

Mathematical Formulae Recognition using 2D Grammars

Abstract We present a method for off-line mathematical formulae recognition based on so called structural construction and two-dimensional grammars. In general, this approach can be successfully used in the analysis of images containing objects that exhibit rich structural relations. An important benefit of structural construction is not treating symbol segmentation in the image and structural analysis as two separate processes. This allows the system to avoid errors usually done during the segmentation phase. We have developed and tested a pilot study proving that the method is computationally efficient, practical and able to cope with noise.

1 Introduction

The paper serves two main purposes. First, it intends to point the reader's attention to the widely unknown theory of two-dimensional (2D) context-free grammars having the potential to cope with structural relations in images. Second, the paper demonstrates on a pilot study concerning recognition of off-line mathematical formulae that the 2D grammars have the potential to deal with real-life noisy images.

The enthusiasm for grammar-based methods in pattern recognition from the 1970's [6] has gradually faded down due to inability to cope with errors and noise. Even mathematical linguistics, in which the formal grammar approach was pioneered [4], has tended to statistical methods since the 1990s.

M.I. Schlesinger from the Ukrainian Academy of Sciences in Kiev has been developing the 2D grammar-based pattern recognition theory in the context of engineering drawings analysis since the late 1970s. His theory was explicated in the 10th chapter of the monograph [15] in English for the first time.

The first author of this paper studied independently the theoretical limits of 2D grammars [12] and proved them to be rather restrictive.

The main motivation of the authors of the reported work is to discover to what extent the 2D grammars are applicable to practical image analysis.

This paper summarizes results achieved by implementing

a pilot study aiming at off-line recognition of mathematical formulae. We have chosen this application domain because there is a clear structure in formulae and works of others exist which can be used for comparison. We have already presented an introduction into 2D CF grammars and results of the method for off-line recognition of black and white, handwritten images of formulae in [13].

Let us categorize the approaches to mathematical formulae recognition along two directions:

- *on-line* recognition (the timing of the pen strokes is available) versus *off-line* recognition (only an image is available).
- *printed* versus *hand-written* formulae.

We deal with off-line recognition of hand-written and printed formulae in this contribution. Gray-scale images are considered.

2 State-of-the-Art

Formulae recognition has been a widely studied task. Several approaches from pattern recognition were adopted as well as new methods were motivated and designed by this particular task. A taxonomy of the methods is given in [3]. There are only a few commercial products performing formulae recognition. The most prominent is probably xMath-Journal software [18] for Tablet PCs which uses on-line recognition. This software serves as a sophisticated calculator allowing inputs to be written by hand.

Most of the known methods follow two-phase procedure:

1. Detection of individual symbols by image segmentation and labelling symbols using pattern recognition techniques.
2. Structural analysis of relations among labelled symbols.

Classical approaches known from Optical Character Recognition were adopted to perform the symbols recognition phase. Images, resp. pen strokes are segmented and a classifier is used to assign symbols to segments. There are also works performing symbol detection, segmentation and labelling during a single simultaneous process using Hidden Markov Models [17].

While the methods for segmentation differ depending on whether on-line, resp. off-line recognition is considered, this is not the case for structural analysis. The goal is to detect relations among recognized symbols considering their positions, sizes, fonts, etc., i.e. to construct a tree corresponding to the represented formulae. Formalisms related to the analysis include various types of grammars like geometric grammars [7] or graph grammars [10]. The notion of context-free grammars appears in [11]. There are also methods not based on parsing, but rather on different principles like finding minimal spanning tree [5] which is probably the most known and used of them.

Criticism of the commonly used methods concerns the image segmentation which is usually done without any knowledge of the formulae structure. It is hardly possible to recover from errors made during symbol segmentation phase. There have been attempts to employ additional error corrections schemes. However, this postprocessing corrections do not fit naturally into the pattern recognition process.

