COMPILATEURS DE GRAMMAIRES DE FORMES

Thomas C. HENDERSON [x] et Larry DAVIS [xx]

Résumé - On étudie une technique syntaxique utilisant des contraintes et appliquée au problème de la reconnaissance de formes. Une classe de grammaire est définie permettant la représentation de contours à 2-D. Les contraintes syntaxique et sémantique dérivées d'une grammaire de formes sont utilisées par un analyseur hiérarchique basé sur la relaxation discrète. Nous proposons une méthode automatique pour déterminer les contraintes entre les symboles d'une grammaire ; cette méthode correspond à une extension des systèmes d'écriture de traducteurs classique, mais appliquée à la reconnaissance des formes.

Cette recherche a été financée en partie par la "National Science Foundation" par le contrat ENG-7904037.

[x]INRIA, Domaine de Voluceau - Rocquencourt BP. 105, 78153 LE CHESNAY

[xx]Computer Science Department University of Texas, Austin, Texas

I. - INTRODUCTION

Les méthodes de l'intelligence artificielle peuvent être appliquées aux problèmes qui se posent dans le domaine de la compilation comme l'a déjà montré Deransard [3]. Mais la démarche inverse est également possible. C'est à dire que certaines techniques de compilation efficaces et rapides peuvent également fournir une méthode pour l'analyse syntaxique des formes. Dans le cas des compilateurs classiques, on restreint la classe des grammaires, par exemple, aux grammaires LR(k), et on utilise un contexte bien défini dans l'analyseur syntaxique (voir Aho et Ullmann [1]). Un tel analyseur a une partie (les tableaux) synthétisée automatiquement à partir d'une grammaire de contexte libre. L'approche utilisée ici consiste à intégrer des méthodes analogues dans l'analyse des formes.

Plusieurs modèles syntaxiques ont été proposés [5,11,12], mais notre point de départ relève plutôt d'un problème d'étiquetage, et non d'une analyse syntaxique. Etant donné le contour d'un objet (un codage de Freeman, par exemple), on fait une approximation par segments de droites ; chacun de ces segments élémentaires, appelé une primitive, peut correspondre à un symbole terminal, mais on ne sait pas lequel. Le problème est donc d'analyser toutes les combinaisons possibles. S'il y a n symboles terminaux et m primitives, il existe n^m possibilités. Bien entendu, n^m est un nombre trop grand pour pouvoir appliquer les techniques standard.

On propose de faire l'hypothèse de tous les symboles pour chaque primitive et de faire une analyse ascendante et d'éliminer les hypothèses qui ne satisfont pas au contexte syntaxique et sémantique. Cette méthode est assez efficace pour les formes à 2-D, parce que, en général, une telle forme à 2-D a une structure plus riche que dans les langages classiques. Les autres méthodes (voir Masini et Monr [9]) utilisant les contraintes de contexte ont été proposées, mais il s'agit presque toujours d'une analyse syntaxique séquentielle.

Le schéma général d'une grammaire de formes est montré dans la Figure 1a, tandis que l'analyse traditionnelle est schématisée dans la Figure 1b. Un compilateur de grammaires de formes produit un mécanisme d'analyse de formes (ou compilateur de formes) à partir de la description d'une grammaire de formes. Le mécanisme d'analyse de formes impose une organisation des primitives de formes selon la grammaire.

La plupart des méthodes avancées pour l'analyse des formes se sont beaucoup intéressées au modèle, tandis que les algorithmes d'analyse correspondants ont été choisis ad hoc à partir du point de vue des grammaires classiques ; par exemple, You et Fu [14] ont utilisé l'agorithme de Earley. Le mécanisme d'analyse syntaxique a été souvent construit manuellement. Enfin, dans la plupart des formalismes, le processus suivant lequel on produit une grammaire de formes est très complexe pour n'importe quelle classe intéressante. Le problème se divise en deux.

(1) la construction des langages simplifiant la description d'une grammaire de formes,

et

(2) la construction de compilateurs efficaces acceptant ces langages.

Pour donner un exemple de cette approche à l'analyse syntaxique des formes, nous proposons une classe de grammaires de formes et une méthode avec laquelle la dérivation du mécanisme d'analyse syntaxique est automatique. En particulier nous définissons :

(1) un formalisme grammatical qui facilite la description de la plupart des formes à 2-D,

(2) une méthode pour calculer automatiquement les contraintes entre les symboles du vocabulaire d'une grammaire

(3) un mécanisme qui utilise les contraintes pour reconnaitre une forme.

