Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

DIRK DRAHEIM [1], WINFRIED NEUN, DIMA SULIMAN

Searching and Classifying Differential Equations on the Web

[1] Freie Universität Berlin, Institute of Computer Science

ZIB-Report 04-22 (Juni 2004)

Searching and Classifying Differential Equations on the Web

Dirk Draheim[2], Winfried Neun[1], and Dima Suliman[1]

[1] Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustr. 7, 14195 Berlin, Germany
{neun,suliman}@zib.de
[2] Freie Universität Berlin, Institute of Computer Science
Takustr. 9, 14195 Berlin, Germany
draheim@acm.org

Abstract. In this paper we describe the semantic analysis of differential equations given in the ubiquitous formats MathML and OpenMath. The analysis is integrated in a deployed Web indexing framework. Starting from basic classifications for differential equations the proposed system architecture is amenable to extensions for further reconstruction of mathematical content on the Web. The syntactic analysis of mathematical formulae given in the considered formats must overcome ambiguities that stem from the fact that formula particles may have different encodings, which are in principle completely arbitrary. However, it turns out that the syntactic analysis can be done straightforward given some natural heuristic assumptions.

1 Introduction

The motivation [15] for this work stems from several tasks we have been dealing with during recent years. We are working in the field of computer algebra, a community that has substantially contributed to the definition of mathematical encodings like MathML [12] or OpenMath [16]. We are also working in the Math-Net project [14], which collects mathematical information worldwide and tries to provide mathematicians with high quality information on mathematical subjects. Beside information on the organizational structure of the mathematical institutes, preprints and other papers on the Web are collected and analyzed to extract relevant information. Since MathML and OpenMath have gained wide acceptance by institutions like W3C and software manufacturers, it can be expected that documents will use these encodings for formulae to a larger extent.

An important detail is that MathML comes in two different styles, called Presentation Markup and Content Markup. The latter describes the mathematical content of an expression, whereas the first defines the rendering of the expression. Hence this widely used style does not contain rich information about the mathematical meaning.

We have started to reconstruct mathematical content from formulae given in MathML and OpenMath with the concrete problem of analyzing differential equations.

Our current implementation supports the following orthogonal classifications:

- order
- ordinary vs. partial
- linear vs. non-linear
- homogeneous vs. non-homogeneous

Modeling with differential equations is common in science. Therefore, if a researcher looks at a collection of mathematical papers the information on the type of differential equations used therein is interesting for him or her. The information may support the researcher with respect to his or her own modeling and may lead to methods for solving considered problems. A typical example is Drach's differential equation. This is not well-known even among researchers in the field. An application for our approach could be to find all occurrences of Drach' equation in the mathematical literature.

The paper proceeds as follows. Sect. 2 discusses the problems encountered in parsing MathML and OpenMath formulae. The several classifications of differential equations are defined in a succinct declarative pseudo code style in Sect. 3. The architecture of the proposed implementation is described in Sect. 4. The paper finishes with a discussion on further work, related work, and a conclusion in Sects. 5, 6, and 7.

2 Parsing MathML and OpenMath

Problems arise in parsing formulae given in MathML and OpenMath formats, because different amount of semantic clarification is needed for the several formats prior to semantic processing. MathML comes along with two versions, i.e., MathML Content Markup and MathML Presentation Markup. Whereas MathML Content Markup and OpenMath are oriented towards meanings of formulae from the outset, MathML Presentation Markup is designed for the task of rendering formulae. Therefore, if targeting semantic analysis eventually, parsing MathML Presentation Markup poses significantly more problems.

2.1 MathML Presentation Markup

In general the several possible particles of a formula have arbitrary appropriate encodings in MathML Presentation Markup – an encoding is appropriate as long as the intended visualization is achieved. Fortunately, it is possible to assume that the vast majority of MathML Presentation Markup documents found on the Web is not directly written or edited by the authors, but produced with one of the ubiquitous mathematical software tools, i.e. Maple [11], Mathematica [18], or WebEQ [4]. Therefore, the problem of ambiguity boils down to a thorough analysis of the common tools' encodings of formula particles. Furthermore the MathML Presentation Markup standard [12] makes a couple of recommendations on how to encode certain constructs and, fortunately, the common tools more or less adhere to these recommendations. We delve into the topics of encoding derivatives, characters, and operator and function applications. Table 1 summarizes the encodings yielded by the three leading mathematical software tools.