There are some works that partially deal with this problem. Local properties of mathematical structures are considered when computing the best segmentation for formulae on-line recognition in [16]. Furthermore, more best matches returned by an OCR tool for a segmented region are passed into the structural analysis in [10]. This allows to minimize global penalty of recognition while possibly increasing local penalty of recognized symbols. However, there are still many cases these two approaches cannot deal with.

The approach based on the two-dimensional context-free (2D CF) grammars and a general structural construction tries to solve the stated problem [15]. It has been already applied to images of musical scores [14] or electrical circuits [9].

3 Idea of Structural Construction

Principles related to the structural construction are explained in this section. The recognition process works with regions of the input image. A *region* is a connected set of pixels having some common property. The region is assigned a *label* determining which structure was recognized to be represented by the region (e.g., a fraction line as a part of a formula) and also a penalty giving the cost of derivation of the region. We consider two finite sets of labels: V_T is a set of *terminals* and V_N is a set of *nonterminals*. There is also one distinguished label $S_0 \in V_N$ corresponding to the whole structure we want to recognize.

At the beginning, there are so called terminal regions only, labelled by terminals. These regions can correspond to single pixels of the input image or to regions detected by an external tool. Usually, the property of regions is that they decompose an image into disjoint components. We relax this requirement and allow that regions to share some pixels.

A set of *rules* specifying how labelled regions can be combined to produce larger regions (their unions) is defined. A rule $N \rightarrow N_1, \dots, N_k$ is interpreted as follows. If there are regions $R_1, \dots, R_k$ labelled by $N_1, \dots, N_k$, their sizes and positions fulfil some constraints connected to the rule then their union $R = \bigcup_{i=1}^k R_i$ with the label N can be

derived. The *penalty* of this derivation is computed from penalties of particular regions R_i . The rules are applied during an iterative process to derive larger and larger regions. The process ends when all possible regions have been derived. If the whole image was assigned by S_0 and the penalty of related derivation fits into some limit then the desired structure in the image was successfully recognized.

The described recognition process is too general and would be highly complex in time and space. Because of this, we need to seek some convenient unifications of rules and region shapes that will lead to an acceptable implementation. The formalisms resulting in these unifications can be based on 2D CF grammars [15]. Permitted regions are only rectangles to suppress the complexity of derivations. Rules (or *productions* – to be consistent with the grammar terminology) are of the following three types:

P1. (column concatenation) $N \rightarrow A|B$

P2. (row concatenation) $N \rightarrow \frac{A}{B}$

P3. (renaming) $N \rightarrow A$

N is a nonterminal, A and B are terminals or nonterminals. The production P1 resp. P2 can be applied to touching rectangles of the same number of rows, resp. columns. The well known Cocke-Younger-Kasami algorithm [8, 19, 1] for recognition of languages generated by one-dimensional context-free grammars can be generalized on 2D CF grammars [15]. Time complexity of the algorithm is

$$\mathcal{O}(m^2 n^2 (m + n)),$$

where m , resp. n denote the number of rows, resp. columns of the input image.

Two observations can be made: The complexity is still high and the constructs supported by 2D CF grammars are not rich enough to model relationships among symbols in mathematical formulae. We will address these two issues in the following section by introducing a suitable extension of the grammars.

4 Mathematical Formulae Recognition

The main ideas taken from the structural construction we follow in our approach can be expressed in the following way:

Perform ‘rough segmentation’ of the input image. For each possible elementary symbol, find all occurrences of it and let the structural analysis decide, structure of which formula fits the input image best and how does the image segmentation looks like.

4.1 Pilot Study Overview

The pilot study supports elements and constructs as numbers, variables, brackets, subscripts, superscripts, basic unary and binary operators, power to operations, fractions, sums, integrals and square roots.

