2009 10th International Conference on Document Analysis and Recognition

Syntactic detection and correction of misrecognitions in mathematical OCR

Akio Fujiyoshi
Department of Computer and Information Sciences, Ibaraki University
4-12-1 Nakanarusawa, Hitachi, Ibaraki, 316-8511, Japan
fujiyosi@mx.ibaraki.ac.jp

Masakazu Suzuki
Graduate School of Mathematics, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan
suzuki@math.kyushu-u.ac.jp

Seiichi Uchida
Graduate School of Information Science and Electrical Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka, 819-0395, Japan
uchida@is.kyushu-u.ac.jp

Abstract

This paper proposes a syntactic method for detection and correction of misrecognized mathematical formulae for a practical mathematical OCR system. Linear monadic context-free tree grammar (LM-CFTG) is employed as a formal framework to define syntactically acceptable mathematical formulae. For the purpose of practical evaluation, a verification system is developed, and the effectiveness of the method is demonstrated by using the ground-truthed mathematical document database InftyCDB-1 and a misrecognition database newly constructed for this study. A satisfactory number of misrecognitions are detected and delivered to the correction process.

1. Introduction

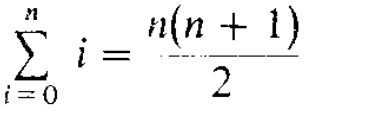
Mathematical notations have been developed in order to express mathematical information clearly and concisely. Fortunately, there exist notational conventions that are globally shared with mathematicians. In mathematical OCR [1], it is natural to think that such notational conventions help to detect and correct misrecognitions of characters and structures in mathematical formulae. This paper proposes a syntactic method for detection and correction of misrecognized mathematical formulae for a practical mathematical OCR system.

In order to define syntactically acceptable mathematical formulae, we employ linear monadic context-free tree grammar (LM-CFTG) [2] as a formal framework. As shown in Figure 1, a mathematical OCR system offers a tree representation of a mathematical formula from a scanned image. The most suitable tree representation needs to be selected from numerous recognition candidates. Therefore, we need a grammar formalism that can define a set of tree structures. An LM-CFTG defines a set of tree structures by arranging fan-out rules and context-free rules, where fan-out rules are used to describe the structural growth of a tree, and context-free rules are used to describe linear growth. For example, some fan-out rules and context-free rules are illustrated in Figure 2.

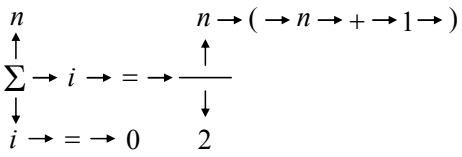
There are other syntactic approaches in mathematical OCR. Anderson [3] introduced a syntactic recognition method with a coordinate grammar. A coordinate grammar takes as input a set of symbols attributed with bounding-box and center coordinates. Chou [4] proposed the use of a two-dimensional stochastic context-free grammar for syntactic recognition so that a grammar directly takes pixels of an original scanned image. Graph rewriting techniques [5, 6, 7] are also syntactic approaches because graph rewriting rules can be organized into a graph grammar. These approaches differ from the formal framework used to define mathematical notations and the stage where syntactic analysis starts during a recognition process.

The employment of LM-CFTG allows us to build a very fast verification system. We can use fast parsing algorithm [8] for LM-CFTG, and, moreover, the parsing algorithm can be speeded up by implementing well-established parsing techniques [9, 10] for context-free grammar (CFG).

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.150

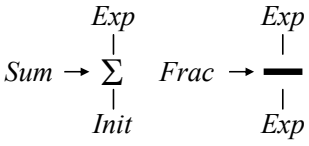


Scanned image



Tree representation

Figure 1. A result of structural analysis of a mathematical formula.



Fan-out rules

$Init \rightarrow Exp = Exp$ $\quad Term \rightarrow Factor$

$Exp \rightarrow Exp + Exp$ $\quad Factor \rightarrow 1$

$Exp \rightarrow Term$ $\quad Factor \rightarrow n$

$Term \rightarrow Term\ Term$ $\quad Factor \rightarrow (\ Exp\ )$

Context-free rules

Figure 2. Rules of an LM-CFTG.