Ce processus peut être considéré comme généralisation des grammaires de précédence parce qu'elles utilisent aussi les contraintes entre les symboles. Avec les grammaires classiques, une analyse ascendante est effectuée par la recherche d'un "handle". On obtient de bons résultats en ce qui concerne les grammaires classiques, mais les grammaires de formes posent beaucoup de problèmes à cause des relations qui peuvent être très compliquées entre les symboles.

II. - GRAMMAIRES DE FORMES

De nombreux chercheurs se sont intéressés aux modèles syntaxiques de formes. En opérant quelques modifications, ces modèles peuvent être intégrés d'une façon naturelle en utilisant les contraintes de contexte. Une extension des grammaires géométriques de Vamos [13] sera utlisée pour modéliser les formes.

Une grammaire à contexte libre stratifiée (G.C.L.S.), G, est un quadruplet (T,N,P,S) où T est l'ensemble des terminaux ; N est l'ensemble des non-terminaux ; P est l'ensemble des règles de production et S est le symblol de départ. A chaque symbole $v \in V = (N \cup T)$, on associe un numéro de niveau,
$\ell n(v) : V \rightarrow \{0,1,\ldots,n\}$, où $\ell n(S) = n$ et $\forall\, v \in T$, $\ell n(v) = 0$: De plus, $\forall\, v \in V$ v = <nom> {les points d'attaches} [la partie sémantique], où <nom> est le nom unique du symbole, {les points d'attaches} est un ensemble de positions particulières du symbole où d'autres symboles peuvent être attachés, et où [la partie sémantique] est un ensemble de propriétés du symbole. Les règles de production ont la forme

$(v := v_1, v_2 \ldots v_k, A,C,G_a,G_s)$, où v est composé des symboles

$v_i, i = 1,k$, et où $v \in N$, $v_i \in V$ et $ln(V_i) = ln(v) - 1$.

A et C sont les conditions de consistance entre les éléments respectifs d'attaches et de sémantique. Ces conditions décrivent d'une part comment les symboles doivent être placés et d'autre part donnent les relations existant entre leurs propriétés.

G_a et G_s sont les règles suivant lesquelles on construit {les points d'attaches } et [la partie sémantique] de v, respectivement. C'est-à-dire qu'il faut expliquer comment on peut calculer les propriétés d'un nouveau symbole.

Pour des explications détaillées sur ces grammaires, voir Henderson et Davis [7]. Les G.C.L.S. donnent de nombreuses contraintes contextuelles entre les symboles. Celles-ci correspondent au contraintes

d'organisation d'une forme. Le problème est d'extraire ces contraintes d'une grammaire et de les utiliser pour reconnaître une forme.

III. - LES CONTRAINTES

Deux sortes de contraintes sont utlisées : l'une syntaxique et l'autre sémantique. Etant donné un symbole, les contraintes syntaxiques de ce symbole consistent en un ensemble de symboles qui peuvent être rattachés à ce symbole, ou plus précisemment, les voisins possibles de ce symbole. Si le voisin d'un symbole n'est pas dans cet ensemble, alors il n'existe pas de dérivation du symbole de départ. Les contraintes sémantiques sont des relations entre les propriétés de deux symboles du vocabulaire. Par exemple, soit

v = < v > {a,b} [s] et w = <w> {c,d} [t] ; il se peut que s soit parallèle à t dans toutes les dérivations du symbole de départ. Il est possible de définir les contraintes syntaxiques et sémantiques de manière à les calculer automatiquement.
Soit G = (T,N,P,S) et v, w et x $\in$ V ; at (v) dénote les points d'attaches de v, et av est un élément quelconque de at (v). Les contraintes syntaxiques sont définies en donnant pour chaque symbole v un ensemble de voisins. En général, une définition des voisins d'un symbole est une fonction du point d'attaches. Définissons :

(1) (v,av) Ancêtre (w,aw) si et seulement si
$\exists p = (R,A,C,G_a,G_s) \in P$ où R est v:=...w... et $\exists aw \in at(w) \ni$ aw est connecté avec av dans G_a de p. On dit que v est un ancêtre de w à travers le point d'attaches de v et aw de w, où av et aw correspondent au même emplacement physique.

