using (5a) and (5b) and $S(\lambda_i)$'s were computed using (6). A prespecified number (n) of a (λ_i)'s and $b(\lambda_i)$'s were retained, normalized and quantized, using 5 bits for each coefficient. The reconstructed boundaries using $n = 32, 16, 8$, and 4 where $2n$ denotes the number of real (n in number) and imaginary coefficients retained are in Fig. 1(b)–(e). Fig. 1(b) is very similar to the original since 64 (real and imaginary) coefficients a retained; the small dissimilarity is due to the error introduced by the quantizer. The quality of reconstruction goes down as n is decreased. To get an idea of the compression involved consider the case when $n = 8$. There are totally, 16 transform coefficients and 13 parameters of CAR model to be stored. Since we have used 5 bits for each coefficients, we need 496 bits to store Fig. 1(d) (assuming that each parameter is of the CAR model represented by 32 bits) compared to $64 \times 32 = 2048$ bits required to store Fig. 1(a), the compression factor being close to 4.12.

IV. Conclusion

A mathematical technique has been given for transform coding of image boundaries using Fourier computations. The scheme given for univariate representation can be easily estended for multivariate representation given in Section II.

References

[1] D. N. Graham, "Image transmission by two-dimensional contour coding," *Proc. IEEE*, vol. 55, pp. 336–346, Mar. 1967.
[2] P. A. Wintz, "Transform picture coding," *Proc. IEEE*, vol. 60, pp. 809–820, July 1972.
[3] R. Bellman, *Introduction to Matrix Analysis.* New York: McGraw-Hill, 1960.
[4] R. L. Kashyap and R. Chellappa, "Stochastic models for closed boundary analysis: Representation and reconstruction," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 627–637, Sept. 1981.
[5] L. H. Koopmans, *The Spectral Analysis of Time Series.* New York: Academic, 1973.
[6] J. Max, "Quantization for minimum distortion," *IRE Trans. Inform. Theory*, vol. IT-6, pp. 7–12, Mar. 1960.

A Syntactic Approach for Handwritten Mathematical Formula Recognition

ABDELWAHEB BELAID AND JEAN-PAUL HATON

Abstact—**Mathematical formulas are good examples of two-dimensional patterns as well as pictures or graphics.**

The use of syntactic methods is useful for interpreting such complex patterns. In this paper we propose a system for the interpretation of 2-D mathematic formulas based on a syntactic parser. This system is able to recognize a large class of 2-D mathematic formulas written on a graphic tablet. It starts the parsing by localization of the "principal" operator in the formula and attempts to partition it into subexpressions which are similarly analyzed by looking for a starting character. The generalized parser used in the system has been developed in our group for continuous speech recognition and picture interpretation.

Manuscript received July 22, 1982; revised March 15, 1983.

The authors are with the Pattern Recognition and Artificial Intelligence Group, C.R.I.N., University of Nancy 1, 54506 Vandoeuvre, France.

Index Terms—**Graphic tablet, hand-drawing segmentation, handwritten characters, mathematic formulas interpretation, primitives, structural recognition.**

I. Introduction

Syntactic pattern recognition methods have been used in many cases where complex patterns can be described as assembly of subpatterns [7], [14].

In this case the decision processor is no longer a classifier which works on a set of pertinent measures extracted from the pattern and projected in a decision space. It is rather an interpreter which is able to perceive the structural organization of the complex pattern and to analyze it.

Mathematics formulas represent a good example for the application of such syntactic techniques. The formula interpreter works here in conjunction with a character recognition system using a structural approach [2]: the character is isolated in the formula text, then each of its drawings is segmented into basic primitives. From results obtained at this level by comparing there primitives to the ones located in a decision tree and from a formula description model (i.e., a coordinate grammar such as Martin's [10]), the interpreter selects the list of characters that gives a good matching with the reference pattern.

The top-down or bottom-up parsing schemes from left to right or right to left commonly used for linear strings (natural language, programming languages, $\cdots$) are not adapted to our case since the notion of end points is not clear. We have developed [8] a generalized parsing algorithm starting from any point within a pattern. The choice of the starting point is function of its importance in the grammar and of its recognition score. This algorithm was tested in the Mirabelle system for the recognition of sketch patterns [12] and adapted to connected speech recognition [11]. In this paper we present its application to the recognition of 2-D mathematical formulas written on a graphic tablet with a view to realize a computer-assisted teaching system. Moreover, this application constitutes an excellent frame for testing our models in character recognition and in the use of the context [16].

Many researchers have taken an interest in character recognition and several methods have been proposed and many systems built during the past few years in the field of character recognition. On the contrary, to our knowledge, few people have been interested in the recognition of handwritten 2-D mathematics formulas. Anderson proposed in 1968 a solution to this problem [1]. His recognition algorithm was first implemented in 1968 on the CTSS time-sharing system at MIT. Anderson used a top-down parsing scheme to partition a two-dimensional character configuration into subproblems. Like in our system, the characters are entered from a graphic tablet and recognized by a character recognition algorithm. It starts with the ultimate syntactic goal and attempts to partition the problem into subgoals until either every subgoal is reached or all possibilities have failed. The algorithm is also syntax directed and aided by a coordinate grammar. Each production maps a set of symbols, located at given coordinates, into a new set the coordinates of which are given by a set of functions associated with the given production.