Our main goal was to investigate whether the structural construction can be successfully applied to off-line formulae recognition. In particular, we were interested how the method can deal with the following situations.

$$\begin{array}{ll} \text{a) } \frac{2-3}{6} & \text{b) } \sum_{k=1}^N A_k \\ \text{c) } \frac{1}{2}-3 & \text{d) } \frac{2}{3} + A \frac{1}{9} \end{array}$$

Figure 1: Corner cases we would like to handle.

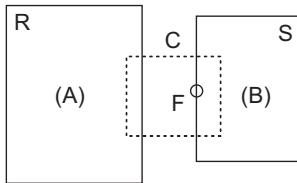


Figure 2: General production scheme of $N \rightarrow A \oplus B$.

- a) Symbols touching vertically or horizontally.
- b) Symbols split into several components.
- c) Ambiguities.
- d) Misplaced symbols.
- e) Noise.

Examples of these situations are given in Figure 1. Cases a) and b) demonstrate standard formulae. Case c) illustrates a fraction line and a minus sign represented by the same image, the meaning is being given by the context. And finally, case d) includes an additional symbol A that is by mistake placed into the formula. We require our method to exclude such a misplaced symbol and recognize the formula composed of the other symbols.

4.2 Extension of 2D Context-free Grammars

We use an extension of 2D CF grammars to model relationships between mathematical symbols. In this section, we give a description of the productions form and their application.

Let $N \rightarrow A \oplus B$ denote a production of our 2D CF grammar extension. The interpretation is similar to the interpretations of productions $N \rightarrow A|B$ and $N \rightarrow \frac{A}{B}$, regions labelled A and B can be united, producing a region labelled N . But this time we do not require the regions to touch each other. Permitted mutual positions of the regions are defined by a constraint that is connected to the *production*. The form of the constraint is depicted in Figure 2.

R and S are two regions in the image, F is a *feature point* of S (the center of left border in this particular case). C is a dashed rectangle the size and position of which is given relative to the size and position of R . It represents the mentioned constraint. The considered production can be applied when the following conditions are fulfilled:

- R , resp. S can be labelled by A , resp. B .

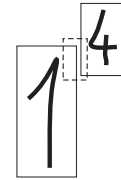


Figure 3: Example of the production usage.

- feature point F is located inside C .

The resulting region is the smallest rectangle containing R and S . Feature points are some specific points of a region. We usually use corners and centers of bounding edges. We also compute baseline (when a new region is derived) and take the intersection with left or right bounding edge.

Figure 3 shows usage of a production to model ‘power to’ relationship where the constraint is defined for the bottom-left corner of the region storing the symbol four.

The penalty of the derived region is computed based on the following factors:

- Used production.
- Penalties of R and S .
- Sum of pixel’s values in the new region that are neither in R nor S .
- Relative sizes and positions of R and S .

To finish the description we will make three remarks. First, note that R and S can overlap. Second, we have defined the constraint on the location of S , assuming we know the location and size of R . In our structural analysis, we will also need the opposite case, i.e., to look for S while knowing R . We slightly extend the production form by attaching one more constraint C' of the mentioned meaning. And finally, in some cases of formulae constructs, it is more convenient to compose a new region from three regions instead of two (consider, e.g., a variable with a subscript and superscript). This can be easily supported by introducing productions of the form $N \rightarrow A_1 \oplus A_2 \oplus A_3$.

4.3 Implementation

Our software consists of two independent layers. The first layer performs *terminals detection*, while the second one is responsible for *structural analysis*. An external OCR tool is used within the first layer. Structural analysis is driven by a grammar defining supported mathematical formulae. The parsing algorithm is of a general nature and it can be used to recognize another types of structures if supported by a proper grammar.

4.3.1 Terminals Detection We have developed an OCR tool for classification of image regions. The tool is based on a simple extraction of features from the input picture. The k -nearest neighbor classifier is implemented to classify the extracted vectors. The tool is trained for two sets of symbols – handwritten, resp. printed symbols.

