Chapter 2

Related Work

2.1 Introduction

When considering common issues in the recognition problem, authors have divided the recognition of mathematical expressions into the stages shown in Fig. 1.7. As mentioned in the last chapter, we concentrate on the stages corresponding to *symbol recognition* and *structural analysis*. These steps can be further divided as shown in Fig. 2.1. Most of the differences between recognition methods for mathematical notation and their improvements are generated by variations of these major steps.

In this chapter we offer a survey of the existent work related to this thesis. In Sect. 2.2, we describe the symbol recognition task step by step. We also review some relevant work in this field. We give an overview of recent and the most important techniques for structural analysis in Sect. 2.3. The most remarkable work on user interfaces for on-line handwriting is described in Sect. 2.4.

The review given in this chapter intends to give the reader an idea of the state of the art of the corresponding areas and should not be considered exhaustive. We refer the reader to [89, 67, 70] for more detailed descriptions and references in the area of on-line handwriting recognition. In the area of recognition of mathematical expressions, we refer the reader to [9, 55, 14].

2.2 Symbol Recognition

2.2.1 Segmentation

Segmentation is the process of decomposing, grouping, or isolating the data into classifiable units which represent single symbols. In the case of on-line run-on hand-

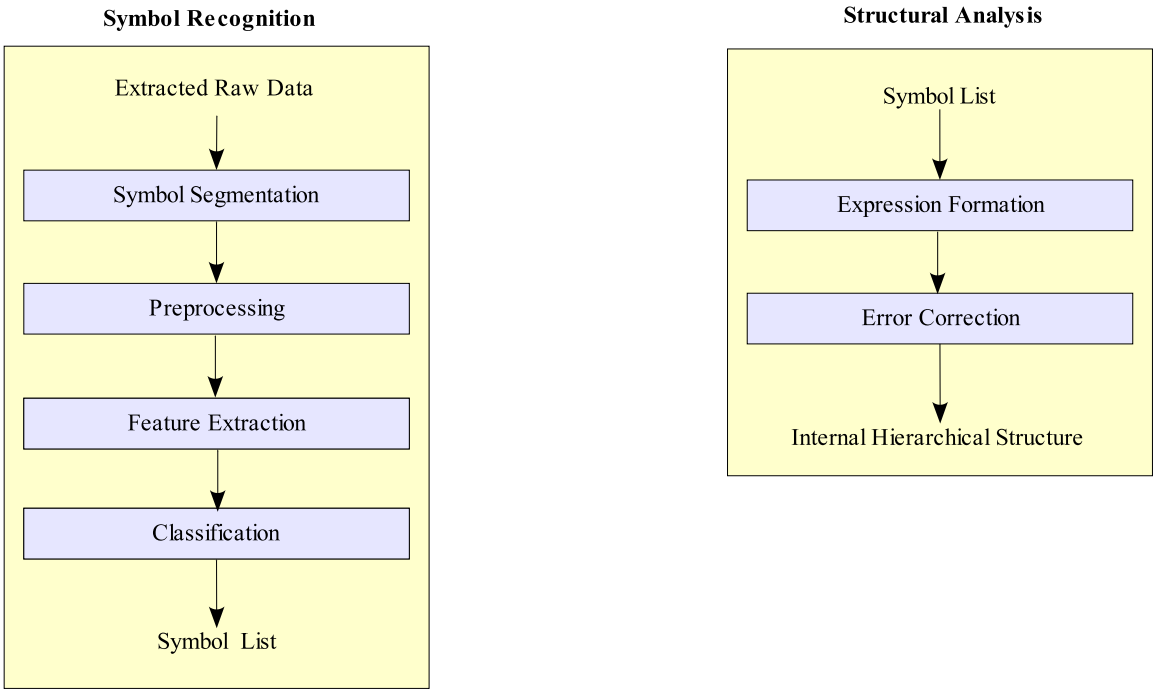


Figure 2.1: *Diagrammatic representation of the stages to recognize symbols and of the structural analysis.*

writing recognition, the segmentation problem consists of properly grouping a given list of strokes. For the recognition of cursive handwriting, segmentation consist of decomposing the main stroke, which forms an entire word, into strokes representing symbols and transitions from one to another.