In Section II we will describe the hierarchy of parsers in our system with the function of each of them as well as the interaction between them.

The kind of formula patterns will be described in Section III. We will then show, through a set of examples, the use of con-

level	form	structure	operation
I	point		acquisition
II	segment		sampling
III	feature		segmentation
IV	character		recognition
V	formula	$\int_a^b y\,d\,y$	interpretation

Fig. 1. Processing levels of the system.

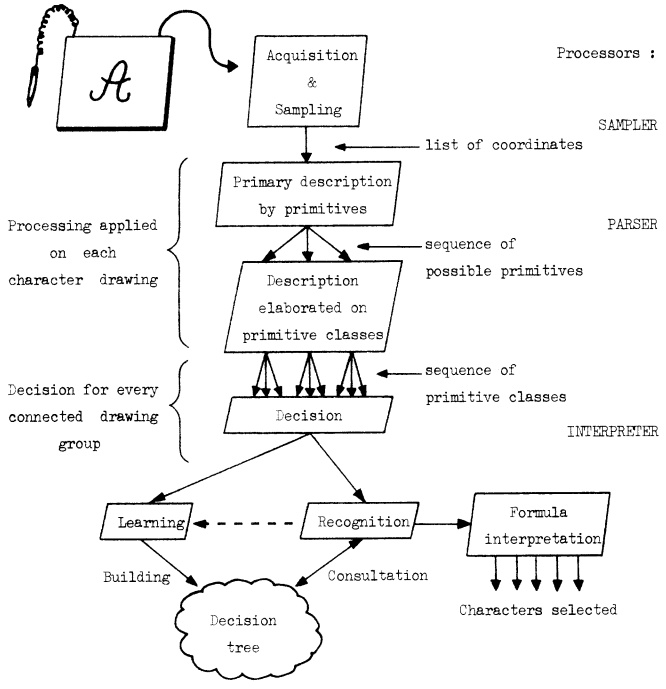


Fig. 2. General structure of the recognition system.

text in order to improve the interpretation and solve the ambiguities.

II. System Overview

The system that we have realized is made up of a sequence of five processing levels corresponding to five elements, respectively called: point, segment, feature, character, and formula. To each of these levels is attached a transformation function applied to the pattern structure as shown in Fig. 1. These levels are called at intervals in the system and their hierarchy well describes the evolution from the point, which is the basic information, up to formula [15].

Fig. 2 shows the general system organization. The system is implemented on a minicomputer SEMS MITRA 125.

The various processing levels will now be described in detail.

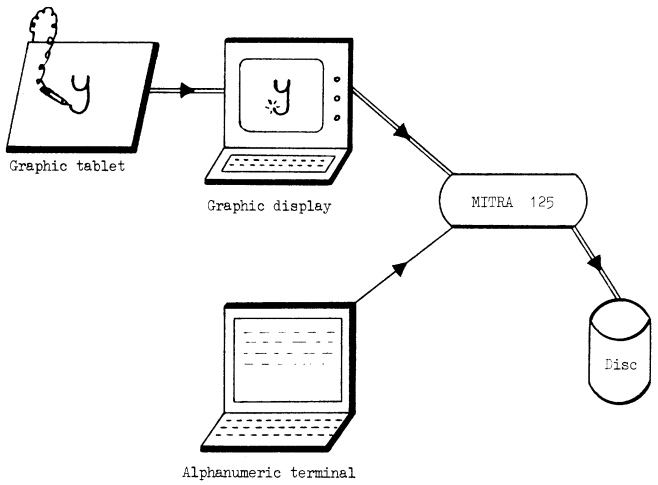


Fig. 3. Overview of the acquisition system.

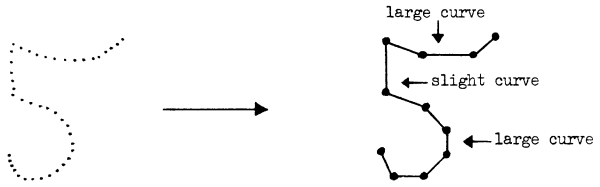


Fig. 4. Curves associated with digit 5.

III. Acquisition and Sampling

This paragraph presents the acquisition process together with the operation of sampling which represents the first transformation applied to the set of coordinates given by the acquisition.

A. Acquisition

Fig. 3 shows the interactive acquisition facility used in our group for graphic processing.

The character is drawn on a graphic tablet and transmitted to the computer under the form of a sequence of point coordinates and indications on pen's risings. This process presents, with regard to other acquisition processes (optical reading, . . .), the advantage of faithfully restoring the sense of the outline and the order in which the different parts of the form have been drawn as shown in [3]. This information is important in the framework of the structural approach we have chosen.

