Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition Letters 29 (2008) 647–655
Pattern Recognition Letters
www.elsevier.com/locate/patrec

Progressive structural analysis for dynamic recognition of on-line handwritten mathematical expressions

Ba-Quy Vuong [a], Siu-Cheung Hui [a], Yulan He [b,*]

[a] Computer Sciences Department, University of Wisconsin-Madison, Madison, WI 53706-1685, USA
[b] Informatics Research Centre, School of Business, The University of Reading, Reading RG6 6AY, UK

Received 1 July 2007; received in revised form 14 September 2007
Available online 15 December 2007

Communicated by M.-J. Li

Abstract

Structural analysis in handwritten mathematical expressions focuses on interpreting the recognized symbols using geometrical information such as relative sizes and positions of the symbols. Most existing approaches rely on hand-crafted grammar rules to identify semantic relationships among the recognized mathematical symbols. They could easily fail when writing errors occurred. Moreover, they assume the availability of the whole mathematical expression before being able to analyze the semantic information of the expression. To tackle these problems, we propose a progressive structural analysis (PSA) approach for dynamic recognition of handwritten mathematical expressions. The proposed PSA approach is able to provide analysis result immediately after each written input symbol. This has an advantage that users are able to detect any recognition errors immediately and correct only the mis-recognized symbols rather than the whole expression. Experiments conducted on 57 most commonly used mathematical expressions have shown that the PSA approach is able to achieve very good performance results.

Keywords: Progressive structural analysis; Structural analysis; Mathematical expressions recognition; Mathematical expression tree; Grouping determination

1. Introduction

Mathematical expression recognition (Chan and Yeung, 2000a) consists of two major processes: *symbol recognition* and *structural analysis*. Symbol recognition involves the recognition of individual handwritten symbols, whereas structural analysis interprets the sequence of the recognized symbols using geometrical information such as the positions and relative sizes of the symbols. Symbol recognition is a very common problem in most recognition systems which has been tackled for the last few decades. Structural analysis of mathematical expressions has also been studied for years. The symbols written in mathematical expressions are usually arranged in a complex two-dimensional structure, possibly of different sizes and in recursive manner. This makes structural analysis a challenging problem even when all symbols are recognized correctly. In addition, for online mathematical expression recognition, the processing speed is also an important issue for consideration.

Many techniques have been investigated for structural analysis for handwritten mathematical expression recognition. These include grammar-based approaches (Chou, 1989; Fateman et al., 1996; Chan and Yeung, 2000a; Toyota et al, 2006), tree transformation (Zanibbi et al., 2002), hidden Markov models (HMMs) (Kosmala and Rigoll, 1998) and minimum spanning tree (Tapia and Rojas, 2003, 2005). Some of the techniques such as the grammar-based approaches are slow, while others are sensitive to users' writing errors. Apart from their underlying

* Corresponding author. Tel.: +44 (0) 118 378 4422; fax: +44 (0) 118 378 4421.
E-mail addresses: baquy@cs.wisc.edu (B.-Q. Vuong), asschui@ntu.edu.sg (S.-C. Hui), Y.L.He.01@cantab.net (Y. He).

0167-8655/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2007.11.017

processing methods, all these techniques share one common characteristic. They require the entire mathematical expression with all the recognized symbols gathered as one input for structural analysis. As such, structural analysis can only start processing after the user finishes writing his expression, and the corresponding recognition result can then be known. This will cause delay and frustration to users when recognition errors occurred especially for long expressions.

In this paper, we propose the progressive structural analysis (PSA) approach for dynamic recognition of handwritten mathematical expressions. The PSA approach recognizes a user's handwritten mathematical expression dynamically while he is writing that expression. This approach has an advantage that it helps a user to identify any recognition errors after he finishes writing a symbol, and enable him to correct the error immediately. As such, users do not need to wait until they finish writing the whole expression before knowing any recognition errors. Compared with traditional structural analysis approaches, the proposed PSA approach is much more efficient and user-friendly.

The rest of this paper is organized as follows. First, Section 2 reviews the related work on structural analysis of mathematical expressions. Next, the layout structure of expressions and the definitions on mathematical expression trees are given in Section 3. Then, the proposed progressive structural analysis (PSA) approach is discussed in Section 4. In Section 5, the system architecture of WebMath which incorporates the PSA approach is briefly discussed. Experimental results are then presented in Section 6. Finally, Section 7 concludes the paper.

2. Related work

Earlier approaches to structural analysis of mathematical expressions relied on a two-dimensional stochastic context-free grammar (Chou, 1989) to parse mathematical formula with a generalized Cocke–Younger–Kasami (CKY) algorithm. However, when applied to two-dimensional grammars, the CKY algorithm is extremely slow. Faster methods were later proposed to work on two-dimensional grammars such as left-to-right recursive descent (Fateman et al., 1996) and more recently tree transformation (Zanibbi et al., 2002). In (Fateman et al., 1996), lexical analysis is first performed to group adjacent digits or names to form numbers and symbols such as sin, cos, log, exp, etc. Then, the two-dimensional symbols are ordered as linear structures and a recursive descent-based parsing is performed such that each parse of a sub-expression is confined to a region of the surface. Zanibbi et al. (2002) proposed tree transformation for both online and offline recognition. The key idea is to first construct a baseline structure tree (BST) describing two-dimensional arrangement of input symbols. Then, this initial BST is transformed to the so-called Lexed BST by grouping tokens comprised of multiple input symbols which include decimal numbers, function names, fractions, accents, etc. Finally, the Lexed BST is translated into an operator tree which describes the order and scope of operations in the input expression. A survey of earlier methods can be found in (Chan and Yeung, 2000b).

