit que la représentation à reconnaître est absente.

Nous avons appliqué cette stratégie à notre analyseur en choisissant d'abord des éléments de p appartenant à $C(A) \cap O(A)$ tel que det (A, p) soit maximum, si $C(A) \cap O(A) \neq \emptyset$; sinon nous nous sommes restreints aux éléments de O(A) . Le temps d'analyse a été divisé par un coefficient 3 sur les quelques essais effectués.

Note :

le calcul effectif de det (A, p) n'est pas fait dans la version actuelle du compilateur de MIRABELLE et det(A, p) est remplacé par $\frac{1}{nb(p)}$ où nb(p) est le nombre d'occurrence de p dans la description ; c'est une approche très simplifiée du critère de déterminisme : le premier appel de REMONTE a choisi parmi nb(p) règles possibles. Ce critère simplifié , a cependant l'avantage de rester utile même dans le cas d'un dessin bruité où les tests sur le contexte ne sont pas toujours possibles (primitives absentes....).

4.- CONTEXTE DANS LE CAS DES RELATIONS TOPOGRAPHIQUES.

Au §.2. nous n'avons défini le contexte d'une sous-forme que lorsqu'il y avait coïncidence aux pôles avec d'autres sous-formes ; dans le cas particulier de la règle $A \longrightarrow R(B, C)$, le contexte de C est donc vide.

Or la procédure d'analyse REMONTE a besoin d'indications lui permettant de choisir les règles à mettre en oeuvre :

après avoir analysée la représentation $c \in S(C)$, supposons que la procédure REMONTE ait à choisir entre les règles

$$A \longrightarrow R_1(B, C) \quad \text{et} \quad D \longrightarrow R_2(E, C) \quad .$$

Le contexte qui peut permettre de décider est

- d'une part l'existence d'une représentation $b \in S(B)$ telle que $R_1(b, c)$ ou d'une représentation $e \in S(E)$ telle que $R_2(b, c)$,

- d'autre part l'apparition de l'assemblage des deux formes dans un contexte qui est celui de A ou celui de D .

La détermination de b ou de e peut être longue et si lors de leurs analyses les mêmes choix se reposent, on aboutit à une recherche exhaustive s'éxécutant en un temps exponentiel.

Nous allons ici donner une définition du contexte permettant de guider le choix par un nombre limité de tests.
L'utilisation de cette dernière définition n'a pas été intégrée dans MIRABELLE et nous ne pouvons donc pas juger de son efficacité.
Son intégration ne nécessiterait que l'introduction de relations supplémentaires pour que les hypothèses de la propriété énoncée au §.4.1.1. soient vraies.

4.1.- DÉFINITIONS.

4.1.1.- RELATIONS INDUITES.

Nous dirons qu'une relation R portant sur des formes *induit une relation* S si et seulement si :

$\forall f, f', g, g'$ (f' sous-forme de $f \wedge g'$ sous-forme de $g \Rightarrow (R(A, B) \Rightarrow S(f, g))$)

On notera $R \overset{*}{\rightarrowtail} S$

Intuitivement la relation induite est une relation encore vraie pour les sous-formes des formes initialement en relation.

De la définition même des sous-formes, on constate que $\overset{*}{\rightarrowtail}$ est transitive.

On notera T (comme Toujours vraie ou True) la relation toujours vraie qui vérifie donc :

$$\forall R \qquad R \overset{*}{\rightarrowtail} T$$

L'ensemble Rel des relations sera muni de l'intersection et :

$$(R \text{ et } S)\ (A, B) \iff R(A, B) \wedge S(A, B)$$

Propriété :

Si l'ensemble Rel de relations est fini et stable par intersection, alors pour toute relation R , il existe R' unique telle que

$R \overset{*}{\rightarrowtail} R'$ et

$\forall S \in Rel \qquad R \overset{*}{\rightarrowtail} S \quad \Rightarrow (R' \Rightarrow S)$

On notera $R \rightarrowtail R'$ et l'on dira que R' est *directement induite* par R .

Démonstration :

il suffit de prendre $R' = (\underline{et}\ S)$: $R \overset{*}{\succ\!\!-} S$

comme $A \overset{*}{\succ\!\!-} B$ et $A \overset{*}{\succ\!\!-} C \Rightarrow A \overset{*}{\succ\!\!-} B$ et C

et que $(B$ et $C) \Rightarrow B$, on obtient le résultat.

Exemple :

introduisons en plus des relations définies au §.1.3. du chapitre 2 la relation

$$AUDESSUS(A, B) \iff m_y(A) \geq M_y(B)$$

il est clair que PSUR n'induit que deux relations :

$$PSUR \overset{*}{\succ\!\!-} AUDESSUS$$

$$PSUR \overset{*}{\succ\!\!-} T$$

donc $PSUR \succ\!\!- AUDESSUS$

de même $INT \succ\!\!- T$

$AUDESSUS \succ\!\!- AUDESSUS$

4.1.2.- CONTEXTE TOPOGRAPHIQUE.

On supposera dans cette partie que toute règle faisant intervenir des relations sera sous la forme $A \longrightarrow R(B, C)$ avec $A \in N$, B et $C \in V$, $R \in Rel$. Cette restriction n'est pas fondamentale et on peut toujours se ramener à ce cas ; elle simplifie par contre considérablement les définitions.

Comme on l'a vu dans l'introduction du §.4., le contexte d'une forme B est dû en partie au fait que A est en relation avec une forme C - et ceci sera exprimé par d_r dans ce qui suit - et d'autre part, aux contextes dans

lesquels apparaissent les formes contentant A : ceci explique qu'une fois de plus, nous aurons une définition récursive par un système à point fixe.

Soit d_r la relation entre V et l'ensemble des parties de

$(Rel \times V) \cup (Rel^{-1} \times V)$ définie par :

soit $A \longrightarrow R(B, C)$ la règle r alors

$B\ d_r\ \{R'(C)\}$ et $C\ d_r\ \{R'^{-1}(B)\}$ avec $R \succ\!\!- R'$

$D\ d_r\ \emptyset$ pour $D \neq B$ ou $D \neq C$.

Notons $p_r(B) = A$ le premier membre de la règle r si B apparaît dans le second membre

$p_r(B)$ = indéfini si B n'apparaît pas dans le second membre de règle r .

On définit alors le *contexte topographique* $\mathcal{C}$ par :

$$\mathcal{C}(A) = \bigcup_{r \in \mathcal{R}} \mathcal{C}_r(A)$$

$$\mathcal{C}_r(A) = \mathcal{C}(p_r(A)) \dot{\cup} d_r(A)$$

$$\emptyset \in \mathcal{C}(A_1) \qquad (A_1 \text{ est l'axiome})$$

avec comme §.2. : $A \dot{\cup} B = \{\alpha \cup \beta \mid \alpha \in A, \quad \beta \in B\}$

Exemple :

Reprenons le système donné au §.1.3.2. du chapitre 2 et modifions-le de la façon suivante :

scène = PSUR (voiliergrée, eau) /1/ $\cup$ PSUR (voilier dégréé, quai) /2/

quai = h o -v /3/ $\cup$ v × h /4/

voilier gréé = PSUR (voiles, coque) /5/

voilier dégréé= PSUR (mat, coque) /6/

coque = g + h + -d /7/

..... (inchangé) .....

Calculons partiellement $\mathcal{C}$:

on a $\emptyset \in \mathcal{C}$(scène)

donc { AUDESSUS (eau)} $\in \mathcal{C}_1$(voilier gréé)

car PSUR $\rightarrowtail$ AUDESSUS

donc { AUDESSUS (eau), AUDESSUS^{-1}(voiles)} $\in \mathcal{C}_5$(coque)

de même on trouve :

coque $\mathcal{C}_6$ { AUDESSUS(quai) , AUDESSUS^{-1}(mat)}

h $\mathcal{C}_7$ { AUDESSUS(quai) , AUDESSUS^{-1}(mat)}

h $\mathcal{C}_7$ { AUDESSUS(eau) , AUDESSUS^{-1}(voiles) }

4.2.- Utilisation des contextes topographiques.

Le contexte $\mathcal{C}_r$ indique toutes les formes qui peuvent être en relation avec une forme donnée. Comme nous l'avons déjà dit au début de §.4., nous n'allons pas rechercher la présence de toutes ces formes en relation avec la forme indiquée, ceci pour éviter une recherche exhaustive trop longue. Nous pouvons nous limiter à tester la présence de sous-formes caractéristiques ou obligatoires : comme le contexte a été défini en utilisant les relations directement induites par les relations concernées, il reste valable pour les sous-formes et donc en particulier, pour des primitives d'une forme.

4.2.1.- Contexte vérifié ou satisfait par une forme.

Soit $b \in S(B)$ une forme présente dans la forme à reconnaître.

Soit $\alpha = \{ R_1(A_1), \ldots , R_k(A_k)\}$ un ensemble de couples de

$(Rel \cup Rel^{-1}) \times V$.

On dira que b *vérifie* α si et seulement si :

$\forall i \in [1, k]$ $\exists p \in C(A_i)$, $\exists q$ représentation de p présente dans la forme à reconnaître et telle que $R_i(b, q)$.

On dira que b *satisfait* α si et seulement si :

$\forall i \in [1, k]$ $\forall p \in O(A_i)$, $\exists q$ représentation de p présente dans la forme à reconnaître et telle que $R_i(b, q)$.

Exemple :

si nous reprenons l'exemple précédent nous avons :

C (voiles) = {f, a, b}

O (voiles) = {h, v, a, d}

C (eau) = O(eau) = {e}

coque vérifie { AUDESSUS (eau) , AUDESSUS^{-1}(voiles) } s'il existe une représentation q de f , de a ou de b vérifiant AUDESSUS (q, coque) et s'il existe une représentation q de e vérifiant AUDESSUS (coque, e) .

coque satisfait { AUDESSUS (eau), AUDESSUS^{-1}(voiles)} s'il existe des représentations de h, v, a, d en relation AUDESSUS avec coque ainsi qu'une représentation q de e vérifiant AUDESSUS (coque, e) .

4.2.2.- MISE EN PLACE DES TESTS SUR $\mathcal{C}$ DANS REMONTE

La procédure REMONTE , ayant identifié $b \in S(B)$ doit déterminer s'il est raisonnable d'espérer terminer l'analyse en choisissant la règle r Or

$$B \ \mathcal{C}_r \ \alpha_1, \ldots, \alpha_n$$

S'il existe i tel que B vérifie α_i , il y a *présomption* que les formes devant être en relation avec B pour terminer l'analyse soient présentes.

S'il n'existe pas de i tel que B satisfasse α_i , alors il y a *certitude* que l'analyse ne peut pas aboutir en utilisant la règle r .

Reprenons l'exemple précédent :

nous avions coque $\mathcal{C}_5$ α = { AUDESSUS (eau), AUDESSUS^{-1}(voiles)}

coque $\mathcal{C}_6$ β = { AUDESSUS (quai), AUDESSUS^{-1}(mat)}

supposons que coque soit identifié. REMONTE doit choisir entre les règles 5 et 6 pour la poursuite de l'analyse.

Si coque ne satisfait pas α , alors la règle 5 peut être éliminée. On remarquera que cela nous conduit à rechercher des occurrences de cinq primitives : h, v, a, d et e . Tester si coque satisfait β conduit à rechercher une occurrence de v pour mat et deux occurrences pour quai : l'une de h et l'autre de V .

Une stratégie possible est donc de limiter le choix aux seules règles r telles qu'il existe $\alpha \in \mathcal{C}_r(B)$ tel que B satisfait r . Cette stratégie est sûre et n'écarte que des règles dont l'emploi rend impossible la réussite de l'analyse.

Une stratégie différente sera de rechercher si B vérifie un α tel que $B \ \mathcal{C}_r \ \alpha$. A partir de cette indication , on choisira en premier lieu les règles r vérifiant cette condition, mais en cas d'insuccès, les autres règles devront être aussi considérées.

Le désavantage apparent de cette seconde stratégie est compensé par les considérations suivantes :

a) le test B vérifie α est plus simple et n'implique pas (en général) une recherche d'un grand nombre de primitives ; dans l'exemple traité, tester si coque vérifie α implique la recherche d'une primitive parmi f, a ou b et la recherche d'une primitive e ; tester si coque vérifie β est immédiat car $C(mat) = C(quai) = \emptyset$: coque vérifie toujours β .

b) si une règle r est choisie, l'analyse va redémarrer pour la sous-forme en relation par la recherche d'une primitive caractéristique, or cette recherche a déjà été faite pour la vérification.

c) cette stratégie peut parfois *combiner* les avantages de la précédente : si $O(A) \cap C(A) \neq \emptyset$ on testera la présence d'une des primitives de $O(A) \cap C(A)$: son abscence implique que B ne vérifiera pas α (donc rejet à coup sur), sa présence nous ramène au cas précédent.

Cette seconde stratégie deviendra donc particulièrement intéressante lorsque la recherche des primitives est coûteuse, c'est le cas du traitement d'images en particulier.

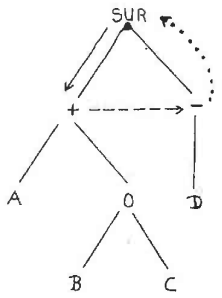
5.- MISE_EN_OEUVRE_DU_COMPILATEUR.

Le compilateur de MIRABELLE permet de prendre en compte de façon conversationnelle la définition des primitives et des différentes règles de la description de la classe de formes que l'on considère. Après une phase d'accueil où l'utilisateur peut demander au système différents rappels sur le mode de travail, il analyse la définition des primitives puis l'ensemble des règles de la description

Les résultats sont

- un fichier des primitives

- un fichier des règles; l'expression associée au second membre de chaque règle est représentée par son arbre habituel (cf. figure 3.3) et de façon à permettre un cheminement dans tous les sens à partir d'un noeud quelconque (cf. la procédure REMONTE) : ce point se réalise facilement en



SUR(A + (B o C), -D)

figure 3.3.

prenant deux liens : un lien du père vers son premier fils, un lien du fils x vers son cadet y ; si y n'a pas de cadet (il est benjamin) alors le deuxième lien désigne le père et est marqué pour indiquer ce fait.

- pour chaque règle r ses initiales : $\mathcal{I}^*(\theta_r(r))$, $\mathcal{I}^*(\zeta_\alpha(r))$

- un fichier des contextes immédiats :

$\forall A \in N \cup T \quad (\forall \alpha, \beta \ \{ r \mid \mathcal{C}_r(A) = (\alpha, \beta)\})$

- un fichier des sous-formes de chaque forme :

$\forall A \quad \Gamma(A)$ (cf. §.3.)

- un fichier des "bons points de départ" pour tout A de N

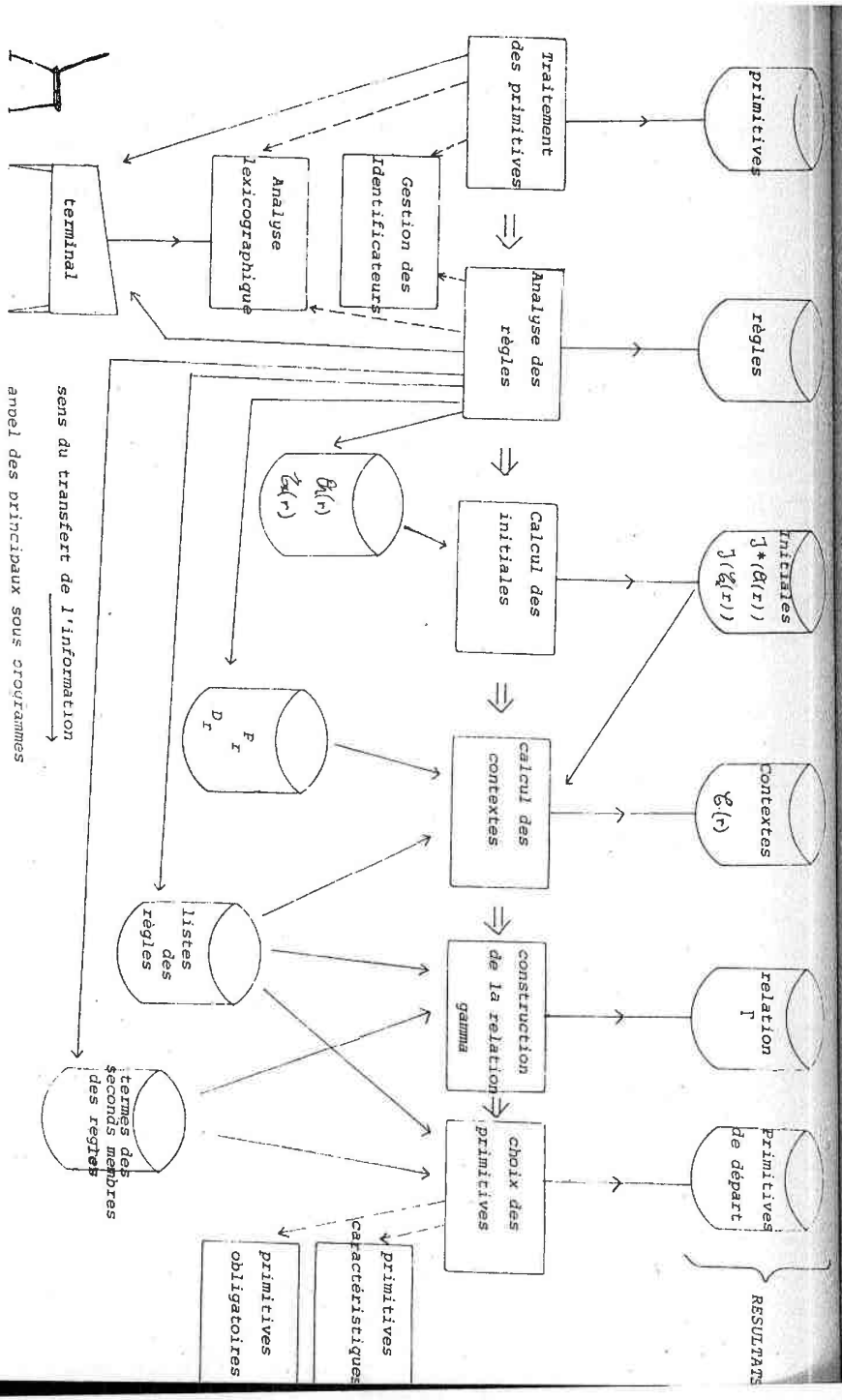
- pour chaque $A \in N$, la liste des règles qui le définissent.

Le programme travaille ligne par ligne et teste au fur et à mesure la correction syntaxique des définitions. Il signale en outre, en fin de définition si chaque forme est effectivement définie (ou productive, ou encore la réduction inférieure en termes de théorie des langages), et si chaque sous-forme est utilisée (ou accessible ou encore la réduction supérieure en terme de théorie des langages).

Le schéma de la page suivante décrit sommairement l'organisation générale du système et l'enchaînement des principales étapes.

Le temps de calcul pour une description de 18 règles, 9 primitives, 12 inconnues est de l'ordre de 10 s. sur IRIS 80, l'espace utilisé de 23 K mots (sans recouvrement). Les programmes ont été rédigés en FORTRAN en utilisant quelques procédures en langage d'assemblage pour des cas particuliers (manipulations de bits). L'ensemble est facilement transportable et a été effectivement implanté sur MITRA 125 (fonctionnement en mode 15).

On trouvera à la suite du schéma un exemple complet de compilation de description, mais sans le listage de tous les renseignements.




```
:roger
PAGES?46
DISQUE?56
EXECUTION F4LIB VERSION V10-05 11*12*06*
  BONJOUR    ICI L'ANALYSEUR DE DESCRIPTION

JE SUIS CHARGE D'ANALYSER SUCCESSIVEMENT ET DANS CET ORDRE LES
DEFINITIONS QUE VOUS ALLEZ ME FOURNIR A SAVOIR:
DEFINITIONS DES PRIMITIVES,DES RELATIONS,DE LA DESCRIPTION(GRAMMAIRE)
      SOUHAITEZ VOUS AVOIR:UN RESUME SUR LA DEFINITION DES PRIMITIVES
                           UN RESUME SUR LA DEFINITION DES RELATIONS
                           UN RESUME SUR L'ECRITURE DE LA GRAMMAIRE
                           UN EXEMPLE
                           LES GENERALITES
                           TOUT(RESUMES+EXEMPLE+GENERALITES)
?juste un exemple
EXEMPLE :
*DEFINITION DES PRIMITIVES
HOR:T(0,5) , VERT T(90,10),ARC:C(0,10)  .
*DEFINITION D''UNE RELATION
 INTERIEUR 5,0,0,0,0
 .
*LA DESCRIPTION SYNTAXIQUE
SCENE : INTERIEUR(BOITE,BOITE)
BOITE : -VER + HOR +VER )@HOR
BOITE: (- VER+HOR+VER)@ ARC
 .

 DEFINITIONS DES TERMINAUX
? hor:t(0,5) , vert:t(90,5) , obd:t(45,30) obg t(135,30)
?a>c\\:c(90,20)  b:c(270,20)  e c(180,20)  f c(0,200)
?k:c(135,20)  f:c(0,20).
              I

 PRIMITIVE 2 FOIS DEFINIE,DERNIERE DEFINITION RETENUE (52)
       1       0       5
       1      90       5
       1      45      30
       1     135      30
       2      90      20
       2     270      20
       2     180      20
       2       0      20
       2     135      20
 ACTUELLEMENT VOUS NE POUVEZ PAS DEFINIR DE NOUVELLES RELA TIONS
 REGLES DELA GRAMMAIRE:
?*
?scene     : psur(voiliergree , eau)          1
?scene     : psur( voilier , quai)            2
? eau      : e+ eau                           3
? eau      : e                                4
?voilier   : psur(mat , coque)                5
?voiliergree: psur(voiles , coque)            6
?quai      : vert + hor                       7
?quai      : hor * -vert                      8
?voiles    : base +( mat@grandevoile)         9
?voiles    : base +(mat@grandevoile * foc)    10
?base      : vert                             11
?mat       : vert                             12
?grandevoile: (hor*cordage +obd) @ a          13
? foc      : cordage+(((f*cordage)+k)@b)      14
?cordage   : vert                             15
?corg\dage: obd                                 16
?cordage   : obg                              17
?coque     : obg + hor + -obd                 18
?.
```

```
 VOULEZ-VOUS LA CODIF DES REGLES?
?n
  VOULEZ VOUS LES FILS DE CHAQUE N.T. ?
?oui
 NT = 1
    2      VOILIERGREE
    3      EAU
    4      VOILIER
    5      QUAI
    6      MAT
    7      COQUE
    8      VOILES
    9      BASE
   10      GRANDEVOILE
   11      FOC
   12      CORDAGE
   61      HOR
   62      VERT
   63      OBD
   64      OBG
   65      A
   66      B
   67      E
   68      F
   69      K
 NT = 2
    6      MAT
    7      COQUE
    8      VOILES
    9      BASE
   10      GRANDEVOILE
   11      FOC
   12      CORDAGE
   61      HOR
   62      VERT
   63      OBD
   64      OBG
   65      A
   66      B
   68      F
   69      K
 NT = 3
    3      EAU
   67      E
 NT = 4
    6      MAT
    7      COQUE
   61      HOR
   62      VERT
   63      OBD
   64      OBG
 NT = 5
   61      HOR
   62      VERT
 ..................
```