An example of a simple heuristics-based segmentation is the method used by Mandler [60]. He associates the input strokes with a predetermined thickness by dilating them with a rectangle. With this criterion, two strokes belong to the same symbol if their dilated versions intersect.

Sometimes, segmentation methods use the information given by the classifier. Winkler et al. [46, 100] generate a symbol hypothesis net from the given strokes. The hypothesis net generates all possible combinations of strokes in order to handle ambiguity during symbol grouping. The different combinations of strokes in the net, which represent potential symbols, are classified using a Hidden Markov Model (HMM). The probabilistic output of the classifier serves to associate a likelihood value to each combination of strokes, and the one with the highest likelihood value is selected as the final segmentation result. Examples of classifier-based segmentation

are used by some of the systems we describe in Sect. 2.4.

2.2.2 Preprocessing

Preprocessing is necessary to eliminate noise, to reduce the amount of information, and to normalize handwriting [89, 34].

An example of preprocessing to eliminate noise is simply the application of a Gaussian filter to a handwritten stroke. In this case, processed points in the stroke are the result of a weighted average of itself and its neighbors. Dehooking, point clustering, filtering, and stroke connection are useful preprocessing procedures to eliminate unnecessary information from the stroke. Examples of normalization for on-line data are scaling, slant correction, and equidistant resampling. Chapter 3 gives detailed descriptions of some of these preprocessing methods.

2.2.3 Feature Extraction

Feature extraction means deriving measures and characteristics from the raw data which are useful in making predictions. It is common that feature extraction methods are based on invariance, reconstruction, and expected distortions.

Features may be *local* or *global*. The source of information of on-line data are points and strokes. When we talk about local features, we refer to characteristics of a specific point in strokes derived from its neighboring points. Global features normally refer to topological (morphological) characteristics of the stroke based on the trajectory it describes.

The first local feature we consider are the coordinates of a point. They serve, for example, to derive important local characteristics, like curvature or direction. Points having a high curvature are important to decompose strokes in static elements. This decomposition is a structural description of the underlying shape of strokes [34].

Global features normally serve to obtain a categorical (not numerical) classification of strokes. For example, the ratio of the distance between a stroke's end points and its length is a global feature, which serves to determine if the stroke is closed or open, when comparing this ratio against a predetermined threshold. Similarly, when considering a symbol formed by two strokes, we can say that they cross or does not cross, or only touch. This kind of information serves to construct a decision tree which reduces the given data into a small set used in a later analysis using more complex features [89].

2.2.4 Symbol Classification

Symbol classification means the use of a certain method to obtain the symbol's label. Actually, we can potentially use any of the great number of existing classification methods to such purpose. In this section we review some of such methods which were specially developed for symbol classification.

Similarity Measures

Similarity measures for strokes allow the use of distance-based classifiers. Generally, unknown symbols are matched against stored references (prototypes), and the label corresponding to the unknown symbol is the same as that of the symbol which best matches it. Similarity measures are normally defined at stroke (curve) level.

Tappert [86] uses *elastic matching* as a similarity measure. The procedure to calculate it is based on the alignment of points which constitute the stroke using the dynamic time warping (DTW) algorithm. The optimal matching corresponds to the minimum sum of matching costs, which are normally the distance between strokes' points obtained by dynamic programming. This algorithm allows the comparison of strokes having a different number of points. In this case, the feature-extraction step can be skipped.

Li and Yeung [57] also use time warping to calculate similarities between strokes. Their method converts strokes into a character string. The string describes the stroke as a sequence of eight directions. They find the distance between two string sequences by calculating costs of three string transformations, namely compression, expansion, and substitution.