Instead of using two-dimensional grammars, Chan and Yeung (2000a) first transformed a mathematical expression into a one-dimensional representation and then parsed recursively using the so-called define clause grammar (DCG) to analyze the structure of the expression. More recently, Toyota et al (2006) converted a tree representation of a mathematical structure into a one-dimensional representation and then parsed by a formula description grammar.

A more complicated grammar-based approach, graph grammar rewriting technique (Grbavec and Blostein, 1995; Lavirotte and Pottier, 1997; Raja et al., 2006), was proposed by first creating an initial graph using compass point directions from each symbol. Then, hand-crafted grammar rules are used to specify a graph fragment for matching and a non-terminal graph fragment for replacing it with. This essentially generalizes string rewriting from standard string parsing to a graph parsing model. This approach was used on optical character recognition (OCR) formula rather than handwritten mathematical expressions.

In (Kosmala and Rigoll, 1998), it proposed using hidden Markov model (HMM) for simultaneous recognition and segmentation of mathematical expressions. The results from the Viterbi decoder are the sequence of recognized symbols as well as the start- and end-frames of each symbol which allow the extraction of geometrical features such as the center of the recognized symbol and the size and position of its bounding box. This effectively facilitates the interpretation of geometrical structure or structural analysis without the introduction of a two-dimensional grammar. The proposed approach achieved 97.7% symbol recognition rate. However, no performance evaluation results on structural analysis was reported.

Tapia and Rojas (2003, 2005) proposed a method for structural analysis based on a minimum spanning tree construction and symbol dominance. In (Tapia and Rojas, 2003, 2005), symbols are considered as nodes of a weighted graph. The weight of an edge joining two symbols is the minimum distance between attractor points (located in the boundary of the symbol bounding box) of symbols if they satisfy some mathematical relationship, or the distance between the center of each symbol bounding box if no mathematical relationship is satisfied. A minimum spanning tree (MST) for the graph can then be constructed. The proposed method can also handle some layout irregularities.

3. Mathematical expression tree

In mathematical expressions, input symbols are related. The relationships could be superscript like a^2, subscript like b_3, and above and under like $\sum_a^b$, etc. And one symbol can form different relationships with other different symbols. There could also be different ways of interpreting a relationship between two specific symbols. Therefore, we need

to identify the most possible relationship between input symbols. In addition, mathematical expressions are organized in a recursive structure. One expression may contain symbols and in turn each symbol may contain other expressions as its superscript, subscript, prescript, etc.

In the proposed progressive structural analysis approach, we focus on determining the relationships between two symbols and the grouping of related symbols. In this section, we define mathematical expression tree (MET), which will be used in the PSA approach to represent mathematical expressions. In the PSA approach, whenever a new symbol from the input mathematical expression is processed, the relationship and grouping properties of that symbol will be identified. These properties are then updated into the MET of the corresponding expression.

Definition 1. A symbol is the *parent symbol* of an expression if it contains that expression as its argument. Inversely, the expression is called *child expression* of that symbol.

For example, in the expression "$a^{m+n} - b$", 'a' is the parent symbol of the expression "$m + n$" and "$m + n$" is the child expression of 'a'.

Definition 2. Two symbols in an expression have *row relationship* if they are horizontally aligned to each other.

For example, in the expression "$a^{m+n} - b$", the symbol pairs $(a, -), (-, b), (m, +)$ and $(+, n)$ have the row relationship.

Row relationship has the following properties:

- *Commutability*. For symbols a, b; if a is row-related to b, then b is row-related to a.
- *Transitivity*. For symbols a, b, c; if (a, b) and (b, c) have row relationship, then (a, c) also has row relationship.

As illustrated from these properties, every mathematical expression can be divided into groups of symbols where all symbols in one group have row relationships. For example, in the expression "$a^{m+n} - b$", there are two such groups which are $(a, -, b)$ and $(m, +, n)$.

Definition 3. A *baseline* of an expression is a virtual line such that all symbols on this line have row relationship.

One expression can have more than one baseline. For example, there are two baselines in the expression "$a^{m+n} - b$". They correspond to the two groups given in the previous example.

Definition 4. The *dominant baseline* of an expression contains all symbols which do not belong to any child expression.

For example, the dominant baseline of the expression "$a^{m+n} - b$" is the one which contains $(a, -, b)$ since 'a', '$-$' and 'b' do not belong to any child expression. We also realize that one expression has only one dominant baseline.

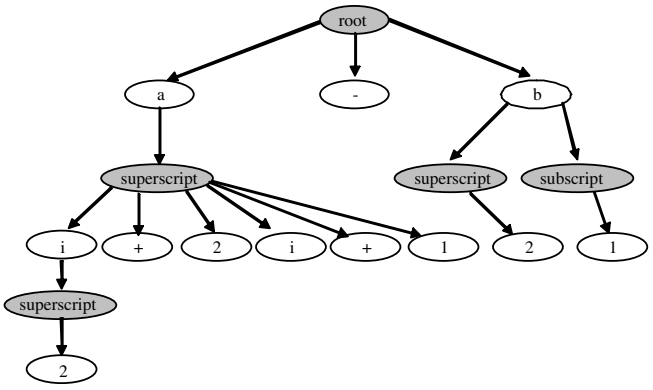

Fig. 1. Expression tree of $a^{i^2+2i+1} - b_1^2$.