We have tested two methods for terminal symbols detection. The first method works with rectangular scanning windows of some specific sizes. Each of the windows is being moved through the input image. Whenever it is in a new position, its content is evaluated by the OCR tool (provided that the sum of pixel's values exceeds some defined threshold). The result of the evaluation is a set of terminals assigned by a penalty, where the penalty corresponds to the belief that the scanning window stores a particular symbol. Only the terminals with a sufficiently low penalty are included in the result. For example, we have a window to detect subscripts and another window to detect variables and numbers. Sizes of the windows are determined by expected sizes of symbols in the formulae which means the method requires tuning for typical inputs.

It would be ideal in theory if we could use views of all sizes to scan the image, however, this approach is time expensive. Limiting the sizes of used views results in an acceptable performance but can miss some symbols in the image when their size differs from the expectation. Because of this we have tested the second method based on preprocessing the content of the image by computing connected components. Selection of views that are evaluated by the OCR tool is driven by these components. The bounding rectangles of the following areas are chosen:

- the components themselves,
- divisions of the components – up to two splitting horizontal and vertical points are considered,
- combinations of neighboring components.

4.3.2 Structural Analysis We describe the algorithm that is used for the structural analysis phase. To simplify the description, we do not explain how information needed to track feature points and derivation trees is being updated. Just note that whenever a region R is labelled by N , we record by which production this was derived. This exactly information needed to build the derivation tree.

1. Let $\mathcal{R}$ be a list of triples (R, N, p) , where R is a region, N label assigned to R and p penalty of this assignment. Initialize $\mathcal{R}$ by results of the terminals detection phase.
2. Iterate through $\mathcal{R}$. Let $(R, L, p) \in \mathcal{R}$ be the current element. For each production $N \rightarrow A \oplus B$ such that $L = A$ or $L = B$, take the rectangular area C defined by the constraint of the production and find subset $\mathcal{S} \subseteq \mathcal{R}$, where for each $(R', L', p') \in \mathcal{S}$ the production can be applied on R and R' . Let $\bar{R}$ be the derived region, $\bar{p}$ penalty of the derivation. If $\bar{p}$ is greater than some threshold then continue by the next iteration, otherwise check whether $\bar{R}$ has been already labelled by N . If not then append $(\bar{R}, N, \bar{p})$ at the end of $\mathcal{R}$. If there is already $(\bar{R}, N, p_2)$ in $\mathcal{R}$ and $\bar{p} < p_2$ then remove $(\bar{R}, N, p_2)$ from $\mathcal{R}$ and append $(\bar{R}, N, \bar{p})$, otherwise ignore the new derivation.
3. A formula is successfully recognized when the bounding rectangle of the input is labelled by S_0 . If not then we can look for the largest region in $\mathcal{R}$ labelled by S_0 .

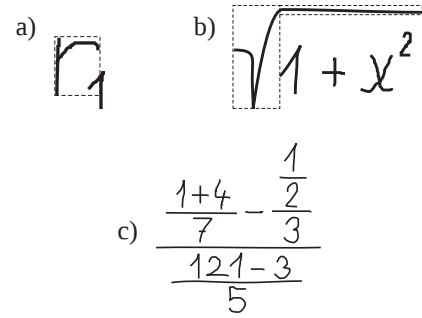


Figure 4: a) Bounding box of r contains a part of the subscript. b) Square root symbol composed of two parts. c) Formula with several fraction lines.

To make the algorithm fast, we use a data structure storing points in a plane, allowing it to effectively evaluate queries of the type: for a rectangle, return the points that are located inside the rectangle (so called *orthogonal range searching*). A suitable data structure allowing to search in time $\mathcal{O}(\log n)$ can be constructed [2].

Our conclusions on time complexity are based on empirical data, we do not give an exact formula since it depends on many factors, including the number of the terminals detected during the first phase. Compared to the generalized Cocke-Younger-Kasami algorithm, time complexity is lower because the algorithm does not process all rectangles in the input. It would be possible to derive some upper bound on time, but it does not give a good idea about expected time. Instead of doing it, we rather discuss the most problematic case in the section regarding results.