Schwenk [79] proposes a similarity measure based on constrained tangent distance. The distance is locally invariant under the following local transformations: translation, scale, axis deformation, rotation, diagonal deformation, and slope transformation. This method tries to incorporate knowledge about geometrical variations of handwriting. In Chap. 3 we explain these transformations in more detail.

Other similarity measures for curves which can be useful for on-line symbol classification can be found in [95].

Structural Methods

Structural methods are based on the assumption that handwriting is made up of some elementary or primitive shapes, also known as allographs. They describe the

morphological characteristics of strokes. For example the symbol '6' can be describes through two primitives: a descending curve and a loop.

Parizeau and Plamondon [65] construct a set of basic allographs using a fuzzy syntactic approach to model handwriting. For example, a primitive can be of type 'c' or 'ci', which corresponds to stroke's segments with a shape similar to the letter 'c' where the 'i' indicates "inversion", i.e. a horizontal or vertical reflection of the shape. In this way, they consider twenty allographs, which correspond to i-shapes, c-shapes, t-crossings, loops, etc. A symbol is coded using a grammar formalism as a sequence of these primitives and the similarity between symbols is measured by using a sequence distance. Similar approaches are used in [11, 74, 102].

Neural Network and Statistical Models

Guyon et al. [35] developed a neural network system for the recognition, based on so-called time delay neural networks. The system integrates recognition and segmentation in the same neural network architecture. The network is used to estimate a posteriori probabilities for characters in a word. One of the layers of the network converts the temporal information, point position, curvature, and direction in a two-dimensional image representation. Similar approaches are used in [59], which have been specially developed for the recognition of on-line handwritten words.

One of the most recent statistical approaches used for on-line recognition are Hidden Markov Models (HMM). They use dynamic programming to find the optimal alignment which performs character segmentation and recognition simultaneously. Bellegarda [5] uses local features, point's positions and curvature, to feed a simple two-state HMM. Sin and Kim [80] also uses a HMM to model inter-letter relations, also known as ligatures. They train a HMM for each letter and ligature type. Schenkel et al. [75] uses a hybrid neural-network-HMM system for on-line word recognition.

Shwenk [79] uses an autoencoder neural network to construct a model for each symbol class. This autoencoder network actually describes each symbol class by finding the principal components that describe the class population. New symbols are fed into each network and projected to a hyperplane generated by the principal components. The tangent distance between the encoded (projected) vector and the original one is calculated. The class label assigned to the new vector corresponds to the one where the minimum distance is reached.

Clustering Methods

Some of the above recognition approaches were developed in a user-dependent manner. Clustering methods in on-line handwriting are developed to remedy this situation. They try to model intra-class variations caused by different writing styles found in large databases. Vuori [96] uses a self-organizing map to cluster writing styles. She uses variations of the elastic matching method as a similarity measure. Her clustering algorithms can be used for prototype selection, which serves to classify characters according to the k Nearest Neighbors (k-NN) rule. The recognition system adapts to new writing styles by modifying its prototype set. In the same fashion, Connell and Jain [20] use also elastic matching to measure intra-cluster and inter-class measures, to be used in a hierarchical clustering algorithm.

2.3 Structural Analysis

2.3.1 Expression Formation

Expression formation consists of translating the results obtained in the symbol recognition step into a tree which describes the relations between symbols of the mathematical expression. The nodes that constitute this tree are a data structure which stores the symbol's label and other numerical parameters, for example size and location. Nodes are a compact representation of the original raw symbol. After an analysis of the relative positions of symbols, the nodes in the tree are linked to each other depending on which one of the spatial relations –above-left, above, superscript, right, subscript, below, below-left, and subexpression– they satisfy [9, 55]. As we will see, most of the methods described in this section rely on this principle.

Fateman et al. [29] developed a recognizer for typeset mathematical expressions. The recognized symbols are grouped into box tokens representing numbers, function names, and variables. They use a bottom-up recursive descent parser to obtain the final result as a lisp expression.