Definition 5. A symbol on the dominant baseline of an expression is called *child symbol* of that expression. Inversely, the expression is called *parent expression* of that symbol.

Using the concepts of *parent symbol, child expression, dominant baseline* and *relationship* between symbols and expression, a mathematical expression can be represented by a hierarchical tree structure called mathematical expression tree. MET consists of two types of nodes: expression node and symbol node.

Definition 6. *Expression node* of a mathematical expression tree represents an expression. The MET root is an expression node. Every other expression node in MET has a parent node which represents a symbol in which the expression is an argument.

Definition 7. *Symbol node* of a mathematical expression tree represents a symbol. A symbol node always has a parent expression node which represents the expression that contains the symbol.

To illustrate the structure of MET, Fig. 1 shows the MET representation of the expression "$a^{i^2+2i+1} - b_1^2$". The shaded nodes are expression nodes. The root node of a MET is always an expression node. All other expression nodes represent sub-expressions in that expression. Every expression node except the root has a symbol node as its parent. The unshaded nodes are symbol nodes. Every symbol node has an expression node as its parent. All symbols represented by sibling symbol nodes have row relationship. These symbols are on the dominant baseline of their parent expression.

4. Progressive structural analysis

Progressive recognition is quite different from other handwritten mathematical expression recognition approaches. For example, in Fig. 2, it shows a written mathematical expression "$a^2 + b^2 = c^2$". The progressive recognition process performs progressively symbol recognition and

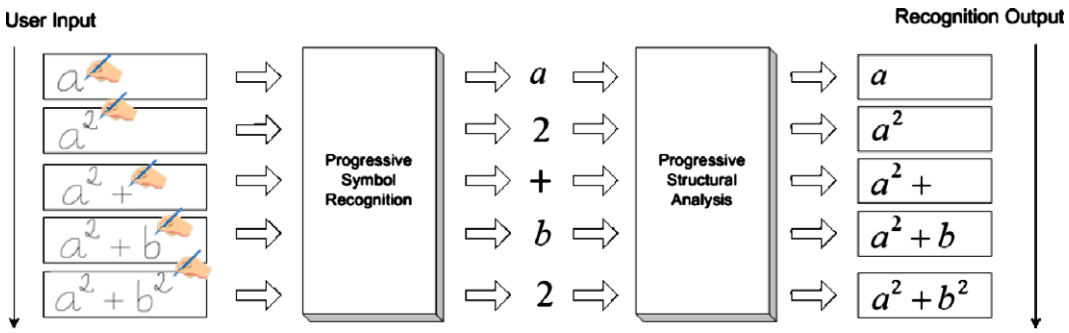

Fig. 2. Progressively recognizing the mathematical expression "$a^2 + b^2$".

structural analysis repeatedly for each symbol entered by the user as follows:

(1) *Progressive symbol recognition:* It gathers the latest written stroke and recognizes the latest written symbol in real-time. A modified elastic matching method based on original elastic structural matching (Chan and Yeung, 1998) is implemented for mathematical symbol recognition. To recognize multi-stroke symbols, a list of historical written strokes for possible groupings is maintained.
(2) *Progressive structural analysis:* It accepts the latest recognized symbol, updates the symbol in the corresponding MET and displays it on the screen. The user is able to make corrections if the symbol is not recognized correctly. At the end of user input, the MET is then converted into an appropriate representative format such as MathML (W3C, 2007a) or Latex (Lamport, 1985).

The PSA approach comprises three sub-processes, namely *related symbol identification, MET update* and *representative format conversion*. *Related symbol identification* identifies the related symbol of the latest input symbol from the current MET. *MET update* updates the latest input symbol into the corresponding position of the MET. It also groups meaningful consecutive symbols in MET into mathematical units such as trigonometric functions sin, cos and tan. *MET to representative format conversion* converts the MET into a representative format such as MathML or Latex. Fig. 3 gives the overall progressive structural analysis (PSA) algorithm.

Algorithm Progressive_Structural_Analysis (T, S)

Input: T – the current Mathematical Expression Tree
S – the latest written symbol
Output: Exp – the graphical presentation of the expression which contains S
Process:

```
1.  R ← Related Symbol Identification(S)
2.  if R is null then
3.    add S as the first child of T
4.  else if R is special then
5.    Special Symbol Update(T)
6.  else
7.    append S to T
8.  end if
9.  Grouping Symbols(T)
10. Exp ← MET To Representative Format Conversion(T)
11. return Exp
```

Fig. 3. The progressive structural analysis algorithm.

4.1. Related symbol identification

Before discussing this process in details, we first define *previous neighbor* and *related symbol* as follows.

Definition 8. The *previous neighbor* of a symbol is the symbol which was written just before it chronologically.

For example, assuming the expression "$a - b$" is written from left to right, the previous neighbor of '$-$' is 'a', and the previous neighbor of 'b' is '$-$'. Every symbol in the expression has only one previous neighbor except the first written symbol which has none.