B. Drawing Description

The drawing obtained from the tablet contains an important number of points. We are only interested in a small number of them necessary for distinguishing between the outline elementary patterns such as straight lines (slight curve) and arcs (large curve). In the following paragraphs, we will describe in what manner we extract these elementary patterns from a few selected points in the drawing. Fig. 4 illustrates this principle in the sketch in digit 5.

Local deformations due to the operator or to the graphic tablet are frequent. This makes it very difficult to carry out the extraction of representative points. The deformation made by the operator appear in the form of zigzags or wire-edges introduced at the end of the sketch; the tablet, moreover, introduces parasite points in the drawing.

C. Sampling

The technique used by the sampler is similar to the one developed in [4] and tested on characters of different sizes. The basic principle was described in [9]: the algorithm starts from the beginning of the drawing and keeps only the points which are within a tolerance predetermined region.

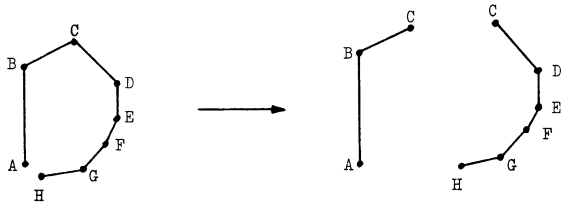
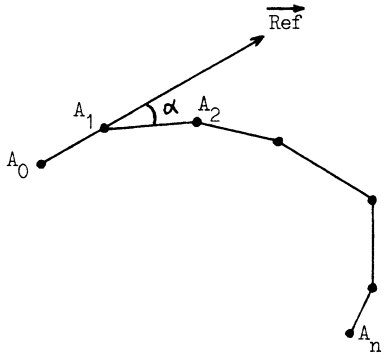


Fig. 5. Decomposition of character D into two homogeneous zones.

We complete, in our method, this angle criterium by a relative length one in order to eliminate interference dots and the wire-edges at the ends of outlines [2].

Let us consider a drawing made up of n points $A_0, A_1, \cdots, A_n$. The sampling algorithm is as follows:

$X0 \leftarrow A0$; $X1 \leftarrow A1$; $X2 \leftarrow A2$; $\overrightarrow{\text{Ref}} = \overrightarrow{X0\ X1}$
FOR $I := 2$ TO $N - 1$ DO
BEGIN $1 \leftarrow$ length $(\overrightarrow{X0\ X1})$
$\alpha \leftarrow$ angle $(\overrightarrow{\text{Ref}}, \overrightarrow{X1\ X2})$
IF $\alpha >$ SANGLE AND $1 >$ SLONG
THEN % the intermediate point is kept %
$X0 \leftarrow X1$; $X1 \leftarrow X2$; $X2 \leftarrow Ai + 1$; $\overrightarrow{\text{Ref}} = \overrightarrow{X0\ X1}$
ELSE % intermediate point is discarded %
$X1 \leftarrow X2$; $X2 \leftarrow Ai + 1$
discard $(Ai - 1)$ % in the entry list %
END



SANGLE is an angle threshold (it was fixed at 25° in our application); SLONG is a length threshold (it was fixed at 1/20 of the drawing length).

D. Component Extraction

In our system a character is split into a sequence of features like in [4] or [13]. In order to take into account the previous definition the system looks for homogeneous parts composed of strokes making angles between them with a same sign and with an absolute value lower than a given threshold. For instance, the character in Fig. 5 will be decomposed into two zones ABC, $CDEFGH$ on the basis of the significant curvature found at point C.

Each zone is then separately segmented from local considerations on the measures of angle and length of their consecutive strokes. Several possible solutions will arise for the first zone of the character in Fig. 5. It should be possible for instance to segment it into only one straight line although it is composed of two strokes.

In fact two problems appear. From what angle and length boundaries can we attest that a straight line does not belong to an arc? And, how can we distinguish between two characters which have the same sequence of components?

For the first problem, we introduce tolerance intervals in order to allow the system to give all possible solutions for the same zone. Each user can adjust these intervals according to the case.

