Structural Analysis of Mathematical Formulae with Verification Based on Formula Description Grammar

Seiichi Toyota[1], Seiichi Uchida[2], and Masakazu Suzuki[3]

[1] Graduate School of Mathematics, Kyushu University
ma204035@math.kyushu-u.ac.jp
[2] Faculty of Information Science and Electrical Engineering, Kyushu University
uchida@is.kyushu-u.ac.jp
[3] Faculty of Mathematics, Kyushu University
suzuki@math.kyushu-u.ac.jp

Abstract. In this paper, a reliable and efficient structural analysis method for mathematical formulae is proposed for practical mathematical OCR. The proposed method consists of three steps. In the first step, a fast structural analysis algorithm is performed on each mathematical formula to obtain a tree representation of the formula. This step generally provides a correct tree representation but sometimes provides an erroneous representation. Therefore, the tree representation is verified by the following two steps. In the second step, the result of the analysis step, (i.e., a tree representation) is converted into a one-dimensional representation. The third step is a verification step where the one-dimensional representation is parsed by a formula description grammar, which is a context-free grammar specialized for mathematical formulae. If the one-dimensional representation is not accepted by the grammar, the result of the analysis step is detected as an erroneous result and alarmed to OCR users. This three-step organization achieves reliable and efficient structural analysis without any two-dimensional grammars.

1 Introduction

In this paper, a reliable and efficient structural analysis method for mathematical formulae is proposed for realization of practical mathematical OCR [2]. The purpose of the proposed method is to provide a tree representation for each mathematical formula together with an estimate of its reliability. In case of low reliability, that is, when the tree representation result is suspicious, users can have an alarm from the proposed method.

As shown in **Fig.1**, the proposed method consists of three steps:

1. representation of the two-dimensional structure of each mathematical formula as a tree,
2. conversion of the tree representation to a one-dimensional representation, and

H. Bunke and A.L. Spitz (Eds.): DAS 2006, LNCS 3872, pp. 153–163, 2006.

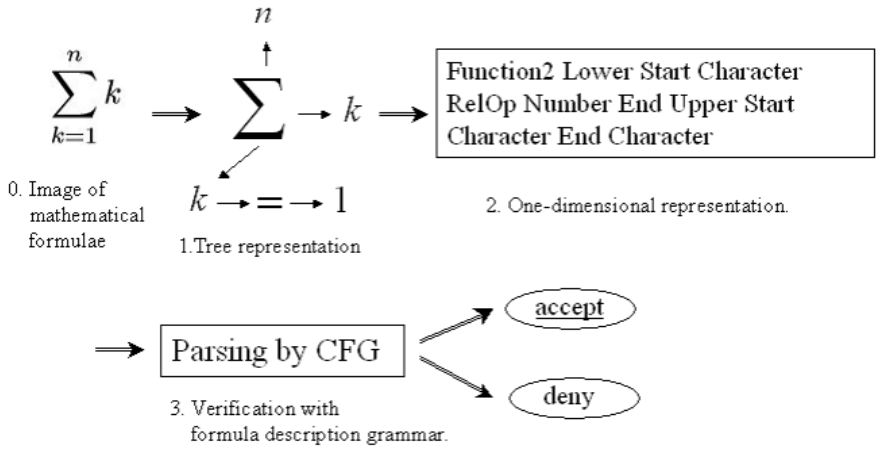


Fig. 1. Flow of the proposed method

3. verification of the one-dimensional representation with a formula description grammar.

In the first step, hereafter called the *analysis step*, a tree representation is constructed for each mathematical formula by considering positional relations between component characters/symbols. This tree construction can be performed very efficiently using the technique proposed in [4, 8]. Furthermore, this analysis step is robust to errors in character/symbol recognition results by a preceding character/symbol recognition engine. For example, the tree representation allows unmatched parentheses. Thus, the following steps should detect two kinds of errors, namely character/symbol-level error (i.e., misrecognitions) and struc ture-level error.

In the second step, hereafter called the *conversion step*, the tree representation by the analysis step is converted into an equivalent one-dimensional representation. The resulting representation is similar to LaTeX representation of mathematical formulae.