Definition 9. A symbol R is the *related symbol* of another symbol S if S is written after R chronologically; and at least one of the following two conditions is satisfied:

(1) S and R are two adjacent symbols on the baseline of an expression; or
(2) In MET, S is represented by the first child symbol node of an expression node of R.

Here, S and R are not necessary to be consecutively written and the *related symbol* relationship is not commutable.

For example, in Fig. 1, 'a' is the related symbol of '$-$' and '$-$' is the related symbol of 'b' according to condition (1). Also in this figure, 'b' in "b_1^2" is the related symbol of '1' according to condition (2).

The *related symbol identification* process has made the two assumptions on user writing habits. First, users always finish writing the sub-expressions at lower levels in any mathematical expressions before continuing the higher

level sub-expression. Second, users do not make any correction on previously written sub-expressions. Once the sub-expression is written, it is already at its final stage. From these assumptions, it can be inferred that the related symbol of a symbol S must reside at the path from the tree root to S's previous neighbor. For example, in the expression "a^{i^2+2i+1}", we assume that the latest input symbol is '1'. Its previous neighbor is '+'. According to the above assumptions, the related symbol of '1' must be either '+' or 'a'. In fact, it is 'a' as will be shown later.

For any symbol S, the *related symbol identification* process builds its candidate related symbol list by searching the previous neighbor N of S. If no such a neighbor can be found, it means that S is the first input symbol. Otherwise, a list L of all the symbols along the path from the root to N (including N) can then be built. Then for each candidate symbol in the list L, its relative relationship with S is identified and a confidence value for this relationship is computed. The symbol with the highest confidence value is considered as the related symbol of S. Details of these steps are discussed in Sections 4.1.1 and 4.1.2.

4.1.1. Symbol relationship determination

This step aims to determine the possible relationship between two symbols. Since mathematical expressions are arranged in a two-dimensional structure, we adopt a geometrical approach which is based on the concept of bounding boxes. The bounding box of one symbol can be defined as the smallest rectangle enclosing it. One typical relationship is row relationship. Other types of relationships are superscript, subscript, inside/outside, prescript, above and under. Fig. 4 illustrates the geometrical layouts of these symbol relationships.

The following rules are used to determine the relationships between two symbols:

- *Row relationship:* Two symbols have row relationship if the difference between the y-coordinates of the two typographic centers of the corresponding bounding boxes do not exceed a threshold. In our approach, this threshold is taken as one third of the maximum height of two bounding boxes.
- *Above/under:* If the x-projection of one box is mostly contained in that of the other, then the relationship is considered as above or under, based on the relative positions of y-coordinates of the two typographic centers.
- *Inside/outside:* If the x-projection and y-projection of one box is mostly contained in that of the other, then the relationship is considered as inside/outside.
- *Superscript/subscript/prescript:* If the angle between the horizontal line and the line connects two bounding boxes' typographic centers is close to $\pi/4$, then the relationship is superscript, subscript or prescript, based on the relative positions of x- and y-coordinates of these two centers.

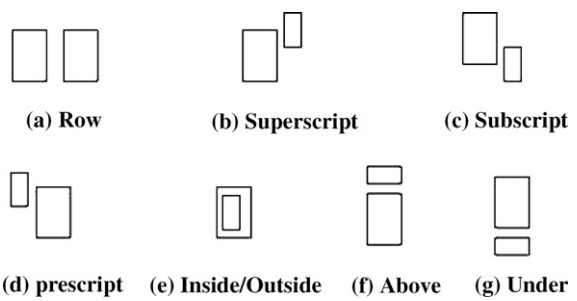

Fig. 4. Different types of symbol relationships.

4.1.2. Confidence determination

This step assigns a confidence value for each relationship identified from the previous step. First, a set of rules are used to determine whether a confidence value of zero should be assigned to a relationship:

- *Symbol property:* Based on the symbol properties, some symbols cannot have certain relationships. For example, some operators like '+' or '→' will not have arguments as superscript, subscript or prescript. Also, they will not be superscript or subscript of other symbols. If superscript or subscript relationship is found between the symbol '+' or '→' and some other symbol, then the confidence value of such a relationship is set to zero.
- *Adjacency:* In order to detect adjacency, a virtual line which connects two bounding boxes' centers of the two symbols is drawn. If this line intersects with any other symbols then these two symbols are considered as not adjacent and any relationship found between them has a confidence value of zero. For example, in the expression "$a - b = c$", 'a' and 'b' are separated by '−'. Thus, the row relationship found between them has zero confidence value.
- *Existing argument:* If a symbol already has an argument of relationship type of superscript, subscript, prescript, above, under or inside, then it will not have the same relationship with other symbols and the confidence value of such a relationship is set to zero.

If a confidence value of zero is not assigned to a relationship and the relationship is determined as row, superscript, subscript, prescript, inside or outside, then the confidence value is calculated as confidence $= 1/|C - \frac{h_1}{h_2}|$ where C is a constant and its value depends on the relationship determined by the bounding box algorithm, h_1 and h_2 are the smaller and the larger height respectively of the two bounding boxes of two symbols. Here, we use the height of a symbol's bounding box to represent its size. This idea comes from the observation that symbols with row relationship often have the same height, while symbols with superscript, subscript, above, under or prescript relationship often have smaller height in comparison with the parent symbol.