Lavirotte and Pottier [52] define a context-sensitive graph grammar to represent mathematical formulas and to remove ambiguities during structural analysis. Rules in graph grammars consist of collapsing a subgraph in a node, if they satisfy a given condition. They define a set of rules to describe the mathematical relation and apply a bottom-up parsing algorithm to obtain the final result. If we consider the expression 'e^x+1', a rule is defined to collapse the two nodes representing the symbols 'e' and 'x'.

They now represent a single token, and the process can be continued by collapsing the new tokens 'e^x' and '1' to construct the final result.

Miller and Viola [62] use a generative model approach based on a stochastic-free grammar. The recognition process of the structure consists in finding the productions of the grammar with the highest probability when considering the overall probability of generating the expression. They use a convex hull condition to prune the searching space of the grammar productions. A production in the grammar is not admissible if the convex hull of a symbol group intersects a symbol not belonging to the group.

Winkler et al. [100] used HMMs for segmentation and classification of symbols. Using a soft-decision approach, they construct a hypothesis net of all possible segmentations of strokes. They store the information of the possible spatial relations between symbols in a directed graph, and a set of alternative interpretations of the expression is given.

Kosmala et al. [48] also present a statistical approach based on the application of HMMs. The system allows simultaneous segmentation and symbol classification, as well as the interpretation of the symbols' spatial relationships. One of the conditions for correct segmentation and recognition is that the expression be written from left to right and from top to bottom. Delayed symbols are not allowed, as when first writing an expression and then enclosing it in parenthesis. In an posterior work [47] they use graph rewriting for the structure analysis to avoid the above-mentioned restrictions of handwriting.

Chan and Yeung [13] propose the use of a definite-clause grammar (DCG) to define precise replacement rules when parsing mathematical expressions. They use DCG because a grammar expressed through this formalism is declarative and easy to maintain and extend. They use a parsing scheme using left-factoring to reduce the time for the expression's interpretation.

Eto and Suzuki [26] use a weighted graph to represent spatial relations between symbols. The nodes in the tree represent the symbols obtained during the segmentation stage. They are connected depending on whether they satisfy the superscript, subscript, or right relations. Because of the multiple results obtained during the recognition step, nodes have multiple edges which store a symbol's label and weight. They generate a set of spanning trees of the initial graph, which result from the minimization of the edge costs. If they result in an admissible (not contradictory) mathematical structure, one of them is selected as the final result if it minimizes a global cost criterion.

Zanibbi et al. [103] describe a mathematical expression as a structure of nested baselines, a so-called *baseline structure tree*. Baselines constitute horizontal arrangements of symbols. The first step in the procedure recursively constructs the baseline tree in the handwritten expression by handling irregular or poor horizontal layout structures. The second step converts the structure tree into an operator tree, which is further processed by applying tree transformations using the TXL language. Tree transformations consists of locating patterns in the tree structure and applying replacing rules on them. Defining different replacing rules helps to define a particular dialect or recognition scope of mathematical notation.

2.3.2 Error Correction

Most of the approaches mentioned before convert a two-dimensional representation of the expression into a one-dimensional one, a string of characters, which represents the expression in a certain computer language. Erroneous symbol segmentation and classification can generate incorrect results during structural analysis.

Lee and Wang [55] propose heuristics for the correction of the most common errors derived by false recognized symbols. They consider four heuristic rules. One of them is "for every binary operator there must exist two operands". By applying this rule the expression '$/ < i < n$' is corrected to '$1 < i < n$'. They also define a rule to consider errors encountered during the construction of function names. An expression like '$5inx$' should be corrected to '$\sin x$'. Numerals with subindexes are considered as an error. Expressions like '1_2' and '5_b' are corrected to 'l_1' and 'S_b' respectively. The last rule considers the correction derived by case confusion of letters and other character properties, considering whether it is italic, bold, etc., because letters forming a determined group in the expression are similar. Expressions like '$3Pqr$' and '$\mathrm{x} \cdot y$' are transformed to '$3pqr$' and '$x \cdot y$'.