Theoretically, the worst-case running time of a verification process for a mathematical formula is $O(n^3)$, where n is the number of symbols of a formula. Since the parsing algorithm can be speeded up, our experimental results show that the average-case running time of a verification process is proportional to the number of symbols of a formula.

One of the final aims of this study is to completely define acceptable mathematical notations. In other words, we want to have a grammar to verify any mathematical notation that has appeared, or will appear, in a long-term build-up of mathematical documents. As mathematicians constantly invent new mathematical notations and many dialects coexist, it seems impossible to establish such a grammar. However, we think it is possible to introduce an LM-CFTG that improves a practical mathematical OCR system for all mathematical documents concerning detection and correction of misrecognitions.

The effectiveness of the proposed syntactic method for detection of misrecognized mathematical formulae is evaluated. A new database is constructed for this study by collecting misrecognized mathematical formulae generated by a mathematical OCR system. We execute the verification system on the misrecognition database, and a satisfactory number of misrecognitions are detected and delivered to the correction process.

This paper is organized as follows: In Section 2, the grammar defining syntactically acceptable mathematical formulae is explained; in Section 3, the misrecognition database is described; in Section 4, the experimental results are shown; and in Section 5, the conclusion is drawn and future work discussed.

2 Syntactically acceptable mathematical formulae defined by an LM-CFTG

We employ linear monadic context-free tree grammar (LM-CFTG) [2] as a formal framework for the definition of syntactically acceptable mathematical formulae. To choose an appropriate grammar formalism, it is necessary for a grammar formalism to have sufficient descriptive power to process a diversity of mathematical formulae. In addition to descriptive power, we also require that a grammar formalism be accompanied by very fast parsing algorithm. We can use fast parsing algorithm [8] for LM-CFTG, and, moreover, the parsing algorithm can be speeded up by implementing well-established parsing techniques [9, 10] for context-free grammar (CFG).

An LM-CFTG is defined by arranging *fan-out rules* and *context-free rules*. Fan-out rules are used to define possible structural configuration of mathematical formulae. We should arrange them for symbols which are possibly connected with adjunct symbols. Examples of those symbols are “capital sigma” for summation, “capital pi” for product, “radical sign” for square root, “long bar” for fraction, “integral sign” for definite integral, etc. Because any variable may have a subscript, we arrange a fan-out rule for all italic alphabet symbols. Context-free rules are used to define possible linear sequences of symbols of mathematical formulae. Context-free rules constitute a CFG, and thus we can use well-established parsing techniques for CFG.

The grammar consists of 39 fan-out rules and 214 context-free rules. On the development of the grammar, we try to arrange context-free rules so that they constitute a deterministic context-free grammar (DCFG) [9] because we can take advantage of a linear-time parsing technique for DCFG [10]. Unfortunately, we need to add some context-free rules, which break the condition of DCFG, and this is the reason why a verification process of some exceptional mathematical formulae requires cubic time. Most of those context-free rules are related to the vertical-bar symbol be-

cause the usage of vertical bar is too diverse: absolute value, divides, conditional probability, norm of a vector, etc.

3. Misrecognition database

A new database was constructed for this study by collecting misrecognized mathematical formulae generated by a mathematical OCR system. The misrecognition database consists of $2,000$ misrecognized mathematical formulae collected from 32 pure mathematical articles. We also constructed a database of correctly recognized mathematical formulae, which consists of $3,000$ formulae from the same articles. We collected only formulae consisting of 10 or more symbols because syntactic analysis requires a certain formula size. We plan to publish both databases on the website of Infty Project [11].

The misrecognized mathematical formulae in the misrecognition database are classified into the following six types of misrecognitions:

(1) character misrecognitions,

(2) inaccurate selections of a parent symbol,

(3) unsuitable relationships with a parent symbol,

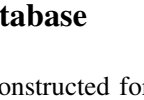
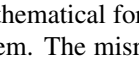
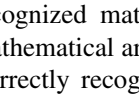
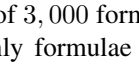
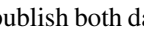
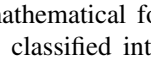
(4) incorrect separations of math-area and text-area,

(5) erroneous deletions of a symbol as noise, and

(6) mistaken recognitions of noise as a symbol.