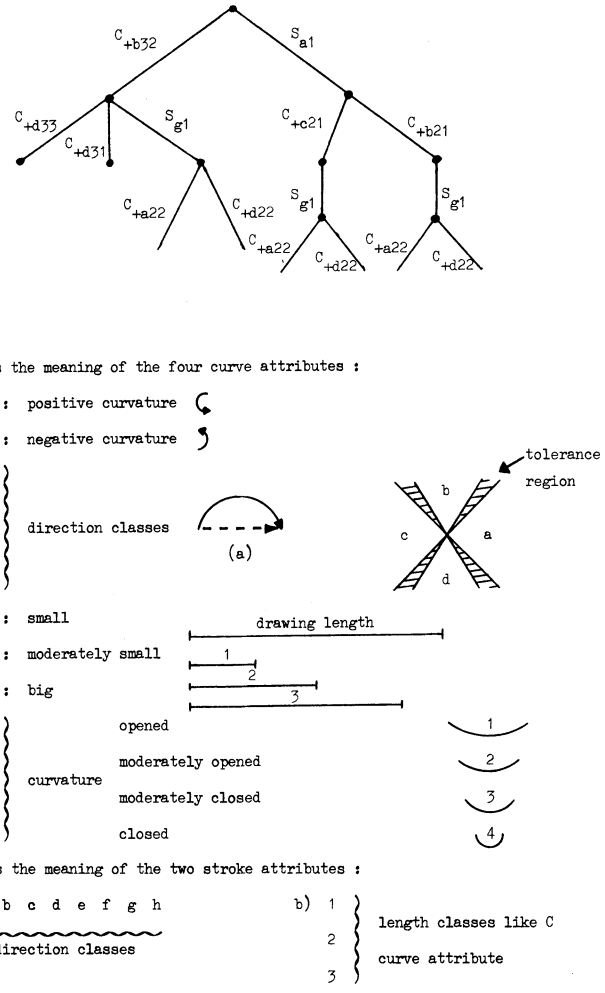


Fig. 6. Example of tree segmentation representation of character C.

For the second question, we reinforce the features extracted by topographic attributes corresponding to their morphology (relative length with regard to the character size, direction from Freeman code, angular variations). There too, many possible attributes are admitted for the same feature. At the end of this segmentation the character will be represented by a tree of several possible features as illustrated in Fig. 6. Each path in the tree describes a possible sequence. A segmentation score is given for each of these solutions. This multiple segmentation is very useful in our approach. It reinforces the interaction between the extraction and recognition processes by proposing as many solutions as possible for segmentation to the recognizer and allows a safe recognition.

IV. Character Recognition

The role of the character recognition module consists of comparing the sequence of primitive classes representing an unknown character to the reference sequences stored during the learning phase in a decision tree. It should be noticed that since the segmentation is carried out completely independently of the recognition it can be used for other graphic processing applications such as sketch interpretation. In fact several different characters can still have identical representations at this level as illustrated in Fig. 7. This means that the recognition process has to take into account other types of information in order to solve these ambiguities.

Except for the last two cases, all the characters can be separated by looking at the position of the crossing point and the extremities of the character. Therefore, whenever these am-

Codes	Ambiguous characters
$S_{a3} + L + S_{g3}$	(+) (T) (L)
$S_{g3} + L + S_{a3} + S_{a3}$	(F) (I)
$S_{g3} + L + C_{g33}$	(D) (P) (b)
$S_{g3} + L + S_{h3}$	(X) (Y) (V)
$S_{g3} + L + C_{-g23} + S_{h3}$	(R) (k)
$S_{g3} + L + S_{a3} + L + S_{g3}$	(H) (4) (U)
$S_{g3} + L + S_{h3} + L + S_{a3}$	(A) (H) (*)
C_{-d33}	(c)
$S_{a2} + C_{-a33}$	(h) (n)

. The capital letters designate primitive patterns C = curve S = segment
. The subscripts in the codes designate names of classes attributes (Ref. fig. 6)
. L designates a pen's rising.

Fig. 7. List of ambiguous primitive descriptions of characters.

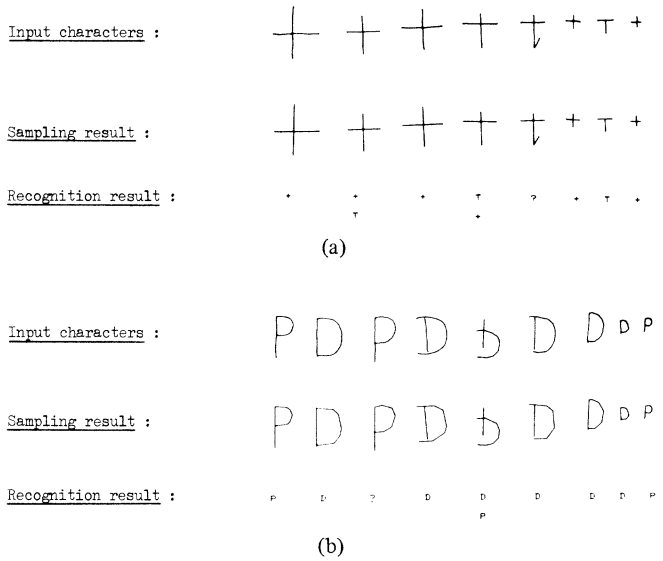


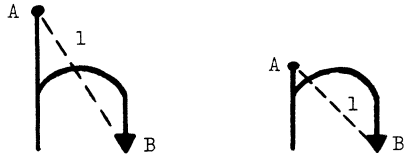
Fig. 8. (a) Example of character recognition. (b) Example of character recognition.

biguities appear in the decision tree a test is introduced to separate the two ambiguous classes. Fig. 8 shows two practical examples of character recognition. Fig. 8(a) shows the results obtained from a sequence of "*T*" and "+" characters with different positions of the crossing point. Only one bad recognition for the character which has a wire-edge was found. Fig. 8(b) gives the answers to a sequence of "*P*" and "*D*" letters. Only one ambiguity remains.

The system was tested with 35 different characters (digits, letters, and mathematical characters). These characters are written in no special way. One hundred acquisitions were made for each character class during the learning phase, allowing us to define up to five different reference patterns for a class. The results obtained are the following:

- recognition rate: 93 percent
- confusion rate: 2 percent
- rejection rate: 5 percent.