Γ (scène)

Γ (voiliergree)

Γ (eau)

Γ (voilier)

Γ (quai)


```
 LISTE DES INITIALES ?
?oui

SCENE
 :  E
 :  HOR
 :  VERT

VOILIERGREE
 :  OBG

EAU
 :  E

VOILIER
 :  OBG

QUAI
 :  HOR
 :  VERT

MAT
 :  VERT

COQUE
 :  OBG

VOILES
 :  VERT

BASE
 :  VERT

GRANDEVOILE
 :  HOR A

FOC
 :  OBG
 :  OBD
 :  VERT

CORDAGE
 :  OBG
 :  OBD
 :  VERT
 LISTES DES FINALES?
?non
 VOULES VOUS LES CONTXTES?
?oui                                  sortie des contextes.

     SCENE
CET IDENTIFICATEUR APPARAIT EN POSITION  ØDANS LA REGLE  Ø
CONTEXTES POSSIBLES :
 :                          contexte vide introduit pour l'axiome SCENE par
 :                          l'intermédiaire d'une règle factice de numéro 0.

     VOILIERGREE
CET IDENTIFICATEUR APPARAIT EN POSITION 13DANS LA REGLE  1
CONTEXTES POSSIBLES :
 :
 :
```

```
     EAU
CET IDENTIFICATEUR APPARAIT EN POSITION 15DANS LA REGLE  3
CONTEXTES POSSIBLES :
 :  -E
 :

CET IDENTIFICATEUR APPARAIT EN POSITION 15DANS LA REGLE  1
CONTEXTES POSSIBLES :
 :
 :


     VOILIER
CET IDENTIFICATEUR APPARAIT EN POSITION 13DANS LA REGLE  2
CONTEXTES POSSIBLES :
 :
 :


     QUAI
CET IDENTIFICATEUR APPARAIT EN POSITION 15DANS LA REGLE  2
CONTEXTES POSSIBLES :
 :
 :


     MAT
CET IDENTIFICATEUR APPARAIT EN POSITION 19DANS LA REGLE 10
CONTEXTES POSSIBLES :
 :  HOR A -VERT
 :  -OBD -A -B -K

CET IDENTIFICATEUR APPARAIT EN POSITION 17DANS LA REGLE  9
CONTEXTES POSSIBLES :
 :  HOR A -VERT
 :  -OBD -A

CET IDENTIFICATEUR APPARAIT EN POSITION 13DANS LA REGLE  5
CONTEXTES POSSIBLES :
 :
 :
 .....
```



```
     E
CET IDENTIFICATEUR APPARAIT EN POSITION 11DANS LA REGLE  4
CONTEXTES POSSIBLES :
 :  -E
 :

 :
 :

CET IDENTIFICATEUR APPARAIT EN POSITION 13DANS LA REGLE  3
CONTEXTES POSSIBLES :
 :
 :  E

 :  -E
 :  E


     F
CET IDENTIFICATEUR APPARAIT EN POSITION 21DANS LA REGLE 14
CONTEXTES POSSIBLES :
 :  B -OBG
 :  K -OBG

 :  B -OBG
 :  K -OBD

 :  B -OBG
 :  K -VERT

 :  B -OBD
 :  K -OBG

 :  B -OBD
 :  K -OBD

 :  B -OBD
 :  K -VERT

 :  B -VERT
 :  K -OBG

 :  B -VERT
 :  K -OBD

 :  B -VERT
 :  K -VERT


     K
CET IDENTIFICATEUR APPARAIT EN POSITION 25DANS LA REGLE 14
CONTEXTES POSSIBLES :
 :  -OBG -F
 :  -VERT -OBD -A -B

 :  -OBD -F
 :  -VERT -OBD -A -B

 :  -VERT -F
 :  -VERT -OBD -A -B

*STOP* 0
```

CHAPITRE 4

MODELES DE DESCRIPTION POUR LA RECONNAISSANCE DES FORMES

CHAPITRE 4

MODÈLES DE DESCRIPTION POUR LA RECONNAISSANCE DES FORMES

Nous allons étudier dans ce chapitre les différentes approches utilisées pour la description de formes et leurs conséquences sur les algorithmes de reconnaissance. Après une étude bibliographique critique nous essayerons de dégager les idées essentielles de cette approche.

1.- ETUDE BIBLIOGRAPHIQUE.

Nous n'aborderons pas dans ce paragraphe les algorithmes qui ont été développés pour une application précise et limitée, on conçoit en effet qu'un programme opérant sur des photographies de visage, après avoir localisé le contour de la tête, recherche au milieu de ce contour les deux yeux, etc... ; ces programmes opèrent à l'aide des connaissances que le programmeur y a insérées directement et de façon ad hoc ; s'ils sont spécialisés , ils ont par contre l'avantage d'être efficaces et rapides.

Notre objectif, à l'opposé de ces réalisations, est de développer des méthodes générales : stratégie d'analyse s'appuyant sur les renseignements concernant l'univers décrit et sur les identifications déjà opérées sur la forme. Ainsi, nous espérons en même temps apporter une contribution à la

connaissance des méthodes de perception visuelle élaborée.

1.1.- MÉTHODE_DE_DESCRIPTION_DYNAMIQUE : ESP^3

Nous appelons dynamique une méthode de description dont les paramètres ne sont pas figés dès le départ, mais peuvent être fixés en cours d'analyse. De fait, il n'existe qu'une seule réalisation correspondant à cette catégorie : il s'agit d'ESP^3 (Extended Snobol for Picture Pattern Processing, ([SHA 77])). Donnons-en brièvement le principe qui est inspiré de SNOBOL.

Les formes reconnues et décrites sont un assemblage de segments de droites pouvant , par exemple, provenir de l'échantillonnage de tracés continus.

Une description en ESP^3 est en fait un programme au sens de SNOBOL. On peut y rencontrer des tests et des affectations telles que :

P = POINT (FIG , LEFTTOP)

qui affecte à P le point "en haut à gauche" de FIG.
C'est essentiellement cet aspect qui rend dynamique la description.

Mais l'originalité de SNOBOL a été d'offrir des procédures d'analyse automatique sur des descriptions ; cette particularité a été étendue ici aux figures ; ainsi considérons la description suivante d'un + (francisé pour permettre une compréhension intuitive immédiate) :

PLUS = DROITE / Tolérance de l'angle -20, +20 / ℵ L 1 ℓ /

+ DROITE / Tolérance de l'angle 70, 110/ ; intersection avec L1/

L1 est défini par effet de bord lors de la découverte d'un segment de droite dont l'angle avec l'axe de référence est ± 20° ; si un tel L1 est identifié le système se met à la recherche d'un segment de droite vertical qui coupe L1 ; en cas d'échec on recherche un nouvel élément L1 et le processus s'itère jusqu'à une identification complète d'un + ou jusqu'à un épuisement des possibilités.

Une autre description pourrait être celle d'une suite finie de + alignés : l'angle α de l'alignement sera fixé par l'identification des deux premiers + rencontrés et l'alignement sera ensuite testé sur les autres.

A l'aide de prédicats plus sophistiqués tels que "au-dessus", "parallèle", ..., on arrive ainsi à décrire des formes très complexes. La seule limitation tient aux champs qui sont accessibles pour chaque figure ; ainsi pour une figure de type cercle sont accessibles le rayon et le centre ; pour des formes plus générales, seules les coordonnées des huit points extrêmes en X et Y sont accessibles. Ce point mineur mis à part, la puissance de description, et donc celle potentielle de reconnaissance, est illimitée : c'est celle d'un langage de programmation.

Les mécanismes de reconnaissance nécessitent cependant des réserves. L'exploration de la description doit obligatoirement se faire de la gauche vers la droite et ceci présente deux inconvénients :
d'une part, il faut avoir découvert le *premier élément de la forme*, et donc il devra être obligatoirement présent : ESP^3 ne pourra donc analyser que des figures complètes ; d'autre part, ceci exclut une *stratégie d'analyse* élaborée permettant d'analyser la figure dans un ordre plus naturel.

Le temps d'exécution d'une reconnaissance par ESP^3 est long : jusqu'à une dizaine de minutes d'unité centrale sur gros calculateur pour une figure complexe d'une vingtaine de segments de droite. Ceci est dû à trois points :

- le temps d'exécution de l'interprète SNOBOL
- l'impossibilité de mettre en oeuvre une stratégie élaborée comme nous venons de le mentionner,
- le fait que l'analyse est effectuée par essai-abandon sans tenir compte des causes d'erreurs et des aspects intéressants des formes abandonnées : ceci peut conduire à des temps d'exécution exponentiels.

1.2.- Les méthodes syntaxiques.

Nous décrivons ici les méthodes se réclamant de l'approche syntaxique nous verrons que les autres méthodes décrites §.1.3. et §.1.4. s'en rapprochent souvent.

Les descriptions syntaxiques ont d'abord été développées pour décrire et reconnaître les langages (langue naturelle ou langages de programmation), c'est-à-dire des chaînes de caractères. L'approche "génération par dérivation" généralement utilisée pour les chaînes de caractères fut naturellement la première mise en oeuvre pour le passage à plusieurs dimensions (§.1.2.1.). Cependant les développements les plus intéressants ont été obtenus lorsque la notion de concaténation à plusieurs dimensions s'est détachée de ces origines linguistiques pour s'attacher plus spécifiquement à la description de formes pluridimensionnelles (§.1.2.2.). Enfin, nous verrons (§.1.2.3.) les grammaires de graphes et comment elles peuvent intervenir pour la description d'images.

1.2.1.- Méthodes issues du principe de "dérivation".

Freeman [FRE 61] a donné pour les courbes une technique de codage qui permet de les représenter par une chaîne ; ce codage est fait par un suivi de la courbe avec des segments élémentaires orientés d'angle 0 , $\frac{\pi}{4}$, $\frac{\pi}{2}$, ..., $\frac{7\pi}{4}$.

Toutes les techniques utilisées pour les chaînes s'appliquent donc immédiatement. Dans ce cadre, l'étude de Feder [FED 68] a montré que l'on était obligé d'utiliser des outils complexes (grammaires contextuelles) dès que l'on veut décrire des courbes intéressantes ; ainsi la classe des courbes fermées n'est pas un langage à contexte libre. Nous pensons que cette

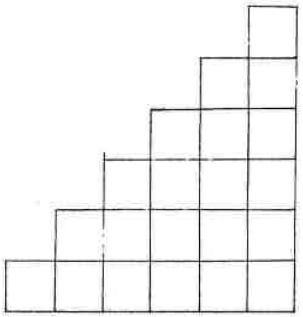
complexité apparente des courbes est due à un mauvais choix au niveau du codage : un segment de droite ou un arc de cercle sont aisés à reconnaître et forment bien mieux des primitives de codages de courbes. D'autre part, si l'on oublie l'aspect génératif pour s'intéresser à l'aspect descriptif, il est facile d'ajouter une condition de fermeture sans savoir comment obtenir cette condition à la génération , alors qu'il suffit de la constater à la reconnaissance.

C'est ainsi que Leydley et ses collaborateurs [LEY 64] ont procédé pour décrire le contour de chromosomes ; d'une part on impose au contour d'être toujours fermé, ce qui évite de faire apparaître ce point dans la description syntaxique ; d'autre part, le choix judicieux de cinq segments de courbes comme formes primitives conduit à décrire simplement tout chromosome. Ce fut la première application réelle des méthodes syntaxiques en reconnaissance des formes. Mais les techniques ci-dessus restent dans le domaine unidimensionnel des chaînes. Dès 1964, Kirsch [KIR 64] proposait un système de réécriture pluridimensionnelle :

on considère des règles du type suivant

en imaginant que l'on travaille sur un quadrillage infini correspondant à l'image discrétisée. Toute une étude de cette approche a été faite sous la direction de Rosenfeld ([ROS 71] , [OTA 73]) mais les résultats (théoriques) obtenus sont décevants : dès que l'on passe à des figures quelque peu complexes (triangles, ...), il faut avoir recours à des grammaires contextuelles difficiles à utiliser . Par exemple, pour engendrer la classe des triangles de côté $n (n > 0)$ du type de celui de la figure 4.1. , il faut utiliser 11 règles. Ceci est à rapprocher de ce que nous avons dit au sujet de la

complexité syntaxique des courbes par un codage de Freeman : le choix des primitives est mauvais ; elles sont trop élémentaires et ne correspondent pas à un concept manipulable dans une définition de haut niveau.



n = 6

figure 4.1.

On peut, cependant, imaginer des processus syntaxiques de reconnaissance travaillant au niveau "point" de l'image et permettant de décrire des formes simples telles que des tracés de rivières dans une zone où il y a des lacs. Mais, dans ces cas simples, pour être crédibles, les méthodes syntaxiques devraient être au moins aussi performantes que les méthodes classiques.

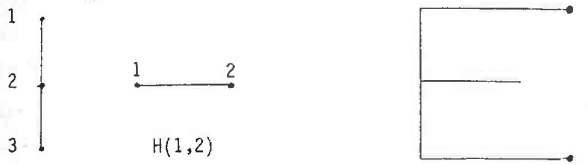
1.2.2.- MÉTHODES UTILISANT DES OPÉRATIONS DE CONCATÉNATIONS ÉTENDUES.

C'est dans ce cadre qu'entre l'approche que nous avons utilisée pour MIRABELLE : une forme est considérée comme un assemblage de sous-formes

au moyen de diverses opérations de concaténation. La grammaire à contexte libre correspond alors à un système à point fixe.

Le premier travail majeur dans ce domaine fut fait par Shaw [SHA 69] avec la définition de PDL ; nous nous en sommes inspirés pour MIRABELLE qui généralise ces idées, et nous ne reviendrons pas sur sa description.

Quelques années avant Shaw, Narasimhan [NAR 66] proposait un outil de description d'images et l'appliquait à la description de l'alphabet majuscules ; les opérations de concaténation se font par coïncidence de points privilégiés de la forme, mais ce nombre de points est quelconque et non plus limité à une origine et une extrémité. Pour chaque opération de concaténation il faut donc préciser quels sont les points à faire coïncider et ceux devant rester comme points de l'assemblage résultant (cf. figure 4.2.). Plus général que le modèle de Shaw, mais plus lourd aussi, ce modèle a été généralisé et étudié par J. Feder [FED 71] . L'idée essentielle ici encore est que les formes appartiennent à une classe (invariante par translation en particulier) et des opérations permettent de construire de nouvelles classes de formes en opérant par mise en coïncidence de point.



V(1,2,3)

Conc(V,H,H,H) $((1_1,1_2),(2_1,1_3),(3_1,1_4);(2_2,2_4))$

les trois premiers couples précisent les points qui doivent coïncider ; le dernier n-uplet précise les points retenus pour la figure résultante.

figure 4.2.

Divers travaux ([CLO 68] , [GRO 74]) utilisent comme outil de concaténation des relations indiquant des positions telles que par exemple "au dessous" . La formule $(x + y * z)/(x + 1)$ peut par exemple être décrite par la description :

formule = SUR (num, SUR(——, den))
num = AGAUCHE (x, AGAUCHE (+ , AGAUCHE(y, AGAUCHE(*, z))))
den = AGAUCHE (x, AGAUCHE (+, 1))

où SUR et AGAUCHE sont des fonctions qui, si les positions de leurs arguments vérifient une relation spécifiée, les réunissent en une forme unique.

La figure 4.3. montre trois formes s'accordant avec une telle description en l'absence de précision supplémentaire dans ces relations ; une normalisation adéquate permet facilement d'éliminer le dernier cas. Cette approche a aussi été introduite dans MIRABELLE .

$$\frac{x + y * z}{x + 1} \qquad \frac{x + y * z}{x + 1} \qquad x \; + \; \frac{y * z}{x + 1}$$

figure 4.3.

Le point commun de ces trois approches est de permettre de définir une classe de formes C à partir de classes de formes A et B en posant $C = A \perp B$ où $\perp$ est soit une *opération de concaténation*, soit une *relation de concaténation*. Le résultat est défini par :

$$C = \{ a \cup b \mid a \in A \;\; \underline{et} \;\; b \in B \;\; \underline{et} \;\; a \perp b \}$$

Dans toutes ces approches C peut parfois être une classe vide sans que A ou B soit vide, comme on l'a vu au chapitre 2 §.1..

Il faut aussi citer ici les "grammaires de coordonnées" [MIL 72] . Chaque forme possède un ensemble de valeurs (le plus souvent les coordonnées de points caractéristiques) et l'assemblage se fait par union sous réserve

qu'une relation spécifiée soit satisfaite entre ces valeurs. Ce n'est donc qu'un cas particulier de relations. Remarquons qu'on peut faire entrer P.D.L. dans ce type : par exemple l'opération de concaténation o (coïncidence des origines) est l'union avec la relation $(x_1 = x_2 \;\; \underline{et} \;\; y_1 = y_2)$ si (x_1, y_1) et (x_2, y_2) sont respectivement les coordonnées de l'origine de la première et de la seconde forme.

Les travaux d'Anderson [AND 77] se réfèrent explicitement à cette approche de grammaire de coordonnées. Pour décrire les formules mathématiques , les règles de concaténation utilisent des relations portant sur les coordonnées extrêmes des sous-formules ainsi que sur les coordonnées "centrales" (cf. figure 4.4.) .

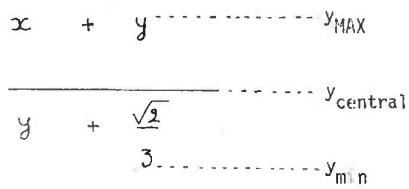


figure 4.4.

Si l'on considère que les coordonnées sont des valeurs particulières d'attributs attachés à une forme, rien n'interdit d'en introduire des nouvelles et dans ce cas, les travaux de Gallo [GAL 75] relèvent encore de la même approche :
elle associe à chaque forme des valeurs : coordonnées du centre pour un arc de cercle, angles, points particuliers. Les règles de description spécifient pour chaque forme la valeur des attributs en fonction des attributs des sous-formes, et indiquent l'assemblage des sous-formes par une relation portant sur

les attributs : par exemple : coïncidence de deux points et angle entre deux directions inférieures à 90° . Le système est utilisé pour décrire et reconnaître des dessins de pièces mécaniques.

Moayer et Fu [MOA 77] , décrivent des mosaïques carrées à l'aide d'une concaténation qui est différente de celles introduites jusqu'ici. Si X1, X2, X3 et X4 sont quatre mosaïques de même taille, $X_1X_2X_3X_4$ désigne l'assemblage des mosaïques ainsi qu'il est indiqué figure 4.5..

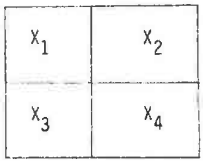


figure 4.5.

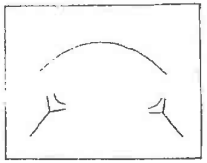


figure 4.6.

Les auteurs donnent alors la description de cinq classes d'empreintes digitales, une des classes étant représentée figure 4.6.. Chaque image est une mosaïque de 8 x 8 éléments, chacun d'eux indiquant la direction générale des empreintes dans la zone concernée et éventuellement le type d'intersection....

290 règles de description sont nécessaires pour des images d'une classe d'empreintes comme celle de la figure 4.6.. L'inconvénient majeur d'une telle description est évidemment sa complexité, et par voie de conséquence le peu de contrôle que l'on peut avoir sur elle : ainsi, cette description accepte la mosaïque présentée à la figure 4.7. qui ne correspond pas à une empreinte du type de celle de la figure 4.6. ; elle ne correspond peut-être pas non plus à une image d'empreinte digitale, ce qui expliquerait cette anomalie.

Les auteurs font cependant justement remarquer que , malgré la complexité de la description , le temps pris par l'analyse est 100 fois inférieur à celui du prétraitement (extraction des directions principales dans un élément de la mosaïque et codification de ces directions).

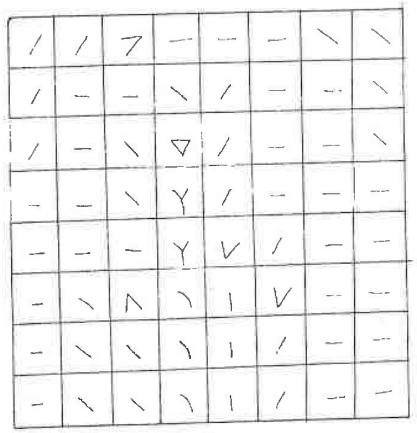


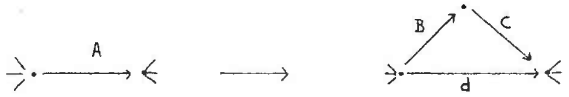
figure 4.7.

1.3.- GRAMMAIRE DE GRAPHES.

Une forme complexe peut souvent être abstraite en un graphe ; si l'on est capable de décrire le graphe , on est capable de décrire la forme : cette idée est la base de cette approche. Nous allons discuter des deux façons d'associer un graphe à une forme.

1.3.1.- Les arcs sont des formes.

On peut associer un graphe à une forme en considérant que chaque arc est une primitive et que les points du graphe symbolisent des coïncidences de points des primitives. C'est déjà ce point de vue que nous avons adopté §.1. du chapitre 2 ; et ainsi pour la règle de "réécriture" suivante :



avec les notations du chapitre 2 nous écrirons $A \longrightarrow (B + C)\ \underline{f}\ d$.

Cette approche a été généralisée par Cho [CHO 74] pour permettre aussi la prise en compte des résultats de Feder [FED 71] et Narasimhan [NAR 66] (cf. §.1.2.2.) : les arcs représentent des parties de primitives et les règles de "réécriture" font intervenir plus de deux noeuds. Cette approche a déjà été considérée dans les paragraphes précédents (§.1.2.2.) . L'approche "graphe" de ce problème permet à Cho de démontrer un théorème sur les limitations intrinsèques des grammaires "à contexte libre" dont les concaténations sont des coïncidences simples ou multiples. En particulier, on ne pourra décrire des maillages du type de celui de la figure 4.8. et dont la taille soit arbitrairement grande.