In the third step, hereafter called the *verification step*, the one-dimensional representation is parsed by a formula description grammar, which is a context-free grammar (CFG) specialized for mathematical formulae. The prepared grammar is based on a content base interpretation of mathematical formulae (see Section 4). Therefore, the third step corresponds to the final spell check process after recognition in usual OCR. If the one-dimensional representation is not accepted by the grammar, the result of the analysis step is detected as an erroneous result and alarmed to OCR users. Consequently, this step verifies the tree representation provided by the analysis step. Note that the verification step has the potential to detect both of the structure-level errors and the character/symbol-level errors.

Several researchers have considered structural analysis of mathematical formulae. However, most of these methods assume that all the characters/symbols

are correctly recognized. Anderson's technique [1] parses using a precedence matrix. Chou [3] devises a two-dimensional stochastic context-free tree grammar, and parses with a generalization of the Cocke–Younger–Kasami algorithm. Applied to two-dimensional grammars, the CYK algorithm is slow on a single processor, though it can be parallelized. Zanibbi [11] has proposed a faster method using two-dimensional grammars, based on tree transformation. Fateman [5] has achieved acceptable speed on a mathematical grammar with left-to-right recursive descent. Okamoto's approach [7] is fast and works solely on the basis of layout, without any grammar whatsoever. A survey of some other methods can be found in [2, 6, 10].

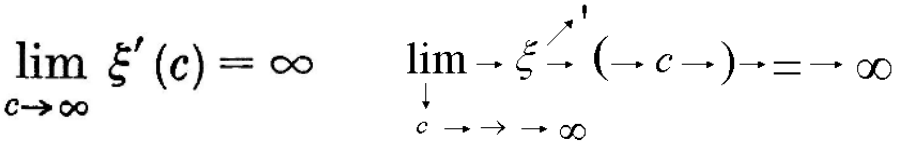
Our technique represents mathematical structure as a tree, using the output of the Eto-Suzuki algorithm [4, 8] applied to individual character recognition results from OCR. This algorithm takes OCR recognition candidates on a page, and arranges them into a tree, optimizing a cost associated to each arrangement. The algorithm can produce several candidate results. The results might include misrecognized characters, or a strange formula structure.

Structural analysis procedures based purely on grammar will fail and provide no structural analysis result when character/symbol-level errors are included. Thus, in the case of failure, OCR users would need to build an *entire* structural analysis result manually. In contrast, the proposed method always can provide some structural analysis result as a tree because it analyzes characters/symbols using only layout information of them. It returns many candidates and their costs after layout analysis. Therefore, if the first candidate is mistaken, it searches for a correct result from other candidates. It rejects candidates that do not conform to a verification grammar that we introduce. This grammar of the verification is based on content rules similar to content mark-up in MathML. By inferring the roles of the symbols, we expect to raise the accuracy of structural analysis.

The remaining part of this paper is organized as follows. Section 2 describes the analysis step for a tree-based structural analysis. Section 3 describes the conversion step for converting the tree representation by the analysis step to a one-dimensional representation. Section 4 describes the verification step using the formula description grammar. Experimental results are provided in Section 5 and in Section 6 a conclusion is drawn.

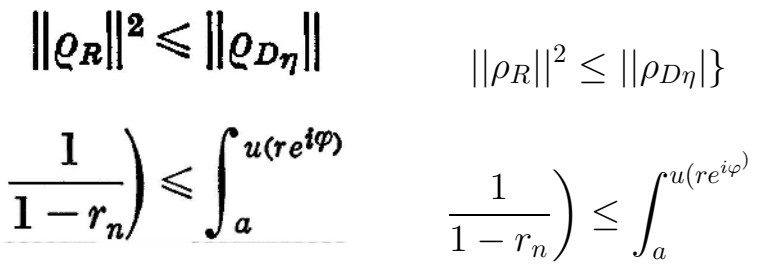
2 Analysis Step

The analysis step is based on the efficient structural analysis method proposed in [4, 8]. The analysis step provides a tree representation for each mathematical formula. Each component character/symbol of a formula is a leaf of the tree. If two component characters/symbols are "adjacent" (e.g., horizontally adjacent like "2" and "x" of "$2x$", or diagonally adjacent like "x" and "2" of "x^2", or vertically adjacent like "$\sum$" and "x" of "$\sum_x$"), their corresponding leaves are connected by a link. The tree is built as an minimum path algorithm where adjacency, positional relation, and size relation are used as costs. It should be



(a) Original image (b) Tree representation

Fig. 2. A formula whose structure is perfectly analyzed by the analysis step



(a) Original image (b) Analysis result

Fig. 3. Failures at the analysis step. The upper example includes misrecognition of "|" as "}". The lower example includes a structure-level error where the rightmost parenthesis ")" is treated as a superscript of "e".

noted that this tree can be built with few computations. **Fig. 2** shows several tree representations of formulae provided by the analysis step successfully.