The important rate of rejection essentially comes from characters "*h*" and "*n*" (0.7 percent).



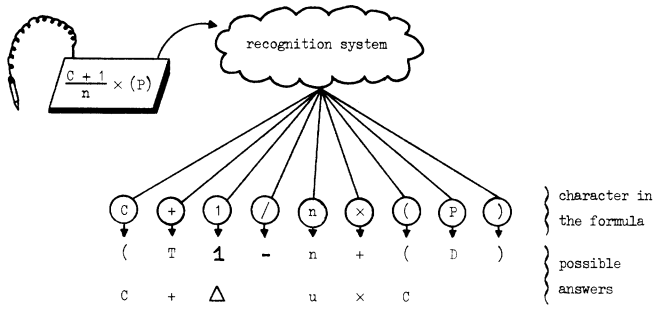


Fig. 9. Ambiguities introduced by the character recognizer.

For such characters composed of only one drawing, the distinction is made by comparing the distance which separates both extremities of the drawing to the length of the frame which contains the character. In the case of "*h*" and "*n*," this heuristics is not sufficient.

As a matter of fact the accuracy of our character recognizer could certainly be improved by considering new features. However, our goal was not to design a highly sophisticated character recognition system but to study the use of *a priori* and contextual information in the frame work of an hierarchical interpretation system. The use of such information makes it possible to a certain extent to solve ambiguities introduced by the character recognition level and similar to the ones appearing in Fig. 9. The way in which the interpretation of a mathematical formula is carried out from the results of the recognition will be presented in the next section.

V. Mathematical Expression Interpretation

A. General Presentation

We have seen that the structural character recognition method described previously yields erroneous strings of character. The role of the interpretation level is to allow for an increase in the recognition rate by taking into account contextual information. This process makes it possible to obtain a very good accuracy in the recognition of mathematical expressions even though the character recognition rate is not very high.

A mathematical notation can be considered as a coherent combination of operators and operands connected together by their positions and by relationships. The analysis of a formula consists of locating the subpatterns which compose it and finding relations which bind them together.

B. Analysis Method

Our interpreter is composed of two syntactic parser (top-down and bottom-up) as in the general algorithm described in [8]. It starts the analysis from a priority operator in the expression to be analyzed and tries to divide it into subexpressions or operands which are then analyzed in the same manner and so on until identificators. If the zone explored by the interpreter is the goal zone, then the analysis is terminated; otherwise, it is reiterated on the nonterminal found and so on until the final goal.

The bottom-up parser chooses from the starting character and from the neighboring subexpressions the corresponding rule in the grammar. This rule gives instructions to the top-down parser to delimit the zones of neighboring operands and operators.

C. Analysis Procedure

The analysis procedure is the following (*C*1 and *C*2 are, respectively, the left and right contexts as given by the operator):

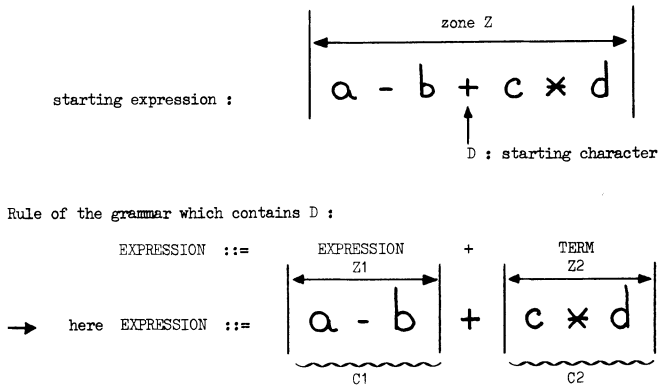


Fig. 10. First analysis of the expression $a - b + c * d$.

PROCEDURE ANALYSIS (Z, GOAL, D): tree(t);
% D is the starting character; z is the zone (space) of the starting expression %
WHILE $D \neq$ GOAL DO
BEGIN choose a rule $B ::= C1\ D\ C2$;
delimit the zone $Z1$ of $C1$;
delimit the zone $Z2$ of $C2$;
detect $D1$ in $Z1$, $D2$ in $Z2$
construct $t1$ by ANALYSIS($Z1$, $C1$, $D1$)
construct $t2$ by ANALYSIS($Z2$, $C2$, $D2$)
$t \leftarrow B \times (t1 + D + t2)$
% the operator $\times$ means that the left term B is the root of tree composed by the right term %
$D \leftarrow B$
END
result T
END ANALYSIS

A terminal D is first located in the formula according to its position and importance in this formula. We will come back to this problem later. (D is not necessarily the beginning or the end of the mathematics notation. In the example of Fig. 10, D is the operator "+" detected in the formula.) Then, we look for the syntactic rule which contains D (the rule found is under the form $B ::= C1\ D\ C2$). We then define the zones $Z1$ of $C1$ and $Z2$ of $C2$ in the space Z of the mathematical notation by looking at the context. $C1$ and $C2$ are then analyzed in the same manner than the first expression. If the resulting B is not the axiom of the grammar, the analysis is iterated on B ($D \leftarrow B$) until description of all terms of the notation. In the example, B is "expression" like the axiom of the grammar but its zone does not cover all the space of the starting expression.