Chan and Yeung [15] extend their system based on definite-clause grammar to handle lexical, syntactic, and semantic errors. They implement some rules to identify common errors in string grammar, namely substitution, deletion, and insertion. They also incorporate rules for the correction of some common syntactic errors, like incorrect implicit operators, missing binding and fence symbols, or missing operands. The semantic errors they consider are corrected heuristically, similar to the way Lee and Wang do. The lexical errors they consider take place at the recognition and segmentation level, caused by poor digitalization or irregular writing. They also propose some performance measures that concern the recognition of expressions, symbols, and

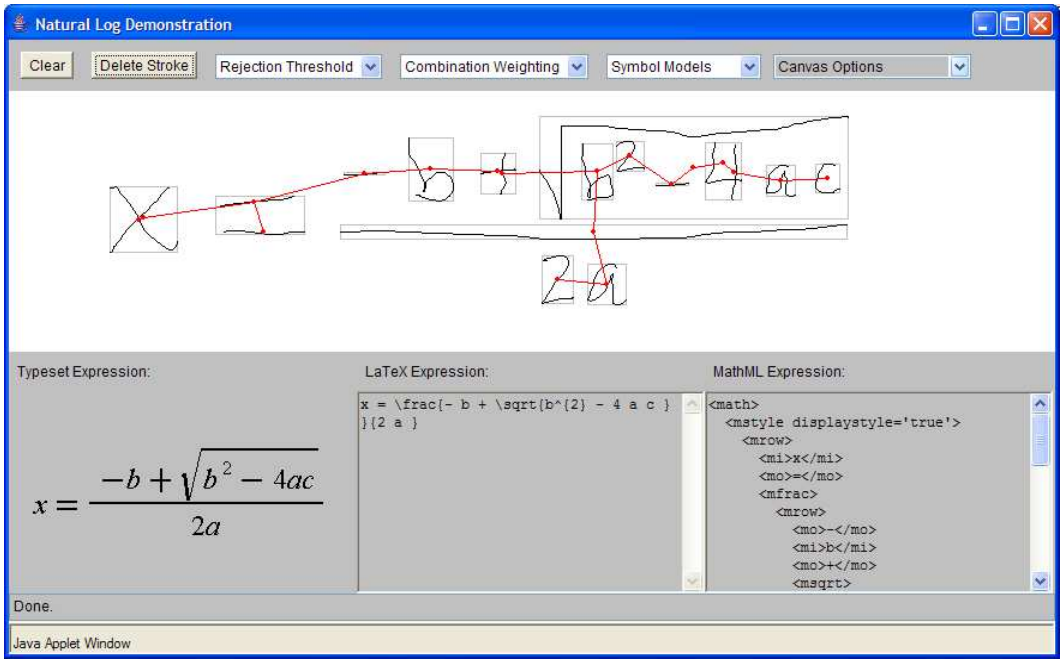


Figure 2.2: *The Natural Log system.*

operators, as well as an integrated performance measure.

Actually, not much effort is expended on the error correction task. Some other authors does not handle automatically the correction of errors. For this purpose, they offer instead user-friendly interfaces which allow immediate feedback and have undo-redo and visualization capabilities, as we will see in the next section.

2.4 User Interfaces

2.4.1 The Natural Log System

The Natural Log system is a user-dependent system developed by Matsakis [61]. The system was written in Java and is only available on-line as an applet on the internet. See Fig. 2.2.