Some formulae are classified into plural types of misrecognitions. In Table 1, examples of original images of a part of misrecognized mathematical formulae and corresponding misrecognized results concerning each type of misrecognition are shown. As for (1), the fraction '$\frac{1}{2}$' is misrecognized as a single asterisk. We classified this type of misrecognition as a character misrecognition. In reference to (2), the parent of the symbol 'j' is misrecognized. The symbol 'j' should have been horizontally connected to the symbol 'i'. With regard to (3), the relationship between the vertical bar and the symbol 'n' is misrecognized. The symbol 'n' should have been a superscript of the vertical bar. Concerning (4), the quotation dash is misrecognized as a minus sign. The quotation dash should have been separated as in text-area. Regarding (5), the accent bar over the symbol 'E' is deleted as noise. And about (6), the dirt between the right parenthesis and the colon is misrecognized as a symbol 'ρ'.

Table 1. Six types of misrecognitions.

Type	Original Image	Misrecognition
(1)		$(\log \frac{1}{1-\lvert z\rvert})^*$
(2)		$\sum h_i^{\mu_j} e_\mu$
(3)		$\lvert K_i(p,e)\rvert n$
(4)		$-q_k\delta_k \in Z$
(5)		$E_x^u(\delta) \to \bar{E}_x^s(\delta)$
(6)		$P_{Y_d}(t)_\rho :=$

4. Experimental results

In order to evaluate the proposed syntactic method based on LM-CFTG, we experimentally built a verification system in accordance with the method. For implementation of the system, the program was written in C language, and GNU Bison [12], a parser generator for CFG, was utilized. The verification system produces verification results in XHTML format with MathML inclusions and displays error messages on a web browser such as Mozilla Firefox [13]. An error message identifies the position and type of error as enlarged and colored red.

4.1 Speed of the method

For evaluation of speed of the method, we executed the verification system on the ground-truthed mathematical document database InftyCDB-1 [14]. The verification of $20,243$ formulae in InftyCDB-1 (size: 39.8MB) was finished within 10 seconds by a PC (CPU: Pentium4 3.06GHz; RAM: 1GB). The speed of the method was confirmed.

Figure 3 is a scatter diagram showing the relationship between the number of symbols and the number of parsing steps for each mathematical formula. We can see that the number of parsing steps is proportional to the number of symbols for most mathematical formulae.

4.2 Detection of misrecognitions

For evaluation of detection of misrecognitions, we executed the verification system on the misrecognition database. The number of misrecognitions detected by the verification system for each type is shown in Table 2. (1), (2), (3), (4) and (6) in Figure 4 are error messages produced by the verification system and the corresponding original images. The respective error detections are as follows: (1) character misrecognition of the summation symbol; (2) inaccurate selection of a parent symbol for the letter 'p'; (3) unsuitable relationships between the plus sign and the fraction. Looking at the original image, we notice that the frac-

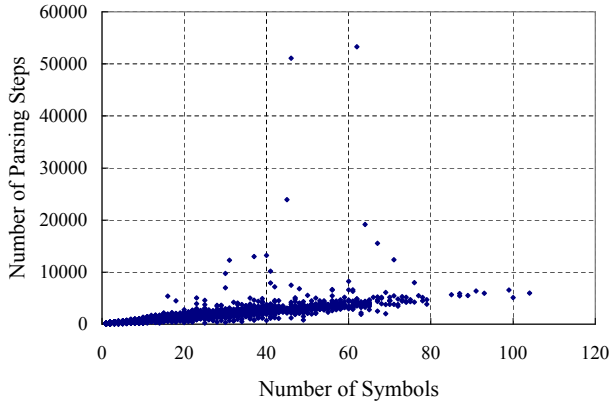


Figure 3. Scatter diagram showing the relationship between the number of symbols and the number of parsing steps.