Encoding of Derivatives There are two well-known presentations for derivatives in mathematical ad-hoc notation. Accordingly, the MathML standard proposes two different Presentation Markup encodings of derivatives. For example, the non-partial derivative of a function f can be written in the following two ways:

$$f' \tag{1}$$

$$\frac{d}{dx}f \tag{2}$$

The MathML Presentation Markup encoding for (1) is the following:

```
<mrow>
  <msup>
    <mi> f </mi>
    <mo> &#8242; </mo>
  </msup>
</mrow>
```

The MathML Presentation Markup encoding for (2) is the following:

```
<mrow>
  <mfrac>
    <mo> &DifferentialD; </mo>
    <mrow>
      <mo> &DifferentialD; </mo>
      <mi> x </mi>
    </mrow>
  </mfrac>
  <mi> f </mi>
</mrow>
```

Character Encoding Unicode characters or entities can be used to encode the same character. For example, the prime character in (1) is encoded by a Unicode character:

```
<mo> &#8242; </mo>
```

However, an alternative encoding for this is the following:

```
<mo> ' </mo>
```

The problem to deal with many encodings for the same symbol is not difficult. The various encodings just have to be reflected by the grammar. The problem is rather to be aware of all possible encodings of a symbol. An allied problem is described in Sect. 4.1: a symbol with a predefined meaning may give rise to unsolvable ambiguity if it is not used with the intended meaning by the formula's author.

Operator and Function Application It makes a difference whether the produced MathML code contains *invisible operators* or not. Invisible elements can affect the appearance of the formula in browsers and may prevent ambiguity. Consider the following MathML code fragment:

```
<mi>x</mi>
<mi>y</mi>
```

Tools may visualize this code fragment as xy. However, it is not clear which of the following meanings the author has had in mind in writing this fragment:

- $x \times y$
- $x \wedge y$
- $x(y)$

The ambiguity of just *sequencing* the variables x and y can be overcome, if the MathML invisible operators InvisibleTimes or applyFunction are utilized. Inserting one of these elements *explicitly* between the identifiers x and y makes the meaning obvious. Even *bracketing* for function application cannot always adequately replace the utilization of invisible elements. Though $f(x, y)$ is unlikely to encode something different than the application of f to x and y, the formula $f(x+1)$ still can have two different meanings, i.e. the application of f to $x+1$ or otherwise the multiplication $f \times (x+1)$. Unfortunately, the tools Mathematica [18] and WebEQ do not enforce the production of necessary invisible operators. Therefore heuristics must be employed in parsing operator and function application, e.g. based on the assumption that certain names are rather used as function names than basic variable names and vice versa.

Table 1. Different encodings of formula particles by different formulae editing tools.

Encoding	Maple	Mathematica	WebEQ
derivatives	mfrac	mfrac/msup	mfrac/msup
character encoding	entity	Unicode	Unicode/entity
multiplication	explicitly	explicitly/sequencing	sequencing
function application	explicitly	explicitly/bracketing	explicitly/sequencing/bracketing

2.2 MathML Content Markup and OpenMath

MathML Content Markup focuses on the mathematical semantics of formulae. With Content Markup it is easy to distinguish whether fx means $f(x)$ or $f * x$, which makes the analysis of the formula much easier for our purposes. However, it is common sense that only a too limited subset of mathematical notation is covered by MathML Content Markup by itself. Therefore the csymbol element, which is available since the MathML version 2.0, allows Content Markup to embed elements from other notations, e.g. from OpenMath or Mathematica. In such a case heuristics must be used to find out which element is actually meant. Another problem is that MathML Presentation Markup and Content Markup are often connected in so-called mixed or parallel markup. This way both rendering aspects and mathematical content are included in the representation of a formula. Again MathML Content Markup comes in different dialects depending on the producing tool.

OpenMath is an alternative for the encoding of mathematical formulae that is older than MathML and attempts to solve the problem in a much more rigorous way than MathML. OpenMath is strictly content oriented and enables semantically rich encodings. Each object that is used in an OpenMath encoding includes a reference to a *content dictionary*, which is a mathematical definition of the object. For example, the operator *plus* used in different rings refers to different objects in separate OpenMath content dictionaries. Thus we have to integrate content dictionaries into the parsing process. Content dictionaries are created and collected by the OpenMath Society.