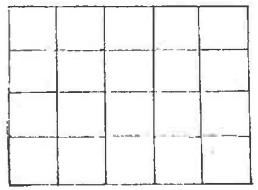
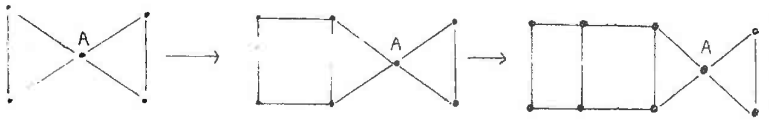


figure 4.8.

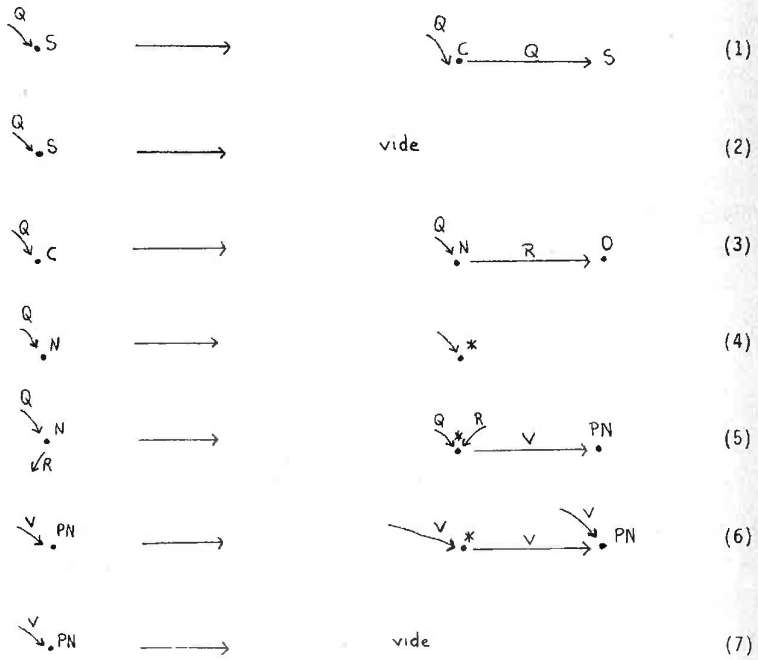
1.3.2.- Les formes sont les points.

Le point de vue précédent, parce qu'il se ramène à des systèmes d'équations du type de ceux du chapitre 2, est celui qui est le plus rarement adopté par les chercheurs travaillant sur les grammaires de graphes (cf. [FU 74] p. 279) : ce sont les noeuds qui jouent le rôle de non terminaux et on peut ainsi avoir la règle de "réécriture" suivante :

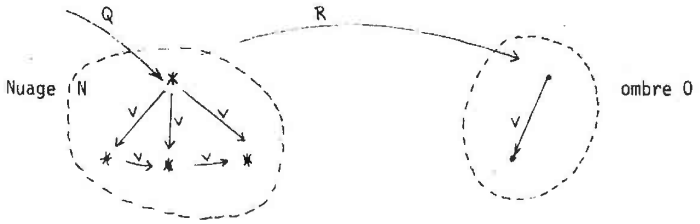
appliquant , on peut avoir la dérivation suivante :



Les arcs entre les points symbolisent ici des relations entre les formes ; prenons par exemple la grammaire décrivant les nuages et leurs ombres sur une image de satellite [BRA 77] : une relation Q (comme quelconque) relie les nuages entre eux, une relation V (voisin) les points d'un nuage ou ceux de son ombre, et une relation R associe son ombre à chaque nuage.



Nous n'avons pas décrit les règles permettant de décrire l'ombre (0) elles sont symétriques à celles des nuages (N) ; exemple de graphe ainsi généré :



Cet exemple, malgré sa simplicité, met en valeur les inconvénients de cette représentation de la forme par un graphe : un point du nuage est en relation R avec un point de son ombre alors qu'il est plus correct de dire que le nuage entier est en relation avec son ombre ; dans ce cas, on pourrait introduire une grammaire contextuelle qui permettrait à chaque point du nuage d'être en relation avec un point de l'ombre (voire avec tous les points de l'ombre) mais d'une part, nous obtenons ainsi une grammaire complexe, et d'autre part, ce raisonnement ne se généralise pas : telle forme peut être en relation avec telle autre, la relation étant vraie au niveau global sans que ce soit vrai au niveau des éléments.

Par exemple, considérons la figure 4.9. : B est un cercle réunion de quatre arcs de cercle b ; A est une succession de primitives a Une description structurée d'une telle forme conduit à définir cette forme comme étant constituéed'un A à l'intérieur d'un B , lequel A est une succession de a et B est un cercle constitué par la mise bout à bout de quatre b . On aboutit ainsi à la description de la figure 4.10. Mais si nous voulons n'introduire des relations qu'entre les primitives, nous ne pouvons plus utiliser la relation "intérieure" : aucun a n'est "intérieur" à un b !

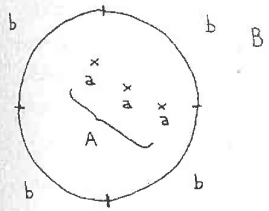


figure 4.9.

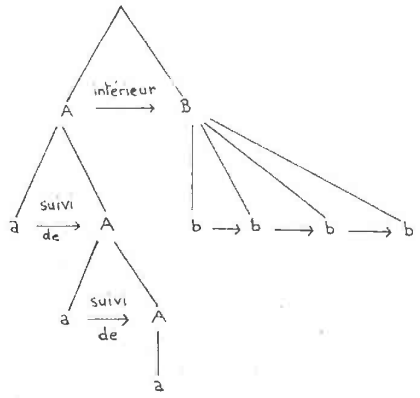


figure 4.10

Cet aspect des grammaires de graphes est par conséquent mal adapté à nos objectifs. Une seule application en a été faite : la reconnaissance nuage - ombre de nuages sur image de satellite qui nous a déjà servie d'exemple auparavant. Notons cependant, que cette approche a permis d'obtenir d'intéressants résultats théoriques sur les générations de graphes [MON 70] et en particulier un algorithme original de vérification de planéarité de graphe [AZE 75] .

1.3.3.- AUTRES APPROCHES STRUCTURELLES INSPIRÉES PAR LES GRAPHES.

Brayer [BRA 77] qui assure prendre une grammaire de graphes , travaille en fait sur une structure syntaxique du même type que celle de la figure 4.11 : les lignes obliques y figurent, le découpage en sous-formes et les autres liaisons spécifient l'arrangement de ces sous-formes ; on peut bien sûr considérer le tout comme un unique graphe , mais il nous semble conceptuellement maladroit de considérer que "être une partie de" est une relation de même nature qu'une relation de position. Comme pour la description associée à la figure 4.10., nous classerons plutôt cette approche avec celles proposées §.1.2.2.

La description de l'univers de Bajcsy et Lieberman [BAJ 74] est du même type, même si les auteurs ne font pas référence aux mots "structurels" et "syntaxiques" habituellement utilisés. En plus d'une décomposition structurée , présentée sous forme de graphe, ils ont dressé la liste des propriétés caractéristiques qui leur permettent, au niveau du traitement d'images, d'identifier leurs formes primitives telles que feuillages, troncs, ...

Herot [HER 77] utilise une description structurée qui n'est plus un arbre (cf. figure 4.11.) car il peut y avoir partage de composants : ainsi deux pièces voisines peuvent avoir le même mur en commun, deux segments de droite peuvent partager une extrémité.

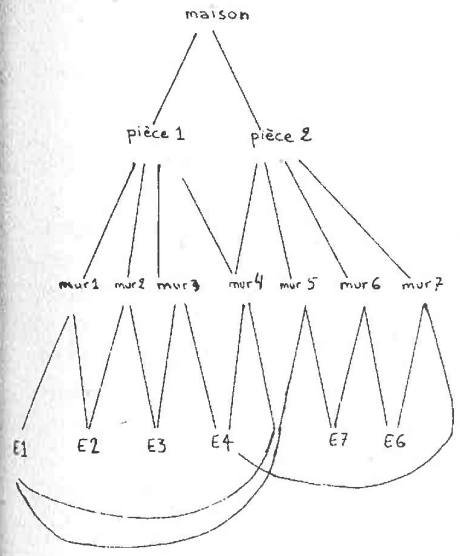
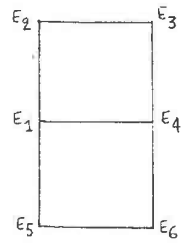


figure 4.11



Dans un tel cas, on peut encore se ramener au cas du §.1.2.2. , en considérant que les mises en commun sont spécifiées par les relations d'assemblages. Ainsi, les opérations de PDL spécifient la mise en commun de points, de même des opérations plus complexes pourraient spécifier la mise en commun de segments. L'avantage de la représentation graphique est de pouvoir exprimer ces mises en commun sans directement formaliser des opérations d'assemblages qui risqueraient d'être complexes ; son inconvénient est qu'un graphe ne décrit qu'une seule forme et non une classe avec des variantes comme c'est possible dans un système d'équations avec des opérations de concaténation.

1.4.- AUTRES MÉTHODES.

L'univers ne peut pas être compris comme une collection d'atomes séparés, mais comme une structure de sous-univers ayant chacun leurs *propriétés* et liés entre eux par des *relations*. Cette observation banale conduisit des chercheurs à obtenir des modèles de description structurée permettant la compréhension ou la reconnaissance d'objets. Les travaux de Winston [WIN 75] fournissent un bon exemple d'une telle description. Les objets élémentaires de son univers sont des volumes prismatiques qui peuvent être qualifiés par leur forme (brique, cube, pyramide, ...) et par des attributs (debout, couché, long, ...) Des relations peuvent indiquer des positions relatives entre des objets (se touchent, parallèles, posé sur, ...) ou des relations entre leurs formes (plus long, de la même forme, ...).

Ainsi la forme de la figure 4.12. sera par exemple décrite par la structure indiquée à la figure 4.13.. Réciproquement, un tel schéma peut permettre l'interprétation de scène de la figure 4.12. comme étant un arche. Ce modèle a été utilisé pour l'apprentissage de concepts à partir d'exemples : un premier exemple engendre un schéma modifié par les exemples et les contre-exemples successifs.

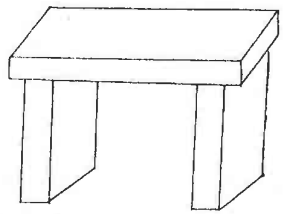


figure 4.12.

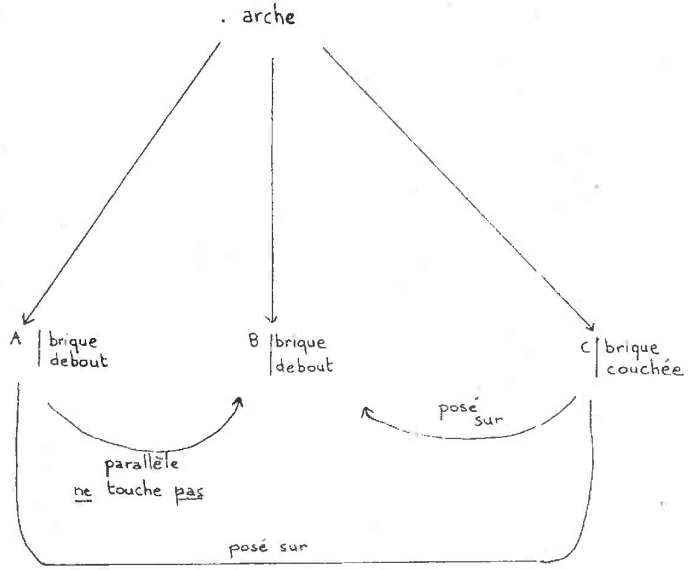


figure 4.13.

A la fois orienté vers la reconnaissance et l'apprentissage, HPL ([WIL 77]) s'est limité aux dessins bidimensionnels formés de segments de droite. On peut "apprendre" au système des figures élémentaires (figure 4.14.) ainsi que des scènes composées de figures élémentaires (figure 4.15.). Les scènes complexes sont représentées de façon hiérarchisée avec des poids attachés à chaque sous-forme, le total de poids des sous-formes reconnues

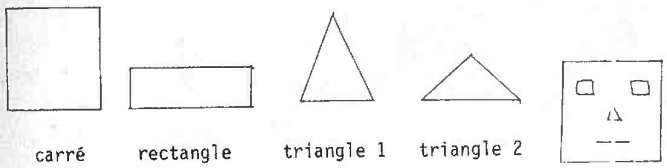


figure 4.14

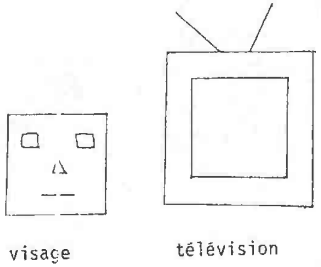
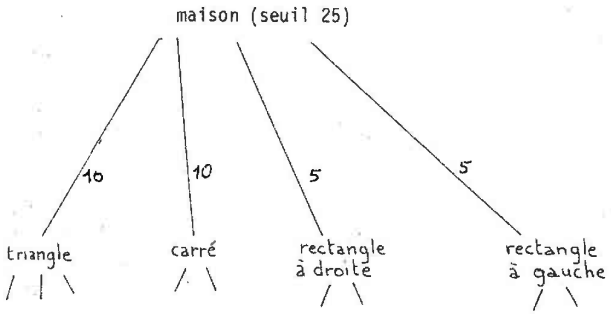


figure 4.15

devant atteindre un seuil minimum. Par rapport au modèle de [HER 77] (§.1.3.3.) cette technique présente l'avantage de permettre la description de plusieurs formes : supposons que dans la description de maisons la porte puisse apparaître à gauche ou à droite , on pourra avoir la description suivante :



On remarquera que toutes les maisons de la figure 4.16. conviendront car le total des poids sera respectivement de 25, 25, 30. Mais comment fait-on si l'on souhaite n'avoir qu'une seule porte ?

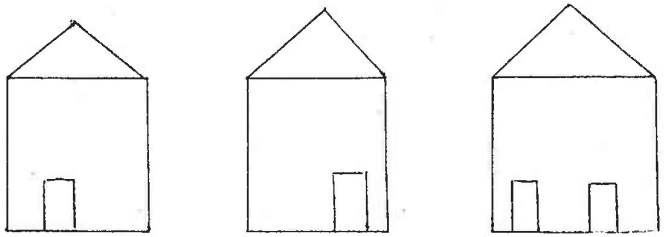


figure 4.16.

Les inconvénients d'HPL sont, selon l'auteur, d'une part le temps d'exécution des programmes dû en partie à l'écriture en LISP (4 à 30s sur Univac 1100), d'autre part, l'impossibilité de désigner des sous-formes en tenant compte de leur contexte d'occurrence. Ainsi si un oeil est un carré (figure 4.15.), à chaque occurrence d'un carré le système reconnaîtra un oeil.

Les connaissances sur une forme présentent des aspects très variés. Ainsi, on peut préciser qu'un navire est environ six fois plus long que large, qu'il est à quai avec une probabilité p ou qu'il est en mer avec la probabilité q . S'inspirant des travaux faits en intelligence artificielle [WIN 72] pour la représentation des connaissances et pour la prise de décision, Ballard et al. [BAL 77] associent une base de données complexes aux connaissances ; chaque noeud correspondant à un objet élémentaire est aussi associé à un ensemble de procédures qui vérifient les conditions, établissent les liaisons entre des différents objets, ... ; si l'on voulait établir un parallèle avec le travail décrit au chapitre 2 et 3 : ce sont ces procédures qui détectent les primitives, construisent l'arbre syntaxique et vérifient les propriétés particulières des objets. Un superviseur décide au fur et à mesure des objets à localiser, ceci en fonction de la facilité du traitement que l'on a à mettre en oeuvre et des renseignements que l'on peut tirer des identifications déjà faites :
à chaque procédure va être associée (entre autres) une précondition qui doit être valable pour qu'elle soit activée, ainsi qu'une valeur indiquant le coût associé à la procédure.

Au M.I.T. Freuder [FRE 77] a entrepris en parallèle un travail sur les mêmes idées :
description à l'aide de liens entre les différents objets, pour chaque objet ses caractéristiques radiométriques et géométriques, une stratégie sur l'action à poursuivre en tenant compte du coût de l'analyse.

L'auteur a appliqué ce réseau à la reconnaissance d'images de marteau sur des fonds divers et dans des orientations diverses. A partir d'un algorithme global de regroupement de régions, on essaie de déterminer approximativement une des formes primitives, le système mettant en oeuvre l'algorithme ad hoc pour la déterminer de façon certaine puis recherchant à partir de là la forme manquante.

Plus généralement, tous les travaux portant sur la représentation des connaissances en vue d'une compréhension peuvent être adaptés. Citons simplement le "blackboard" du projet Hearsay ([RED 77]) développé pour la compréhension du discours continu ; les auteurs essaient maintenant d'appliquer leurs idées à la compréhension d'images. Il s'agit d'un système de compréhension guidé par les données : l'apparition de certains types d'informations active des procédures qui recherchent et modifient des informations du "tableau noir". Une telle structure permet des modifications aisées des sources d'informations et des procédures de traitements ; son déroulement n'étant plus contrôlé de façon stricte , sa mise en oeuvre demandera des précautions si l'on souhaite obtenir le résultat voulu en un temps raisonnable.

2.- CRITÈRES POUR LE CHOIX D'UN MODÈLE.

2.1.- CONSIDÉRATIONS GÉNÉRALES.

La description de l'univers dans lequel l'on veut opérer la reconnaissance sert d'interface entre :

- l'utilisateur qui détaille les formes possibles, leurs structures , leurs propriétés,
- le système de reconnaissance qui va chercher les informations nécessaires pour décider si la forme identifiée intéresse l'utilisateur, mais aussi toutes les indications pouvant le guider dans sa recherche.

Pour l'utilisateur les qualités d'un modèle seront donc la *puissance* et la *simplicité*. On peut arguer que la simplicité n'est pas essentielle si l'on considère que l'utilisateur va faire une fois pour toute l'effort de réflexion et de codification nécessaire pour son problème : il faut cependant éviter que l'on arrive à une description aussi complexe que le programme ALGOL résolvant le même problème, avec les mêmes risques d'erreurs lors de la conception et de la codification. Puissance et simplicité sont d'ailleurs deux critères antagonistes ; considérons par exemple les grammaires programmées ; simplement pour engendrer l'ensemble des $a^n b^n c^n$ il faut sept règles (cf. [FU 74] page 44) et la correction de cet ensemble de règles n'est pas évidente, car les définitions ne sont pas statiques dans ce cas.

Une description dynamique comme ESP^3 (cf. §.1.1) est à la fois puissante et, si on se fixe des limites à l'utilisation des outils offerts, elle est aussi claire. Les descriptions statiques qui sont issues de la généralisation des grammaires à contexte libre (cf. §.1.2.2. et les méthodes pouvant s'y ramener) sont simples, structurent bien la description, mais sont limitées en puissance ; l'introduction d'opérateurs de concaténations multiples tels l'opération $\underline{f}$ (chapitre 2 §.1.2.) permet cependant d'accroître leurs possibilités ; comme nous l'avions déjà signalé de tels opérateurs sont en fait contextuels.

La description fournit au système de reconnaissance le but à atteindre : reconnaître et structurer une forme en accord avec les objectifs de l'utilisateur. Comme nous l'avons vu au chapitre 3, le système peut aussi y puiser des indications lui permettant de lever des ambiguités sur les formes à reconnaitre, ceci en tenant compte des contextes dans lequel cette forme est identifiée.

De plus, une identification partielle pourra lui fournir des indications sur la façon de poursuivre la reconnaissance : renseignements d'ordre géographique (localisation d'une forme) ou d'ordre typologique (type de forme recher-

chée, particularité de ces formes).

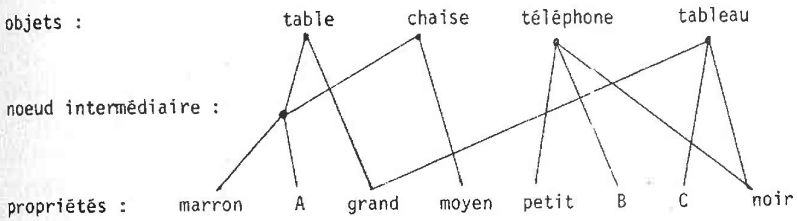
Par exemple, Williams [WIL 77] (cf. §.1.3) note que la reconnaissance d'une maison composée d'un triangle et d'un carré est à peine plus longue que la reconnaissance d'un triangle, car le triangle étant reconnu, le système est piloté par cette information et localise rapidement le carré. Son système n'utilise cependant pas les informations contextuelles pour lever les ambiguités comme nous l'avons vu.

ESP^3 par contre ne fera pas de confusion entre deux formes identiques dont l'interprétation est différente selon son contexte ; il fournit aussi des indications concernant la poursuite de l'analyse. Malheureusement, la stratégie de la reconnaissance est limitée par l'obligation d'explorer la description de la gauche vers la droite. Cette contrainte est due à l'aspect dynamique de la description : il faut lier au fur et à mesure des identificateurs à des formes ou des valeurs trouvées lors de l'exploration de la forme, et,comme dans un programme, ce déroulement est séquentiel.

Dans certains cas une reconnaissance partielle est possible pour détecter la présence d'une forme. Comme le notent Bajcsy et Tidhar ([BAJ 77]) si le téléphone est la seule forme noire de l'univers, la détection d'une forme noire suffit à conclure à la présence d'un téléphone. Ces auteurs proposent ainsi une description d'univers sous la forme d'un treillis. Cette structure permet alors facilement de vérifier si l'information est suffisante pour décider de la présence par calcul d'une plus grande borne inférieure un peu particulière. Illustrons ce point par l'exemple donné par les auteurs ; si on définit les objets suivants :

table	:	forme A , grand, marron
chaise	:	forme A , moyen, marron
téléphone	:	forme B , petit, noir
tableau	:	forme C , grand, noir

Ces définitions se traduisent dans le treillis schématisé ci-après :



La "plus grande borne inférieure" de marron et forme A donne un noeud intermédiaire qui ne permet pas de décider, alors que la seule donnée de forme B permet de conclure à la présence de téléphone.

Dans une structure statique donnée sous forme de graphe telle celle de Herot (§ 1.3.3.), il est facile d'exprimer toutes les liaisons entre les objets, élémentaires ou non ; ces liaisons peuvent très bien ne pas respecter le découpage structuré de la forme complexe comme nous l'avons vu sur l'exemple donné. Il y est aussi très facile, ayant identifié une sous-forme, de découvrir les sous-formes qui doivent être en relation avec elle, en déterminant celles qui lui sont directement liées par un arc du graphe. Nous avons vu au chapitre 3 que cette dernière possibilité nous était aussi offerte dans une description plus complexe comme celle que nous avions choisie au chapitre 2, mais au prix d'un calcul parfois complexe. Les avantages de facilité qu'offre la description par un graphe sont cependant contrebalancés par l'impossibilité de décrire plus d'une forme par un graphe ; on peut bien sûr enrichir le modèle en imposant que certaines liaisons ou parties peuvent être optionnelles mais cela ne résoud que très partiellement le problème.