Table 2. Number of misrecognitions detected by the verification system.

Type	Misrecognitions	Detections	Ratio
(1)	1,196	518	43.3 %
(2)	206	98	47.6 %
(3)	155	103	66.5 %
(4)	493	186	37.7 %
(5)	34	0	0 %
(6)	164	139	84.8 %
Total	2,000	899	45.0 %

tion line is a little below the horizontal center of the formula, which may be the cause of the misrecognition; (4) incorrect separation of math-area and text-area before the vertical bar. A part of the mathematical expression before the vertical bar was recognized as text; and (6) the irregular appearance of the period before the equal sign. The dart has been recognized as a period. (Since no misrecognitions were detected for "erroneous deletions of a symbol as noise," (5) is not listed in Figure 4.)

We also counted the number of false alarms occurring for correct recognition results. We executed the verification system on the database of correctly recognized mathematical formulae and counted the number of false alarms as shown in Table 3. Most false alarms resulted from separations of formulae by insertions of a new line. In this case, a faulty correspondence of parentheses may have occurred.

The detection ratio 45.0% is a satisfactory number because a misrecognition doesn't always result in a violation of syntactic rules. The misrecognition database includes a

Table 3. Number of false alarms occuring for correctly recognized mathematical formulae.

Formulae	False Alarms	Ratio
3,000	71	2.4 %

lot of misrecognized mathematical formulae that are syntactically acceptable. For example, the misrecognitions in Table 1 are syntactically acceptable. If we want to detect syntactically acceptable misrecognitions, the inclusion of semantic analysis or coordinate analysis needs to be considered.

The experimental result reveals forte and defect of the proposed syntactic method for detection of misrecognized mathematical formulae. The verification system detects 84.8% of misrecognitions of "mistaken recognitions of noise as a symbol," while no misrecognition of "erroneous deletions of a symbol as noise" is detected.

The false-alarm ratio, however, 2.4% is not small enough. If we improve the verification system so that separated formulae can be verified together, the false-alarm ratio will be smaller.

4.3 Correction of misrecognitions

Once a mathematical formula is identified as syntactically unacceptable, it is delivered to the correction process. In the correction process, the verification system asks a mathematical OCR system to offer the next recognition candidate of the formula. If the next recognition candidate is syntactically acceptable, then the verification system returns it as a corrected result, otherwise, the verification system asks another recognition candidate. The followings are the three possible end of the correction process:

1. The correct recognition result is obtained;
2. A syntactically acceptable but misrecognized result is obtained; and
3. All recognition candidates are identified as syntactically unacceptable.

We executed the correction process for syntactically unacceptable misrecognized mathematical formulae. The correct recognition results were obtained for some mathematical formulae, for example:

- $\boxed{\max_{\xi_{o\leq\xi\leq\xi_2}} u(\xi)} \rightarrow \boxed{\max_{\xi_0\leq\xi\leq\xi_2} u(\xi)}$,
- $\boxed{\Theta_{S;}(c)} \rightarrow \boxed{\Theta_{S_j}(c)}$, and
- $\boxed{u(r_n e^{i\theta_{nt)}})} \rightarrow \boxed{u(r_n e^{i\theta_{nj}})}$.

(1) STRUCTURAL ERROR: "two" may not have a subscript.

$W(p) = 2_{i=1}^{n}(\nu_i(p) - (i-1))$

$$\mathbf{W(p) = \Sigma_{i=1}^{n}(\nu_i(p) - (i-1))}$$

(2) STRUCTURAL ERROR: "equal" may not have a subscript.

$m_n(f) = p \max_{\in c_n} |f(p)|$

$$m_n(f) = \max_{p \in C_n} |f(p)|$$

(3) STRUCTURAL ERROR: "plus" may not have a subscript.

$+ \frac{1}{2\pi} \int_0^{2\pi} \log^+ |D_1(\tau^{-1}(re^{i\theta}))| d\theta$

$$+ \frac{1}{2\pi} \int_0^{2\pi} \log^+ |D_1(\tau^{-1}(re^{i\theta}))| \, d\theta$$

(4) SYNTAX ERROR: "vert" is in unexpected position.