If the relationship is above/under, then the confidence value is calculated in a similar way as confidence =

$1/|C - \frac{w_1}{w_2}|$ where C equals to 0.8 and w_1, w_2 are the width of the smaller bounding box and larger bounding box, respectively.

4.2. MET update

The *MET update* process aims to update the latest input symbol into its corresponding position in the MET. It first checks for its related symbol which is obtained from the *related symbol identification* process. If there is no related symbol, it means that the latest input symbol is the first symbol written by the user and it is then appended as the first child node of the tree root of the MET. If a related symbol is found, the latest input symbol is added into the MET as follows:

- If the input symbol has row relationship with the related symbol, then it is added into the MET as the next sibling of the related symbol;
- Otherwise, a new expression node is created and appended to the related symbol according to the relationship between the latest input symbol and the related symbol (superscript, subscript, above, under or inside/outside). The input symbol is then appended as the first child of this newly created expression node.

4.2.1. Special symbol update

Special treatment is required for *special symbols* during MET Update as they may cause the positions of other symbols change in the MET. In our approach, we handle two types of special symbols, the fraction sign and square root sign. We observe that all the *affected symbols* always lie on some spatial regions relative to these special symbols. We call these regions *affected region*. For example, the affected regions of a fraction sign are areas under or above it. For square root sign, there is only one region which is the area inside it. Furthermore, the previous neighbor of a special symbol is an affected symbol if and only if it lies on the special symbol's affected regions. To determine all affected symbols in the tree with respect to a special symbol, we follow the two steps below:

(1) If the special symbol's previous neighbor lies on its affected regions, add this neighbor into the affected symbol list. Otherwise no affected symbols exist.
(2) If affected symbols exist, find the largest tree branch which contains the previous neighbor and all symbols in this branch lie on the same affected region as this neighbor. Nodes in this branch are all affected symbols.

Therefore, when the latest written symbol is a special symbol, we need to update the positions of the affected symbols in the tree. If the related symbol exists, it is added to the expression tree following the steps below:

(1) Detach all affected symbols and their child expressions from the tree.
(2) Add the latest written symbol to the tree as non-special symbols.
(3) Create a new expression node, append this node to the latest written symbol according to the affected region which contains all affected symbols.
(4) Append all detached symbols and expressions to the new expression node. The relative positions among these symbols and expressions are kept unchanged.

4.2.2. Grouping symbols

The last step in *MET update* is grouping symbol nodes into one unit if they can form either a single numeric number or some mathematical operators such as sin, cos, tan, cotan, lim, etc. Since our approach is progressive, we only consider the grouping possibility with the latest written symbol. To do this, we create combinations of the latest written symbol together with its previous sibling nodes such that no node has a child expression.

4.3. MET to representative format conversion

The final process is to convert the MET into a representative format MathML in order to render the expression graphically for viewing. A recursive algorithm was implemented for the conversion. One should note that the MET has an interleaving layer structure of expression nodes and symbol nodes. Nodes at odd levels are expression nodes and nodes at even levels are symbol nodes.[1] We start the conversion by creating a MathML root corresponding to the tree root and analyzing the tree root's children. For each child C which is an expression node, we create a corresponding MathML tag with arguments on C's children and append this tag to the MathML root. Each of C's children is then treated as the root of another sub-tree and this sub-tree is analyzed in the same manner as the containing tree. The process ends when all nodes have been processed.

5. System architecture

A Web-based handwriting mathematics system, called WebMath, has been developed. Fig. 5 shows the overall architecture of the WebMath system which consists of the following components: Handwriting Mathematical Expression Editor, Mathematical Computation, AJAX-Based Communicator. These components are organized in a client-server architecture according to its functionalities.

- *Handwriting Mathematical Expression Editor*. This is the major component of the WebMath system which provides a writing pad interface for client users. It first collects data from the clients in the form of point

[1] Root is at level zero.

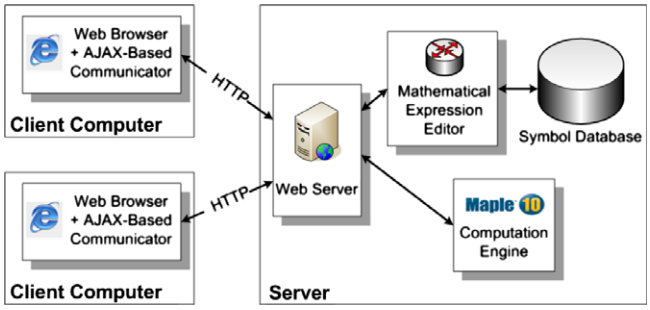

Fig. 5. System architecture of WebMath.