To classify on-line symbols, he constructs a high-dimensional normal distribution, which describes the population of each class. The symbol label corresponds to the class that has the maximum probability. Low probability values are used to reject symbols which can represent potential errors in the handwriting. The procedure to recognize a given mathematical expression begins by finding an optimal grouping of the written strokes into isolated symbols. The final grouping of stroked is determined

by evaluating all possible groupings and taking the one which minimizes a sum-cost function. This function is the sum of the log likelihood of the classifier's output of each symbol in the current partition. To make the optimization of the cost function manageable, its evaluation is constrained by the minimum spanning tree of strokes, considering the centers of strokes' bounding boxes as nodes of a completely connected weighted graph. Different combinations of subtrees of the minimum spanning tree are evaluated and the optimal one is taken as the final segmentation result.

The structural analysis in this system consists of locating a "key" symbol usually an explicit mathematical operator. Once the key symbols are located, the parse algorithm proceeds to find their corresponding operands, and partial subexpression are formed. The procedure is applied recursively until no more key symbols are found. The algorithm is extended to support parsing of superscripts, i.e. to non-explicit operators, but no support for subindexes is offered.

2.4.2 Free Hand Formula Entry System

The Freehand Formula Entry System is a pen-based equation editor developed by Smithies and Norvins [81] and distributed under the GNU General Public License. The program runs under Linux and MacOS X platforms.

The classification of symbols is done by using the nearest-neighbor method. The developers use confidence information supplied by the classifier to group strokes. Their method proceeds by generating all possible combinations of a fixed number of strokes (by default they take a maximum of four strokes), which potentially can constitute a single symbol. Once single symbol is classified, the confidence level of a combination correspond the lowest output of the classifier. Finally, the group with the highest confidence is taken and the first symbol in the group is returned and considered a correctly recognized character. The procedure is repeated, once again, when a fixed number of input strokes is reached.

For the structural analysis, they first used a method based on graph rewriting, similar to the method developed by Lavirotte and Pottier [52]. Figure 2.3 shows a version of their program, modified by Zanibbi [103], which uses the structural analysis method he developed.

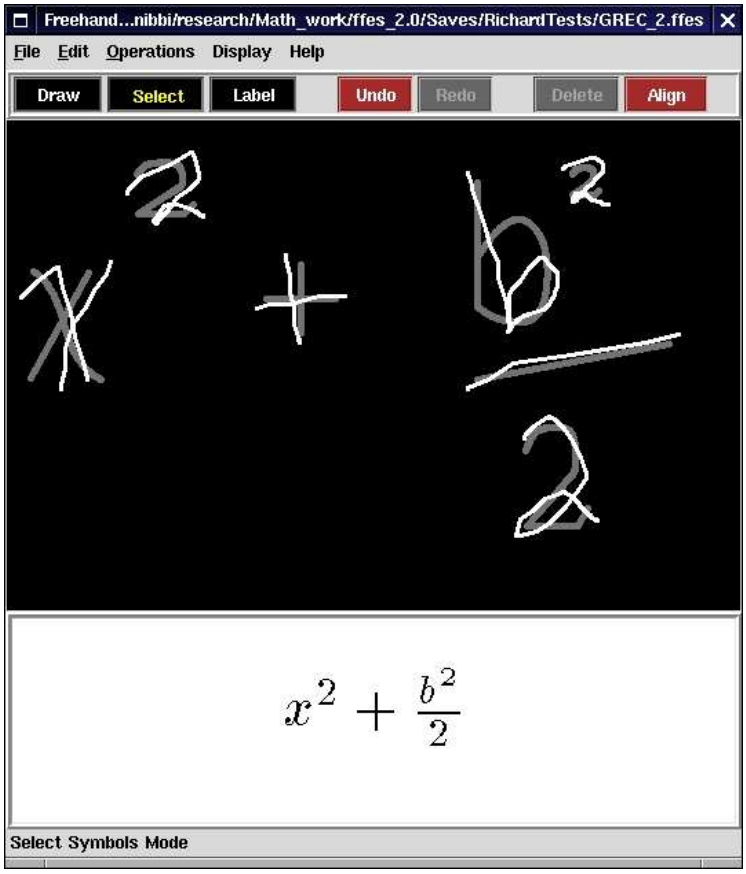


Figure 2.3: *Free Hand Formula Entry System*

2.4.3 Infty Editor

Infty Editor [63] is a commercial system specialized for creating mathematical documents. The editor is linked to the computer algebra system Mathematica by Mathlink. It also supports input and output of expressions in TEX format. The editor contains an real-time recognition system for mathematical expressions. See Fig. 2.4.