D. Mathematics Notation Pattern Description

The description language we use is valid for a large number of formulas containing variables (reduced to one letter), numerical constants, linear arithmetic expressions, trigonometric functions, and fractions.

It is of course possible to integrate new operators such as exponential, integral, etc., . . . as Anderson did in his system [1]. It is obvious that the resulting system would be far more complicated than ours.

The characters of the formula are presented to the interpreter with the following:

- Name ($A, B, +, *, \cdots$).
- Position in the formula:

  entry order ($1, 2, \cdots, n$) (writing order)

  position in the answer list (the characters are ordered according to the recognition score).

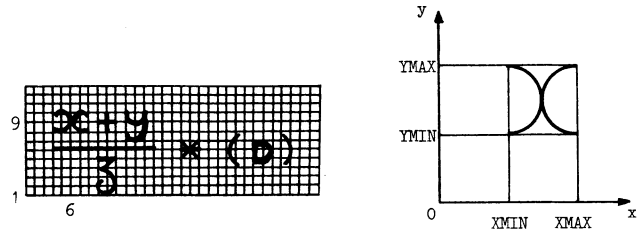


Fig. 11. Formula representation.

EXPRESSION → TERM / 1 / EXPRESSION + TERM / 2 /
EXPRESSION - TERM / 3 /
TERM → FACTOR / 4 / TERM * FACTOR / 5 /
FACTOR → -PRIMARY / 6 / PRIMARY / 7 /
PRIMARY → $\frac{\text{EXPRESSION}}{\text{EXPRESSION}}$ / 8 / PR-SIMPLE / 9 /
PR-SIMPLE → IDENTIFIER / 10 / FUNCTION / 11 /
NUMBER / 2 / EP.PARENTHESEID / 13 /
IDENTIFIER → LETTER / 14 /
FUNCTION → NAME EP.PARENTHESEID / 5 /
NAME → SIN / 16 / COS / 7 / TAN / 18 /
NUMBER → NUMBER DIGIT / 19 / DIGIT / 20 /
EP-PARENTHESEID → (EXPRESSION) / 21 /

Fig. 12. The grammar of mathematical formulas.

- Position in the tablet space:

  Frame: XMIN, YMIN, XMAX, YMAX.

- Position in the formula space:

  Center: Xcenter, Ycenter.

The character center is related to a squaring realized on the space formula in order to have a good precision on the character position. Xcenter and Ycenter will be given by the nearest horizontal and vertical lines of the actual center of the character:

$$\frac{\text{XMIN} + \text{XMAX}}{2}, \quad \frac{\text{YMIN} + \text{YMAX}}{2}.$$

Fig. 11 gives an example.

E. Description Grammar

The formula grammar we use can be formally defined as a 4-tuple:

$$G = (T, N, P, S)$$

where

T is a set of terminal symbols:

$$A, B, \cdots, Z, a, b, \cdots, z, +, -, *, /, (,)$$

N is a set of nonterminal symbols: $T \cap N = 0$

P is a finite set of productions

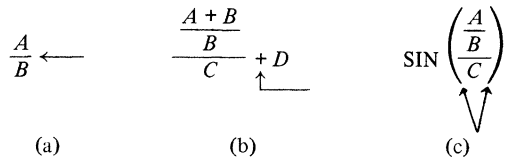
S is the grammar axiom ($S \in T$).

This grammar is composed of context-free rewriting rules as indicated in Fig. 12 (a similar formulation can be found in [17]).

Here are some examples of formulas generated by the grammar:

$$A + B * C - D$$

$$A + \frac{C * D}{E - C} - \frac{B - \dfrac{C}{\text{SIN}(A + B)} * \left(\dfrac{A}{B} - \dfrac{C}{D}\right)}{\dfrac{C * \text{TAN}(-A)}{2 + \text{COS}(C)}}.$$



(a) (b) (c)

Fig. 13. Examples of interesting starting character.

F. Analysis of Ambiguities Introduced at the Recognition Level

These ambiguities can be of three types which will now be illustrated by some examples.

1) *Multiple Answers:* In example a), the syntax of the trigonometric expression enables us to choose the letter C in the first list and the parenthesis in the second. In example b) the symbol + has been chosen since a variable is reduced to only one letter.

a) $C \rightarrow \begin{matrix} C & \underline{\mathrm{COS}} & \underline{(} & \underline{A} & \underline{+} & \underline{B} & \underline{)} \\ (& : & C & & C & & \\ & & (& & (& & \end{matrix}$

b) $+ \rightarrow \begin{matrix} L & \underline{T} & \underline{+} & \underline{T} \\ T & : & T & \\ + & & + & \end{matrix}$

2) *Misrecognition:* In example c) the opening parenthesis will be cancelled because the corresponding closed parenthesis does not appear. In that case a backtracking to the recognition level will be necessary.

c) $C \rightarrow (\begin{matrix} \underline{T} & \underline{+} & \underline{T} & \underline{+} & \underline{C} & \underline{+} & \underline{D} \\ (& & & & & & \end{matrix}$

3) *Confusion:* A decision is impossible in the two examples d) an e) because both characters are likely to appear in the formula. An interaction with the user is necessary to solve this ambiguity.

d) $X \rightarrow \begin{matrix} + & \underline{T} & \underline{+} & \underline{T} \\ \times & & + & \\ & & \times & \end{matrix}$

e) $S \rightarrow \begin{matrix} 5 & \underline{S} & \underline{+} & \underline{T} \\ \mathrm{S} & S & & \\ & 5 & & \end{matrix}$.