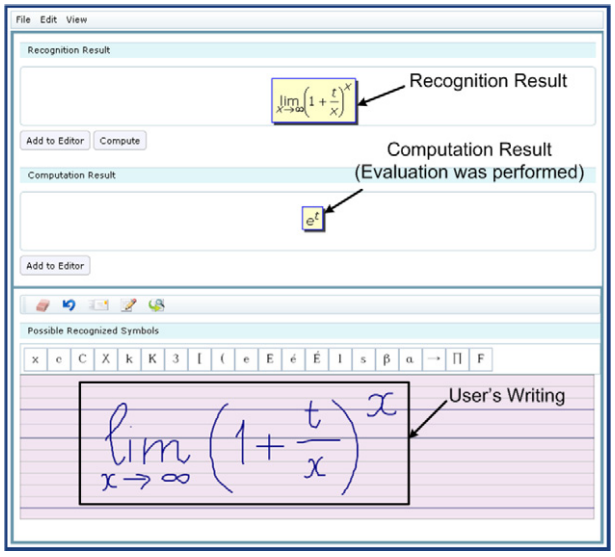

Fig. 6. Handwritten mathematical expression editor.

sequences which are then processed and sent over to the server for recognition. The elastic matching algorithm is used for symbol recognition and the PSA algorithm is used for structural analysis. A symbol database which stores the sample symbol dataset is also maintained at the server for symbol recognition purpose. Both inputs and outputs are displayed in graphical representations in clients using SVG (W3C, 2007b). Fig. 6 shows the user interface of the Handwriting Mathematical Expression Editor.

- *Computation Engine*. This enables the computation of results of mathematical expressions through the computation functions provided by Maple (Maplesoft, 2007). Many mathematical functions such as algebraic simplification and factorization, and differentiation and integration in calculus are supported. The computation results are displayed on Web browser through the MathML mathematical representation.
- *AJAX-Based Communicator*. AJAX (White, 2006; Garrett, 2005; Pascarello and James, 2006) is used for the implementation of the communication mechanism between client and server. It helps to maintain continuous data flow between client and server without the need of refreshing the web page.

6. Performance evaluation

To evaluate the effectiveness of our proposed approach, we have conducted experiments using a number of different expressions extracted from the "CRC standard Math tables and formulae" (Zwillinger, 2003). These expressions are grouped into six domains, namely Elementary Algebra, Number Theory, Trigonometric Functions, Geometry, Differential Calculus and Integration. Each domain contains ten expressions of different types except Number Theory which contains only three expressions. There are totally 53 expressions which are listed in Appendix A.

6.1. Experimental setup

Ten users were involved in the experiments. Each user was first given a short introduction of the WebMath system. Then, they spent 10 min to get familiar with the functions provided by the WebMath's Handwriting Mathematical Expression Editor and experimented it with a few simple expressions of different types which are different from those used for testing. After that, each user was asked to write all the expressions from the test data set of expressions, with each expression written only once. Since we aimed to measure the performance of the PSA algorithm only, we assumed that symbols were recognized correctly in the *Symbol Recognition* phase. To ensure this, after writing each stroke, the user was required to correct manually any mis-recognized symbols with the correct ones. These mis-recognized symbols were not counted towards the evaluation of the performance of the PSA approach.

Since our proposed PSA approach consists of three processes, we measured errors in each process as well as the overall error caused by the approach.

6.2. Experimental results

The experimental results are given in Tables 1–3. We can observe that the overall error rate of the PSA approach is the same as the error rate in the *MET update* process. This indicates that the accuracy of the other two processes, *related symbol identification* and *MET to representative format conversion*, are all 100%.

In the *MET update* process, it has a step on *symbol relationship identification*. The PSA approach has sometimes made a wrong relationship determination (e.g., row relationship was mistaken as superscript) due to writing ambiguity. The more complicated the expression is, the higher chance the errors will occur. The average related symbol relationship identification rate over all the users is 94.53% as shown in Table 1.

Table 2 shows the recognition rates based on mathematical domains. Among the six mathematical domains used in our experiment, Trigonometric functions and differential calculus gave the highest recognition rates of 97% while number theory gave the lowest recognition rate of 90%. On average, the recognition rates for all domains are above

Table 1
Experimental results based on different users

	Accuracy (%)			
	RSI	MU	RFC	Overall
User 1	100	98.11	100	98.11
User 2	100	90.57	100	90.57
User 3	100	92.45	100	92.45
User 4	100	96.23	100	96.23
User 5	100	96.23	100	96.23
User 6	100	94.34	100	94.34
User 7	100	96.23	100	96.23
User 8	100	94.34	100	94.34
User 9	100	92.45	100	92.45
User 10	100	94.34	100	94.34
All users	100	94.53	100	94.53

Table 2
Experimental results based on different mathematical domains

	Accuracy (%)			
	RSI	MU	RFC	Overall
Elementary Algebra	100	94	100	94
Number Theory	100	90	100	90
Trigonometric Functions	100	97	100	97
Geometry	100	93	100	93
Differential Calculus	100	97	100	97
Integration	100	96	100	96
All domains	100	94.53	100	94.53

Table 3
Experimental results based on different numbers of symbols in expressions

# Symbols	Accuracy (%)			
	RSI	MU	RFC	Overall
1–10	100	100	100	100
11–20	100	97.93	100	97.93
20–30	100	91.43	100	91.43
30–40	100	80.0	100	80.0
40–50	100	80.0	100	80.0
All categories	100	94.53	100	94.53

90%. We also observed that expressions in domains with lower recognition rates are more complex in terms of the number of symbols and baselines.

Table 3 shows the recognition rates based on the complexity of expressions in terms of the number of symbols contained. It is observed that the recognition accuracy reduces as the number of symbols increases. For those expressions with less than 10 symbols, the recognition rate is 100%. However, for expressions with more than 30 symbols, this rate is only 80%. This result reflects the fact that longer expressions are more difficult to recognize as it is more difficult to align all the written symbols properly.