3 Supported Classifications of Differential Equations

The supported classifications of differential equations encompass the order of an equation, the question whether an equation is ordinary or partial, whether it is linear, and whether it is homogeneous. A succinct definition of these classifications is provided in Fig. 1. Ad-hoc abstract syntax notation and declarative notation is used for this purpose. In the abstract syntax definitions all items are non-terminals. In particular, we do not specify the set of variables `var` and the set of operations `op`.

Note that the actual LISP code for the classification used in the implementation of our approach – see Sect. 4 – is, in effect not more complex than the declarative definitions given in Fig. 1. For the

derivative ::= var var∗
equation ::= var
| derivative
| op equation$_1$ … equation$_n$

$$order : \boldsymbol{equation} \rightarrow \mathbb{N}$$

$$order\ eq \stackrel{\mathrm{DEF}}{=} \begin{cases} 0 & ,\ eq \in \boldsymbol{var} \\ n & ,\ eq = f\ v_1 \ldots v_n \in \boldsymbol{derivative} \\ max \bigcup_{1 \leq i \leq n} order\ eq_i & ,\ eq = op\ eq_1 \ldots eq_n \end{cases}$$

$$derivations : \boldsymbol{equation} \rightarrow 2^{\boldsymbol{var}}$$

$$derivations\ eq \stackrel{\mathrm{DEF}}{=} \begin{cases} \emptyset & ,\ eq \in \boldsymbol{var} \\ \{v_1 \ldots v_n\} & ,\ eq = f\ v_1, \ldots v_n, \in \boldsymbol{derivative} \\ \bigcup_{1 \leq i \leq n} derivations\ eq_i & ,\ eq = op\ eq_1 \ldots eq_n \end{cases}$$

$$ordinary : \boldsymbol{equation} \rightarrow \mathbb{B}$$
$$partial : \boldsymbol{equation} \rightarrow \mathbb{B}$$

$$ordinary\ eq \stackrel{\mathrm{DEF}}{=} |derivations\ eq| = 1$$
$$partial\ eq \stackrel{\mathrm{DEF}}{=} \neg\ ordinary\ eq$$

`normalized` ::= (`equation`? `derivative`?)∗

$$linear : \boldsymbol{normalized} \rightarrow \mathbb{B}$$

$$linear\ eq_1\ (f\ vars_1) \ldots eq_n\ (f\ vars_n) \stackrel{\mathrm{DEF}}{=} \forall_{1 \leq i \leq n}\ f \notin eq_i$$

`homogeneous` ::= (`equation`? `derivative`)∗

Fig. 1. Definition of the supported classification of differential equations. Non-terminals of the abstract syntax are used in the declarative definitions in bold face in order to denote the respective syntactic categories.

purpose of analyzing a differential equation an equation is simply a term composed of variables, operations, and derivatives. A derivative consists of the target function name and the derivate variables. The list of derivate variables may be empty, this way representing the genuine occurrence of the equation's target function. For the definition of linearity it can be assumed that the differential equation adheres to the common sum-of-product normal form, expressed by the abstract syntax notation `normalized`. Furthermore it can be assumed that the differential equation's target function is unique. In our implementation these assumptions hold because of the employed computer algebra stage and verifier stage. The homogeneity of a differential equation can be specified syntactically without detouring.

4 Solution System Architecture

Separation of concerns leads to a straightforward compiler stage architecture for the solution system, see Fig. 2. After finding a formula encoded in MathML or OpenMath it is parsed and an abstract syntax tree is built. We have used ANTLR [17] to build the parser. Printing the abstract syntax tree yields the formula in LISP format. If the formula represents a supported differential equation it is further processed by the REDUCE [8] computer algebra system and the formula is transformed into a normal form. Afterwards LISP programs classify the formula. The result of the classification is printed out in Summary Object Interchange Format (SOIF) because the classification is embedded into the Harvest system [1, 13], which uses SOIF data for its internal information storage and exchange.

The proposed solution is scalable with respect to functionality in several means. Future formats can be integrated easily – the abstract syntax serves as a mediator between different formats. Because of the computer algebra stage the software system is prepared for projected advanced semantic analysis.