4) *Fraction Line:* Another typical example of confusion is between the fraction line and the minus sign. The distinction between them can only be made according to the context [i.e., presence or absence of expressions above and under the symbol, as illustrated in example f).]

f) $\dfrac{T+T}{C-D}$ contexts.

G. Starting Character Choice and Context Delimitation

The question is what is the best character we choose in the formula which gives the maximum information in order to facilitate the formula interpretation. Let us look at some examples given by Fig. 13. In Fig. 13(a) it is easy to choose the fraction line as a starting character. It gives enough information to localize the two expressions A and B. This is more interesting in Fig. 13(b) where the choice of the operator "+" allows one to fasten on the principal fraction line. It is easy, in Fig. 13(c), to detect from parentheses the name of the trigonometric expression and the principal fraction line of the expression between parentheses.

We choose the starting character among the arithmetic operators as (+, −, ∗) and the mathematical symbols as (/, ( )) due to their important connection role. The priority order observed is the following: parentheses ( ); +; −; ∗; -unary; fraction line. If the expression to analyze contains any of these operators, it comes down to a variable or to a constant and its analysis is obvious. This carving order allows one to respect the expression structure according to the syntactic description. In Fig. 13(b), we choose as "central" operator the last "+." The situation of this operator allows us to directly determine the principal fraction line of the left expression.

1) $T+P$
2) $A+C*T-P$
3) $1+A$
4) $\dfrac{P}{A}-\dfrac{A}{P}$
5) $((((C)-P)-A)-P)$
6) $\dfrac{A}{C-\left(\dfrac{N}{C}\right)}$
7) $C-\dfrac{\dfrac{A}{C}+\dfrac{T}{N}}{\dfrac{C}{T}}$
8) $C+\mathrm{TAN}\,(A)$

Fig. 14. List of formulas used during the test.

Formula No.	Well Recognized	Confused	Rejected
1	40	0	0
2	40	0	0
3	40	0	0
4	40	0	0
5	36	0	4
6	40	0	0
7	38	2	0
8	40	0	0

Fig. 15. Interpretation results (10 writers, 4 productions for each writer). The confusions and errors which appeared in the test correspond to confusions between characters A and ∗, and characters T and +.

VI. Experimental Results

The system has been implemented in the framework of a computer-assisted teaching system. Our goal was to evaluate the interest of a handwritten input as a natural means for the student to write mathematical expressions.

In order to check the efficiency and robustness of the system a set of eight formulas has been selected. This set is given in Fig. 14. Ten different writers (six male and four female) were asked to write each formula four times without any preliminary training of the character recognition system. The total number of characters in the dictionary is 30. The overall recognition results were 98.1 percent good formula interpretation, 1.3 percent rejection and 0.6 percent confusion.

More detailed results are given in Fig. 15. The recognition scores for individual writers are ranged from 95 to 100 percent.

VII. Conclusion

We have realized a formula interpreter based on a structural approach which uses the maximum of contextual information at its three main levels.

At the Preprocessing Level: A primitive extraction automaton yields two kinds of important primitives: straight lines and arcs. This automaton extracts both local and global primitives

by using a limited number of simple tests. These tests are introduced by the operator and may be adjusted on any kind of drawing. The morphology of the drawing is taken into account in order to find all the likely segmentation possibilities.

At the Character Level: An efficient learning process allows the decision procedure a simple and rapid search. A decision tree is constructed automatically and completed during the recognition. Tests are introduced to solve possible ambiguities between similar characters. These tests use global measures such as the relative position of the drawing crossing points in relation to its extremities.

At the Formula Level: The mathematical formulas have provided a good environment for testing our ideas both on character recognition and contextual interpretation. We have used a mixed top-down bottom-up method controlled by a coordinate grammar associated with our formal description. The interpretation result is a syntactic tree with topographic links between operators and operands.

The overall system has shown its efficiency and robustness on a number of practical handwritten formulas.

ACKNOWLEDGMENT

The authors would like to acknowledge the valuable participation of R. Mohr in this work. They also wish to thank C. Floc'h for having made every effort in preparing this paper.

REFERENCES

[1] R. H. Anderson, "Syntax-directed recognition of hand-printed two-dimensional mathematics," in *Interactive Systems for Experimental Applied Mathematics*, M. Klerer and J. Reinfelds, Ed. New York: Academic, 1968.

[2] A. Beláid, "Reconnaissance structurelle de caractères manuscrits et de formules mathématiques," 3rd cycle thesis, Univ. Nancy 1, 1979.