6.3. Common error cases

There are some typical cases in which the WebMath system may fail in analyzing the written mathematical expressions. These include:

- *Row alignment error:* It is easy for a user to write two consecutive row-related symbols but one looks like the superscript or subscript of the other. This type of errors can easily be detected and corrected by human, but it is hard for our system as it does not support contextual analysis. Among all the symbols, the fraction sign has the highest probability of having this type of errors.
- *Prescript alignment error:* Another type of errors may occur when we write prescript with the square root sign (e.g., $\sqrt[y]{a}$). If the prescript is written too close to the square root sign, our system may mistake it as an inside relationship.

7. Conclusions

In this paper, we have proposed the progressive structural analysis approach for dynamic recognition of on-line handwritten mathematical expressions. The proposed approach makes use of the properties of expressions and user's writing habits to provide immediate recognition feedback in order to achieve dynamic and interactive recognition. Experimental results show that the proposed approach has achieved good performance results. Currently, the proposed PSA approach is being extended to support other special symbols including matrices, piecewise defined functions and vectors, in addition to the fraction and square root signs.

Appendix A. Expressions for the experiments

(1) Elementary Algebra
  (a) $a^2 - b^2 = (a - b)(a + b)$
  (b) $(a \pm b)^2 = a^2 \pm 2ab + b^2$
  (c) $(ab)^x = a^x b^x$
  (d) $\sqrt[x]{\frac{a}{b}} = \frac{\sqrt[x]{a}}{\sqrt[x]{b}}$
  (e) $\frac{b^2c^2 - 4b^3d - 4ac^3 + 18abcd - 27a^2d^2}{a^4}$
  (f) $(a \pm b)^4 = a^4 \pm 4a^3b + 6a^2b^2 \pm 4ab^3 + b^4$
  (g) $a^4 + b^4 = (a^2 + \sqrt{2}ab + b^2)(a^2 - \sqrt{2}ab + b^2)$
  (h) $a^{\frac{x}{y}} = \sqrt[y]{a^x} = (\sqrt[y]{a})^x$
  (i) $\sqrt[x]{\sqrt[y]{a}} = \sqrt[xy]{a}$
  (j) $\frac{2x}{x^2-1} = \frac{1}{x-1} + \frac{1}{x+1}$

(2) Number Theory
  (a) $\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{b_i}$
  (b) $x_k + y_k\sqrt{d} = (x + y\sqrt{d})^k$
  (c) $\frac{x}{1-x^2} + \frac{1}{1-x^4} + \frac{x^2}{1-x^4} = \frac{1}{1-x}$

(3) Trigonometric Functions
  (a) $\sin(-\alpha) = -\sin(\alpha)$
  (b) $\tan(\alpha + n\pi) = \tan\alpha$
  (c) $\sin^2 z + \cos^2 z = 1$
  (d) $\sin\alpha = \tan\alpha\cos\alpha$
  (e) $\tan\alpha = \frac{\sin\alpha}{\cos\alpha}$
  (f) $\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$
  (g) $\sin 2\alpha = 2\sin\alpha\cos\alpha = \frac{2\tan\alpha}{1+\tan^2\alpha}$

(h) $\cos\frac{\alpha}{2} = \pm\sqrt{\frac{1+\cos\alpha}{2}}$

(i) $\cot\alpha \pm \cot\beta = \frac{\sin\beta\pm\alpha}{\sin\alpha\sin\beta}$

(j) $\cot^2\alpha = \frac{1+\cos 2\alpha}{1-\cos 2\alpha}$

(4) Geometry

(a) $pq = ac + bd$

(b) $ax + by + c = 0$

(c) $\frac{y-y_1}{x-x_1} = \frac{y_0-y_1}{x_0-x_1}$

(d) $\left(\frac{kx_1+(100-k)x_0}{100}, \frac{ky_1+(100-k)x_0}{100}\right)$

(e) $\frac{1}{4}\sqrt{4p^2q^2 - (b^2+d^2-a^2-c^2)^2}$

(f) $s = \frac{1}{2}(a+b+c+d)$

(g) area $= \sqrt{(s-a)(s-b)(s-c)(s-d)}$

(h) $p = \sqrt{\frac{(ac+bd)(ab+cd)}{(ad+bc)}}$

(i) $r = \frac{1}{2}a\cot\frac{180^\circ}{k}$

(j) $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$

(5) Differential Calculus

(a) $\lim_{x\to\infty}\left(1+\frac{t}{x}\right)^x = \mathrm{e}^t$

(b) $\lim_{x\to 0}\frac{a^x-1}{x} = \log a$

(c) $\lim_{x\to 0}\frac{\sin ax}{x} = a$

(d) $\lim_{x\to 0}\frac{\log(1+x)}{x} = 1$

(e) $\frac{\mathrm{d}}{\mathrm{d}x}(a) = 0$

(f) $\frac{\mathrm{d}}{\mathrm{d}x}(u+v) = \frac{\mathrm{d}u}{\mathrm{d}x} + \frac{\mathrm{d}v}{\mathrm{d}x}$

(g) $\frac{\mathrm{d}}{\mathrm{d}x}(u^n) = nu^{n-1}\frac{\mathrm{d}u}{\mathrm{d}x}$

(h) $\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{1}{u}\right) = -\frac{1}{u^2}\frac{\mathrm{d}u}{\mathrm{d}x}$