4.1 Parsing Problems

First of all we assume that the MathML and OpenMath code under consideration are syntactically correct. Thus we specified non-validating grammars to reconstruct the content of the represented formulae. Parsing is aborted if it can be recognized lexically that the considered formula is actually no differential equation, for example, because it encompasses elements for encoding integrals or matrices. We cannot assume that the rules for the so-called *proper grouping* in Sect. 3.3 of [12] are always followed. Thus unnecessary mrow elements are escaped. Unnecessary mrow elements are those which are not nested within other elements, e.g., mfrac or msup elements. Only attributes relevant to differential equations are recognized.

The Presentation Markup grammar needs to distinguish between two kinds of identifiers, i.e., monad operators and simple variables. If an ambiguity occurs our heuristic solution decides in favor of the monad operator. For example, <mi> sin </mi> always represents the sinus function even if there is no applyFunction element next to it.

A differential equation consists of variables, numbers, differential quotients, and functions, which are bound by operators. In MathML Content Markup and OpenMath the grammar has just to follow the apply structure of the formula in order to reconstruct it. However, with respect to the actual operator structure parsing a formula encoded in MathML Presentation Markup is more complicated because the grammar has to define several internal levels to correctly obtain the priority of arithmetical operations.

4.2 Further Processing

After parsing a verifier checks whether a given formula actually can be considered a proper differential equation. For example, it is checked whether the target function is unique across all the equation's derivatives. If the target function is not unique the equation stems perhaps from a differential equation system, may be interesting, but not amenable to the classifications examined in this paper.

For the purpose to store the result in a unified format, the intermediate result is processed with computer algebra methods. Simplification rules are applied and a normal form is achieved. The processing identifies dependent and independent variables. As a result the derivatives and the target function are factored out so that the analysis of linearity and degree are simplified. REDUCE has been the computer algebra system of choice because it is a proven system and because REDUCE formulae can be denoted by LISP S-expressions. ANTLR generated parsers build abstract syntax trees as two dimensional presentation of the formula that can be printed out as LISP S-expressions and can be forwarded directly to REDUCE.

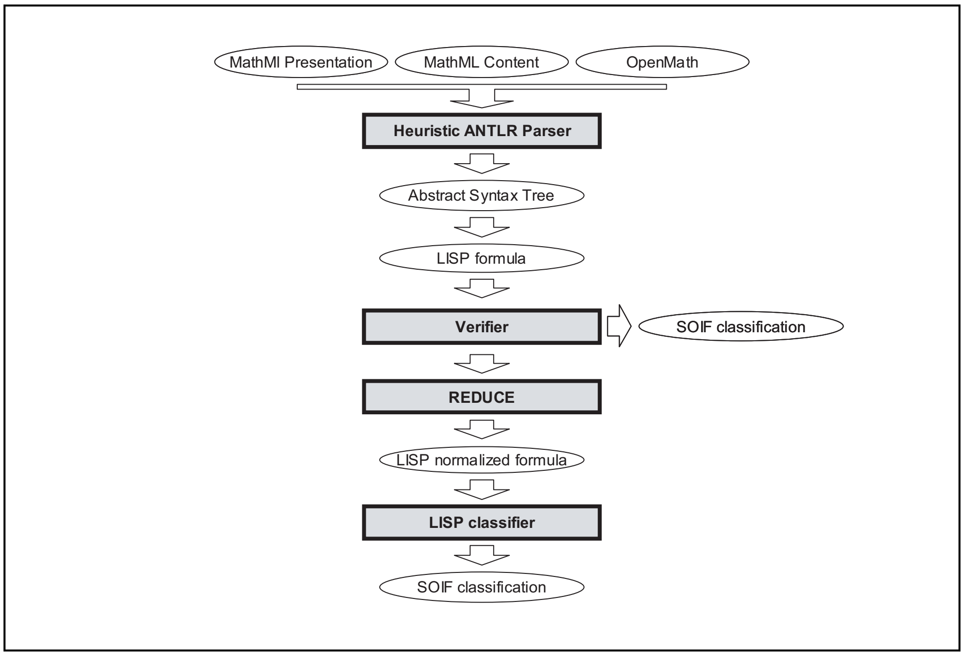

Fig. 2. The compiler stages of the differential equation classifier.

4.3 Integration into Harvest

The resulting classifier is embedded into an instance of the Harvest system – see Fig. 3 – that is hosted by the Math-Net project. Harvest is a distributed search system; the main components of Harvest are the gatherer and the broker. The gatherer collects and summarizes the resources on the Web. The broker indexes the collected information and answers queries via a Web interface. Gatherer and broker communicate using an attribute-value stream protocol, the proprietary SOIF. The gatherer triggers different summarizers for different kinds of documents. The available set of summarizers neither supports MathML nor OpenMath resources. We provided appropriate new summarizers for these resources – MathML can be found on the Web in XML files explicitly tagged as MathML or in XML files; OpenMath can be found in XML files.

