MfrDB: Database of Annotated On-line Mathematical Formulae

Jan Stria
Faculty of Mathematics and Physics
Charles University
Malostranské nám. 25, 118 00 Prague 1, Czech Republic
Email: stria.jan@gmail.com

Martin Bresler, Daniel Průša, Václav Hlaváč
Center for Machine Perception
Czech Technical University
Karlovo nám. 13, 121 35 Prague 2, Czech Republic
Email: {breslmar, prusapa1, hlavac}@cmp.felk.cvut.cz

***Abstract*—This paper announces a ground truthed database of on-line handwritten mathematical formulae. It have recently been collected in our group in connection with the research on methods for structural pattern recognition. Unlike the availability of handwritten characters or texts, collections of structural objects are rather scarce, thus we would like to provide them to the community. We also present the methodology and tools used for data acquisition. Finally, we report on our experiment with the automatic generation of additional samples. The process utilizes the dataset to extract statistical descriptions of symbols alignments and relative sizes.**

***Keywords*-database; mathematical formulae; web application; recognition;**

I. Introduction

Mathematical formulae recognition is a task tackled by several researchers in the past two decades [1], [2], [3], [4], [5]. The initial motivation was driven by demand for an automatic conversion of mathematical texts (printed books, handwritten notes) into an electronic form. The on-line formulae recognition is also becoming increasingly important, especially with the onset of tablets in recent years. Except a tablet pen, there are available other input devices such as a touchscreen, external digitizer or mouse. It is much more convenient for a common user to write a formula by hand rather than composing its structure piece by piece. The proof is the utility called Math Input Panel provided by Microsoft Corporation since Windows 7. This application performs recognition of on-line formulae and allows to transfer results in MathML [6] format to other programs (i.e. OpenOffice, Opera, Maple).

Even the methods and their implementations have reached a state where they can be successfully used in practice, there is still much room for improvement. The applications are usually sensitive to noise, do not take into account the semantics or require entries to be written with a certain level of care. Besides the improvements, there are theoretical questions related to possibilities and limits of formalisms suitable for expressing structure. The portability to other domains like chemical formulae, musical scores or electric circuits is studied as well. Thus, universal, robust and effective approaches are legitimately the subject of further research.

As for our activity in the field of structural pattern recognition, we developed and applied the idea of symbols segmentation and recognition driven by the structural analysis. The general framework was explicated in [7]. We use a two-dimensional coordinate grammar to model the structure. Analysis performed by the grammar is penalty oriented, derivations are assigned by a value determining its reliability. Results concerning our pilot study were published in [8]. Currently we work on the second version featuring new improvements and making the core algorithm accessible via a web-based interface [9].

During our research, we had to deal with the unavailability of suitable training set of annotated expressions. A sufficiently representative database is the basic prerequisite for tuning and testing the recognition method. Despite the fact there are many papers on on-line formulae recognition, as far as we know only one database has been already published as a part of MathBrush project [10]. This is in contrast to the collections of single characters [11], [12]. The situation is also better in the case of off-line formulae, mainly thanks to Infty Project [13] that provides ground truthed databases of mathematical documents as well as single expressions.

The conditions led us to a systematic development of an own database. We have created an infrastructure helping us to simplify the process of data acquisition and processing. It includes a web-based application used to collect formulae and a tool producing annotations. To test the data representativeness, we even designed and trained a statistical model of the formulae structure and used it to generate additional samples.

The purpose of this paper is to describe the database content and the methodology behind its creation. The text is organized as follows. Section II gives details on the database, including its statistics, data format and hosting web page. Section III explains principles and tools used to build the formulae set. Section IV describes the experiment with the generation of artificial samples.

II. Database description

We gathered 2018 formulae written by 232 people. Table I summarizes the overall statistics. The information about input device was not recorded at the beginning of the

978-0-7695-4774-9/12 $26.00 © 2012 IEEE
DOI 10.1109/ICFHR.2012.231

process, therefore it is not specified for a portion of the data. An example of taken samples is given in Figure 1.

Table I
OVERALL STATISTICS

Total number of formulae	2018
Distinct formulae	185
Users participated	232
Users with more than 10 formulae	36
Average strokes count per formula	18.54
Mouse input	604
Stylus input	71
Touchpad input	58
Not specified input	1285

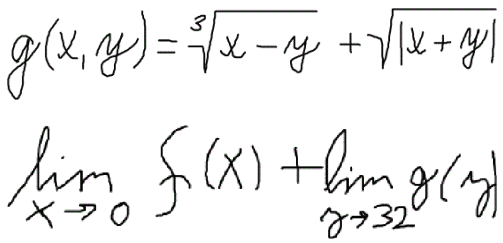


Figure 1. Two samples taken.