Une représentation par réseaux et procédures telle que celle proposée par Ballard [BAL 77] présente l'avantage de la souplesse. Toute information pourra y être mise sous la forme d'une procédure faisant le traitement associé. C'est

cette grande souplesse qu'a exploité Winograd [WIN 72] pour l'analyse de la langue naturelle lorsqu'il a abandonné les modèles des grammaires formelles pour les réseaux de transitions à procédures ; cette approche a eu le succès que l'on sait. Cependant, je pense que si cette approche est prometteuse, il est prématuré de l'adopter dès aujourd'hui ; elle ressemble actuellement plus à un ensemble d'heuristiques où presque tout a été mis dans des procédures ad hoc ; l'approche formelle n'a pas encore été suffisamment défrichée et devrait fournir un cadre et des méthodes générales dans lesquels les procédures particulières devraient être réduites au minimum ; par exemple, simplement au choix de la procédure de recherche de formes primitives en fonction des informations déjà acquises sur la forme.

Compte tenu de ces remarques et des critiques que nous avons formulées lors de l'étude bibliographique, il nous semble qu'un bon compromis soit l'utilisation d'un système d'équations algébriques généralisation des grammaires à contexte libre comme celui que nous avons introduit au chapitre 2 : il est simple à utiliser et si le choix des opérations d'assemblages est judicieux il a une bonne puissance descriptive ; de plus, il est suffisamment formel pour se prêter à une étude systématique du type de celle que nous avons menée au chapitre 3. Ainsi , nous limiterons par la suite nos réflexions à un modèle s'inscrivant dans ce cadre.

2.2.- CAS D'UN SYSTÈME ALGÉBRIQUE.

Dans ce cas une description sera opérée par un système d'équations à point fixe, chaque équation décrivant toutes les formes possibles d'une classe (syntaxique) donnée ; le chapitre 2 en a donné un bon exemple. Nous allons discuter ici des deux choix fondamentaux qui se posent : celui des formes primitives et celui des règles d'assemblages.

2.2.1.- LES FORMES PRIMITIVES.

De façon évidente, leur choix dépend du problème à traiter. Les règles suivantes permettent cependant de guider ce choix :

a) Pour l'algorithme d'analyse chaque forme primitive est considérée comme un objet sans considération sur ses parties. Ceci impose que la définition des primitives intègre les renseignements particuliers que l'on voudra pouvoir utiliser par la suite comme par exemple les coordonnées extrêmes (cf. chapitre 2 ou ESP^3), l'intensité lumineuse moyenne .... Une forme , primitive ou non, ne sera donc pas seulement un ensemble de pointsmais on y attachera aussi un ensemble comprenant les mesures particulières que l'on voudra utiliser. Ces paramètres des formes sont souvent encore appelés attributs. Bajcsy et Liberman [BAJ 74] fournissent un bon exemple d'une telle paramétrisation pour la description de leur univers (scène champêtre) : tous les paramètres nécessaires à la distinction de leurs formes élémentaires ont été recensés ; ainsi pour la forme "zone d'herbe" on trouvera :

. couleur : vert - jaune - marron
. frontière : contraste flou
ligne de frontière de forme quelconque
. forme : arbitraire
. position : fait partie du sol
. taille : arbitraire
. texture : couleur : vert ou jaune
forme : allongée
taille : largeur et longueur
relation spatiale : dense et approximativement parallèle .

b) La reconnaissance d'une forme primitive relève d'un *niveau d'identification* différent de celui, structurel, qui procède à la reconnaissance de la scène complète ; c'est en général la mise en oeuvre d'un ensemble de procédures d'identificationsconnues qui permettra la localisation de la primitive

recherchée. L'identification peut faire elle-même appel à des techniques structurelles.

Ainsi, une surface ayant une texture déterminée peut être mise en évidence par un algorithme de regroupement de région ; une ligne sera localisée par un algorithme de suivi de contour qui travaillera sur le gradient de l'image ou sur l'image elle-même selon que cette ligne est une frontière ou un "fil" ; cette même ligne pourra être divisée en segments élémentaires, arcs de cercles, segments de droite ; ... .
On peut ainsi avoir plusieurs niveaux ; prenons le travail effectué dans notre équipe par Belaid [BEL 79] : un premier niveau procédural segmente un ensemble de tracés en arcs de cercle et segments de droite ; un second niveau (structurel) regroupe ces segments pour identifier des symboles (caractères, symboles mathématiques,...) un troisième niveau (structurel) vérifie l'arrangement de ces symboles afin d'analyser une formule mathématique.

Le modèle de treillis de [BAJ 77] que nous avons adapté auparavant est d'un usage particulièrement souple pour la définition des primitives. Il permet de détecter facilement une primitive avec un minimum de propriétés la caractérisant et cette structure se prête facilement à l'adjonction de nouvelles primitives caractérisées par l'ensemble des propriétés qu'elle doit vérifier.

c) Les primitives doivent être *adaptées* au type de problèmes traités. Segmentsde droite et arcs de cercle sont un choix plausible si nous nous intéressons aux dessins ; dans le cas des images les surfaces et les lignes paraissent mieux adaptées. Un bon contre exemple est l'usage des grammaires de mosaïques(cf. §.1.2.1.) : la primitive est le point élémentaire de l'image (pixel) et à partir de composant atomique, on arrive péniblement à décrire un triangle.

Ces trois aspects ne sont pas indépendants. Une bonne paramétrisation des primitives (a) doit permettre une mise en oeuvre automatique des procédures d'identifications (b), celles-ci pouvant être choisies dans un répertoire en tenant compte de critères de performances, des valeurs déjà connues des paramètres, de préconditions (voir par exemple Ballard et al. [BAL 77] , cf. §.1.3.). Par ailleurs le choix de primitives adaptées à la description (c) implique le choix du paramétrage et des procédures d'identification : une trop grande finesse des primitives conduit à des procédures d'identification complexe et spécialisée et on perd ainsi l'intérêt apporté par l'approche structurelle : simplicité et généralité.

2.2.2.- L'ASSEMBLAGE DES FORMES.

Une forme complexe est une combinaison de formes élémentaires. Les opérations d'assemblage définissent la façon de réaliser ces combinaisons. Elles doivent être adaptées à l'assemblage logique que l'on veut introduire dans la description. Si la concaténation "bout à bout" (telle que nous l'avons décrite au chapitre 2) peut convenir à des dessins plans, elle ne convient plus à l'univers tridimentionnel des blocs.

En effet, logiquement un bloc est un assemblage de facettes et cet assemblage se fait par mise en commun d'une arête et non plus par mise en commun d'une extrémité. Ceci explique la remarque de Stanton au sujet des dessins de la figure 4.17. : "The figures are three-dimensionnally similar but two dimensionnally dissimilar, *particulary* with respect to the concatenation of primitives". [STA 72] .

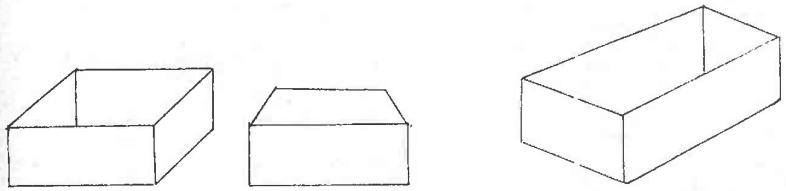


figure 4.17.

Si l'on considère cette boîte comme un objet tridimentionnel composé de faces , elles-mêmes décrites par des arêtes , on obtient la description suivante :

objet : face R face R face face R face R face

où R désigne la mise en commun d'une arête

et face : oblique S verticale // verticale S oblique

où S désigne la concaténation de segments
// désigne la relation parallèle

Une telle description ne tient évidemment pas compte de faces cachées et ne serait correcte que pour un objet transparent. Pour prendre en compte ce point, il faut introduire l'arête interrompue ; une arête interrompue est une arête ayant une concaténation du type suivant avec l'arête l'interrompant : ⊥ ;

notons I la relation de ce type entre deux arêtes.

On peut alors distinguer trois types de faces interrompues :

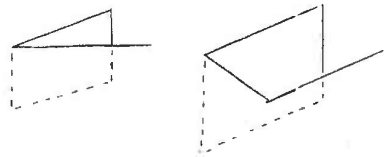
type 1 = oblique S verticale ; oblique S ligne I

ligne = oblique U oblique S oblique

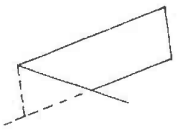
type 2 = oblique S verticale S oblique // oblique S oblique I

type 3 = verticale I oblique I verticale ; verticale S oblique S

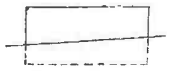
face de type 1 :



face de type 2 :



face de type 3 :



Cet exemple illustre bien la nécessité de l'adéquation des opérations d'assemblage à la "structure logique" de la forme.

Nous diviserons les opérations de concaténation en deux familles :

concaténation par coïncidences :

Il s'agit d'un assemblage précis obtenu par la superposition de parties des formes considérées. Les concaténations par coïncidences de points (cf. §.1.2.2. et chapitre 2) entrent dans cette catégorie ainsi que les assemblages par mise en commun de morceaux de frontières d'une surface, la mise en commun d'une arête (dans l'exemple précédent) . Une grande variété d'applications est couverte par de telles concaténations : on peut imaginer par exemple, qu'un ensemble mécanique est divisé en deux sous-ensembles avec mise en commun de parties : rotule de transmission et pattes de fixation. Cependant, si la coïncidence est d'un usage agréable pour la reconnaissance, elle est trop contraignante pour certaines descriptions, ce qui nécessite une seconde famille :

concaténation par relations :

moins précises que les précédentes, elles sont définies par le *prédicat* que doivent vérifier les formes en relation. La relation SUR du chapitre 2 en est un exemple.

Si les premières peuvent se ramener aux secondes, on aura intérêt à les distinguer cependant : la généralité des relations définies par un prédicat n'autorise pas une statégie d'analyse aussi fine que celle que l'on peut utiliser lors des concaténations précises : l'algorithmes du chapitre 2 distingue soigneusement les deux cas. Rien n'interdit d'ailleurs de distinguer des sous-cas pour lesquels des stratégies de recherche différentes seraient appliquées.

Nous détaillerons §.3.3.1. , ces différentes concaténations et nous les illustrerons par des exemples.

2.2.3.- Le système de description.

Par le choix même du modèle que nous avons fait au §.2.2., la description va consister à indiquer les décompositions possibles de chaque forme en sous-formes et l'agencement de ces sous-formes. Une telle description peut être utilisée dans un but *génératif* ou de *reconnaissance*. Nous avons vu au chapitre 2 (§.1.3.3.) que l'utilisation d'opérations d'assemblage puissantes peut conduire à des formes indéfinies lorsqu'on les utilise pour la génération ; par contre, leur utilisation ne pose aucun problème lors de l'analyse ; bien mieux, la multiplicité des contraintes qu'elles imposent peut limiter les choix possibles. Pour cette raison , nous abandonnons l'aspect génératif des systèmes de description pour n'en retenir que l'utilisation en reconnaissance.

Le modèle que nous allons définir est voisin de celui proposé par Milgran et Rosenfeld [MIL 72] quant à la définition générale, mais s'en distingue par la mise en oeuvre et l'esprit.

Définitions :

On se donne un *espace* E *de formes* muni d'une loi U et un espace A de *"valeurs d'attributs"* , les paramètres des primitives en sont un cas particulier. Une forme est un couple (s, a), $s \in E$, $a \in A$; intuitivement, s correspond à la partie "visible" de la forme et a correspond à un certain nombre de valeurs qualifiant la forme ; en particulier, dans le cas des grammaires à coordonnées, a est la valeur des coordonnées de la forme.

On se donne pour tout $n > 0$ un ensemble $\mathfrak{R}_n$ de relations n-aires sur $E \times A$ et un ensemble $\mathfrak{F}_n$ d'applications de A^n dans A et nous noterons $\mathfrak{R} = \bigcup_{n>0} \mathfrak{R}_n$, $\mathfrak{F} = \bigcup_{n>0} \mathfrak{F}_n$.

Intuitivement, les relations servent à décrire les formes et les applications permettent de construire la valeur d'attribut de la forme construite à partir des valeurs des formes la constituant il ne s'agit donc ici que d'attributs synthétisés au sens de Knuth et Lohro

Nous noterons ω une forme dite "indéfinie". A partir des relations R de $\mathfrak{R}_n$ et des applications φ de $\mathfrak{F}_n$, nous allons construire des applications de

$$(E \times A)^n \longrightarrow (E \times A) \cup \{\omega\} \qquad \text{en posant}$$

$$(R, \varphi)\ ((s_1, a_1), \ldots, (s_n, a_n)) =$$

$$\underline{\text{si}}\ \ R((s_1, a_1), \ldots, (s_n, a_n))\ \ \underline{\text{alors}}\ \ \left(\bigcup_{i=1}^{n} s_i\ ,\ \varphi(a_1, \ldots, a_n)\right)$$

$$\underline{\text{sinon}}\ \ \omega$$

Nous noterons Ω l'ensemble des applications ainsi obtenues et nous prolongerons naturellement ces applications sur $((E \times A) \cup \{\omega\})^n$ en imposant que si l'un des arguments est indéfini $(= \omega)$, alors le résultat est indéfini.

Dans la suite, nous prolongerons toute fonction $f \in \Omega$ à l'ensemble des parties de $((E \times A) \cup \{\omega\})^n$ en posant comme habituellement

$$\varphi(X_1, \ldots, x_n) = \{\, y \mid \exists\, (x_1, \ldots, x_n) \in X_1 \times \ldots \times X_n,\ f(x_1, \ldots, x_n) = y \,\}$$

Nous appelons système algébrique tout système d'équations où chaque équation est de la forme suivante, où N est l'ensemble fini des inconnues

$$A = \bigcup_{j=1}^{P} \omega_j(A_j^1, \ldots, A_j^{a_j}) \quad \text{pour chaque inconnue } A \in N$$

où

soit $a_j \geq 1$ et $\omega_j \in \Omega$ et ω_j est d'arité a_j, $A_j^i \in N$,

soit $a_j = 0$ et ω_j désigne un sous-ensemble de $E \times A$ (ω_j est une primitive).

Un système (S) d'inconnues $N = (A_1, \ldots, A_n)$ admet comme *solution* un n-uple

$$(X_1, \ldots, X_n) \in [\mathcal{P}(E \times A \cup \{\omega\})]^n \quad \text{si pour tout } i \text{ de } 1 \text{ à } n$$

$$X_i = \bigcup_{j=1}^{P_i} \omega_{ij}(X_{ij}^1, \ldots, X_{ij}^{a_{ij}})$$

(S) admet toujours une *solution minimale* sur le treillis $[\mathcal{P}(E \times A \cup \{\omega\})]^n$ (cf. chapitre 1 §.3.1.) et nous noterons $S(A_1)$ la composante de la solution minimale associée à l'inconnue A_i et $L(S) = S(A_1)$ la première composante de cette solution : c'est le "langage" défini par le système.

Beaucoup d'auteurs préfèrent à la notion de système d'équations, celle de règles de réécriture. Il y a un parallèlisme complet entre les notations :

l'équation $A = \omega(A, B) \cup \varphi(C, D, E)$
se traduira par les règles

$$A \longrightarrow \omega(A, B)$$
$$A \longrightarrow \varphi(C, D, E)$$

Le terme "inconnue A" est alors remplacé par "non-terminal A" et souvent "primitive" devient "terminal" par analogie avec la terminologie de la théorie des langages.

Comme nous l'avions déjà signalé au chapitre 2, les deux présentations sont rigoureusement équivalentes, mais on préfère généralement la notion de système d'équations lors d'une présentation générale, alors que l'on

préfère parler de règles (de réécritures) lorsqu'on considère individuellement chacune de ces règles (en particulier lors de l'analyse syntaxique).

Exemple :

Exemple 1 :

Le modèle introduit au chapitre 2 entre dans ce formalisme. En effet, prenons :

$E = \mathfrak{P}(\mathbb{R}^2)$ muni de l'union

$A = \mathbb{R}^2 \times \mathbb{R}^2$.

Une forme sera du type $(s, (M, N))$ où $s \subset \mathbb{R}^2$ est la partie visible de la forme, $M, N \in \mathbb{R}^2$ les points origine et extrêmité. Chaque forme est donc caractérisée par une partie du plan et ses attributs sont en fait les coordonnées de deux points.

$\mathfrak{R} = \mathfrak{B}_2$: toutes les relations sont binaires ; elles sont :

O : O((s, (M, N)) , (s', (M', N'))) ⟺ M = M' (coïncidence des "origines")

E : E((s, (M, N)), (s', M', N'))) ⟺ N = N' (coïncidence des "extrêmités")

S : S((s, M, N)), (s', M', N'))) ⟺ N = M' (coïncidence de l'"origine" du second avec l'"extrêmité du premier)

D : D = O ∩ E (coïncidence des "origines" et des "extrêmités")

I : I((s, (M, N)), (s', (M', N'))) ⟺ s est une ligne polygonale et s' est intérieur à s .

Nous avons enfin deux fonctions d'arité 2 :

f((M, N), (M', N')) = (M, N')

g((M, N), (M', N')) = (M, N)

On remarquera la simplicité de ces fonctions définies sur les valeurs attributs et qui ne jouent qu'un rôle accessoire.

Soit alors les cinq applications suivantes (définies avec les notations précédentes)

+ = (S, f)

o = (O, f)

x = (E, f)

$\underline{f}$ = (D, f) = (D, g)

int = (I, g)

On remarquera que +, x, o et $\underline{f}$ sont les opérations introduites au chapitre 2.

Soit alors le système :

(S) A = int(B, E1) U int (B, E2)

B = $\underline{f}$ (+(U, -h) , +(c, h))

C = + (v, + (h, -v))

U = + (-v, + (h, v))

E1 = $\underline{f}$ (+ (h, v), + (v, h))

E2 = int (E1, A)

h désigne l'ensemble des formes "segment horizontal" défini formellement par la classe obtenue par translations à partir des (s, (M, N)) où s est un segment $\{(\lambda, 0)/0 < \lambda < x\}$ x > 0 et N = (0, 0) et M = (x, 0) (figure 4.18). De même v désigne la forme "segment vertical".

L(S) est alors l'ensemble des "bouteilles étiquetées, l'étiquette pouvant représenter une bouteille". (figure 4.19).

Notons que la définition de "int" est différente de celle du chapitre 2 car nous ne disposons pas ici des valeurs d'attributs que sont les coordonnées extrêmes ; ces valeurs sont cependant faciles à introduire. Cette dernière définition de "int" a été écartée pour MIRABELLE car le calcul de l'intérieur d'une ligne polygonale était trop lourd.

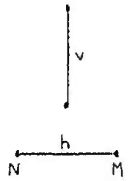


figure 4.18

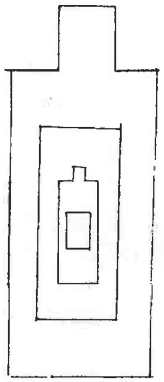


figure 4.19

Exemple 2 : (cas d'image).

Nous supposons disposer de fonctions capables de :

- rechercher des segments de droite ayant une direction donnée
- analyser la texture, la brillance
- fusionner des régions adjacentes ayant mêmes propriétés.

Il est assez naturel de considérer alors comme forme primitive des segments ayant une direction donnée, des surfaces homogènes ayant une texture donnée.

L'espace des formes E est alors l'espace habituel des fonctions à deux variables du plan : $(x, y) \longrightarrow f(x, y)$ et celui A des valeurs d'attributs est $(\mathbb{R}^+ \times \mathbb{R}^+)$, le premier attribut étant la valeur de la brillance moyenne, le second la mesure pour une surface ou la direction pour un segment. Avec de tels attributs une relation pourra porter sur : des tailles de surfaces, des directions, des niveaux relatifs de brillance. On suppose de plus avoir accès à tout le support de la forme pour pouvoir définir des relations "topographiques" telles que "est entouré de".

Une description peut alors être :

scène ⟶ fond ENTOURANT objet

fond ⟶ primitive (surface s , texture t , brillance b)

objet ⟶ cadre CONTENANT suite de barreaux

cadre ⟶ sh C sv C sh C sv C sh

sh ⟶ primitive (segment, direction 0, brillance b') (segment horizontal)

sv ⟶ primitive (segment, direction 90, brillance b') (segment vertical)

suite de barreaux ⟶ sv

⟶ sv AGAUCHE suite de barreaux

C est une concaténation du type "bout à bout".

La définition d'une relation telle que ENTOURANT peut préciser que la brillance du fond est supérieure à celle de l'objet et que la position du second "opérande" est contenue dans la surface englobée dans les frontières du premier opérande.

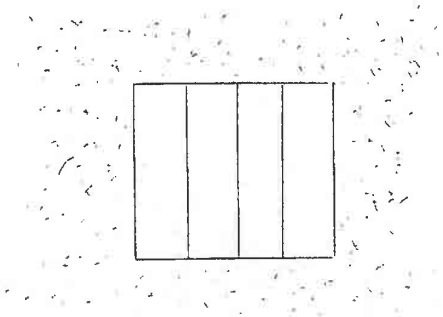


figure 4.20.

3.- ANALYSE DANS UN MODÈLE STRUCTUREL.

3.1.- QUELQUES_MÉTHODES.

La plupart des articles décrivant l'utilisation des méthodes structurelles sont avares de détails concernant l'intéraction entre l'analyse et la structure décrivant l'univers. Une part importante des autres articles détaillent des méthodes directement issues de l'analyse syntaxique et qui balayent de gauche à droite la description. Quelques travaux présentent cependant des méthodes plus adaptées aux formes bidimensionnelles.

3.1.1.- MÉTHODES_DIRECTEMENT_ISSUES_DE_L'ANALYSE SYNTAXIQUE.

Les techniques d'analyse syntaxique s'appliquent directement aux formes transformées en chaîne linéaire par un prétraitement. C'est le cas de signaux linéaires tels que électroencéphalogrammes ou électrocardiogrammes ([HOR 75]) ; la mise en oeuvre des analyseurs ou des automates d'états finis ne pose alors aucun problème. Un contour de formes peut aussi être codé par des éléments de lignes élémentaires mais il se pose alors le problème de la détermination du point de départ ; Leydley [LEY 64] l'a résolu en proposant une grammaire générant toutes les configurations de contour définies à une permutation circulaire près.