Although the analysis step generally provides correct tree representations, it sometimes fails. This is because the analysis step does not care about the classes of adjacent characters/symbols. For example, the analysis step may allow a strange tree where two "+"s are linked together. In addition, the classes themselves may be erroneous due to the failures by a preceding character/symbol recognition procedure.

Typical failure examples are shown in **Fig.3**. The upper example includes a character/symbol-level error, i.e., a misrecognition of character/symbol. The lower example includes a structure-level error, i.e., error in the structure of the tree, and the position of the right parenthesis is erroneous.

3 Conversion Step

The conversion step of the proposed method connects the preceding analysis step and the succeeding verification step. The result of the analysis step is a tree, i.e., a kind of two-dimensional structure, while the verification step will accept one-dimensional sequences because it is based on a one-dimensional (i.e., usual) CFG. Thus, the role of the conversion step is the transformation of the tree representation into an equivalent one-dimensional representation.

In the conversion, the structure and the component characters/symbols of each mathematical formula are represented by the terminals of **Table 1**. Several

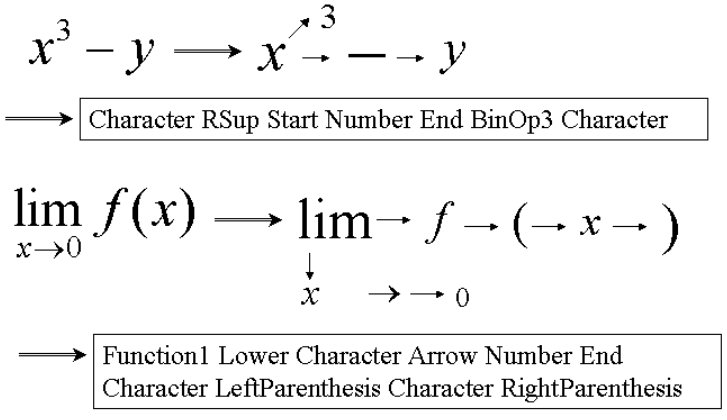


Fig. 4. Example of a one-dimensional representation of a formula. The tree representation of each formula is also presented.

terms, such as Number, Character, BinOp, etc., represent the category of component character/symbol. Other terms, such as LSup (left superscript), RSub (right subscript), etc., represent the structure of the tree. Roman and Greek letters are classified into the same category(Character). Start and End, are used to specify the range of a super/subscript. The conversion step also prepares the category of each parentheses and punctuation. The choice of category is restricted by existing subarea links.

The conversion rule transforms the tree representation into a one-dimensional representation like the LaTeX description of mathematical formula. The details of the conversion are omitted here. **Fig.4** shows examples of the conversion.

4 Verification Step

The role of the verification step is the detection of structure-level errors and character/symbol-level errors in the result of the analysis step. The verification relies on parsing based on a formula description grammar, which is a context-free grammar specialized for mathematical formulae. Thus, we can implement this verification procedure with any conventional parsing technique for CFG. In the following experiment, we used the well-known chart method for parsing. The terminals used in our CFG are listed in the **Table 1**.

Assume that a mathematical formula "(a)" is wrongly analyzed as "$(a]$". Consider the following grammar:

⟨Start⟩ ::= ⟨Expr⟩
⟨Expr⟩ ::= Character
⟨Expr⟩ ::= LeftParenthesis ⟨Expr⟩ RightParenthesis
⟨Expr⟩ ::= LeftBracket ⟨Expr⟩ RightBracket

Table 1. Terminals for representing formulas. A character/symbol terminal corresponds to a component character/symbol of a mathematical formula. A relation terminal is used to represent the positional relation among character/symbol terminals.