A. Procedure

All data were collected via a web-based interface, freely accessible by any Internet browser. A significant group of users was formed of students attending our courses. In general, the users had two choices – to enter a formula at their discretion, or to follow a displayed template. The latter option was more preferred. There were 130 different formulae used as the templates. Gathered inputs were manually examined whether they are syntactically correct, nontrivial and whether their quality is not too poor. Acceptable samples were annotated as described in Subsection III-B.

B. Data format

Two files are used to store one formula sample. Taken strokes are serialized in a InkML [14] file. An informational record and the ground truth resides in an XML file which consists of three top level elements. The origin is specified by `<FormulaInputInfo>`. It gives info on user id, IP address, time of creation and used input device (mouse, stylus, touch screen or unknown). Basic semantics is represented by a pure MathML under element `<mathML>`. Since not all elementary symbols are represented in MathML by standalone items, we designed an extension to MathML suitable for storing info on the strokes segmentation. This is included under element `<annotatedMathML>`. The newly introduced elements are listed in Table II.

`<aString>` groups strokes (forming a word) that cannot be (completely) separated into individual symbols. An example is cos written continuously by one move. Each of the elements has the attribute `strokeIds` which of value is a comma separated list of strokes identifiers.

Table II
MATHML EXTENSION

Element	Represents
`<aChar>`	character
`<aString>`	continuous string
`<aOpen>`, `<aClose>`	parenthesis
`<aFractionLine>`	fraction line
`<aSqrt>`	square root symbol

Since there is a very good support for reading/writing XML, the whole format as well as its parts can be easily translated into any other representation.

C. Web page

The dataset can be downloaded at the web page http://mfr.felk.cvut.cz/Database.html. The page contains details about the database. Some basic functionality is also provided. It is possible to preview any sample or to download a subset filtered out after specifying some parameters.

III. TOOLS FOR DATA CREATION

We have developed two applications supporting data acquisition. One requirement was to provide the participants with an easily accessible interface. The next requirements were easy maintainability of the infrastructure and possible reusability on another domains.

A. Web-based data collector

The application consists of a user interface running in a web browser at the client side which communicates with web services at the server side. The interface (Figure 2) has been developed in HTML5 [15] and Javascript. The canvas is implemented by jQuery UI [16] widget called jQueryInk [17]. It is publicly available for download as an open-source project. The web services have been written in C# using WCF (Windows Communication Foundation) technology and run on Microsoft Windows Server 2008 R2. Entered strokes are serialized to JSON format and sent to server via an Ajax request.

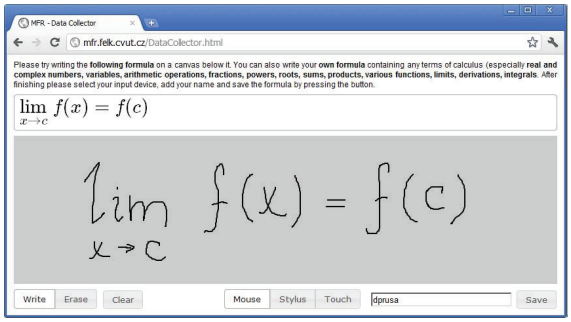


Figure 2. Data collector interface.

The writer should fill in his name first and then write and save several formulae. A hint in the form of a printed formula is provided by the server and displayed to the user. He can rewrite it or write a formula at his discretion. A collection of formulae templates acting as hints are maintained on the server. The hint is changed each time a sample is saved.

The system manages multiple accesses at the same time. This was successfully tested during classes when groups of students comprising 25 people were instructed to write samples. The application even works on the default mobile web browsers found in Android and iOS operating systems. Despite of some editing optimizations it is not always as smooth as in desktop browsers. However, the performance of these mobile devices grows rapidly, thus there is a well-founded reason to believe this will get better in the near future.

B. Formulae annotator

The formulae annotator is a desktop application serving to tag the acquired data. Since this is a laborious process, we tried to simplify it as much as possible. Assuming a collected formula is loaded to the tool, the first task is to create its MathML representation. If the user correctly rewrote a hint, it is possible to assign MathML of the related template. Otherwise, Math Input Panel, which is a COM control in Microsoft Windows 7, is invoked and the strokes are passed to it. The panel tries to continuously recognize inputs and shows the results. If the recognition does not match the expected outcome, there is a possibility to correct symbols or even select interpretation for parts of the formula. The desired recognition result is accessed programmatically as a text string containing MathML and it is copied to our tool.