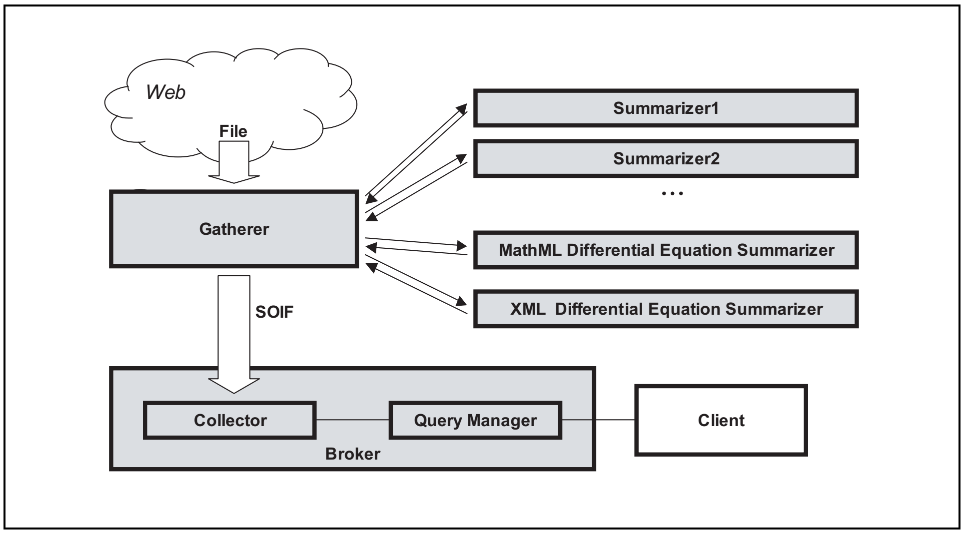

Fig. 3. Integration of the differential equation classifier as a summarizer into the Harvest system.

5 Further Work

Currently we are working on the integration of a general formula search facility into our system. The problem of searching formulae, i.e., querying for formulae that match a given target formula, is different from querying formulae that match given classification options. A merely textual approach for searching formulae is obviously not sufficient. Therefore each approach for searching formulae must fix a notion of similarity between formulae. Different degrees of similarity are conceivable. At a very basic level one wants to consider equivalent formulae as equal. Similarly formulae should be considered equal up to renaming of variables. However, more sophisticated similarities between formulae might be of interest. For example, two formulae are similar, if they can be constructed from each other by general term rewriting. However, subtle pragmatic questions now arise that can only be resolved by heuristic decisions: What is the appropriate granularity of term substitutions ? Should it be possible to search for instances of a target formula ? Or should it rather be possible to search for generalizations of a target formula ?

A great deal of work with respect to questions like these has been done in the *SearchFor* project [3] at the INRIA Sophia-Antipolis Research Center. We are trying to utilize the results of the *SearchFor* system in our project. In principle, there are two basic approaches to this. In the brute force approach a converter from the S-expression format to the CAML data structure format is developed so that the *SearchFor* code can be reused operatively. In the reverse engineering approach the CAML code of the *SearchFor* system is investigated and reimplemented in LISP code.

Furthermore it would be interesting to fix formal semantics for notions of similarity between formulae. Again the knowledge about formula similarity construction that is encapsulated in the *SearchFor* system could serve as a cornerstone for this work.

Based on a facility for searching formulae it would be neat to offer search options for well-known differential equations or differential equation patterns, like Abel's differential equation, the Bernoulli differential equation etc.

6 Related Work

In [9] a PROLOG-based document object model for XML documents is proposed. Transforming XML data into the so-called *field notation* results in an expert system against which rule-based queries can be executed. The approach is elaborated for XML documents in general and is applied to the concrete formats MathML Content Markup and OpenMath in particular. Furthermore [9] outlines the application of the approach to the classification of ordinary differential equations with respect to Kamke's list [10]. The notion of rule-based expert system has gained remarkable consideration in the eighties for building and exploiting knowledge bases – today's working knowledge base mining relies on statistical modeling methods rather than on proof theory. However, in the case of processing mathematical knowledge it is promising to employ a logical programming language, because term rewriting problems that typically occur can be tackled by the logical programming languages' unification mechanisms. Other approaches that employ logic programming language techniques for mathematical knowledge bases are described in [2] and [6, 7].