(a) Character/symbol terminals

Terminal	A typical character/symbol	Succeedable relation terminals
Number	$0, 1$	LSup, LSub, RSup, RSub
Character	x, y, α, β	LSup, LSub, RSup, RSub
RelOp	$<, =, >$	
BinOp1	$\div, \times$	
BinOp2	$*, /$	
BinOp3	$+, -$	
LeftParenthesis	(	
LeftBrace	{	
LeftBracket	[	
RightParenthesis	)	RSup, RSub
RightBrace	}	RSup, RSub
RightBracket	]	RSup, RSub
LeftFloor	$\lfloor$	
LeftCeil	$\lceil$	
LeftAngle	$\langle$	
RightFloor	$\rfloor$	RSup, RSub
RightCeil	$\rceil$	RSup, RSub
RightAngle	$\rangle$	RSup, RSub
OtherParenthesis	$\vert, \Vert$	
Point	$"$, $'$, ;	
Function1	lim	Lower
Function2	$\sum, \Pi$	Lower, Upper
Function3	$\sin, \cos$	RSup
Function4	$\int$	RSup,RSub
Arrow	$\leftarrow, \rightarrow$	Upper, Lower

(b) Relation terminals

Terminal	Meaning
Start	Start for range of script
End	End for range of script
RSub	Right subcript
RSup	Right superscript
LSub	Left subscript
LSup	Left superscript
Lower	Lower script
Upper	Upper script

where $\langle$ $\rangle$ indicates a non-terminal. This grammar will accept the following sequence representing the formula "(a)":

LeftParenthesis Character RightParenthesis

On the other hand, the above analysis result "(a]" is represented as the following sequence:

LeftParenthesis Character RightBracket

and thus will not be accepted by the grammar. Consequently, a user will have an alarm of an erroneous analysis result according to this verification result.

In practice, we prepare more grammar rules for the verification of various kinds of mathematical formulae; the above tiny example, however, shows the basic approach of the verification step. Note that Anderson's grammar [1] will be helpful to understand the entire rule set.

5 Experimental Results

The verification performance of the proposed method was evaluated by using mathematical formula images extracted from the ground-truthed mathematical document database called INFTY CDB-1 [9]. The ground-truth data for each mathematical formula in INFTY CDB-1 is composed of (i) the correct class of each component character/symbol and (ii) the correct tree structure of the formula.

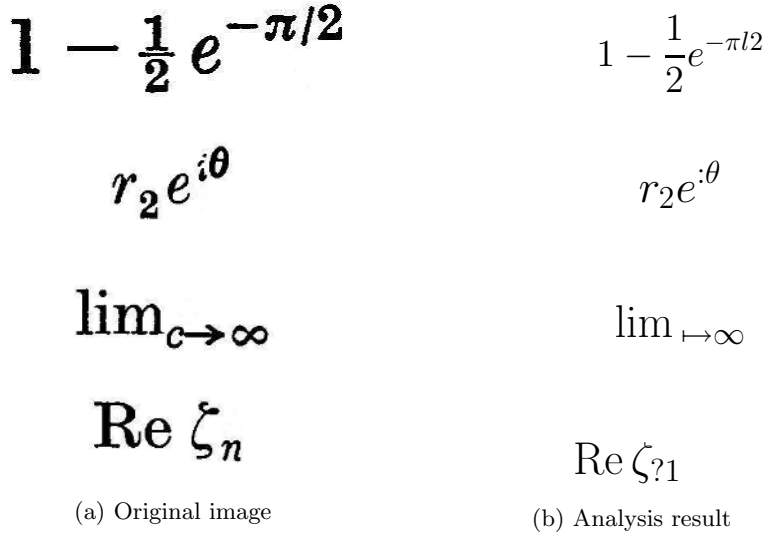
The experiment focused on the ability of the verification step (i.e., the third step) on detecting the failures at the analysis step (i.e., the first step). The failures at the analysis step are classified into the following two types: misrecognition of component character/symbols (e.g., "/"(slash)→ "l"), and wrong analysis of positional relations (e.g., "x^2A"→ "x2A"). The verification step has the potential to detect both failures.

Figures 5–9 show several results of the proposed method. In each of those figures, formula images (left) and their analysis results by the first step (right) are shown. The analysis results are represented as formulae synthesized by applying the analysis results to the LaTeX complier.