(2) (w,aw) Descendant (v,av) ssi (v,av) Ancêtre (w,aw)

(3) (v,av) Voisin (w,aw) ssi

(a) $\exists p = (R,A,C,G_a,G_s) \in P \ni$ R est x:= ...v...w... et aw est connecté avec av dans A de p, ou

(b) $\exists x \in V$ avec $ax \in at(x)$ et $\exists y \in V$ avec $ay \in at(y) \ni$ (x,ax) Ancêtre (v,av) et (y,ay) Voisin (x,ax) et (w,aw) Descendant (y,ay).

On peut facilement calculer ces relations avec une représentation de matrices binaires (voir Aho pour une introduction à ce sujet). Etant donné s, le nombre de symboles dans V, soit la matrice booléenne, Amn, carrée et d'ordre s, l'élément Amn (i,j) est égal à 1 ssi le symbole v_i est en relation A au symbole v_j à travers le point d'attaches m de v_i et n de v_j (on peut considérer que les point d'attaches sont ordonnés). Une relation (qui dépend des points d'attaches) est tout à fait spécifiée par k^2 matrices, où k est le nombre de points d'attaches par symbole. Cependant, si tous les symboles sont symétriques, une seule matrice définit une relation, parce que toutes les k^2 matrices sont les mêmes.

La relation Ancêtre, Amn, est la fermeture transitive d'une matrice booléenne qui correspond au départ à la définition donnée. La relation Descendant, Dmn, est la transposée de Amn. Etant donné Amn et Dmn, la relation Voisine, Nmn, est calculée de cette manière :

$$N_{mn} : N_{mn} + \Sigma_p\{D_{mp} * [\Sigma_q N_{pq} * A_{qn})]\}$$

où + est la fonction booléenne "ou" et * est la fonction "et", et les deux N_{ij} à droite sont les voisins explicites dans les règles de production.

Les contraintes sémantiques peuvent être calculées de la même façon : on définit une relation binaire et on calcule la fermeture transitive, par exemple, les deux orientations de deux symboles sont parallèles si c'est explicitement donné dans une règle de production, ou bien, par transitivité par rapport au troisième symbole. Les lignes et colonnes de la matrice représentant cette relation sont les vecteurs d'orientation associés à chaque symbole.

En général, une relation transitive est donnée par :

$$P := (P(0) + \sim I) * P(0)!$$

où ~I est le complément de la matrice d'identité et P(0)! est la fermeture transitive de P(0), où P(0) est la relation explicitement donnée dans les règles de production. Cette relation n'est pas forcément réflexive ; un symbole, v, n'est pas considéré parallèle à lui-même sauf au cas ou il faut que v apparaisse deux fois dans une forme, et que les orientations des deux v soit perpendiculaires, doivent être calculées avec des algorithmes spéciaux.

IV. - PROCESSUS HIERARCHIQUE DE CONTRAINTES

Nous avons déjà constaté que l'interprétation des primitives est ambiguë. Dans ce cas, l'analyseur doit non seulement résoudre l'analyse syntaxique des primitives données, mais, en plus, déterminer l'interprétation de chaque primitive. Le processus hiérarchique de contraintes (PHC) a été proposé pour résoudre ce problème (voir Davis [6].)PHC :

(1) construit un ensemble, Ip, d'interprétations possibles pour chaque primitive p ;

(2) construit un réseau d'hypothèses initiales, c'est-à-dire que pour chaque élément i_p de I_p, un noeud, $N(i_p)$, est créé dans le réseau ; deux noeuds, $N(i_p)$ et $N(i_{p'})$ sont liés si p et p' sont voisins ;

(3) applique les 3 procédures CONSTRUIRE, CONTRAINDRE et COMPRIMER au réseau d'hypothèse jusqu'à ce que le réseau soit vide ou jusqu'à ce que le symbole de départ soit construit.

Les primitives sont déterminées en calculant plusieurs approximations linéaires à partir du contour de la forme. Une modification de l'algorithme de "Split-and-merge" de Pavlidis [10] calcule les segments de droites en utilisant une mesure de grandeur d'angle proposée par Freeman et Davis [4].