$|_{t=0} = \frac{1}{2\pi i} \int_U \mu_0 K(\zeta, \eta) K(\eta, z_0) \mu(\eta) d\eta \overline{d\eta}$

$$\frac{d}{dt} K^{\psi_t}(\zeta, z_0)\Big|_{t=0} = \frac{1}{2\pi i} \int_U \mu_0 K(\zeta, \eta) K(\eta, z_0) \mu(\eta) \, d\eta \, \overline{d\eta}$$

(6) SYNTAX ERROR: "equal" is in unexpected position.

$D(\sigma)^{G_{t-k}} = D(\sigma(b(0)', b(1)', \ldots, b(2^{t-1} - 1)'))$

$$D(\sigma)^{G_{t-k}} = D(\sigma(b(0)', b(1)', \ldots, b(2^{t-1} - 1)'))$$

Figure 4. Error messages produced by the verification system and corresponding original images.

In most cases, however, the correction process resulted in the third end.

5 Conclusion and future work

We have proposed a syntactic method for detection and correction of misrecognized mathematical formulae for a practical mathematical OCR system. Though we have recognized the usefulness of the method by the experimental results, we realize the necessity of improving the method.

In order to detect syntactically acceptable misrecognitions, the inclusion of semantic analysis or coordinate analysis needs to be considered.

For further improvement of the correction process, we want to internalize a verification system within the recognition engine of a mathematical OCR system so that the information from syntactic analysis can be used to create new recognition candidates.

Because users of a mathematical OCR system may want to extend the grammar for new mathematical notations, we will reflect on ways users can update the grammar by themselves.

References

[1] Kam-Fai Chan and Dit-Yan Yeung. Mathematical expression recognition: a survey. *Int. J. Document Analysis and Recoginition*, 3(1):3–15, 2000.

[2] Akio Fujiyoshi and Takumi Kasai. Spinal-formed context-free tree grammars. *Theory of Computing Systems*, 33(1):59–83, 2000.

[3] R.H. Anderson. *Interactive Systems for Experimental Applied Mathematics*, chapter Syntax-directed recognition of hand-printed two-dimensional mathematics, pages 436–459. Academic Press, 1968.

[4] P. A. Chou. Recognition of equations using a two-dimensional stochastic context-free grammar. In *Proc. SPIE*, volume 1199, pages 852–863, 1989.

[5] Ann Grbavec and Dorothea Blostein. Mathematics recognition using graph rewriting. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition (ICDAR '95)*, volume 2, pages 417–421, 1995.

[6] Stéphane Lavirotte and Loïc Potter. Optical formula recognition. In *Proceedings of the 4th International Conference on Document Analysis and Recognition (ICDAR '97)*, pages 357–361, 1997.

[7] Amar Raja, Matthew Rayner, Alan P. Sexton, and Volker Sorge. Towards a parser for mathematical formula recognition. In *Proceedings of the 5th International Conference on Mathematical Knowledge Management (MKM 2006)*, pages 139–151. LNCS 4108, 2006.

[8] Akio Fujiyoshi. Application of the CKY algorithm to recognition of tree structures for linear, monadic context-free tree grammars. *IEICE Trans. Inf. & Syst.*, E90-D(2):388–394, 2007.

[9] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, Reading, Massachusetts, 1979.

[10] Klaas Sikkel and Anton Nijholt. *Handbook of Formal Languages*, volume 2, chapter Parsing of Contex-Free Languages, pages 61–100. Springer-Verlag, Berlin, 1997.

[11] Infty Project. `http://www.inftyproject.org/en/`.

[12] Charles Donnelly and Richard Stallman. Bison: The yacc-compatible parser generator. Available on: `http://www.gnu.org/software/bison/manual/`, 2006.

[13] Mozilla Firefox. `http://www.mozilla.com/firefox/`.

[14] Masakazu Suzuki, Seiichi Uchida, and Akihiro Nomura. A ground-truthed mathematical character and symbol image database. In *Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR 2005)*, volume 2, pages 675–679, 2005.