5 Results

We have implemented the pilot study in Java. It includes an user interface allowing to browse formulae images, run the recognition on them and display results. Except the results, the interface also provides information helping to understand and tune the process of structural analysis. For example, it is possible to query for all regions labelled by a specific nonterminal, for penalties of related derivations, etc.

The implementation has been tested on two sets of formulae – handwritten and printed, both sets containing over 200 images. We can conclude that after tuning the grammar there were no problems with correctness of the structural analysis. The problems we have encountered are connected mainly to the terminals detection phase.

We have faced some limitations when working with rectangular regions. Not all symbols in a formula can be separated by rectangles. This is mostly case of handwritten formulae. Figure 4 a) shows one of the simplest examples. The bounding box of symbol r contains a part of the subscript. It has an impact on recognition of r , which is assigned a bigger penalty in this case. In general, the recognition does not give good results, when the bounding boxes overlap too much.

We were also forced to compose some elementary symbol from more components due to the mentioned limitation. A typical example is a square root which we consider to be

formed of two parts (the square root argument can be treated as an additional, third part) – see Figure 4 b).

Other problems are connected to fraction lines. Continual subparts of a fraction line are also recognized as fraction lines. This leads to a large number of terminal symbols and possible combinations among them to be checked during the structural analysis. Figure 4 c) shows an image on which the problem starts to be visible. Recognition of this formulae takes about 20 seconds and grows fast for larger fractions (note that the recognition of each formula depicted in Figure 1 takes up to 2 seconds). We have implemented a preprocessing of detected fraction lines that reduces their number. It would be also possible to implement a special method (separated from the OCR tool) for fraction lines detection.

6 Conclusion and Future Work

We have showed that the method of structural construction can be applied for off-line mathematical formulae recognition. It is more suitable for printed formulae since they exhibit more regularities and the impact related to the rectangular regions limitation is not so high.

Our main contribution to the area of formulae recognition is summarized in the following achievements:

- Real segmentation of the image is driven by structural analysis (no error corrections are needed). We took advantage of the rich formula structure which allows this approach.
- Structural analysis is robust. It is penalty oriented and searches for the formula structure that best matches the input image. It can easily deal with noises, including misplaced symbols.
- We have designed an extension of 2D CF grammars powerful enough to express the formulae structure. It can be also effectively parsed (thanks to constraints defined via rectangles and the usage of data structures for orthogonal range searching).

Our future plans include the usage of learning methods. Provided that a sufficiently large set of formulae is collected, the methods can be applied on learning etalons of terminal symbols and productions parameters. The learned etalons can improve the terminals detection phase, while the learned productions parameters will improve tuning of the grammar for a concrete typesetting style of formulae (so far we have tuned these parameters manually).

Our second plan is to adopt ideas of structural construction to implement on-line formulae recognition. Again, we would like to design a method where segmentation of strokes is driven by the structural analysis. It is an advantage that the analysis works regardless the input type, thus it is sufficient to focus on strategies that select candidates for segmented symbols. The results achieved on off-line recognition promise the proposed method will work in the on-line case as well.

Acknowledgement

The authors were supported by the EC project INTAS 04-77-7347 PRINCESS and by the Czech Ministry of Education under project 1M0567.