Figure 5 shows the results that the verification step could detect the failure of the character/symbol misrecognition at the analysis step. Several characters were misrecognized in the analysis results. For example, in the top example, a mathematical symbol "/"(slash) was misrecognized as "l". In the third example, a character "c" and a symbol "→" touch one another. Thus, they were misrecognized as a single symbol "↦" as shown in the analysis result.

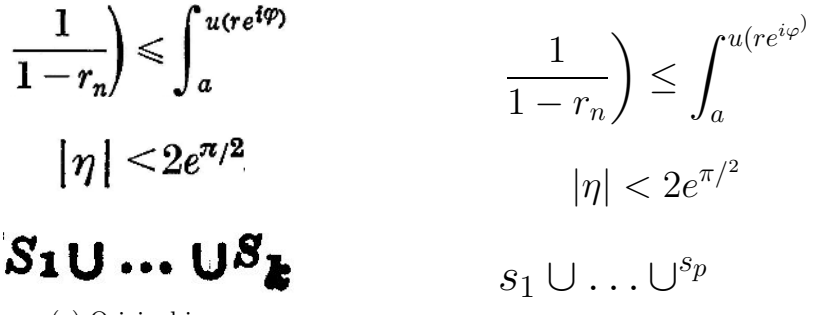
Those results with misrecognitions were successfully detected by the verification step. For example, our CFG does not accept the expression "$\pi l2$" because the CFG requires that digits ("2") should precede letters ("l") if they form a single and horizontally aligned term. The CFG also does not accept the expression "$\lim_{l} \rightarrow$" because the CFG requires that the first character/symbol of a right subscript should not be an binary operator (e.g., "↦").

Figure 6 shows how that the verification step could detect the failure of the wrong analysis of positional relations at the analysis step. In the first example of this figure, a correspondence of a left parenthesis in the right superscript of the mathematical symbol("$\int$") is wrong. In the second example, the point which a



(a) Original image (b) Analysis result

Fig. 5. Formulae whose character/symbol-level errors were successfully detected by the verification step



(a) Original image (b) Analysis result

Fig. 6. Formulae whose structure-level errors were successfully detected by the verification step

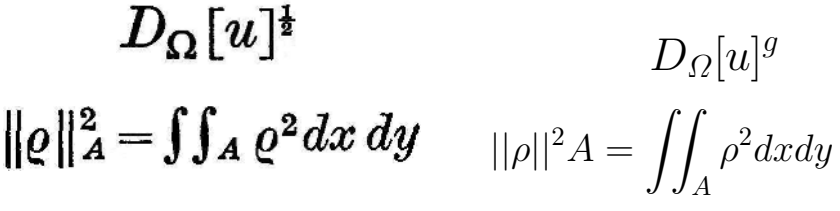
mathematical symbol "/"(slash) has a right superscript of "2" is nonsense as a mathematical formula. In the same fashion, the above failure of the positional misrecognition can be detected by this verification.

Figure 7 shows the case that the results of the analysis step include not only misrecognition but also wrong positional analysis. Like the previous examples, this complex failure is also detected by the verification step.

Figure 8 show formulae whose structure-level errors could not be detected by the proposed method. In the first example, the right superscript letter ("$\frac{1}{2}$") of the right square bracket("]") was misrecognized as "g", but this verification is not detected because both of "$\frac{1}{2}$" and "g" are correct as mathematical formulae. In the second example, the position of "A" is wrong. A mathematical symbol

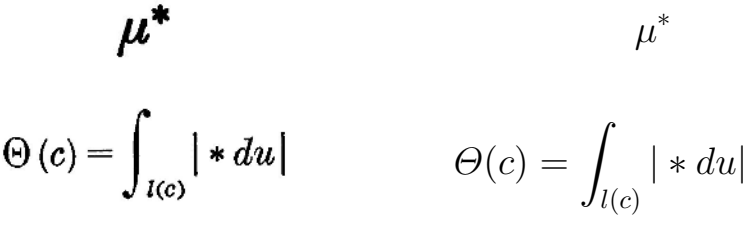
(a) Original image (b) Analysis result

Fig. 7. Formulae whose both character/symbol-level errors and structure-level errors were successfully detected by the verification step



(a) Original image (b) Analysis result

Fig. 8. Formulae whose errors could not be detected by the verification step



(a) Original image (b) Analysis result

Fig. 9. False alarm by the verification step, that is, formulae wrongly detected as erroneous ones

("|") has a possibility of a right subscript like this. Therefore, in this case, there is nothing to detect the failure. The proposed method, however, could not detect this failure because the CFG is insufficient to detect that the character/symbol was misrecognized as a meaningful formula. We can reduce these undetected failures by adding new grammar rules to our CFG.