Shaw [SHA 69] fut le premier à détailler comment une analyse syntaxique descendante (descente récursive) pouvait opérer lorsqu'on abandonnait les chaînes pour aborder des objets réellement multidimensionnels : il propose une analyse descendante classique balayant la description de gauche à droite du même type que l'algorithme ADROITE que nous avons donné au chapitre 2. Les inconvénients de l'algorithme qu'il avait proposé étaient d'analyser obligatoirement en partant du "début de la forme" et d'autre part, de faire tous les choix possibles avec retour arrière sans tester au préalable si les choix peuvent aboutir.

Anderson [AND 77] signale que l'utilisation brutale de la descente récursive conduit dans un cas réel à un temps d'exécution trop long. Il introduit en conséquence des tests sur les initiales (cf. chapitre 2 §.2.) pour limiter les choix lors de l'analyse ; de plus, il utilise une heuristique prenant en compte le fait que, lors de l'écriture des formules mathématiques, l'écrivain écrit généralement de la gauche vers la droite.

Moayer et Fu (cf. §.1.2.2.) utilisent aussi une méthode descendante du même type pour analyser les mosaïques décrivant les empreintes digitales : on procède par analyse successive de chaque quart de la sous-image. L'algorithme stocke de plus les reconnaissances partielles réussies de sous-images qui peuvent apparaître dans d'autres règles, ce qui évite de reprendre leur analyse après retour arrière.

Ces méthodes et leurs améliorations contrôlent complètement le processus de reconnaissance ; de plus, comme elles procèdent toujours de haut en bas, elles ne permettent pas une prise en compte astucieuse de ce qui a pû être reconnu.

3.1.2.- AUTRES_STRATÉGIES.

La représentation des formes par un graphe exprimant d'une part la décomposition de la forme en sous-formes, d'autre part, les différentes relations entre ces sous-formes conduit naturellement à introduire une nouvelle stratégie d'analyse : à partir d'une sous-forme reconnue le graphe indique quelles sont les formes qui peuvent être en relation avec la forme identifiée : l'analyse peut ainsi débuter en n'importe quel point de la forme.

Les travaux de Barrow et Popplestone [BAR 71] séparent complètement les deux étapes : construction du graphe associé à la forme puis vérification de la concordance de ce graphe avec le modèle. Cette approche est bien sûr trop simpliste et ne permet pas d'intéraction entre le modèle et la forme. Cette intéraction a été mise en oeuvre dans les travaux déjà cités au §.1.3.3. et §.1.4. [WIL 77] , [HER 77] .

Comme nous l'avons déjà signalé , William remarque ainsi que HPL reconnaît une maison composée d'un triangle sur un rectangle dans un temps à peine plus long que la reconnaissance d'un triangle et bien inférieur au temps total de la reconnaissance d'un triangle et d'un rectangle séparés. Ceci n'est pas surprenant puisque l'information contenue par la reconnaissance d'une forme guide la reconnaissance de l'autre. Un raffinement supplémentaire dans cette stratégie aurait été la limitation de la recherche des formes en relation avec une forme reconnue en testant des caractéristiques du contexte de la forme reconnue .

La même approche peut être étendue à des descriptions faites sous forme de règles : c'est cette stratégie que nous avons utilisée dans MIRABELLE et c'est aussi de cette façon que procède Gallo ([VAM 77] ,[GAL 75] l'analyse est une analyse ascendante ; l'organisation est cependant complexe : la stratégie d'analyse est déterminée par l'ordre des règles et des primitives caractéristiques servant à guider l'analyseur ; les primitives caractéristiques sont déterminées pendant une période d'apprentissage durant laquelle on estime la probabilité P (p|O) pour qu'il y ait un objet O sachant qu'on observe une primitive p .

Malheureusement, les références indiquées ne donnent que peu de détails sur les intéractions des différents sous-programmes utilisés lors de l'analyse.

Cette stratégie permet en outre de choisir les algorithmes de reconnaissances adaptés ([BAL 77] , [FRE 77]) : ainsi un algorithme de recherche global et approximatif , recherche les primitives les plus évidentes,

puis après validation des primitives correctes, l'analyse démarre de ces points pour rechercher les autres ; Freuder souligne par exemple que l'algorithme de recherche du manche de son marteau est différent selon que l'on recherche un manche après avoir localisé la tête ou que l'on cherche à valider un manche à partir de zones qui pourraient le représenter.

3.2.- DÉCIDABILITÉ DE LA RECONNAISSANCE DANS LE MODÈLE PROPOSÉ.

3.2.1.- DÉCIDABILITÉ.

Nous allons montrer que sous certaines hypothèses le problème de la reconnaissance est décidable dans notre modèle. Cette démonstration est nécessaire pour deux raisons :

- Milgran [MIL 72] a montré que les grammaires de coordonnées avaient la puissance des machines de Turing et que par conséquent le problème de la reconnaissance est indécidable. Notre modèle englobe les grammaires de coordonnées et nous ne savons pas laquelle de nos hypothèses met en défaut la démonstration de Milgran ; l'article est en effet très succint et ne donne que la ligne directrice de la démonstration.

- la discussion que nous aurons pour justifier chacune de nos hypothèses mettra en valeur les dangers et les principes du modèle.

Nos hypothèses sont :

H1- Il est possible de décomposer toute forme en ses formes primitives qui la constituent et cette décomposition est unique (hypothèses de décomposition).

H2- Chaque occurrence de forme élémentaire n'intervient qu'une fois et une seule dans la construction d'une forme plus complexe.

H3- Pour chaque n-uple $(f_1, \ldots, f_n)$ de représentations de formes et pour chaque prédicat $R \in \mathfrak{B}_n$, il est possible de décider si $R(f_1, \ldots, f_n)$ est vérifié ou non.

H4- Chaque terme de l'équation définissant le système (c.a.d. chaque second membre de règle) est soit de la forme a où a est une primitive, soit de la forme $\omega(A_1, \ldots, A_n)$ où $n \geqslant 2$.

Sous ces hypothèses le problème de la reconnaissance est décidable et nous donnons l'algorithme effectuant la reconnaissance ; cet algorithme est un algorithme d'analyse ascendante globale.

Algorithme

% initialisation %

$m \leftarrow 1$;

$E_1 = \{ (f, a, \{f\}) \mid f$ est une représentation de la primitive a apparaissant dans la forme F $\}$ % hypothèse H1 % .

% le triplet $(r, A, \{p_1, \ldots, p_q\})$ est lié dans ce programme par la propriété :
la représentation r appartient à la classe de forme A et $p_1, \ldots, p_q$ sont les représentations des primitives ayant une occurrence dans r %

p = le nombre de primitives apparaissant dans F

% itération %

tant que $p > m$ faire

% propriété invariante : toute représentation f composée de k primitives $(k \leqslant m)$ et appartenant à la classe A décrite par le système vérifie $(f, A, Q) \in E_m$, Q étant l'ensemble des représentations de primitives constituant r %

$E_{m+1} \leftarrow E_m$;

pour chaque équation $(i = 1, \ldots, n)$ faire

pour chaque terme $\omega(B_1, \ldots, B_k)$ de la i équation faire

pour chaque k-uple $((f_1, C_1, P_1), \ldots, (f_k, C_k, P_k))$ d'éléments de E_m faire

% hypothèse H2 % si $C_j = B_j (j = 1, \ldots, k)$ et $\bigcup_{j \neq \ell} (P_j \cap P_\ell) \neq \emptyset$

% hypothèse H3 % et $\omega(f_1, \ldots, f_k) = g * \omega$

alors $E_{m+1} \leftarrow E_m \cup \{(g, A, \bigcup_{j=1}^{k} P_j)\}$ fsi

fait fait fait

$m \leftarrow m + 1$

fait

On démontre alors le résultat suivant :

Théorème :

Si F appartient à l'ensemble des formes décrit par le système, alors

$(F, A_1, \{f \mid f$ est une représentation de primitive de F $\}) \in E_p$

Démonstration :

En utilisant l'hypothèse H4 le lecteur familier avec la méthode d'analyse syntaxique ascendante globale se convainc facilement du résultat. La démonstration est donnée dans l'annexe 4.

3.2.2.- Discution des hypothèses.

Hypothèse H1. :

Cette hypothèse de segmentation est une des bases de l'approche structurelle. Si plusieurs segmentations sont possibles, la décidabilité reste acquise : il suffit de considérer toutes les possibilités ; mais, s'il n'est plus possible de distinguer toutes les primitives alors le problème de la reconnaissance devient indécidable ([MOH 73]).

Le problème posé par l'absence ou la mauvaise identification des primitives est de même nature. Si les différentes possibilités d'adjonction et de correction sont en nombre fini, le problème est résolu sur le plan théorique. De façon pratique il faudra éviter cette explosion combinatoire et nous aborderons ce point §.3.3.

Hypothèse H2. :

L'hypothèse H2 peut être remplacée par "chaque forme élémentaire n'intervient qu'au plus k fois, ..." k étant une constante fixée. Dans ce cas le test d'arrêt de l'itération de l'algorithme précédent donné en 1.1. sera $pk > m$.

Cette généralisation est intéressante en particulier lorsqu'une même sous-forme appartient à la frontière de deux formes ; elle devra donc figurer deux fois dans la description de la forme globale. Il semble que le cas $k > 2$ soit très rare dans les exemples concrets.

L'absence de cette hypothèse pose de façon cruciale le problème de l'arrêt ; elle rend probablement le problème de la reconnaissance indécidable.
Sur un exemple considérons ce qui peut se passer dans le cas contraire.

Considérons le système :

$$A = a + b + - a + - b + A \cup -b$$

interprété dans le modèle que nous avons défini au chapitre 2; a, b sont des segments de longueurs unité décrit à la figure 4.21.

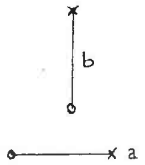


figure 4.21.

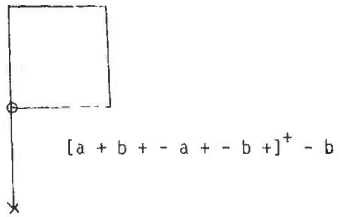


figure 4.22.

La forme représentée à la figure 4.22 est une forme décrite par le système : une de ses décompositions possibles est $[a + b + - a + - b]+ - b$ mais si l'on accepte de "faire n fois le tour du carré", une décomposition possible de cette forme est $[a + b + - a + - b+]^n - b$; un algorithme d'analyse syntaxique analysant cette forme pour le système précédent ne sera donc pas borné, ayant à fournir un résultat infini. On remarquera que l'ambiguité est due ici à l'interprétation des formules de ce système, et n'est pas intrinsèque au système.

Hypothèse H3. :

Cette hypothèse ne fait que spécifier que les concaténations que nous choisissons sont calculables.

Hypothèse H4. :

Cette hypothèse n'est pas fondamentale et dans la pratique il est souvent utile d'avoir des opérateurs unaires jouant un rôle de sélection sur la forme donnée : supposons que R désigne la classe des rectangles et que chaque forme rectangle possède deux attributs : longueur et largeur ; on peut alors définir la classe RA des rectangles allongés en posant

$$RA = (\mathcal{R}, id)\ (R)$$

où id la fonction identité

$\mathcal{R}$ est le prédicat portant sur les attributs de longueur et largeur et qui est vrai lorsque la longueur est supérieure à deux fois la largeur.

Examinons donc ce qui se passe si l'hypothèse H4 n'est plus vérifiée. Dans ce cas, nous avons des termes $A_i = \omega_i(A_j)$ dans le système ; notons $A_i \longrightarrow A_j$ ce fait.

Distinguons alors deux cas :

a) il y a un circuit dans le graphe de $\longrightarrow$; soit $A_1, \ldots, A_p, A_1$ un tel circuit ; donc nous avons pour les solutions $X_1, \ldots, X_p$

$$X_1 \supset \omega_1(X_2)$$

$$X_2 \supset \omega_2(X_3)$$

$$\ldots\ldots\ldots\ldots$$

$$X_p \supset \omega_p(X_1)$$

donc $X_1 \supset \omega_1(\omega_2(\ldots(\omega_p(X_1)\ldots)$.

Soit alors un $f \in X_1$ tel que $g = \omega_1(\omega_2 \ldots (\omega_p(f) \ldots)$ soit défini ; comme f et g ne différent que par la valeur de leurs attributs, cela

signifie qu'une même forme peut avoir deux interprétations donc que le système est ambigu. Or, on peut espérer que le système ne l'est pas.

b) Nous supposons donc qu'il n'existe pas de circuit dans la relation $\rightarrow$ dans ce cas, on se ramène à l'hypothèse H4 selon le processus de contraction habituel ; si, par exemple, le système est :

$$A = \omega(B) \cup \varphi(A, B)$$

$$B = \psi(C) \cup \{a\}$$

$$C = \gamma(B, A)$$

on obtient par contraction le système équivalent

$$A = \omega(a) \cup \omega(\psi(\gamma(B, A))) \cup \varphi(A, B)$$

$$B = \psi(\gamma(B, A)) \cup \{a\}$$

$$C = \gamma(B, A)$$

Il faut alors considérer $\{a\}$ et $\omega(a)$ comme les deux primitives du système.

Donc, si la grammaire initiale n'est pas ambiguë, on peut se ramener au cas de l'hypothèse H4.

3.3.- Analyse effective.

Nous allons ici montrer comment l'on peut généraliser le principe de l'algorithme décrit au chapitre 2. Nous verrons comment il peut s'adapter à un modèle plus général et comment on peut y intégrer les idées de [FRE 77] et [BAL 77] pour tenir compte de l'existence de différents algorithmes d'identifications de primitives et de leurs coûts respectifs. Le modèle décrit §.2. étant cependant trop général , nous allons le préciser un peu (et par conséquent le restreindre), afin de pouvoir distinguer les deux types de concate-

nations que nous avions déjà signalés : celles, précises, mettant en contact des formes, et les autres.

3.3.1.- Distinctions dans les concaténations.

Nous retiendrons deux types de concaténations par mise en contact de formes : concaténation par mise en coïncidence de pôles particuliers des formes et concaténation par "mise en commun" de sous-formes.

concaténation par coïncidence de pôles.

Elle a déjà été illustrée au chapitre 2 mais cette fois, pour plus de généralité, nous introduisons n pôles de concaténations comme dans l'exemple final donné dans [NAR 66] . Pour chaque forme f , $P_k(f)$ désignera le $k^{ième}$ pôle de cette forme.

Pour certaines formes f le $k^{ième}$ pôle pourra ne présenter aucun intérêt car n'étant jamais utilisé, dans ce cas $P_k(f)$ pourra être indéfini.

Les concaténations (R, φ) doivent préciser les pôles en coïncidence et les pôles retenus pour la forme résultante :

R sera donc du type : $\bigwedge_{i=1}^{k} P_{i_1}(f_{i_2}) = P_{i_3}(f_{i_4})$

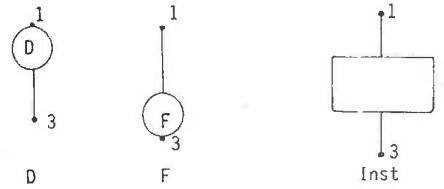
et φ précisera les n points retenus comme pôles de la forme résultante : ce sera donc un n-uplet $(P_{a_i}(f_{b_i}),\ i = 1, \ldots, n)$.

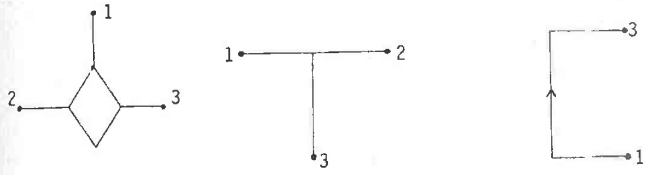
Exemple :

nous allons illustrer les possibilités offertes par la concaténation à plus de deux pôles sur un exemple qui pourrait aussi être décrit - mais de façon non naturelle - par le modèle du chapitre 2.

Les formes ont trois pôles, le 2ème étant parfois fictif ne sera pas représenté.

Formes primitives :





Test Joint Iter

Concaténations :

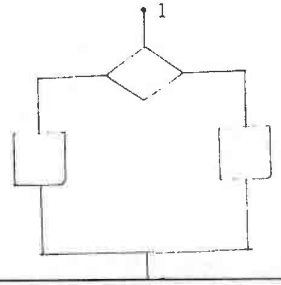
suite (f_1, f_2) : $((P_3(f_1) = P_1(f_2))$,

$(P_1(f_1)$, indéfini, $P_3(f_2)))$

cond (f_1, f_2, f_3, f_4) : $((P_2(f_1) = P_1(f_2) \wedge P_3(f_1) = P_1(f_3) \wedge P_3(f_2) = P_1(f_4)\ \ P_3(f_3) = P_2(f_4))$, $(P_1(f_1)$, indéfini, $P_3(f_4)))$

boucle (f_1, f_2, f_3, f_4) : $((P_3(f_1) = P_1(f_2) \wedge P_2(f_2) = P_1(f_3) \wedge P_3(f_3) = P_1(f_4) \wedge P_3(f_4) = P_1(f_1))$, $(P_1(f_1)$ indéfini $P_3(f_4)))$

par exemple : cond (Test, Inst, Inst, Joint) :



Système de description :

il est alors naturel de vouloir décrire des organigrammes, d'où le système suivant :

Programme = suite (D, suite (Séquence, F))

Séquence = Instruction U suite (Instruction, Séquence)

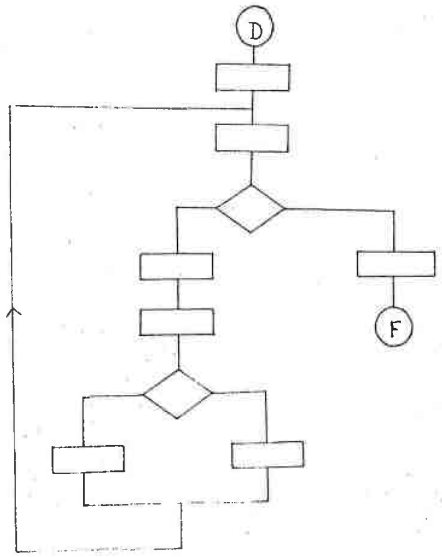
Instruction = Inst U Conditionnelle U Itération

Conditionnelle = cond (Test, Séquence, Séquence, Joint)

Itération = boucle (Séquence, Test, Séquence, Iter)

On remarquera que le choix des primitives et des concaténations nous permet, dans le système, de respecter la structure logique que nous souhaitons mettre sur ces organigrammes structurés.

Exemple de forme appartenant à Programme :



Concaténation par mise en commun de sous-formes :

A chaque forme sera associée m de ses sous-formes que nous appelerons *joints* ; comme pour les pôles, la concaténation devra préciser lesquels des joints seront retenus dans la forme résultante, certains d'entres eux pouvant ne pas être définis. $F_k(f)$ désignera le kième joint associé à f ;

un tel type de concaténation (R, φ) $(f_1, \ldots, f_p)$ précisera donc les joints mis en commun :

$$\bigwedge_{i=1}^{k} F_{i_1}(f_{i_2}) = F_{i_3}(f_{i_4})$$

φ doit bien sûr préciser le n-uples de pôles retenus, enfin donner le m-uples de joints retenus dans la forme résultante :

$$(\alpha_i,\ i = 1, \ldots, m)$$

avec α_i étant soit : un $F_k(f_j)$

: un f_j

: indéfini

NOTE :

pour que deux $F_i(f_j)$ puissent coïncider , il faut, bien sûr, que ces deux formes soient les mêmes. On imposera donc que ces joints correspondent à des formes d'une même classe ; ceci signifie que pour une classe A de formes du système son $i^{è}$ joint sera toujours, soit indéfini , soit une forme d'une classe B fixée. Cette contrainte statique peut être facilement vérifiée par le système qui prend en compte la définition.

Exemple :

nous prendrons n = 2 et m = 1

Les formes primitives seront les suivantes :

Assemblage : 

H : 

V : 

il est évident que Assemblage devrait être décrit à l'aide de H et de primitives obliques, mais nous ne l'avons pas fait pour simplifier la description.

Concaténations:

ajuste (f_1, f_2) : $(F_1(f_1) = F_1(f_2)$, coïncidence de joints
(indéfini, indéfini) , pôles retenus
indéfini joints retenus

$+ (f_1, f_2)$: $(P_2(f_1) = P_1(f_2)$ coïncidence de pôles
$(P_1(f_1)$, $P_2(f_2)$, pôles retenus
indéfini joints retenus

fermeture(f_1,f_2) : $(P_1(f_1) = P_1(f_2)$ $P_2(f_1) = P_2(f_2)$, coïncidence de pôle
$(P_1(f_1)$, $P_2(f_2))$, pôles retenus
f_2 joints retenus

$- (f_1)$: (vrai ,
$(P_2(f_1)$, $P_1(f_1)$,
indéfini)

système de description :

Pièce = ajuste (Haut, Bas) /1/

Haut = fermeture (+ (+(V, H) , -(V)) , Mortaise) /2/

Mortaise = Assemblage /3/ ∪ + (Assemblage, Mortaise) /4/

Bas = fermeture (+(+(- (V), H), V), Mortaise) /5/

exemple de forme décrite : 

On remarquera que cette forme peut aussi être facilement décrite par le modèle du chapitre 2 mais si l'on considère que le dessin représente l'assemblage logique de deux pièces usinées, le modèle du chapitre 2 ne permet pas de prendre en compte logiquement cet assemblage ; alors que ici ce point est rendu naturellement par l'opération "ajuste" .

autres concaténations :

Nous les restreindrons en les supposant seulement binaires. De plus, comme au chapitre 3, nous supposerons que pour une telle concaténation conc (f, g) , il sera possible de déterminer la zone où doit figurer g connaissant f (et réciproquement) mais aussi, connaissant une sous-forme f' de f il soit possible de déterminer la sous-zone dans laquelle se trouvera toute sous-forme g' de g (et réciproquement).

3.3.2.- ALGORITHME.

Un algorithme suffisamment souple doit être dirigé par les formes qu'il reconnaît et doit utiliser les connaissances tirées du modèles pour piloter la progression. Ceci exclut l'approche d'analyse descendante

proposée par les premiers travaux sur le sujet et qui est entièrement controlée par le modèle, ainsi que les travaux de Barrow et Poppelstone qui construisent la structure de la forme sans s'appuyer sur le modèle, ne vérifiant qu'après coup l'adéquation de la forme au modèle.

L'algorithme du chapitre 2 s'appuie sur un principe général que nous allons généraliser ici. L'analyse dans la procédure REMONTE s'appuyera sur les formes reconnues pour choisir les règles qui conviennent dans le modèle ; la procédure d'analyse DESCEND par contre, s'appuie sur le modèle pour rechercher des formes complètant une partie analysée. Il est nécessaire de supposer pour la correction de cet algorithme que le système de description *n'est pas ambigü* ; effet lors de la concaténation de deux formes A et B par la mise en commun d'un joint C , si la forme A est déjà analysée, l'analyse de la forme B se fera à partir du joint C déjà analysé : il faut donc que ce joint ait la même structure comme sous-forme de A et comme sous-forme de B .

procédure d'analyse ascendante : REMONTE.