[3] A. Beláid and G. Masini, "Segmentation de tracés sur tablette graphique en vue de leur reconnaissance," *Tech. et Sci. Inform.*, vol. 1, no. 2, pp. 155–168, 1982.

[4] M. Berthod, "Une méthode syntaxique de reconnaissance de caractères manuscrits en temps réel avec apprentissage continu," 3rd cycle thesis, Univ. Paris 6, 1975.

[5] M. Berthod and P. Jancenne, "Le pré-traitement des tracés manuscrits sur une tablette graphique," presented at the Congr. AFCET, Reconnaissance des Formes et Intelligence Artificielle, Toulouse, Sept. 1979.

[6] M. Berthod and J. P. Maroy, "Morphological features and sequential information in real-time hand-printing recognition," presented at the 2nd IJCPR, Aug. 1974.

[7] K. S. Fu, *Syntactic Methods in Pattern Recognition*. New York: Academic, 1974.

[8] J. P. Haton and R. Mohr, "A new parsing algorithm for imperfect patterns and its application," presented at the 3rd IJCPR, San Diego, CA, Nov. 7–11, 1976.

[9] M. R. Ito and T. L. Chui, "On line computer recognition of proposed standard ANSI (USASI) hand-printed characters," *Pattern Recognition*, vol. 10, no. 5/6, pp. 341–349, 1978.

[10] W. A. Martin, "Computer input-output of mathematical expressions," in *Proc. ACM 2nd Symp. Symbolic and Algebraic Manipulation*, New York, 1971.

[11] J. F. Mari, "Contribution à l'analyse syntaxique et à la recherche lexicale en reconnaissance du discours continu," 3rd cycle thesis, Univ. Nancy 1, 1979.

[12] G. Masini, "Réalisation d'un système de reconnaissance structurelle et d'interprétation de dessins," 3rd cycle thesis, Univ. Nancy 1, 1978.

[13] R. Narashiman and V.S.N. Reddy, "A syntax-aided recognition scheme for handprinted English letters," *Pattern Recognition*, vol. 3, pp. 345–361, 1971.

[14] T. Pavlidis, "Structural pattern recognition: Primitives and juxtaposition relations," in *Frontiers of Pattern Recognition*, M. S. Watanabe, Ed. New York: Academic, 1972.

[15] T. Pavlidis and F. Ali, "A hierarchical syntactic shape analyzer," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, Jan. 1979.

[16] G. T. Toussaint, "The use of context in pattern recognition," *Pattern Recognition*, vol. 10, pp. 189–204, 1978.

[17] A. Aho and J. D. Ullman, *Principles of Compiler Designer*. Reading, MA: Addison-Wesley, 1977.

A Similarity Measure Between Patterns with Nonindependent Attributes

TETSURO ITO, YOSHIFUMI KODAMA, AND JUNICHI TOYODA

***Abstract*—A generalized version of a set-theoretical measure for obtaining similarities between patterns with nonindependent attributes is presented. The dependence here is given by the pairwise correlation. Since the proposed measure needs no assumption of attribute independence, the resulting similarity values can reflect directly the relationships between the attributes.**

***Index Terms*—Information retrieval, ordering, pattern classification, pattern matching, similarity measure.**

I. INTRODUCTION

One of the major interests in the study of pattern analysis is how to compute the similarity between any pair of patterns (e.g., pictures, strings of symbols, phoneme sequences, documents). Once the similarity is obtained, it becomes possible to determine the classes of the patterns, or to order the similar patterns according to their relational intensity. Various authors have proposed similarity functions for the purpose of pattern classification, speech understanding, database organization, etc. Among them a set-theoretical measure [1] has often been used because of its simple process of computing the number of common attributes to both and the total number of distinct ones possessed by either of the patterns. Findler and Leeuwen have settled in [2] a measure essentially equivalent to a set-theoretical one for the quantitative comparison of two strings.

These previously proposed measures, however, are valid in the case where the attributes of the patterns are so chosen that they are mutually exclusive or independent. The difficulty of choosing independent attributes is seen in [1]: sometimes it is required that a combination of attributes should be listed as itself an attribute. Salton [3] stated that the assumption about the independence appears true for about 70 percent of the index terms (i.e., attributes for specifying documents). Recently, Lam and Yu [4] have devised a file search method incorporating the term dependence.

In this paper, a new similarity measure S, a generalized version of a set-theoretical measure T, which takes pairwise correlation between attributes into account, is presented to compare the patterns with nonindependent attributes.[1] The discussion about the validity of the proposed formalism is based on the theoretical consideration and on the computational experiment.

Manuscript received July 27, 1979; revised November 29, 1982.

T. Ito is with the University of Library and Information Science, Ibaraki-ken 305, Japan.

Y. Kodama is with Nippon Electric Company, Ltd., Abiko-shi 270-11, Japan.

J. Toyoda is with Osaka University, Ibaraki-shi 567, Japan.

[1] Bonner [5] indicated briefly the necessity of such a measure.

0162-8828/84/0100-0111$01.00