(i) $\lim_{x\to a}\frac{f(x)}{g(x)} = \lim_{x\to a}\frac{f'(x)}{g'(x)}$

(j) $\frac{\mathrm{d}^2}{\mathrm{d}x^2}(f(u)) = \frac{\mathrm{d}f}{\mathrm{d}u}(u)\cdot\frac{\mathrm{d}^2u}{\mathrm{d}x^2} + \frac{\mathrm{d}^2f}{\mathrm{d}u^2}(u)\cdot\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)^2$

(6) Integration

(a) $\int a\,\mathrm{d}x = ax$

(b) $\int\frac{1}{x}\,\mathrm{d}x = \log x$

(c) $\int \mathrm{e}^{ax}\,\mathrm{d}x = \frac{\mathrm{e}^{ax}}{a}$

(d) $\int\frac{1}{a^2+x^2}\,\mathrm{d}x = \frac{1}{a}\tan^{-1}\frac{x}{a}$

(e) $\int\frac{1}{\sqrt{x^2\pm a^2}}\,\mathrm{d}x = \log\left(x+\sqrt{x^2\pm a^2}\right)$

(f) $\int_1^\infty\frac{1}{x^m}\,\mathrm{d}x = \frac{1}{m-1}$

(g) $\int_0^\infty\frac{x^{p-1}}{1+x}\,\mathrm{d}x = \frac{\pi}{\sin p\pi}$

(h) $\int_a^a f(x)\,\mathrm{d}x = 0$

(i) $\int_a^b f(x)\,\mathrm{d}x + \int_b^c f(x)\,\mathrm{d}x = \int_a^c f(x)\,\mathrm{d}x$

(j) $\int_0^\infty\frac{\mathrm{d}x}{(1+x)\sqrt{x}} = \pi$

References

Chan, K.-F., Yeung, D.-Y., 2000a. An efficient syntactic approach to structural analysis of on-line handwritten mathematical expressions. Pattern Recognition 33 (3), 375–384.

Chan, K.-F., Yeung, D.-Y., 2000b. Mathematical expression recognition: a survey. Int. J. Doc. Anal. Recognit. 3 (1), 3–15.

Chan, K., Yeung, D., 1998. Elastic structural matching for on-line handwritten alphanumeric character recognition. In: The 14th Internat. Conf. on Pattern Recognition. Brisbane, Australia, pp. 1508–1511.

Chou, P., 1989. Recognition of equations using a two-dimensional stochastic context-free grammar. In: Pearlman, W.A. (Ed.), Visual Communications and Image Processing IV. SPIE Proceedings Series, vol. 1199, pp. 852–863.

Fateman, R., Tokuyasu, T., Berman, B., Mitchell, N., 1996. Optical character recognition and parsing of typeset mathematics. J. Vis. Commun. Image Represent. 7 (1), 2–15.

Garrett, J., 2005. Ajax – a new approach to web applications. Daptive-path.com,adaptivepath.com/publications/essays/archives/000385.php.

Grbavec, A., Blostein, D., 1995. Mathematics recognition using graph rewriting. In: 3rd Internat. Conf. on Document Analysis and Recognition. Montreal, Canada, pp. 417–421.

Kosmala, A., Rigoll, G., 1998. On-line handwritten formula recognition using statistical methods. In: Internat. Conf. on Pattern Recognition (ICPR). Brisbane, pp. 1306–1308.

Lamport, L., 1985. LATEX – A Document Preparation System – User's Guide and Reference Manual. Addison-Wesley, Reading, MA.

Lavirotte, S., Pottier, L., 1997. Optical formula recognition. In: 4th Internat. Conf. on Document Analysis and Recognition. pp. 357–361.

Maplesoft, 2007. Maple. <http://www.maplesoft.com/products/maple/index.aspx>.

Pascarello, D., James, D., 2006. Ajax in Action. Manning Publications Co.

Raja, A., Rayner, M., Sexton, A., Sorge, V., 2006. Towards a parser for mathematical formula recognition. In: 5th Internat. Conf. on Mathematical Knowledge Management. LNCS 4108. Springer-Verlag.

Tapia, E., Rojas, R., 2003. Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance. In: 5th Internat. Workshop on Graphics Recognition, Recent Advances and Perspectives (GREC). LNCS 3088. pp. 329–340.

Tapia, E., Rojas, R., 2005. Recognition of on-line handwritten mathematical expressions in the e-chalk system – an extension. In: 8th Internat. Conf. on Document Analysis and Recognition. pp. 1206–1210.

Toyota, S., Uchida, S., Suzuki, M., 2006. Structural analysis of mathematical formulae with verification based on formula description grammar. In: 7th Internat. Workshop on Document Analysis Systems. LNCS 3872. Nelson, New Zealand, pp. 153–163.

W3C, 2007a. Mathml. <http://www.w3.org/Math/>.

W3C, 2007b. Scalable Vector Graphics (SVG). <http://www.w3.org/Graphics/SVG/>.

White, A., 2006. Measuring the benefits of ajax. Developer.com, <www.developer.com/xml/article.php/3554271>.

Zanibbi, R., Blostein, D., Cordy, J., 2002. Recognizing mathematical expressions using tree transformation. IEEE Trans. Pattern Anal. Machine Intell. 24 (11), 1455–1467.

Zwillinger, D., 2003. CRC Standard Math Tables and Formulae, 31st ed. CRC Press, Boca Raton.