The recognition system combines segmentation and recognition of characters to remedy difficulties in structural analysis due to irregular symbol position and size. The rewriting puts symbols into extendable symbols and unextendable ones. The former are extended to form other symbols by adding more strokes and the latter are written with only one stroke. For example, F can be extended into E. When a stroke is classified as unextendable, the classification result is rewritten by the computer in the drawing area using a predefined prototype. If a stroke is classified as an extendable character, the system waits for the next strokes. The classification result is written automatically if a predetermined time interval has elapsed or the expected number

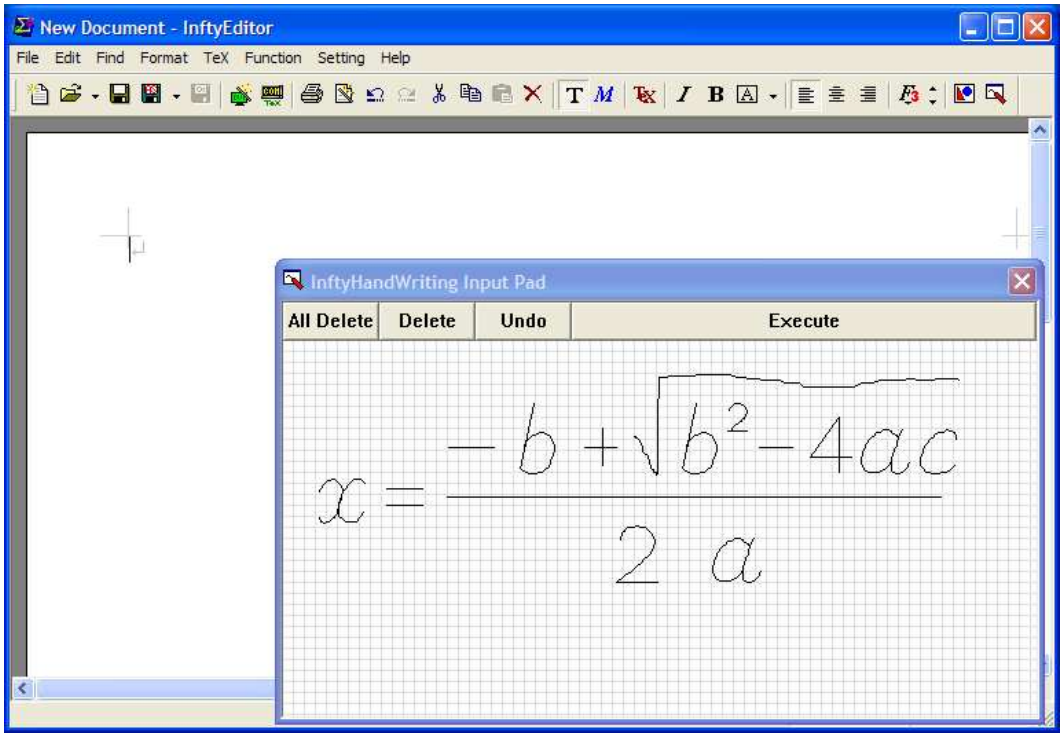


Figure 2.4: *Infty editor*

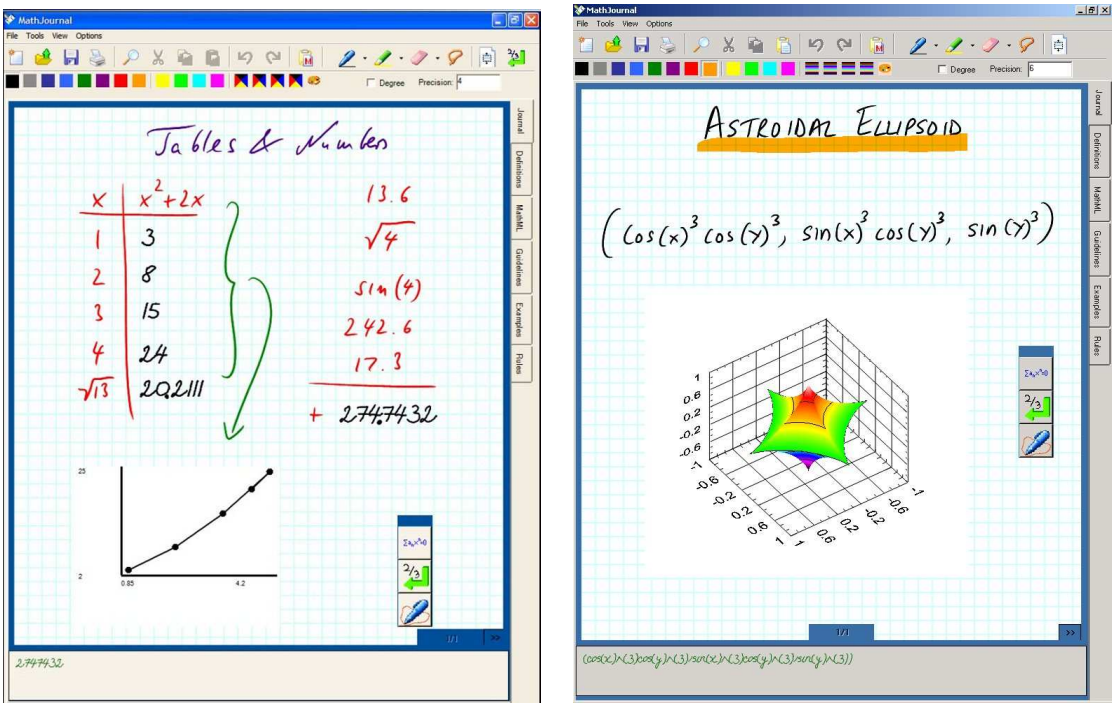
of strokes is reached.

During structural analysis, delayed symbols generate structural layout errors. The same occurs when super-indexes are added after the whole expressions is entered. For example, when writing the expression '$(-x)^n$' in the sequence 'x', '$-x$', '$(-x$', '$(-x)$', '$(-x)^n$', it is recognized as '$\_x()^n$'.

2.4.4 MathJournal

Wenzel and Dillner [98] describe another commercial product, MathJournal, developed for the Tablet PC version of the Windows platform. The interface is very similar to Microsoft's Journal program, which is included in the operating system, see Fig. 2.5. Apparently, the program is still under development and only descriptions of it are given. The developers also offer the xThink Calculator as an evaluation program. The recognition capabilities of this program are similar to the ones of MathJournal. It operates as a normal pocket calculator, the operations are done by recognizing a handwritten arithmetical expression.

MathJournal uses the recognizer integrated in the operating system for the classification of isolated handwritten characters. Although it is possible to recognize special



(a) (b)

Figure 2.5: *MathJournal.*

mathematical symbols and constants (square root, calculus operators, and the constant π), the system does not recognize Greek letters. The symbols recognized by the system are limited to the ones recognized by the Microsoft API.

In the description of the system, they mention that heuristics of graph rewriting and, when required, a minimum spanning tree construction are used during structural analysis.

The most relevant aspects of this system are its "solution engines". They process the recognized expressions in numeric, graphic, or symbolic formats. Diagrams, such as function tables, are processed and plotted by using curly braces and arrows as gestures. Similar gestures are used for the solution of equation systems or for plotting functions.