Cette procédure analyse une forme But localisée dans une zone Z sachant qu'une sous-forme de But a déjà été reconnue et que sa structure est donnée par un arbre t ; le résultat sera :

l'arbre de la forme reconnue, ses pôles, ses joints.

```
procédure   REMONTE   (% zone d'analyse : % Z ,
                       % but de l'analyse : % But ,
                       % arbre de la sous-forme reconnue : % t ,
                       % ensemble des pôles associés à cette sous-forme : % P ,
                       % ensemble des joints associés à cette sous-forme : % J ,
                       % ensemble de sous-formes déjà analysés pouvant avoir
                         une occurrence commune avec la forme à analyser : % A )
```



```
résultat  (% il s'agit d'un triplet :
             arbre de la forme reconnue % arbre,
           % ensemble des pôles de la forme % ensemble de points,
           % ensemble des joints de la forme % ensemble de formes)

% toutes les variables sont locales à la procédure et leur type
  est implicitement défini par leur utilisation % .

soit B la racine de t ;
si B = But alors % le but est atteint % résultat : (t, P, J)
sinon choix d'une règle D ⟶ φ (B1,..., Bq) telle qu'il existe i tel que Bi = B

    cas . φ est une règle de concaténation par joints alors

                ti ← t
                Joints ← J ;
                Reste ← {B1, ..., Bi-1, Bi+1, ..., Bq} ;
                tant que Reste ≠ ∅ faire
                    choix de x ∈ Joints ;
                    choix de Bj ∈ Reste et devant coïncider
                     avec x dans la concaténation φ .
                    J' ← {y | y ∈ Joints et y coïncide avec Bj
                     dans φ } ; tx ← arbre syntaxique de x ;
                    (tj, Pj, Jj) ← REMONTE (Z, Bj, tx, pôles de x
                                            joints de x , A U J' ) ;
                    Reste ← Reste - Bj
                    Joints ← Joints U Jj ;
                fait
                si toutes les coïncidences de joints indiquées
                 par φ ne sont pas vérifiées alors erreur fsi ;
```

P' ← ensemble des pôles indiqués par φ ;
J' ← ensemble des joints indiqués par φ ;

. φ est une règle de concaténation par pôles alors

t_i ← t ;
Pôles ← P ;
Reste ← $\{B_1, \ldots, B_{i-1}, \ldots, B_q\}$
tant que $B_1, \ldots B_{i-1}, B_{i+1}, \ldots, B_q$
choix de x ∈ Pôles
choix de B_j ∈ Reste tel que un $P_k(B_j)$ coïncide avec le pôle x dans φ ;
(t_j, P_j, J_j) ← DESCEND (Z, B_j, x, k, A) ;
Reste ← Reste -$\{B_j\}$;
Pôles ← Pôles U P_j
fait
si toutes les coïncidences de pôles indiquées par φ ne sont pas vérifiées alors erreur fsi ;
P' ← ensemble de pôles indiqué par φ ;
J' ← ensemble de joints indiqué par φ

. φ est une concaténation par relation (R, φ) (B_1, B) alors

t_2 ← t ;
% dans ce cas q = 2 %
Soit Z' la sous-zone de Z dans laquelle doit être localisé B_1 pour être en relation R avec B ;
choix de y forme primitive ou forme de A, ayant une occurrence dans Z' ; soit t_y son arbre associé, P_y et J_y ses pôles et ses joints

(t_1, P_1, J_1) ← REMONTE (Z', B_1, t_y, P_y, J_y, A)
P' ← ensemble des pôles spécifié par f ;
J' ← ensemble des joints spécifié par f .

fin cas

résultat : REMONTE (Z, But, D x $(t_1 + \ldots + t_q)$, P', J', A) ;

fsi

fin procédure

La procédure d'analyse descendante du chapitre 2 se généralise de la même façon :

DESCEND recherche une forme D dans la zone Z à partir d'un point x qui correspond au $k^{ième}$ pôle de D .

procédure DESCEND (% zone de travail % Z ,
% nom de la sous-forme à analyser : % D ,
% pôle point de départ de l'analyse : % x ,
% numéro du pôle de D devant coïncider en x : % k ,
% sous-forme déjà analysée pouvant avoir une occurrence commune avec la forme à analyser : % A)

résultat (% il s'agit d'un triplet :
arbre de la forme reconnue % arbre ,
% ensemble de pôles de la forme % ensemble de points
% ensemble de joints de la forme % ensemble de formes).

choix entre

. s'il existe y une forme A appartenant à la classe syntaxique D et ayant son $k^{è}$ pôle en x ;
% la forme recherchée a déjà été analysée %
soit t_y son arbre syntaxique, P_y son ensemble de pôles, J_y son ensemble de joints ;
résultat (t_y, P_y, J_y)

. si D ∈ T alors % D est primitive %

si il existe une représentation y de D ayant son $k^{è}$ pôle en x

alors soit P_y ses pôles, J_y ses joints ;

résultat (y, P_y, J_y)

sinon erreur fsi

sinon choix d'une règle $D \rightarrow \varphi (B_1, \ldots, B_q)$;

cas . φ est une concaténation par pôles alors

Pôles ← x ;
Reste ← $\{B_1, \ldots, B_q\}$

tant que Reste ≠ ∅ faire

choix de y ∈ Pôles
choix de B_j ∈ Reste tel que un pôle $P_n(B_j)$
coïncide avec y dans φ ;
(t_j, P_j, J_j) ← DESCEND (Z, B_j, y, n, A) ;
Reste ← Reste -$\{B_j\}$
Pôles ← Pôles U P_j

fait ;
si toutes les coïncidences de pôles indiquées
par φ ne sont pas vérifiées alors erreur fsi ;
P ← ensemble des pôles indiqué par φ ;
J ← ensemble des joints indiqué par φ ;

. φ est une concaténation par joints alors

choix de B_j dont le pôle $P_n(B_j)$
doit coïncider avec x ;
(t_j, P_j, J_j) ← DESCEND (Z, B_j, x, n, A) ;

Joint ← J_j
Reste ← $\{B_1, \ldots, B_{j-1}, B_{j+1}, B_j \ldots, B_q\}$

(2)

tant que Reste ≠ ∅ faire

choix de x ∈ Joint
choix de B_i ∈ Reste tel que x doit
coïncider avec un joint de B_j dans φ ;
soit t_x, P_x, J_x l'arbre, les pôles et
les joints associés à x ;
(t_i, P_i, J_i) ← REMONTE(Z, B_j, t_x, P_x, J_x, A)
Joint ← Joint U J_i ;
Reste ← Reste - $\{B_i\}$;

fait

si toutes les coïncidences de joints indiquées
par φ ne sont pas vérifiées alors erreur fsi
P ← ensemble de pôles indiqué par φ ;
J ← ensemble de joints indiqué par φ

. φ est une concaténation habituelle (R, f) B_1, B_2) alors

soit B_i la forme ayant un pôle $P_n(B_i)$
en coïncidence avec x ;
(t_i, P_i, J_i) ← DESCEND (Z, B_i, x, n, A) ;
soit Z' la sous-zone de Z en relation R
avec B_i ;
choix de y primitive ou sous-forme de A
dans Z' ; soit t_y, P_y, J_y son arbre, ses
pôles, ses joints ;
$(t_{2-i}, P_{2-i}, J_{2-i})$ ← REMONTE(Z', B_{2-i}, t_y,P_y,J_y,A)
P ← ensemble des pôles exprimé par f
J ← ensemble des joints exprimé par f

fin cas ;

résultat : (D x (t_1 + ... + t_q) , P, J)

fsi fin du choix

fin procédure

Exemple d'analyse :

Reprenons le système de description de pièce donné §.3.3.1., et supposons que nous ayons à analyser la forme de la figure 4.24.

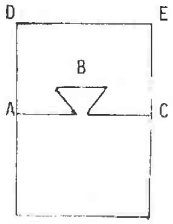


figure 4.24.

Supposons que la primitive Assemblage A - B - C soit choisie comme point de départ.
Un premier appel de
REMONTE (le plan, pièce, Assemblage,({A, C},{indéfini} , conduit à choisir entre les règles 3 et 4. L'absence d'Assemblage à l'extrémité C conduit à préférer la règle 4 ; on a ainsi identifié Mortaise ; un nouvel appel de
REMONTE (le Plan, pièce, Mortaise, {A,B},{indéfini} , ∅ nous amène à choisir entre les règles 2, 4, 5.

La règle 4 est exclue puisque cette mortaise ne se trouve pas à l'extrêmité d'un assemblage et les choix 2 et 5 sont indifférents. Choisissons alors la règle 2 par exemple. C'est une règle avec concaténation par pôles en A et C sur la mortaise déjà reconnue. Trois appels successifs de DESCEND vont alors identifier AD, CE et DE . Ainsi la forme Haut est analysée.

L'appel de REMONTE (le plan, scène, Haut, {A,C} , {Mortaise} , ∅) va alors choisir la seule règle possible : la règle 1 ; il s'agit d'une concaténation par joints et donc il se produit un nouvel appel à

REMONTE (le plan, Bas, Mortaise, {A,C} , {indéfini} , ∅)

qui va analyser Bas .

On remarquera que la forme Mortaise, commune aux deux sous-formes Haut et Bas, n'est analysée qu'une seule fois.

3.3.3.- RÉDUCTION DE L'INDÉTERMINISME.

En marge des algorithmes décrits dans le paragraphe précédent, nous avons fait apparaître les cinq principaux types de choix. Pour éviter l'explosion combinatoire due à une simple exécution avec retour arrière en cas d'échec , il faut pouvoir réduire au maximum ces choix par une bonne stratégie permettant d'une part d'éliminer des impasses prévisibles, d'autre part de deviner le choix qui le plus probablement conduira à un succès.
Examinons successivement ces choix :

choix (1) :

ayant une sous-forme B il faut choisir la façon de l'assembler avec des formes environnantes selon une des règles d'assemblages. Le choix d'une règle $D \longrightarrow \varphi (B_1, \ldots, B, \ldots, B_q)$ ne devrait être fait que si $B_1 \ldots B_q$ sont présentes et assemblées par la concaténation φ mais aussi si l'analyse pourra se poursuivre avec D .

Ce problème est exactement le même que celui que nous nous étions posé au chapitre 3 ; y répondre par la recherche effective de $B_1, \ldots, B_q$, et tester si l'analyse peut se poursuivre avec D revient à poursuivre complètement l'analyse et à la valider si elle est réussie. Une solution intermédiaire satisfaisante sera donc de tester de façon limitée un contexte environnant et de se limiter aux règles $D \rightarrow \varphi$ tel que D est une sous-forme du but à atteindre.

Si nous ne considérons que les concaténations par coïncidence de pôles le calcul du contexte (contexte immédiat) se généralise aisément par le calcul des ensembles de formes primitives pouvant apparaître au $k^{\text{è}}$ pôle de la forme reconnue. Les concaténations se faisant aussi par coïncidence de joints, il faut étendre cette notion de contexte immédiat aux formes présentes aux joints.

Un contexte limité tenant compte des joints pourrait être un contexte prenant en compte aussi les formes primitives présentes aux pôles des joints considérés.

Pour le contexte introduit par les relations topographiques on peut reprendre les définitions du chapitre 3 §.4. : le problème n'a pas changé de nature.

Le contexte présente donc deux aspects :
contexte topographique indiquant les primitives pouvant être en relation avec la forme considérée,
contexte de coïncidence indiquant les ensembles de primitives pouvant être présentes aux p pôles de la forme ainsi qu'aux p pôles de chacun des j joints .

Pour une description complexe et lorsque p et j deviennent grands ($p.j > 6$ par exemple) on va obtenir des ensembles de contextes possibles énormes :
ainsi pour $p = 3$, $j = 2$, si à chaque pôle une forme parmi deux possibilités est présente, comme $3 + 3 \times 2$ pôles sont considérés, cela nous fera 2^9 cas. Conserver tous les contextes possibles de façon exhaustive ne devient plus raisonnable ; un algorithme devra donc sélectionner que les seules informations permettant de lever le choix ; son résultat peut se présenter de la façon suivante pour chaque fous-forme A :

si présence de α en η alors règle r_1 ou r_2

si présence de β en η et γ en ρ alors règle r_3 ....

Une autre façon de présenter ces résultats est de les organiser en treillis au sens de Bajcsy et Thidar (cf §.2.1.) : ce treillis risque lui aussi d'avoir une taille combinatoire si l'on conserve toutes ses possibilités : mais sa structure même permet par un examen simple d'en éliminer les noeuds redondants pour ne retenir que ceux qui utilisent des propriétés discriminantes.

Choix 2 :

il s'agit de choisir l'ordre d'analyse lorsqu'une règle doit être utilisée. Ce choix se fait simplement en choisissant un B_j en coïncidence avec la partie déjà analysée, et x est selon le cas le joint ou le pôle de coïncidence.

Choix 3 :

Nous voulons ici choisir le point de départ de l'analyse d'une forme B_1 devant être en relation avec B ; ce choix peut être celui d'une forme caractéristique ou obligatoire au sens du chapitre 3 §.3. et tout ce qui a été développé à ce sujet tient encore ici.

Nous avons cependant un cas particulier : il est possible que la forme B_1 soit construite à partir de l'une des formes de l'ensemble A ; ce dernier cas peut se produire lorsque une forme B est concaténée par deux coïncidences de joints avec une forme C ; B étant analysée, l'analyse de C démarre à l'un des joints ; supposons alors que C se décompose en C_1 et C_2 réunis par une relation R et que chaque C_i contient un des joints ; lorsque DESCEND a analysé C_1 , l'analyse de C_2 peut reprendre avec le second joint qui est conservé dans l'ensemble A .

Choix 4 :

Ce choix se résoud de façon évidente : il est possible que la sous-forme D qu'il nous reste à analyser soit un joint commun à une autre forme et que pour cette raison elle soit déjà analysée ; il suffit donc de vérifier si nous sommes dans ce cas et sinon l'analyse reste à faire.

Choix 5 :

Etant donné un pôle x il nous faut choisir qu'elle est la décomposition de la forme D . Un test simple peut être fait en observant les primitives présentes au point x . Les initiales définies au chapitre 3 §.1., permettaient de connaître pour chaque règle choisie, les ensembles de primitives pouvant apparaître à chaque pôle. C'est donc cette notion qu'il faut généraliser ici. La multiplicité des pôles n'est pas un obstacle ici ; une simple précaution doit être prise lorsque sont utilisées des règles avec coïncidence de joints : lors de la coïncidence sur un joint j de B et C en un pôle n

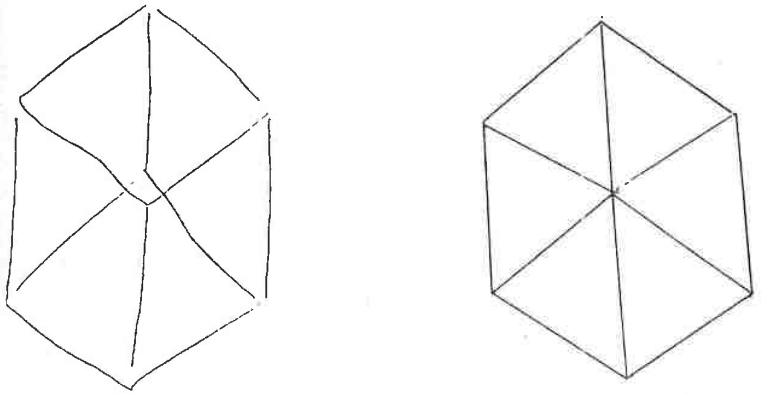
du joint apparaissent les primitives de ce joint occurrent en x réunies à celles de B et C apparaissant en x et n'appartenant pas au joint. En tenant compte de ce point le calcul des initiales se généralise de façon immédiate.

3.3.4.- Analyse d'une forme entachée de bruits.

Le monde réel ne présente que rarement des formes pures : les déformations et les bruits dûs aux capteurs, aux conditions d'acquisition à la digitalisation rendent la reconnaissance plus difficile, voire impossible.

Pour s'affranchir de ces problèmes une première voie consiste à effectuer un prétraitement : filtrage ou lissage d'images [CAS 77] , raccordement d'extrémités de segments .... Il s'agit d'opérations globales mais aveugles ; quand le modèle du bruit est connu et régulier (bruit blanc uniforme par exemple) ces techniques permettent une restauration suffisante pour permettre les traitements ultérieurs.

Ces traitements généraux ne sont pas toujours satisfaisants ; comme le fait remarquer Herot [HER 76] , son système fait parfois des corrections malheureuses en voulant raccorder des segments (figure 4.24) . Par ailleurs, pour des accidents relativement graves (occlusion d'une partie de la forme par exemple) ces techniques sont impuissantes alors que ces erreurs peuvent être corrigéespar l'être humain en tenant compte de partie non accidentée de la forme. On peut donc espérer que l'information contextuelle permettra à l'analyseur de se passer des parties trop déformées si celles-ci ne sont pas essentielles. Nous avions vu au chapitre 1 , un exemple de caractère ambigü : il pouvait être reconnu comme un A ou comme un R : le contexte du mot englobant ce caractère permettait facilement de trancher.



original corrigé

figure 4.24.

Avant de voir comment ces idées se généralisent dans l'analyse des formes structurées, distinguons les différents cas de détériorations locales :

a) destruction d'une sous-forme ; par exemple son occlusion

b) déformation d'une primitive ne permettant sa reconnaissance avec certitude (confusion avec d'autres primitives). C'était le cas des caractères A - R précédents

c) détérioration des caractéristiques d'une primitive ; par exemple obscursissement d'une tache ou segment de droite trop court

d) insertion de formes inutiles.

Lorsque dans le cas b) la primitive initialement reconnue n'est plus identifiable , quand par exemple un X est reconnu en H ou A , on considérera qu'on est en présence simultanément de d) et a) . Ainsi, si le caractère X est identifié comme étant A ou H on constatera une disparition de X et une apparition de A ou H .

Considérons alors comment peut opérer l'algorithme d'analyse :

. Lorsque l'on cherche une primitive de départ (choix 3) il faut tenir compte des risques d'erreurs d'identification et il faut donc choisir une primitive qui soit identifiée avec le maximum de certitude. Ce critère doit donc se concilier avec celui de nature purement syntaxique qui indiquait le type de choix de départ .

. Lorsque l'analyseur doit choisir la forme qu'il doit analyser de façon descendante (choix 2), on retiendra comme pôle au joint de départ celui réalisant au mieux la concaténation. Ainsi si dans une description de MIRABELLE nous avons A f B et que B est reconnue, l'analyse descendante peut, pour analyser A, choisir soit de partir de son extrémité, soit de partir de son origine puisque ici il y a double coïncidence ; si donc on n'observe pas de primitives correspondant à l'origine d'une forme A à l'origine de B on tentera une analyse à partir de son extrêmité.

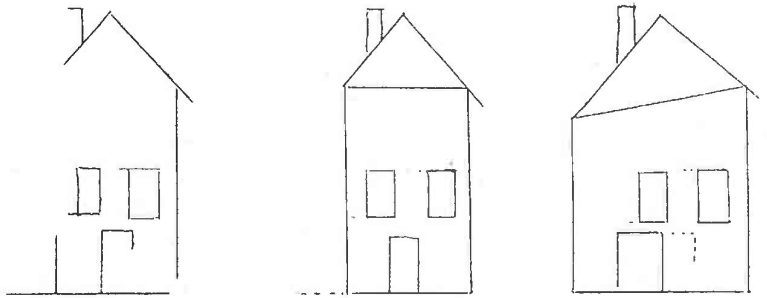
. Lorsque les formes primitives sont identifiées avec incertitude, l'analyse limitera l'ambiguité en ne considérant que les identifications compatibles avec la poursuite de l'analyse.

. Le modèle impose des conditions sur les attributs des sous-formes et peut ainsi permettre des corrections simples des attributs des formes primitives. Par exemple, si un segment de droite doit avoir son extrémité en un point, on pourra le raccourcir ou l'allonger, voire modifier son tracé afin que cette correction soit possible ; On obtient ainsi une méthode de recollement contextuelle qui contraste avec celle trop générale proposée par Herot. La figure 4.25 donne un tel exemple.

. Si une sous-forme absente n'est pas jugée essentielle, l'analyseur peut l'admettre comme "oubliée" et poursuivre l'analyse. Ainsi, si l'on a une description du type :

C ⟶ intérieur (A, B) et si, A étant reconnu , toutes les tentatives d'analyser B échouent, l'analyseur peut poursuivre l'analyse avec une forme C incomplète. Cela n'est évidemment admissible que si B n'est pas essentielle, et n'est pratiquement possible que si la suite de l'analyse ne doit pas s'appuyer sur un pôle ou un joint de B .

Dans des cas simples la correction peut être faite par l'insertion de primitives supposées absentes. C'est ce qu'a réalisé Masini [MAS 78] dans le cas des segments de droite ; mais ces corrections doivent être contrôlées pour éviter des résultats comme ceux qui ont été obtenus dans une réalisation précédent celle de MIRABELLE :
la forme originale de la figure 4.25 a été reconnue correctement ou avec des corrections surprenantes selon le choix du point de départ ([HAT 77]) .



forme originale — analyse correcte — analyse incorrecte.

figure 4.25.

. L'analyseur ignore naturellement les primitives qui ne lui sont pas utiles. Cela semble résoudre le problème créé par la création de nouvelles primitives par le bruit.

La situation n'est cependant pas aussi simple :
la présence de formes parasites ou mal identifiables fausse les tests qui permettaient de limiter les choix possibles. Ces tests ne seront donc plus

impératifs mais seulement indicatifs : ceci va accroître de façon exponentielle le temps de reconnaissance.

Autoriser l'analyseur à ignorer des formes supposées inutiles, revient à lui permettre de se restreindre à une sous-forme de la forme initiale il risque donc de se satisfaire d'une forme très restreinte par rapport à la forme initiale. Cette possibilité, combinée aux possibilités de corrections, nous amènera à voir l'algorithme d'analyse reconnaître une forme totalement différente de la forme initiale à partir d'une seule primitive de la forme de départ. Il nous faut donc introduire un contrôle sur l'analyseur. Ce contrôle se fait de trois façons :

- contrôler a posteriori le nombre de primitives qui n'ont pas été prises en compte et invalider la reconnaissance si ce nombre est trop élevé.

- limiter au fur et à mesure de l'analyse le nombre de corrections. Un seuil raisonnable sera un rapport entre le nombre de corrections et le nombre de primitives correctement utilisées ; ce nombre de corrections doit bien sûr être pondéré : une correction mineure sur un attribut de primitives n'a pas le même poids que l'insertion factice d'une sous-forme absente.