Figure 9 shows the false alarms by the proposed method. The analysis step could analyze the structure of those formulae successfully. The verification step, however, judged that they are wrong. This is because "*", which is usually a binary operator, appears without a left operand. We will reduce these false alarms by revising our CFG; however, false alarms are less serious than undetected

failures. Our method is a verification method and therefore the analysis results detected by the proposed method will be checked by users just as "suspicious" results.

6 Conclusion

We have proposed a reliable and efficient structural analysis method for mathematical formulae, where a verification procedure based on formula description grammar is utilized. First, we assume that the structure of each mathematical formula is analyzed and represented as a tree. Although this analysis can be done efficiently, its result often includes structure-level errors and/or character/symbol-level errors. Thus, the formula description grammar, which is a context-free grammar specialized for mathematical formulae, is used to parse and verify the tree representation. This verification also can be done efficiently, because the tree representation is converted into a one-dimensional representation before parsing. If the one-dimensional representation is not accepted by the grammar, the failing portion will be included in the tree representation but alarmed to users. Experimental results showed that the proposed method can detect some erroneous tree representations successfully.

In the future, we plan to quantitatively evaluate our verification procedure, using data from InftyCDB-1 and *Infty-Reader* [12] which implemented Eto and Suzuki's algorithm. The verification procedure should accept correctly-analyzed formulas and reject incorrectly-analyzed formulas at a high rate. Also, we plan to study how additional semantic information can be used to select structural recognition results from the candidates allowed by our verification grammar.

Acknowledgement

This research is supported by the Kyushu University 21st Century COE Program: "Development of Dynamic Mathematics with High Functionality".

References

1. R.H. Anderson, "Syntax-directed recognition of hand-printed two-dimensional mathematics," Interactive Systems for Experimental Applied Mathematics, M. Klerer and J. Reinfelds, Eds. Academic Press, pp. 436-459, 1968.
2. K. -F. Chan, D. -Y. Yeung, "Mathematical expression recognition: a survey," Int. J. Doc. Anal. Recognit. vol. 3, no. 1, pp. 3-15, 2000.
3. P. A. Chou, "Recognition of equations using a two-dimensional stochastic context-free grammar," Proc. SPIE, vol. 1199, pt. 2, pp. 852-863, 1989.
4. Y. Eto and M. Suzuki, "Mathematical Formula Recognition Using Virtual Link Network," Proc. ICDAR, pp. 430-437, 2001.
5. R.J. Fateman, T. Tokuyasu, B.P. Berman, N. Mitchell, "Optical character recognition and parsing of typeset mathematics," Journal of Visual Communication and Image Representation vol 7 no. 1, pp. 2-15, 1996.

6. U. Garain and B. B. Chaudhuri, "A syntactic approach for processing mathematical expressions in printed documents," Proc. ICPR, vol. 4 of 4, pp.523-526, 2000.
7. M. Okamoto, B. Miao, "Recognition of mathematical expressions by using the layout structure of symbols," Proceedings of First International Conference on Document Analysis and Recognition Saint Malo, pp. 242-250, 1991.
8. M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori, "INFTY — An integrated OCR system for mathematical documents," Proc. ACM Symposium on Document Engineering, pp.95-104, 2003.
9. M. Suzuki, S. Uchida, and A. Nomura, "A ground-truthed mathematical character and symbol image database," Proc. ICDAR, vol. 2 of 2, pp. 675-679, 2005.
10. J. -Y. Toumit, and S. Garcia-Salicetti, H. Emptoz, "A hierarchical and recursive model of mathematical expressions for automatic reading of mathematical documents," Proc. ICDAR, pp. 119-122, 1999.
11. R. Zanibbi, D. Blostein, J.R. Cordy, "Recognizing mathematical expressions using tree transformation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 11, pp.1455-1467, 2002
12. *Infty-Reader*, `http://www.inftyproject.org/en/download.html`.