A similar approach was used in the TILU [5] project by Einwohner and Fateman. This project provides an alternative to algorithmic integration algorithms by integral table lookup. Special conditions which arise often in integration problems, e.g., special functions in the integrand, can be used to optimize the search of the formulae.

7 Conclusion

We expose the following assumptions and observations:

- The Web is a huge repository; this is particularly true for mathematical knowledge.
- Sophisticated search capabilities for mathematical Web resources are required by research and education.
- MathML and OpenMath can be expected to become de-facto standards for mathematical Web resources.
- Subtle notation issues arise in handling MathML and OpenMath.

We have undertaken the following steps:

- We have chosen the classification of differential equations as a challenging starting point of our project.
- Due to its separation of concerns, the chosen system architecture is scalable with respect to further formats and further classifications.
- The implementation is used in a working instance of the Web indexing framework Harvest.

It is further work to integrate a general formula search facility into our approach.

References

1. C.M. Bowman, P.B. Danzig, D.R. Hardy, U. Manber, M.F. Schwartz, and D.P. Wessels. Harvest: A Scalable, Customizable Discovery and Access System. Technical Report CU-CS-732-94, University of Colorado, Boulder, USA, 1994.

2. Stéphane Dalmas, Marc Gaëtano, Claude Huchet. A Deductive Database for Mathematical Formulae. In: Proceedings of Design and Implementation of Symbolic Computation Systems, LNCS 1128, pp 287-296. Springer, 1996.
3. Stéphane Dalmas, Marc Gaëtano: Indexing Mathematics with SearchFor. Presented at: International MathML Conference 2000 - MathML and Math on the Web, 2000.
4. Design Science Inc.: WebEQ version 3.5, 2003.
5. T.H. Einwohner, R.J. Fateman: Searching Techniques for Integral Tables, Proceedings ISSAC 1995, ACM, New York, 1995.
6. Andreas Franke, Michael Kohlhase. System Description: MBASE, an Open Mathematical Knowledge Base. In: Proceedings of CADE 2000 - International Conference on Automated Deduction, pp. 455-459. LNCS 1831, Springer, 2000.
7. Andreas Franke, Michael Kohlhase. MBase: Representing Knowledge and Context for the Integration of Mathematical Software Systems. Journal of Symbolic Computation, vol. 32, no. 4., pp. 365-402, 2001.
8. Anthony C. Hearn. REDUCE 2: A System and Language for Algebraic Manipulation. In: Proceedings of the 2nd ACM Symposium on Symbolic and Algebraic Manipulation. ACM Press, 1971.
9. Bernd D. Heumesser, Dietmar Seipel, Ulrich Güntzer. An Expert System for the Flexible Processing of XML-Based Mathematical Knowledge in a PROLOG-Environment. In: Proceedings of the Second International Conference on Mathematical Knowledge Management. Springer, 2003.
10. Erich Kamke. Differentialgleichungen: Lösungsmethoden und Lösungen. Akademische Verlagsgesellschaft, 1967.
11. Maplesoft Inc.: Maple version 9, 2003.
12. Mathematical Markup Language (MathML) Version 2.0, W3C Recommendation, 21 February 2001.
13. Kang Jin Lee. Development Status of Harvest. In: Proceedings of SINN'03 Conference - Worldwide Coherent Workforce, Satisfied Users: New Services For Scientific Information, 2003.
http://www.isn-oldenburg.de/projects/SINN/sinn03/proceedings.html
14. Math-Net. http://www.math-net.org.
15. Winfried Neun. Harvesting Webpages that Contain Mathematical Information. In: Proceedings of SINN'03 Conference - Worldwide Coherent Workforce, Satisfied Users: New Services For Scientific Information, 2003.
http://www.isn-oldenburg.de/projects/SINN/sinn03/proceedings.html
16. The OpenMath Web Site. http://www.openmath.org
17. T.J. Parr, R.W. Quong. ANTLR: a Predicated-LL(k) Parser Generator. In: Journal of Software-Practice & Experience, vol. 25, no. 7, pp. 789-810. John Wiley & Sons, July 1995.
18. Wolfram Research Inc.: Mathematica version 5, 2003.