Les ensembles I_p sont, en général, égaux à T, parce que tous les symboles terminaux sont a priori possibles. Cependant, on peut utiliser l'information globale pour réduire le nombre initiale de possibilités ; par exemple, on peut éliminer les hypothèses qui associent les terminaux les plus longs aux primitives les plus courtes. Le réseau

d'hypothèses représente toutes les chaînes de terminaux possibles pour les primitives données. Les grammaires définssent les contours simples et fermés, et chaque boucle dans le réseau représent un ensemble distinct d'hypothèses pour les primitives et doit être analysé. Il est évident que les boucles sont trop nombreuses pour être analysées l'une après l'autre.

Le PHC fait une analyse ascendante de toutes les boucles en parallèle. Cette analyse est accomplie suivant l'opération des 3 procédures sur 2 ensembles : le CONTRAINDRE-ENSEMBLE et le COMPRIMER-ENSEMBLE :

CONSTRUIRE - Etant donné le niveau k du réseau, CONSTRUIRE utilise les productions de la grammaire pour construire les noeuds de niveau k + 1. Les arcs sont connectés s'ils sont voisins. On garde l'information pour savoir quels noeuds ont été utilisés pour construire un nouveau noeud de niveau k + 1. Tous les noeuds de niveau k + 1 construits par CONSTRUIRE sont ajoutés à CONTRAINDRE-ENSEMBLE, et tout les noeuds de niveau k sont ajoutés à COMPRIMER-ENSEMBLE (les deux ensembles sont vides au commencement).

CONTRAINDRE - Tant que CONTRAINDRE-ENSEMBLE n'est pas vide, CONTRAINDRE examine chaque élément de l'ensemble ; si un noeud, n, ne satisfait pas les contraintes, alors ses voisins sont mis dans CONTRAINDRE-ENSEMBLE, tous les noeuds construits avec n et tous les noeuds qui ont été utilisés pour construire n sont mis dans COMPRIMER-ENSEMBLE, et n est éliminé du réseau.

COMPRIMER - Tant que COMPRIMER-ENSEMBLE n'est pas vide, COMPRIMER examine chaque élément ; étant donné un noeud n (de niveau k) de COMPRIMER-ENSEMBLE, si un des noeuds utilisé pour construire n n'est plus dans le réseau, ou si n n'a pas contribué à la construction d'un noeud (et le niveau k + 1 a été construit), dans ce cas, les voisins de n sont ajoutés à CONTRAINDRE-ENSEMBLE, tous les noeuds construits avec n et tous les noeuds qui ont été utilisés pour construire n sont mis dans COMPRIMER-ENSEMBLE, et n est éliminé du réseau.

V. - CONCLUSION

On analyse une classe de formes de la façon suivante :

(1) on définit une grammaire G.C.L.S. pour la classe de formes,

(2) on dérive les contraintes syntaxiques et sémantiques entre les symboles du vocabulaire de la grammaire,

(3) on applique le PHC au réseau d'hypothèse construit à partir d'un ensemble de primitives.

Voir Henderson et Davis [8] pour l'application de PHC à la classe d'avions.

Nous avons montré une approche à une théorie de l'analyse syntaxique des formes. On a essayé d'utiliser la théorie classique d'analyse syntaxique, mais l'ambiguité des primitives demande une autre

stratégie. Nous avons décrit un analyseur ascendant de contraintes et nous avons montré la relation entre l'analyseur et la théorie classique.

BIBLIOGRAPHIE

[1] Aho, S. et J.D. Ullman,
The theory of Parsing, Translation and Compiling,
Vol. 2, Prentice Hall, 1973.

[2] Davis, L.,
"Hierarchical Relaxation for Shape Analysis",
Pattern Recog. and Image Proc. Conf., Chicago, I1, 1978, pp 273-279.

[3] Deransart, P.,
"Synthèse Automatique de Traducteurs Définis par Attributs Sémantiques",
Congrès AFCET-IRIA Reconnaissance des Formes, 1979, pp. 111-120.

[4] Freeman, H. et L. Davis,
"A Corner-Finding Algorithm for Chain Coded Curves",
IEEE Trans. On Computers,
Vol. C-26, March, 1977, pp. 297-303.

[5] Fu, K.S. et B.K. Bhargava,
"Tree Systems for Syntactic Pattern Recognition",
IEEE Trans. on Computer, Vol. C-22, déc., 1973, pp. 1081-1099.