Once we have the MathML notation of the formula, it is converted to the extended version by adding the special tags presented in Subsection II-B. Then, it is ready to be annotated. Since Math Input Panel does not provide any segmentation related info, the whole process has to be done manually. Each symbol included in the formula has to be bound with the appropriate strokes. This is typically done in a batch mode when nodes in MathML tree are traversed one by one and corresponding strokes are selected using the mouse. The tool in action can be seen in Figure 3.

C. Distance on MathML trees

When we wanted to count the number of distinct samples in the database, we had to deal with the fact that users sometimes do not rewrite the displayed template precisely. Moreover, formulae written freely by one or more users can be the same or very similar. Thus we based the estimation of distinct formulae on a metric.

Since MathML [6] has a tree structure (Figure 4), it is a logical choice to use the tree edit distance to express the similarity. The distance between two trees is defined

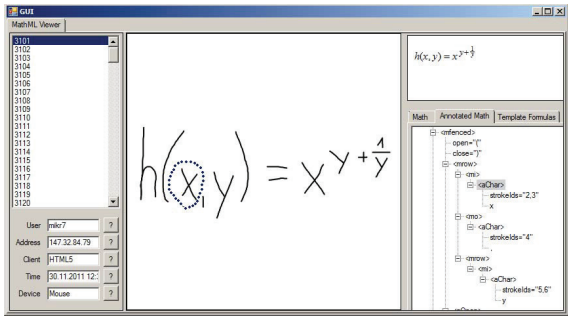


Figure 3. Annotator desktop application.

as the minimum cost of node editing operations (deletion, insertion, renaming) transforming one tree to the other. This is a generalization of the well known Levenshtein distance [18] on strings. To compute the tree distance we employed Zhang's algorithm [19] and set the cost of all operations to 1. In this case, one misrecognized symbol implies the distance 1. When a symbol is missing or the symbol category (variable, number, operator) differs, the distance is 2.

The metric also plays an important role in evaluation of recognition methods. Comparing the recognizer output directly with the ground truth does not grant a sufficiently fine benchmark. One usually wants to know how close the result is to the correct answer rather than to be only notified if the recognition succeeded or not.

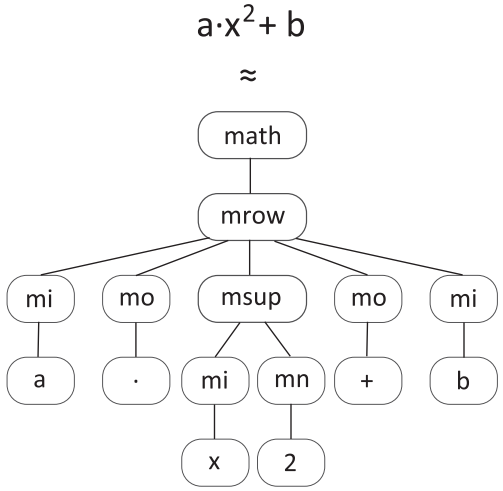


Figure 4. Standard MathML tree.

IV. Experiment

One of information the training set of mathematical formulae provides are the distributions of mutual positions between symbols. Handwriting style is much looser and more flexible than the printed one, thus the distributions are quite crucial for the structure description.

We present our model of the formula structure, comprising a two-dimensional grammar and probability distributions estimated based on the dataset. The accuracy of the model is demonstrated on the task of generating formulae. Artificially generated samples can serve as an auxiliary set for the development. The advantage is that we can cover templates different to those we used when collecting the data. The procedure can also be useful when a fast transcription of printed expressions to a handwritten form is needed. Finally, showing that the data produce a solid model is a justification of their quality as well as a significant step towards the recognition.

A. Grammar

We demonstrate the proposed grammar productions represented in a text form by the following snippet.

```
Sum->[sum]|LowBound@B|UpBound@T|Expr@R
AddSub->AddSubOp|Term@L|Expr@R
AddSubOp->[+]
Expr->Sum
```

Each line corresponds to one production. There is a source nonterminal on the left-hand side and a sequence of target nonterminals and terminals on the right-hand side, terminals being enclosed in brackets. The productions are of two types. When there is only one target element, we speak about a 1-production (the third and fourth line), otherwise we speak about an n-production (first two lines). 1-productions are used to maintain the grammar readable enough, while n-productions model spatial relations in the formula structure. Consider the first production describing a summation. The first target symbol `[sum]`, denoting the summation symbol, is the main element. Positions of remaining elements are defined relative to this main one, using suffixes starting by `@`. E.g., `LowBound@B` defines that the lower bound in a summation is positioned at the bottom of summation symbol. Eight different spatial placements are used: left, right, top, bottom, top-left, top-right, bottom-right and inside.