- en cas de choix, tenter d'abord les règles conduisant aux formes les plus "grandes". Cette dernière heuristique a pour but d'éviter de ne reconnaître qu'une sous-forme trop restreinte de la forme initiale, et par conséquent, de rejeter l'analyse par le test final indiqué ci-dessus.

3.3.5.- GRAMMAIRES STOCHASTIQUES ET FRÉQUENCE D'UTILISATION DES RÈGLES.

Les grammaires stochastiques (voir par exemple [FU 74]) ont été introduites par la considération suivante : certaines règles syntaxiques sont utilisées moins fréquemment que d'autres et donc les mots engendrés apparaissent moins fréquemment.

Pour chaque non-terminal A, l'ensemble des règles de premier membre A est muni d'une densité de probabilité ; si l'on suppose des probabilités indépendantes on associe à un mot α le produit des probabilités des règles utilisées pour dériver α de l'axiome ; on démontre que l'on définit ainsi une densité de probabilité sur les mots du langage engendré (à condition que la grammaire soit réduite). Bien que ces notions n'aient été définies que pour des grammaires décrivant des mots d'un monoïde ou des chaînes codant des contours de formes, cette notion peut très bien se généraliser aux descriptions que nous avons utilisées ici.

Utiliser ces techniques pour estimer la probabilité d'un mot paraît cependant difficile à justifier. En effet, plus une chaîne sera longue, plus elle utilisera de règles pour être définie, et plus sa probabilité sera faible. Une normalisation du type $\sqrt[n]{\ }$ où n est le nombre de règles utilisées serait nécessaire ; dans ce cas le résultat ne serait plus une probabilité, mais une valeur comprise entre 0 et 1 qui serait une mesure de "probabilité" (au sens intuitif du terme).

Par ailleurs l'hypothèse d'indépendance pour les probabilités d'utilisation des règles est fausse, et l'utilisation de cette mesure pour écarter des chaînes trop peu probables n'est pas réaliste . Considérons l'exemple suivant :

la grammaire a deux règles : $A \longrightarrow a$ /probabilité p /

$A \longrightarrow Aa$ /probabilité q /

la probabilité associée à a^n sera pq^{n-1} . La première règle est utilisée une fois et une seule dans tout mot α, donc lors de l'apprentissage la valeur estimée de p ne sera pas négligeable, même si la probabilité de trouver le mot simplement réduit à a est quasi nulle !

Fu propose en particulier l'utilisation de telle probabilité pour tenir compte des erreurs. Ainsi si on a le codage de contour donnée à la figure 4.26 un segment élémentaire horizontal peut être codé par a avec la probabilité p_1 et par b ou c avec la probabilité $p_2 = (1 - p_1)/2$. Considérons

alors un segment de droite, comme une suite de segments horizontaux élémentaires selon la grammaire :

$$L \longrightarrow HL \ / \ H$$
$$H \longrightarrow a \ / \ b \ / \ c$$

la probabilité associée à un segment est alors plus ou moins faible selon qu'elle a ou non beaucoup de segment déformé (b ou c) . Mais remarquons alors que les segments déformés de la figure 4.27 ont même probabilité bien que le premier corresponde plus à une droite déformée que le second.

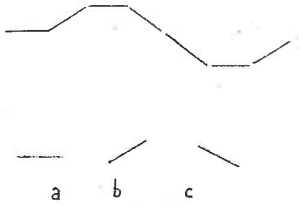


figure 4.26

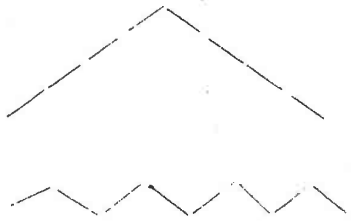


figure 4.27

La fréquence d'utilisation des règles a un autre intérêt cependant. Pierrel [PIE 75] travaillant sur la reconnaissance du discours continu a mis en évidence que l'utilisation d'une recherche en profondeur d'abord était notablement accélérée par l'exploration prioritaire de la voie la plus probable. En effet, supposons qu'à chaque fois, trois choix soient possibles, chacun ayant la probabilité p_i d'être correct ($\Sigma p_i = 1$) . Si l'arbre de recherche est de hauteur n et si les choix sont faits au hasard, le nombre moyen d'étapes nécessaires pour trouver la solution correcte est $E_n = 3^n/_2$. Si à chaque étape les choix sont faits dans l'ordre 1, 2, 3 , le nombre moyen d'étapes devient :

$$E'_n = p_1 E'_{n-1} + p_2(3^{n-1} + E'_{n-1}) + p_3(2.3^{n-1} + E'_{n-1})$$

$$= E'_{n-1} + 3^{n-1} (p_2 + 2p_3)$$

Comme $E'_1 = p_1 + 2p_2 + 3p_3 = 1 + (p_2 + 2p_3)$

on obtient $E'_n = 1 + \frac{3^n-1}{2} (p_2 + 2p_3)$

Si $p_1 > p_2 > p_3$ on aura donc $E'_n < E_n$

Avec par exemple $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{3}$, $p_3 = \frac{1}{6}$

on aura $E'_n \# \frac{E_n}{2}$.

L'utilisation des fréquences des règles est donc évidente lors d'une analyse descendante. Il faut pour l'analyse ascendante une autre probabilité indiquant pour une forme donnée la probabilité que telle règle définisse la forme englobante.

CHAPITRE 5

BILAN ET PERSPECTIVES

C H A P I T R E 5

BILAN ET PERSPECTIVES

Les différents aspects de l'approche structurelle sont illustrés par des exemples d'applications dans deux domaines voisins mais de complexité et de structure très différentes : les dessins et les images. Pour chacun d'eux, nous allons conclure séparément en mettant en évidence l'état des systèmes actuels, les améliorations possibles et les perspectives.

1.- LES DESSINS.

Par dessins nous entendons ici toutes formes composées de lignes, c'est-à-dire aussi bien les dessins manuscrits que ceux simulés en fournissant une liste de segments ou une matrice représentant la discrétisation de lignes digitalisées(y compris les lignes résultant de l'extraction des contours d'images de polyèdres [SHI 75]).

1.1.- LE SYSTÈME MIRABELLE.

A l'origine nous nous proposions de réaliser un outil général nous permettant d'opérer la reconnaissance et l'interprétation de dessins complexes éventuellement incomplets ou erronés. Cet objectif est atteint dans la mesure où le système MIRABELLE est capable d'analyser un dessin manuscrit correct d'une cinquantaine de segments en un temps raisonnable : une à deux minutes sur MITRA 125 (y compris les temps des échanges avec les périphériques); cependant,

les performances du système vont en s'affaiblissant avec le nombre d'omissions autorisées dans le tracé.

Les deux principales idées qui ont présidé à la réalisation de ce système ont une portée plus générale comme nous l'avons vu au chapitre 4. Il s'agit de :

- une analyse ascendante-descendante qui partant de sous-formes reconnues, permet de poursuivre l'analyse en étant plus guidé par la forme que par la syntaxe.
- une description formelle de la classe des dessins à reconnaître ; ce formalisme permet de calculer de façon statique des tables contenant les renseignements pertinents qui vont servir à l'analyseur.

En plus des extensions possibles qui ont été mentionnées chapitre 4 §.2 et §.3, les améliorations suivantes permettraient une meilleure utilisation du système :

a) les informations que l'on désire affecter à des parties d'un dessin ne sont pas toujours de nature graphique ; l'on voudra indiquer que porte et fenêtres d'une maison sont des ouvertures, que le style d'une maison dépend du rapport des surfaces des ouvertures sur celui de la façade, de la pente du toit, etc.... Cela revient à rajouter aux formes des attributs sémantiques, ceux-ci étant définis en fonctions des attributs graphiques de la formes (coordonnées, longueur, ...) et d'autres attributs sémantiques.

Considérons par exemple une suite de bornes et supposons que nous voulons les numéroter. Une description dans MIRABELLE pourrait être :

```
Suite : ADROITE (Borne, Suite)     a
Suite : Borne                      b
```

Pour attribuer un numéro à chaque borne il est nécessaire de faire le compte des bornes et cela peut se définir de la façon suivante :

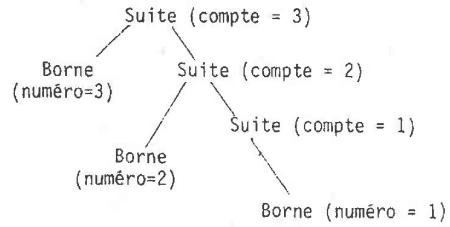
```
règle b : numéro (Borne) = 1  (1ère borne)
          compte (Suite) = 1  (suite d'une seule borne)
```


règle a : comme ici nous avons deux occurrences de "Suite" dans la règle nous les distinguons en notant $Suite_1$ la première et $Suite_2$ la seconde

Numéro (Borne) = compte ($Suite_2$) + 1

compte ($Suite_1$) = compte ($Suite_2$) + 1

La vérification de la correction de ces définitions et leurs évaluation se fait selon des algorithmes connus ([LOR 74]). L'arbre syntaxique ci-dessous montre un exemple de résultat de cette évaluation.



L'introduction de tels attributs dans une description parallèle à la partie syntaxique ne pose aucune difficulté théorique nouvelle : le problème a été complètement résolu dans toute sa généralité et les algorithmes sont disponibles à l'IRIA.

b) La procédure d'analyse est simple mais analyse la forme en un seul bloc à partir du point de départ. Lui permettre d'analyser des parties de formes significatives dispersées en différents endroits semble une modification prometteuse ; ainsi l'analyseur ne sera plus obligé d'abandonner des sous-formes significatives bien reconnues lors d'un retour arrière. Cette modification offre aussi des possibilités de tests plus puissants : nous nous étions restreints aux primitives dans cette version, maintenant nous pouvons tester la présence de sous-formes plus complexes.

c) La segmentation du tracé original en arc de cercle et segments de droite est paramétrée, mais faite de façon figée pour tout un tracé. Or, la façon de dessiner doit influer sur cette segmentation : un arc de cercle de faible courbure peut être corrigé en un segment de droite sauf s'il a été tracé lentement et avec précaution . Souvent aussi des tracés manuscrits sont prolongés par une nouvelle ligne se superposant partiellement à celle qu'elle prolonge, et le tout doit alors être considéré comme un seul tracé. Le système d'acquisition développé au M.I.T. dans l'équipe de Negroponte ([HER 76]) possède ces raffinements qui pourraient aussi être incorporés dans MIRABELLE. Cela revient à augmenter la complexité et l'importance de l'étage de segmentation, et par conséquence les algorithmes de corrections qui modifient et complètent cette segmentation sur des indications de l'analyseur.

1.2.- Autres travaux.

Nous avons discuté au chapitre 4 §.3.1 des différentes méthodes d'analyses utilisées. Mise à part la particularité du système de Bajcsy et Thidar [BAJ 77] qui permet d'opérer une reconnaissance à partir d'informations partielles mais caractéristiques, la représentation de l'univers et la stratégie de reconnaissance que nous avons adoptées se comparent favorablement avec celles des autres systèmes.

Pour la plupart, ces autres systèmes offrent cependant en plus des possibilités d'apprentissage ([HER 76] , [WIL 76]). Cette manière d'enrichir la description de l'univers est bien plus naturelle que celle qui consiste à ajouter des règles. Si l'on ne se restreint pas aux seuls segments de droite, (domaine d'application retenu dans les travaux cités ci-dessus), le problème de la segmentation complique celui de l'apprentissage ; ainsi considérons le dessin de la figure 5.1. a) ; doit-il être segmenté de la façon décrite en 5.1. b), 5.1. c) ou d'une autre façon ?

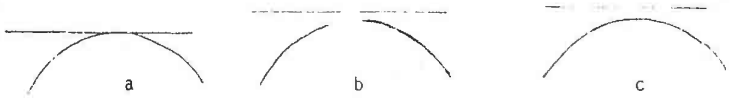


figure 5.1.

Nous avons mentionné en chapitre 4 §.2.2.2 le problème posé par la vision bidimensionnelle d'un objet à trois dimensions. Si nous avons pu donner une réponse structurelle au cas simple que posait Stanton , le problème reste posé pour le cas général. Notons que le problème des polyèdres et de leurs ombres a été résolu entièrement par Waltz [WAL 75] par une approche bien différente : il ne considère que les interprétations possibles pour chaque segment de droite et procède par élimination selon des règles de compatibilité entre segments ayant un sommet commun.

2.- Les Images.

Les images se distinguent des dessins par deux aspects principaux ; ces différences justifient souvent des approches que nous avions récusées au chapitre 4. Examinons ces différences :

a) les dessins représentent un concept produit par l'homme et par conséquent ont une structure plus riche que les images du monde extérieur qui apparaissent au contraire le plus souvent comme une collection d'objets ayant des relations plus ou moins faibles entre eux ; la description structurée d'une image ne dépasse que rarement trois niveaux. Cette faiblesse est compensée par l'extrême richesse des relations qui lient les sous-formes, tant sur le plan des positions que sur celui des rapports entre surfaces, textures, intensité lumineuse ....

Voir la forme comme une collection d'objets liés par des relations justifie alors la vision de l'univers sous forme de graphe, mais comme nous l'avons dit au chapitre 4 §.1, cette approche exclut pratiquement les relations entre deux groupes d'objets.

b) La complexité des formes primitives contraste avec la faiblesse de la structure des images ; la conséquence de cette richesse est que les formes élémentaires sont plus difficiles à identifier, mais bien plus informantes que les segments des dessins. Ainsi, dans un paysage,une surface ayant une texture de tuiles caractérise une construction et la seule présence d'une fenêtre sous ces tuiles permet de conclure à la présence d'une habitation ; mais la détection d'une texture est particulièrement couteuse (20 mn d'IRIS 80 dans le cas étudié par [GAM 79]). C'est pourquoi le temps de recherche des sous-formes est pris en considération dans la stratégie de certains systèmes ([BAL 77]).

Que les réalisations non spécialisées fonctionnant à ce jour n'aient été testées que sur des images très simples s'expliquent par ces considérations. Elles expliquent aussi pourquoi [BAL 77] , [FRE 77] et dans une moindre mesure [BAJ 74] , ont préféré aux outils formels ceux plus souples des réseaux et procédures ; dans un système où la réalisation d'une précondition déclenche une action il est facile d'adapter la stratégie à chaque cas ; on peut ainsi inférer la présence d'une construction à partir de la présence d'un toit de tuiles ; la complexité des relations entre les objets sera rendue par les procédures ad hoc. Cependant, les avantages d'une approche formelle : description statique sans écriture des procédures, possibilité d'un prétraitement permettant de fournir une stratégie unique générale, méritent qu'on travaille à revoir les outils nécessaires afin d'obtenir un système réellement opérationnel sur les images. Ainsi la complexité des relations peut être contournée par une paramétrisation de quelques relations de base ; d'autres outils de descriptions permettraient par exemple d'exprimer la présence de formes choisies dans une

collection d'objets possibles et ayant entre eux une relation de proximité ; ce cas se présente souvent pour des images et s'exprime lourdement avec le modèle du chapitre 4.

Nous avons déjà mentionné §.1.2. le problème de la vision tridimensionnelle. Il se retrouve posé ici, la complexité des images s'y ajoutant. Les structures bidimensionnelles d'un visage vu de face, de profil ou de demi-profil sont différentes : le nez est topologiquement à des endroits différents,.... Peut-on espérer trouver une structure unique qui permette de modéliser ces trois aspects ?

Une nouvelle technique d'interprétation d'images se développe actuellement : l'étiquetage de scènes (scene labelling) ; elle consiste à attribuer à chaque objet une étiquette et à introduire des relations de compatibilité entre les étiquettes en fonction de leurs positions respectives. C'est de cette façon que Waltz a résolu le problème des polyèdres (cf. §.1.2.). Elle s'est développée avec le problème de l'indéterminisme de la reconnaissance et a abouti à des techniques d'étiquetage probabilistes optimales ([ROS 76] , [FAU 78]) . L'approche structurelle telle que nous l'avons défendue ici ne permet pas (sauf par recherche exhaustive) l'interprétation optimale dans ce sens ; la seule façon dont l'algorithme de reconnaissance prend en compte les probabilités de reconnaissance est de choisir à chaque étape la voie la plus plausible d'abord. En revanche, comme pour les graphes, cette technique d'étiquetage ne permet pas d'exprimer de relation entre des objets structurés, de façon simple.

3.- En conclusion

Nous résumons ici les axes principaux de ce travail.

Les méthodes structurelles fournissent un outil naturel de description et de reconnaissance de formes. Afin d'être efficace le choix des formes primitives et des relations qui rassemblent ou lient ces formes est fondamental. Dans une telle approche la stratégie d'analyse peut être déterminée par un prétraitement de la description a priori qui est fournie, et les connaissances

sur le type de forme à reconnaître autorisent des corrections en cas de déformations ou de présence de bruit.

Ces idées sont illustrées par le système MIRABELLE que nous avons écrit en FORTRAN sur MITRA 125. MIRABELLE est capable de prendre en compte une description formelle d'une classe de dessins puis d'analyser un dessin manuscrit appartenant à l'ensemble décrit. Ce système a des performances qui se comparent favorablement aux systèmes existants ; de plus, il est capable de prendre en compte des dessins comportant des omissions et des primitives inutiles.

Notre travail se distingue des autres travaux dans le même domaine par deux particularités :

- un traitement élaboré de la description formelle qui fournit des renseignements utilisables pour piloter l'analyse. Notre formalisme suffisamment strict permet en particulier de justifier théoriquement l'utilisation des différentes tables calculées.

- une technique d'analyse souple qui s'écarte des méthodes habituellement tirées de l'analyse syntaxique classique ; simple et comportant peu d'heuristiques, elle se généralise aisément.

Les extensions et améliorations immédiates de MIRABELLE ont été signalées tout au long de ce travail ;

- introduction de concaténations plus sophistiquées, par exemple, par mise en commun de sous-formes;
- calcul de renseignements supplémentaires pouvant servir à l'analyseur (contexte topographique);
- modification de l'algorithme d'analyse afin de lui permettre d'analyser en parallèle plusieurs ilots;
- enrichissement de la description par l'incorporation d'attributs orientés vers les applications.

La complexité des images du monde réel rend actuellement problématique l'utilisation de ces idées dans ce domaine. La seule issue à court terme semble être l'utilisation d'un outil moins formel : les réseaux de procédures. Mais la recherche de moyens de description adaptés ainsi que celle d'une stratégie d'analyse prenant mieux en compte la richesse de l'information des formes élémentaires rendraient possibles l'extension de ce travail.

BIBLIOGRAPHIE

B I B L I O G R A P H I E

[AHO 73] V.A. AHO, J.D. ULLMAN - The theory of parsing, translation and compiling. Prentice Hall, Vol 1 (1972), vol 2 (1973).

[AND 77] R.H. ANDERSON - Two-dimensional Mathematical Notation. In Syntactic Pattern Recognition (Fu Ed.) Springer-Verlag (1977).

[AZE 75] J. AZEMA - Grammaire de graphes, algorithmes d'analyse et applications. Thèse de 3è cycle, IMAG, Grenoble (1975).

[BAJ 74] R. BAJCSY, L.I. LIEBERMAN - Computer description of real outdoor scenes. 2d IJCPR, Copenhague (1974).

[BAJ 77] R. BAJCSY, A. TIDHAR - Using a structured world model in flexible recognition of two dimensional patterns. Pattern Recognition vol. 9 pp. 1-10.(1977).

[BAL 77] D.H. BALLARD et al. - An approach to knowledge directed image analysis. 5th IJCAI , Cambridge . (1977).

[BAR 71] H.G. BARROW, J.R. POPPLESTONE - Relationnal descriptions in picture processing. In Machine Intelligence 6 pp 377-396. Edinburg Univ. Press. (1971).

[BRA 77] J.M. BRAYER, P.H. SWAIN, K.S. FU - Modeling of Earth Resources Satellite Data. In Syntactic Pattern Recognition Applications, (FU ed.) Springer-Verlag (1977).

[BEL 78] A. BELAID - Segmentation de tracés en vue de leur analyse ; application à la reconnaissance structurelle de caractères. Congrès AFCET- Orsay , sept. 1978.

[BEL 79] A. BELAID.- Reconnaissance en ligne de formules mathématiques manuscrites. Congrès AFCET reconnaissance des formes et intelligence artificielle. Toulouse 1979.

[BLO 76] P.A. BLONIARZ, M.J. FISCHER, A.R. MEYER - A note on the average time to compute transitive closures. In Automata, Languages and Programming (Michaelson, Milner ed.) Edinburg University Press (1976)

[CAS 77] S. CASTAN, G. STAMON - Traitement numérique des images. Cours de l'école d'été de l'AFCET. Montréal 1977.

[CHO 74] Y.E. CHO - The generating properties of context-free picture grammars. 2d IJCPR, Copenhague (1974).

[CLO 68] M.B. CLOWSES - Transformational grammars and the organisation of pictures. In Automatic Interpretation and Classification of Images (Grasseli ed.) Academic Press (1968).

[DER 71] J.C. DERNIAME, C. PAIR - Problèmes de cheminement dans les graphes. Dunod (1971).

[FAU 78] O. FAUGERAS, M. BERTHOD - Scene labeling : an optimization approach. Publication IRIA, décembre 1978.

[FED 68] J. FEDER - Languages of encoded line patterns. Inf. and Control 13 pp. 230-244 (1968).

[FED 71] J. FEDER - Plex languages. Information Sci. 3, pp. 225-241. (1971).

[FRE 61] H. FREEMAN - On the encoding of arbitrary geometric configurations. IEEE Trans. Elect. Comp. 10. pp. 260-268.

[FRE 77] E. FREUDER - A computer system for visual recognition using active knowledge. 5th IJCAI, Cambridge (1977).

[FU 74] K.S. FU - Syntactic Methods in Pattern Recogrition. Academic Press (1974).

[GAL 75] V. GALLO - Geometrical pattern recognition based on linguistic method of description . 4th I.J.C.A.I. , Tbilisi (1975).

[GAM 79] J.P. GAMBOTTO, C. GUEGEN - A multidimensional modeling approach to texture classification and segmentation. IEEE conf. on Acoustics, Speeck and Signal processing. Washington (1969) et publication ENST C-78024.

[GRI 74] M. GRIFFITHS - Introduction to compiler compiler. In Compiler Construction, an advanced course. Lecture Notes in Computer Science. Springer Verlag (1974).

[GRO 74] J. GROS, J.P. MAROY - An axiomatic representation of on-line recognition of two dimensional programming language,IRIA-LABORIA rapport n° 51 (1974).

[HAT 77] J.P. HATON, R. MOHR - Méthodes syntaxiques en reconnaissance des formes. Rapport de contrat IRIA-SESORI. Nancy (1977).

[HER 76] C. HEROT - Graphical imput through machine recognition of sketches. SIGGRAPH 76, Comp. Graphics, Vol.10, n° 2 pp. 97-102 (1976).