[6] Henderson, T.,
"Hierarchical Constraint Processes For Shape Analysis",
Ph. D. Dissertation, U. Of Texas, Dec., 1979.

[7] Henderson, T. et L. Davis,
"Hierarchical Models and Analysis of Shape",
BPRA 1980 Conf. On Pattern Recognition, Oxford, England, 1980, p. 79.

[8] Henderson, T. et L. Davis,
"Hierarchical Constraint Processes for Shape Analysis",
(to appear in IEEE Trans. on Pattern Analysis an Machine Intelligence, May 1981).

[9] Masini, G. R; Mohr,
"Un système de Reconnaissance de Dessins",
Congrès AFCET-IRIA Reconnaissance des Formes, 1978, pp. 107-114.

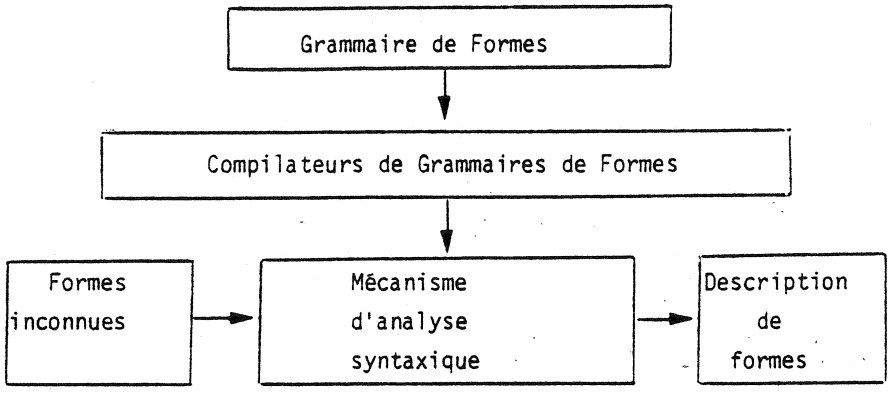
[10] Pavlidis. T. et S. Horowitz,
"Piecewise Approximation of Plane Curves",
Pattern Recognition, 1973, pp. 345-405.

[11] Rosenfeld, A. et D. Milgram,
"Web Antomata and Web Grammars",
Machine Intelligence
(eds. B. Meltzer and D. Michie), 7, Edinburgh U. Press, 1972,
pp. 307-324.

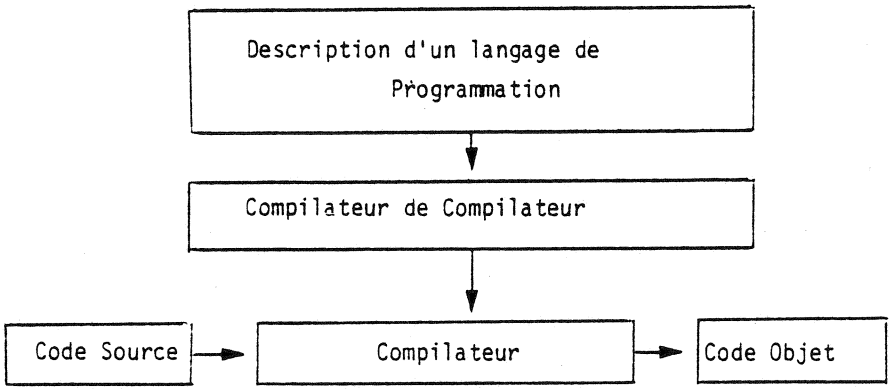
[12] Shaw, A.C.,
"A Formal Picture Description Scheme as a Basis for Picture
Processing Systems",
Information and Control, Vol. 14, 1969, pp. 9-52.

[13] Vamos, T. et Z. Vassy,
"Industrial Pattern Recognition Experiment - A Syntax Aided
Approach",
IJCPR, Wash, 1973, pp. 445-452.

[14] You, K. et K.S. Fu,
"Syntactic Shape Recognition",
in Image understanding and Information Extraction, Summary
Report of Research, March, 1977, pp. 72-83.



1a - Schéma des Grammaires de formes



1b - Schéma traditionnel

Figure 1 - Compilation des Grammaires de Formes

afcet

3ème Congrès

RECONNAISSANCE DES FORMES ET INTELLIGENCE ARTIFICIELLE

NANCY

16, 17, 18

Septembre 1981

●