References

- [1] A.V. Aho and J.D. Ullman. *The theory of parsing, translation, and compiling*, volume 1 – Parsing. Prentice-Hall, Englewood Cliff, New Jersey, 1971.
- [2] M. Berg, O. Schwarzkopf, M. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.
- [3] K-F. Chan and D-Y. Yeung. Mathematical expression recognition: a survey. In *IJDAR*, volume 3, pages 3–15, 2000.
- [4] N. Chomsky. *Syntactic Structures*. Mouton and Co, The Hague, 1957.
- [5] Y. Eto and M. Suzuki. Mathematical formula recognition using virtual network. In *Proceedings of the ICDAR 2001*, pages 762–767, 2001.
- [6] K.S. Fu. *Syntactic Methods in Pattern Recognition*. Academic Press, New York, 1974.
- [7] P. Garcia and B. Couasnon. Using a generic document recognition method for mathematical formulae recognition. In *Proceedings of the SPIE 1998*, number 2390, pages 236–244, Berlin, Germany, 1998. Springer-Verlag.
- [8] T. Kasami. An efficient recognition and syntax analysis algorithm for context-free languages. Scientific report AFCLR-65-758, Air Force Cambridge Research Laboratory, Bedford, Mass., USA, 1965.
- [9] V.M. Kiyko. Recognition of objects in images of paper based line drawings. In *Third International Conference on Document Analysis and Recognition*, pages 970–973, Montreal, 1995.
- [10] S. Laviotte and L. Pottier. Mathematical formula recognition using graph grammar. In *Proceedings of the SPIE 1998*, volume 3305, pages 44–52, San Jose, CA, 1998.
- [11] E.G. Miller and P.A. Viola. Ambiguity and constraint in mathematical expression recognition. In *AAAI/IAAI*, pages 784–791, 1998.
- [12] D. Průša. *Two-dimensional Languages*. PhD thesis, Faculty of Mathematics and Physics, Charles University, Prague, 2004.
- [13] D. Průša and V. Hlaváč. 2d context-free grammars: Mathematical formulae recognition. In Jan Holub and Jan Žďárek, editors, *Proceedings of the Prague Stringology Conference '06*, pages 77–89, Prague, Czech Republic, August 2006. Prague Stringology Club, ČVUT.
- [14] B. Savchynsky, M.I. Schlesinger, and M.O. Anochina. Parsing and recognition of printed notes. In *Proceedings of the conference Control Systems and Computers*, pages 30–38, Kiev, Ukraine, 2003. in Russian, preprint in English available.
- [15] M.I. Schlesinger and V. Hlaváč. *Ten lectures on statistical and structural pattern recognition*, volume 24 of *Computational*

601	<i>Imaging and Vision</i> . Kluwer Academic Publishers,	661
602	Dordrecht, The Netherlands, 2002.	662
603	[16] K. Toyozumi, N. Yamada, K. Mase, T. Kitasaka,	663
604	K. Mori, Y. Suenaga, and T. Takahashi. A study of	664
605	symbol segmentation method for handwritten	665
606	mathematical formula recognition using mathematical	666
607	structure information. In <i>17th International Conference on</i>	667
608	<i>Pattern Recognition (ICPR'04)</i> , volume 2, pages 630–633,	668
609	2004.	669
610	[17] H.J. Winkler and M. Lang. Online symbol	670
611	segmentation and recognition in handwritten	671
612	mathematical expressions. In <i>Proceedings of the IEEE</i>	672
613	<i>International Conference on Acoustics, Speech, and Signal</i>	673
614	<i>Processing</i> , volume 4, pages 3377–3380, Munich,	674
615	Germany, 1997.	675
616	[18] Mathjournal 2.0. a software for formulae recognition	676
617	from the xThink company, 2006.	677
618	http://www.xthink.com/MathJournal.html .	678
619	[19] D.H. Younger. Recognition of context-free languages	679
620	in time n^3 . <i>Information and Control</i> , 10:189–208, 1967.	680
621		681
622		682
623		683
624		684
625		685
626		686
627		687
628		688
629		689
630		690
631		691
632		692
633		693
634		694
635		695
636		696
637		697
638		698
639		699
640		700
641		701
642		702
643		703
644		704
645		705
646		706
647		707
648		708
649		709
650		710
651		711
652		712
653		713
654		714
655		715
656		716
657		717
658		718
659		719
660		720