[HOR 75] S.L. HOROWITZ - A syntactic algorithm for peak detection in waveforms. C.A.C.M. vol. 18, n° 5 (1975).

[KIR 64] R. KIRSCH - Computer interpretation of english text and pictures patterns. IEEE Trans. Elect. Comp. 13 , pp. 363-376 (1964).

[KNU 65] D.E. KNUTH - On the translation of languages from left to right. Inf. and Control 8, pp. 607-639 (1965).

[LEY 65] R.S. LEYDLEY et al. - FIDAC and associated syntax-directed pattern recognition programming system (Bechkowitz, Tippet, ed.) M.I.T. Press, Cambridge (1965).

[LOR 74] B. LORHO - De la définition à la traduction des langages de programmation : méthode des attributs sémantiques. Thèse d'état Université Paul Sabatier, Toulouse (1974).

[LIV 78] C. LIVERCY - Théorie de programmes. Dunod (1978).

[MAR 79] J.F. MARI - Contribution à l'analyse syntaxique et à la recherche lexicale en reconnaissance du discours continu. Thèse de 3è cycle Université de Nancy I, (1979) (à paraître).

[MAS 78] G. MASINI - Réalisation d'un système de reconnaissance structurelle et d'interprétation de dessins. Thèse de 3è cycle. Nancy I (1978).

[MOA 77] B. MOAYER, K.S. FU - Fingerprint classification. In Syntactic Pattern Recognition Applications (FU Ed.) Springer Verlag (1977).

[MOH 71] R. MOHR - Formalisation d'un outil pour la description d'images. RIRO, R-2, pp. 118-123 (1971).

[MOH 73] R. MOHR - Modèle algébrique pour l'analyse syntaxique de figures. Thèse de 3è cycle . Nancy I. (1973).

[MOH 75] R. MOHR - Généralisation des langages à contexte libre. Application à l'analyse syntaxique de figures. RAIRO 5-2 pp. 55-88. (1975).

[MOH 76] R. MOHR, J.P. HATON - A new parsing algorithm and its application to speech and picture analysis. 3d IJCPR San Diego (1976).

[MOH 78] R. MOHR, G. MASINI - Drawing analysis and C.A.D.. In Artificial Intelligence and Pattern Recognition in Computer Aided Design (Latombe ed.) North Holland (1978).

[MON 70] V.G. MONTANARI - Separable graphs, planar graphs and web grammars. Inf. and control , pp. 243-268 (1970).

[MIL 72] D.L. MILGRAN, A. ROSENFELD - A note on "grammars with coordinates" In Graphic Languages (Nake and Rosenfeld ed.) North-Holland (1972).

[NAR 66] R. NARASIMHAN - Syntax directed interpretation of classes of pictures. C.A.C.M. 9,3 pp. 166-173 (1966).

[OTA 73] P.G. OTA - Mosaic grammars. Univ. of Pennsylvanie, Moore School report n° 73-10

[PAI 68] C. PAIR, A. QUERE - Définition et étude des bilangages réguliers. Inf. and Control 13 pp. 565-593 (1968).

[PAI 73] C. PAIR - Cours d'analyse syntaxique. Ecole d'été du Bréau. EDF-IRIA-CEA , sept. 1973.

[PAV 69] M. PAVEL - Fondements mathématiques de la reconnaissance des structures. Hermann 1969.

[PIE 75] J.M. PIERREL - Contribution à la compréhension du discours continu. Thèse de 3è cycle. Nancy I (1975).

[RED 77] K.R. REDDY et al. - Speech understanding systems : a final report. Carnegie Mellon Univ., Comp. Science dept. (1977).

[SCH 78] D. SCHOEVAERT-BROSSAULT - Analyse morphologique et classification automatique des spermatozoïdes humaines. In Perception Automatique et classification par ordinateur. Séminaires IRIA (1978).

[ROS 71] A. ROSENFELD - Isotonic grammars, parallel grammars and picture grammars. In Machine Intelligence 6, pp. 281-294, Edinburg Press Univ. (1971).

[ROS 76] A. ROSENFELD, R.A. HUMMEL, S.W. ZUCKER - Scene labeling by relaxation operations. IEEE trans. on Systems Man and Cybernetics. N° 6 pp. 420-443, (1976).

[SHA 69] A.C. SHAW - A formal picture description scheme as a basis for picture processing systems. Inf. and Control 14. pp. 9-52 (1969).

[SHA 77] L.G. SHAPIRO, R.J. BARON - EPS^3 : a language for pattern description and a system for pattern recognition. IEEE trans. on Soft. Eng. Vol. 3, n° 2, pp. 169-183 (1977).

[STA 72] R.B. STANTON - Interpretation of graphic languages. In graphic Languages (Nake, Rosenfeld ed.) North Holland (1972).

[VAM 77] T. VAMOS - Industrial Objects and Machine Part Recognition. In syntactic Pattern Recognition Applications (FU ed.) Springer Verlag (1977).

[WAL 75] D. WALTZ - Understanding line drawings of scenes with shadows. In The psychology of Computer Vision (Winston Ed.) Mac Graw-Hill (1975).

[WIL 77] H. WILLIAMS - A net-structure learning system for pattern description. Pattern Rec. vol 8, pp. 261-271 (1977).

[WIN 72] T. WINOGRAD - Understranding natural language. Academic Press (1972).

[WIN 77] P.H. WINSTON - Learning structural descriptions from examples. In The Psychology of Compter Vision (Winston ed.) Mac Graw-Hill (1977).

ANNEXES

ANNEXE 1

Evaluation de l'optimisation du calcul du plus petit point fixe (chapitre 1, §.1.3.3)

N'ayant pas connaissance de travaux sur ce point, je livre ici une petite étude simple qui permet d'évaluer ce gain. Cette étude peut être étendue facilement, soit en considérant d'autres types de fonctions (fonctions n-aires au lieu de binaires) soit en considérant d'autres vitesses de convergence des approximations successives. L'évaluation en moyenne de cette optimisation semble par contre plus ardue.

Hypothèses simplificatrices :

On considère que toutes les fonctions f_{ij} du système d'équations sont binaires ou constantes et que les arguments de ces fonctions sont des inconnues ; il y a M occurrences de fonctions binaires dans le système.

Nous supposerons que le point fixe est calculé en p itérations et que, à chaque itération, on calcule un nombre égal de nouveaux éléments pour chaque composante.

Calcul de $X_{k+1} = (X_{k+1,1}, \ldots, X_{k+1,n})$ à partir de X_k

algorithme initial :

$X_{k+1} \leftarrow (\emptyset, \ldots, \emptyset)$

pour chaque f_{ij} faire

si f_{ij} est constante alors $X_{k+1,i} \leftarrow X_{k+1,i} \cup \{f_{ij}\}$

sinon % f_{ij} est binaire et A_r et A_s sont ses arguments %

pour chaque $x \in X_{k,r}$ faire

pour chaque $y \in X_{k,s}$ faire

$X_{k+1,i} \leftarrow X_{k+1,i} \cup \{f_{ij}(x, y)\}$

fait fait fsi

fait

algorithme optimisé :

$Y_{k+1} \leftarrow (\emptyset, \ldots, \emptyset)$

pour chaque f_{ij} non constant faire

% f_{ij} est binaire, A_r et A_s sont ses arguments %

pour chaque $x \in X_{k,r}$ faire

pour chaque $y \in Y_{k,s}$ faire

si $f_{ij}(x,y) \notin X_{k,i}$ alors $Y_{k+1,i} \leftarrow Y_{k+1,i} \cup \{f_{ij}(x,y)\}$

fsi

fait fait

% si f_{ij} est commutative cette seconde partie est inutile %

pour chaque $x \in Y_{k,r}$ faire

pour chaque $y \in X_{k,s}$ faire

si $f_{ij}(x,y) \notin X_{k,i}$ alors $Y_{k+1,i} \leftarrow Y_{k+1,i} \cup \{f_{ij}(x,y)\}$

fsi

fait fait

ait

$k+1 \leftarrow X_k \cup^n Y_k$

Cas 1 : progression arithmétique de la solution approchée.

Nous supposons ici que chaque itération calcule m éléments pour chaque composante. Après p itérations la solution sera donc un n-uple $(E_1, \ldots, E_n)$ avec $\text{card}(E_i) = mp$ $(i = 1, \ldots, n)$.

nombre d'opérations de l'algorithme initial :

nous avons $\text{card}(X_{k,i}) = km$;

M fonctions f_{ij} sont binaires, d'où par conséquent

$M(km)^2$ calculs de $f_{ij}(x, y)$ nécessaires pour

calculer X_{k+1} , soit pour le calcul complet du point fixe :

$$Mm^2 \sum_{k=1}^{p} k^2 = \frac{p(p+1)(2p+1)}{6} \text{ opérations.}$$

nombre d'opérations de l'algorithme modifié :

$\text{card}(Y_{k,i}) = m$, $\text{card}(X_{k,i}) = km$

d'où $2.M.km^2$ calculs de $f_{ij}(x,y)$ nécessaires pour le calcul de X_{k+1} et Y_{k+1} .

Soit pour le calcul de la solution complète :

$Mm^2\, p(p+1)$ opérations

soit environ un gain d'environ $\frac{p}{3}$

(rappelons que p est le nombre d'itérations pour le calcul du point fixe).

Cas 2 : progression géométrique de la solution approchée.

Nous supposons maintenant que la kième itération calcule $m^{(p-k)}$ éléments nouveaux éléments pour chaque composante. La solution calculée après p itérations sera donc un n-uple dont chaque composante aura :

$$\sum_{k=1}^{p} m^{(p-k)} = \frac{m^p - 1}{m - 1} \quad \text{éléments.}$$

Nombre d'opérations de l'algorithme initial.

$$\text{card } (X_{k,i}) = \sum_{i=1}^{k} m^{p-i} = m^{p-k} \frac{m^k - 1}{m - 1}$$

soit donc $M(m^{p-k} \frac{m^k - 1}{m - 1})^2$ calculs de fonctions pour déterminer X_{k+1} , ce qui donne pour déterminer la solution complète :

$$\frac{M}{(m-1)^2} \sum_{k=1}^{p} [m^{p-k} (m^k-1)]^2 = \frac{M}{(m-1)^2} [pm^{2p} - \frac{m^{2p}-1}{m - 1}] \quad \text{opérations.}$$

nombre d'opérations de l'algorithme modifié :

$$\text{card}(X_{k,i}) = m^{p-k} \frac{m^k - 1}{m - 1} \quad ; \quad \text{card } (Y_{k,i}) = m^{p-k}$$

La détermination de Y_{k+1} et X_{k+1} demande donc :

$$2 M m^{p-k} \times m^{p-k} \frac{m^{k-1}}{m - 1} \quad \text{opérations}$$

Soit donc au total

$$\frac{2M}{m - 1} \sum_{k=1}^{p} (m^2)^{p-k} (m^k - 1) = \frac{2M}{m-1} [m^p \frac{m^p-1}{m-1} - \frac{m^{2p}-1}{m^2-1}]$$

$$= \frac{2M}{(m-1)^2} [m^{2p}-m^p - \frac{m^{2p}-1}{m+1}]$$

ce qui représente cette fois un gain d'un facteur $\frac{p}{2}$

Nous nous sommes restreint ici au cas de fonctions binaires ; en effet ce sont des fonctions de ce types que nous utilisons au chapitre 3.

Dans les autres cas le gain reste en $O(p)$. Notons que le cas 2 (convergence géométrique de la solution) est plus proche de la réalité que le cas 1 ; dans tous les cas le gain est un $O(p)$ et on peut estimer qu'il en est de même dans la réalité.

ANNEXE 2

Démonstration du théorème sur les initiales (page 63)

Nous noterons initiale(f) (resp. finale(f)) l'ensemble des primitives ou opposés de primitives de f ayant leur origine coïncidant avec l'origine (resp. l'extrémité) de f .

Dans la partie directe et réciproque nous ne démontrerons que le cas $A \in N$, la cas $A \in -N$ étant symétrique.

partie directe :

Démontrons le résultat par récurrence sur n tel que $A \; \mathcal{J}^n \{a_1, \ldots, a_m\}$. Le résultat est évident pour $n = 0$.

Supposons le résultat établi jusqu'au rang n-1 ; au rang n :

$A \; \mathcal{J} \{B_1, \ldots, B_q\} \; \mathcal{J}^{n-1} \{a_1, \ldots, a_m\}$

il existe donc une règle $A \to \mu$ telle que $\mathcal{G}(\mu) = \{B_1, \ldots, B_q\}$

et $B_i \; \mathcal{J}^{k_i} \{b_1^i, \ldots, b_{n_i}^i\}$ $(i = 1, \ldots, q)$

et $\sum_{i=1}^{q} k_i = n - 1$, et $\bigcup_{i=1}^{q} \{b_1^i, \ldots, b_{n_i}^i\} = \{a_1, \ldots, a_m\}$

L'hypothèse de récurrence permet de conclure qu'il existe $f_i \in S(B_i)$ $(i = 1, \ldots, q)$ tels que $\text{initiale}(f_i) = \{b_1^i, \ldots, b_{n_i}^i\}$

Considérons alors la formule μ' obtenue à partir de μ en substituant aux B_i les f_i et aux autres inconnues A_i une forme de $S(A_i)$

μ' s'interprète (hypothèse H2) en une forme f et l'hypothèse H1 nous garantit qu'à l'origine de f n'apparaissent que les origines des formes spécifiées par μ' , c'est à dire

$$\bigcup_{i=1}^{q} \{b_1^i, \ldots, b_{n_i}^i\} = \{a_1, \ldots, a_m\}$$

d'où le résultat.

Réciprogue.

Soit Φ l'application de $[\mathcal{P}(F)]^n$ dans lui-même associée au système d'équations (cf. chapitre 1 §.3.1.2.).

Si $f \in S(A_i)$ il existe n tel que f appartienne à la ième composante de $\Phi^n(\emptyset, \ldots, \emptyset)$.

Raisonnons alors par récurrence sur n .

Pour $n = 0$, nous avons à faire à l'ensemble vide : le résultat est évident.

Supposons le résultat établi jusqu'au rang $n - 1$.

Donc $f = \mu'$ où μ' est obtenu à partir d'un terme μ de la ième équation en substituant à chaque inconnue A_j un terme de f_j de la jème composante de $\Phi^{n-1}(\emptyset, \ldots, \emptyset)$; par construction on trouve donc à l'origine de f les origines des primitives d'initiale (f_j) si $A_j \in \mathcal{G}(\mu)$, les origines des primitives de finale (f_j) si $-A_j \in \mathcal{G}(\mu)$, et l'hypothèse H1 nous garantit que nous ne trouvons que celles-ci, donc :

$$\text{Initiale}(f) = \left(\bigcup_{B_j \in \mathcal{C}_r(\mu)} \text{initiale}(f_j) \right) \cup \left(\bigcup_{B_j \in \mathcal{C}_r(\mu)} \text{finale}(f_j) \right)$$

or $A_i \; J \; \mathcal{C}_r(\mu)$

par récurrence $B_j \; J^* \text{ initiale } (f_j)$

$- B_j \; J^* \text{ finale } (f_j)$

donc $A_j \; J^* \left(\bigcup_{B_k \in \mathcal{C}_r(\mu)} J^* \; B_k \right)$

donc $A_i \; J^* \text{ finale } (f)$

ce qui clôt la récurrence

ANNEXE 3

Démonstration du théorème sur les contextes (page 76).

Nous noterons cont(f', f) le couple (α, β) , α ensemble des primitives et opposés de primitives apparaissant à l'origine de la sous-forme f' de f et n'appartenant pas à f' , β l'ensemble de celles apparaissant à l'extrémité de f' et n'appartenant pas à f' .

Partie directe :

Nous procéderons par récurrence sur n où n est la longueur du chemin de l'arbre syntaxique de f et allant de la racine à la racine du sous-arbre décrivant f' .

n = 0 ; dans ce cas f' = f , $\alpha = \beta = \emptyset$ et A est l'axiome ; dans ce cas $\mathcal{C}(A, -A) = (\emptyset, \emptyset)$ par définition.

Supposons le résultat établi jusqu'au rang n - 1 ; au rang $n (n \geq 1)$:

soit f" la forme directement englobante de f et soit B sa classe ; par hypothèse de récurrence

cont (f", f) = $(\alpha", \beta")$ avec $(\alpha", \beta") \in \mathcal{C}(B, -B)$;

soit alors r la règle $B \longrightarrow \mu$ telle que μ' obtenue en substituant f' à A vérifie :

$\mu' \xrightarrow{*} \lambda$ = f" (c.a.d. la règle permettant d'obtenir f" à partir de B) .

soit $(\alpha', \beta') = \text{cont}(f', f'')$. L'hypothèse H1 et le théorème des initiales nous permettent d'affirmer que :

$$(\alpha', \beta') \in \mathfrak{J}^*.(D_r(A, -A))$$

Plusieurs cas se présentent alors selon que l'origine ou l'extrémité de f' coïncide avec l'origine ou l'extrémité de f'' . Nous n'en discuterons que deux , les autres sont similaires ;

a) ni l'origine, ni l'extrémité de f' ne coïncident avec l'origine ou l'extrémité de f'' et donc d'après l'hypothèse H1
$\text{cont}(f', f) = \text{cont}(f, f'') = (\alpha', \beta')$

Dans ce cas donc $P_r(A, -A) = (\emptyset, \emptyset)$

Par définition :

$$\mathcal{C}_r(A, -A) \supset \mathfrak{J}^* D_r(A, -A)$$

et donc $(\alpha, \beta) = (\alpha', \beta') \in \mathcal{C}_r(A, -A) \subset \mathcal{C}(A, -A)$

b) l'origine de f' coïncide avec l'extrémité de f'' ;
donc maintenant $\text{cont}(f', f) = (\alpha' \cup \beta'', \beta')$.

Ici $P_r(A, -A) = (-B, \emptyset)$

par définition

$$\mathcal{C}_r(A, -A) \supset \mathcal{C}(P_r(A, -A)) \hat{\cup} \mathfrak{J}^*(D_r(A, -A))$$

or $(\beta'', \emptyset) \in \mathcal{C}(-B, \emptyset)$

donc

$$(\alpha' \cup \beta'' , \beta') \in \mathcal{C}_r(A, -A) \subset \mathcal{C}(A, -A)$$

ce qui clôt la récurrence de la partie directe.

La réciproque est fausse ainsi que le montre le contre exemple suivant :

$X = A \underline{f} B$
$A = \ldots\ldots$
$B = a \cup b$

nous avons $D_r(A, -A) = (\{B\} , \{-B\})$, $P_r(A, -A) = (\emptyset, \emptyset)$

$$\mathfrak{J}^*(B) = \{ \{a\} , \{b\} \}$$

donc $\mathcal{C}(A, -A) = \{ (\{a\} , \{-a\}) , (\{a\} , \{-b\}) , (\{b\} , \{-a\}) , (\{b\} , \{-b\}) \}$,

or il est clair que les seuls contextes possibles ne peuvent être que $(\{a\} , \{-a\})$ et $(\{b\} , \{-b\})$.

ANNEXE 4

Démonstration de la correction de l'analyse syntaxique globale (pp. 149.150).

Soit $\mathcal{F}$ l'ensemble des formes et Φ_s l'application de $[\mathcal{B}(\mathcal{F})]^n$ dans lui-même associée au système à point fixe (cf. §.3.1. chapitre 1) ; la solution du système à point fixe est :

$$\lim_{n \to \infty} \Phi^n (\emptyset, \ldots, \emptyset) \quad .$$

La correction sera une conséquence des trois lemmes suivants :

lemme 1 :

Soit G une sous-forme présente et appartenant à $S(A_i)$ et soit m le nombre de ses primitives ; alors G appartient à $\Phi^k(\emptyset, \ldots, \emptyset)$ pour $k \geq m$.

Démonstration :

Comme Φ est croissante il suffit d'établir le résultat pour $k = m$.

Le résultat est évident pour $m = 0$.

Supposons le vrai jusqu'au rang $m - 1$.

Soit G est une primitive et donc d'après H4 , on a le terme G dans la ième équation et donc le résultat est établi (avec $m = 1$) .

Sinon il existe un terme $\alpha(A_{i_1}, \ldots, A_{i_k})$ dans la ième équation et $G = \alpha(f_1, \ldots, f_n)$ avec $f_j \in S(A_{i_j})$. Comme le nombre de primitives de G est la somme du nombre des primitives des f_{i_j} (hypothèse H4) et que $n \geq 2$ (hypothèse H2) on peut appliquer l'hypothèse de récurrence à $f_1, \ldots, f_n$; f_j appartient donc à la $i_j$ième composante de $\Phi^{m-1}(\emptyset, \ldots, \emptyset)$ et donc $G = \alpha(f_1, \ldots, f_n)$ appartient à la ième composante de $\Phi^m(\emptyset, \ldots, \emptyset)$.

Lemme 2 :

Soit E_k défini à la kième itération de l'algorithme ; si $(f, A_i, Q) \in E_k$ alors f appartient à $S(A_i)$.

La démonstration est une conséquence immédiate de la construction de E_k .

Lemme 3 :

Soient $p \geq 2$ et f une forme de la $i^{ème}$ composante de $\Phi^p(\emptyset, \ldots, \emptyset)$; si f est une sous-forme non réduite à une primitive présente dans la forme à reconnaître, alors $(f, A_i, Q) \in E_p$ (Q étant l'ensemble de ses primitives).

Démonstration :

par récurrence sur p .

$\Phi^1(\emptyset, \ldots, \emptyset)$ ne contient que des formes primitives donc $\Phi^2(\emptyset, \ldots, \emptyset)$ ne contient que des formes primitives ou des formes du type $g = \alpha(f_1, \ldots, f_n)$ où les f_i sont primitives.

Si g est une sous-forme de la forme à reconnaître, les $f_1, \ldots, f_k$ étant des primitives les $(f_j, f_j, \{f_j\})$ appartient à E_1 , et par construction même de E_2 et compte tenu de H3 , $(\alpha(f, \ldots, f_n), A_i, \{f, \ldots, f_n\}) \in E_2$

Le même raisonnement permet alors de passer de $p - 1$ à p .

En conséquence, nous obtenons le théorème : le lemme 2 nous garantit que nous ne construisons dans E_n que les solutions du système, et le lemme 3 nous montre que nous construisons toutes les sous-formes possibles de la forme initiale ; enfin le lemme 1 nous affirme que nous pouvons arrêter nos itérations avec $k = m$.

NOM DE L'ETUDIANT : Mr MOHR Roger

NATURE DE LA THESE : Doctorat Es Sciences

VU, APPROUVE

et PERMIS D'IMPRIMER

NANCY LE 11 JUIN 1979

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

M. BOULANGE