So far, the grammar contains about 80 n-productions and 200 terminal symbols. It describes all samples contained in the database. Contrary to MathML, each formula structure is determined unambiguously by the productions. We have a procedure that reads MathML and creates a derivation tree. This is an inner representation of the structure. The subtree rooted in the node represents a structural part of the formula. Each non-leaf node in the tree is assigned by a production which defines the relation between the node and its children.

B. Distributions

As we can see, the grammar provides only a rough description of elements positioning. A more precise characteristic has to be given statistically. The goal is to generalize the spatial placement of nodes and their relative sizes. Regarding the amount of training data and the memory needed for a representation, it is necessary to categorize nodes to some groups that exhibit the same spatial properties, so that the number of distributions can be reduced and shared per each category.

To achieve this, several types of terminals are distinguished. They relate to the alignment with respect to the ascender line, midline and baseline as it is shown in Figure 5. The types comprehend ascender terminals having the bottom of its bounding rectangle vertically aligned with the baseline and the top with the ascender line (number 2), lower terminals having the bottom aligned with the baseline and the top with the midline (letter x), central symbols having their center aligned with the midline (operator $+$), etc.

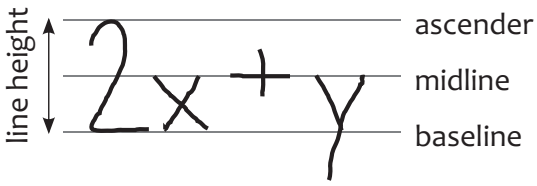


Figure 5. Types of terminals.

The type of the terminal determines the base, mid and ascender segment for the node as in Figure 6. Not all segments can be determined for each terminal node (e.g. there is only the mid segment for the operator $+$ because it is not aligned with the baseline or the ascender line). The horizontal coordinates of the mid segment are coordinates of its strokes' bounding rectangle. The horizontal coordinates of the base and ascender segments are determined based on the vertically local minimum and maximum of the strokes' horizontal coordinates. If two of the segments are defined for the node at least it is possible to estimate its line height as shown in Figure 5.

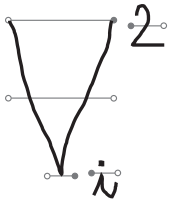


Figure 6. Segments for terminal *V*.

Different relative positions and sizes are extracted for derivation nodes depending on their spatial position in a given n-production. E.g., for the node describing addition $2x + y$ we utilize the relative position of $2x$ and $+$, relative position of y and $+$ and relative position and size of $2x$ and y. While determining the relative positions, the base, mid and ascender segments of the terminal nodes are employed.

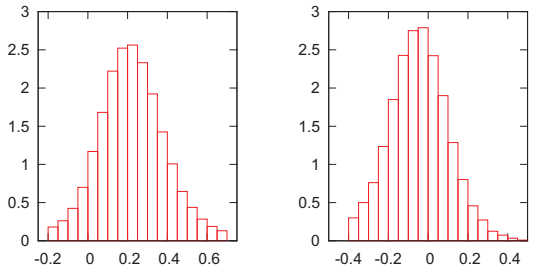
E.g. to describe the placement of $2x$ and $+$, the vertical and horizontal distance of mid segments of nodes x and $+$ are computed and normalized by the line height estimated from $2x$.

We have a set of rules defining which relative distances and sizes are evaluated for a particular type of production and how they are computed depending on the defined segments. Probability distributions of these values are estimated. The other set of rules describes where the same distribution is shared.

The distributions themselves are stored as histograms in one dimensional arrays. Each category includes one to three histograms, for the vertical and horizontal position and for the relative size. In total, we have 26 categories. Figure 7 shows extracted vertical and horizontal distributions describing the placement of $2x$ and $+$ in the working example. It would also be possible to have a two dimensional grid to represent both, horizontal and vertical, distances of two nodes but it would require more data as well. For simplicity, we decided to use in the experiment the one-dimensional variant. This choice is justified by our measurement confirming that the relative horizontal and vertical distance are largely independent. The absolute value of the correlation coefficient was usually about 0.1 and always under 0.2 for all 26 categories.

C. Results

To generate formulae, we have selected 500 MathML files from the Infty database as the drafts. The files were read, parsed and derivation trees were created. The distributions were employed to assign coordinates and sizes to the nodes. We smooth them first using a Gaussian filter, then we generated needed random values. Particular symbols were taken from our database. To produce a consistent output for each generated formula, we used symbols written by one user. Two instances generated from one MathML file are shown in Figure 8. All the samples are available at our web page.



(a) horizontal (b) vertical

Figure 7. Relative distances distributions.

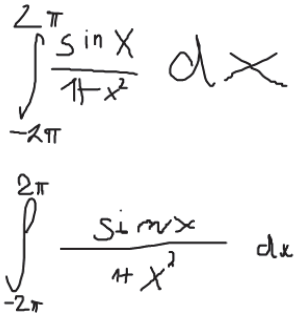


Figure 8. Two samples taken.

V. Conclusion

We have created a database of on-line mathematical formulae and made it publicly available. The release contains over 2000 samples, comprising about 27000 instances of individual symbols. In the future, we would like to continue in the incremental enlargement of this collection. The implemented infrastructure allows us to balance the content. Additional templates for hints will be supplemented to enhance the variety and to cover more mathematical constructs. The server could prefer to offer hints having little instances in the database.

We hope the database will be beneficial for the community and welcome any feedback on it.

Acknowledgment

The first author was supported by the Grant Agency of the Czech Republic under project P202/10/1333, the second author by the Grant Agency of the Czech Technical University in Prague, grant No. SGS12/187/OHK3/3T/13, and the last two authors by the Grant Agency of the Czech Republic under project P103/10/0783.

References

[1] B. P. Berman and R. J. Fateman, "Optical character recognition for typeset mathematics," in *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, ser. ISSAC '94. New York, NY, USA: ACM, 1994, pp. 348–353. [Online]. Available: http://doi.acm.org/10.1145/190347.190438

[2] S. Lavirotte and L. Pottier, "Mathematical formula recognition using graph grammar," in *Proceedings of the SPIE 1998*, vol. 3305, San Jose, CA, 1998, pp. 44–52.

[3] N. Matsakis, "Recognition of handwritten mathematical expressions," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1999.

[4] S. Smithies, K. Novins, and J. Arvo, "A handwriting-based equation editor," in *Graphics Interface*, 1999, pp. 84–91.

[5] R. Zanibbi, D. Blostein, and J. Cordy, "Recognizing mathematical expressions using tree transformation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1455–1467, 2002.

[6] "MathML2 recommendation." http://www.w3.org/TR/MathML2.

[7] M. Schlesinger and V. Hlaváč, *Ten lectures on statistical and structural pattern recognition*, ser. Computational Imaging and Vision. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2002, vol. 24.

[8] D. Průša and V. Hlaváč, "Structural construction for on-line mathematical formulae recognition," in *Proceedings of the Iberoamerican Conference on Pattern Recognition*. Springer Verlag, September 2008, pp. 317–324.

[9] J. Stria and D. Průša, "Web application for recognition of mathematical formulas," in *ITAT 2011: Proceedings of the Conference on Theory and Practice of Information Technologies*, M. Lopatková, Ed., vol. 788. Tilburg, Netherlands: CEUR Workshop Proceedings, September 2011, pp. 47–54.

[10] S. MacLean, G. Labahn, E. Lank, M. Marzouk, and D. Tausky, "Grammar-based techniques for creating ground-truthed sketch corpora," *International Journal on Document Analysis and Recognition*, vol. 14, no. 1, pp. 65–74, Mar. 2011. [Online]. Available: http://dx.doi.org/10.1007/s10032-010-0118-4

[11] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and J. S., "Unipen project of on-line data exchange and recognizer benchmarks," in *Proceedings of International Conference on Pattern Recognition*, 1994, pp. 29–33.

[12] M. Nakagawa and K. Matsumoto, "Collection of on-line handwritten Japanese character pattern databases and their analyses." *International Journal on Document Analysis and Recognition*, pp. 69–81, 2004.

[13] M. Suzuki, "Infty project." http://www.inftyproject.org/en/index.html.

[14] "Ink Markup Language (InkML)." http://www.w3.org/TR/InkML.

[15] "HTML5 working draft." http://www.w3.org/TR/html5.

[16] "jQuery UI library." http://jqueryui.com.

[17] "jQueryInk widget." http://plugins.jquery.com/project/Ink.

[18] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

[19] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," *SIAM J. Comput.*, vol. 18, pp. 1245–1262, December 1989. [Online]. Available: http://dl.acm.org/citation.cfm?id=76071.76082