PERGAMON

Applied Mathematics Letters 15 (2002) 875–879

Applied Mathematics Letters

www.elsevier.com/locate/aml

Learning and Using Mathematics Software the Natural Way

S. TSE AND V. DAHL

Department of Computing Science, Simon Fraser University

Burnaby, British Columbia, Canada V5A 1S6

<stephent><veronica>@cs.sfu.ca

(Received July 2001; accepted August 2001)

Communicated by N. Cercone

Abstract—We motivate the need for more standard while natural ways of accessing the growing number of internet applications of mathematics software. We then identify a subset of natural language appropriate for this task, and describe an efficient logic programming transformation from this subset of language into the desired commands. We use substructural logic for dealing with different kinds of mathematical anaphora. We exemplify our ideas in the context of Maple, an advanced mathematics software for symbolic computing. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords—Mathematics software, Natural language processing, Logic programming, Assumption grammars, Maple.

1. MOTIVATION

The number of internet applications of mathematics software is growing quite dramatically [1–3]. Most notably, applications such as scientific visualization, distance collaboration, tele-learning, etc., allow users from different backgrounds to share information over the network.

Yet the interfaces remain clumsy, non-user-friendly. Even though the provision of a standard protocol and syntax for mathematical input is a remote possibility, users often feel reluctant to learn yet another syntactic convention. One-time visitors, particularly if not computationally inclined or versed, who typically want results immediately, demand a more friendly yet uniform interface.

What better or more natural way of using a new tool than through commands in your own mother tongue? The state of the art in natural language processing is not yet at the point in which it can treat unrestricted text. However, for the restricted domain of mathematics software, we should be able to identify a useful subset of, say, English, and a fairly standard way of transforming it into commands of the particular syntax chosen by the software.

The significance of this research is twofold: on one hand, it enables users to consult mathematics software directly in their mother tongue. On the other hand, it can be of great help to students

This research was made possible by NSERC Research Grant 611024.

PII: S0893-9659(02)00056-3

learning to use the software, who will be able to consult it or command it in English and see the formalized query popping up on the screen as a result. While mathematical software is usually not too difficult to learn, some of its command syntax and user interface can be discouraging to newcomers. Maple [4], for instance, is particularly well-suited to aid university freshmen learn calculus through verifying calculations and plotting complicated graphs. However, understanding calculus concepts while learning to use computer program/mathematics software, and picking up Maple syntax, all at once, can be overwhelming for them. *What better way of learning a new syntax than through picking its commands up one by one in your own mother tongue?*

2. COVERAGE

The first decision to be made concerns the coverage of constructs. Obviously, the wider, the better, within tractability and minimizing of ambiguities. The decision is made harder by the fact that there are simply too few resources on NLP for mathematics software: no reference implementation, no complete corpus for mathematics, and no parser tailored for mathematics.

Instead of aiming at a full blown system with a complete lexicon for mathematics, we concentrate on a restricted yet interesting and tractable domain: that of basic arithmetic with calculus operations. The system is intended to be a prototype for first-year calculus students. A BNF description of the input language of our system can be found on our homepage [5].

3. CHALLENGES AND PROPOSED SOLUTIONS

Anaphora in Mathematics

Mathematics definition can be viewed as a form of anaphora, i.e., as a resource which is defined at some point to be used elsewhere. We use linear implication as embedded in BinProlog [6,7] for a direct implementation of mathematical definitions. For instance, in

```
> assume that f is 'cos(x)' differentiate f
```

the system works as follows: when it encounters `assume that f is 'cos(x)'`, it adds temporarily a clause usable *at most once* in later proofs. In this case, it adds the definition `pre_define(X, Meaning)`, where `X` is `f` and `Meaning` is `'cos(x)'`. Also, in the later proof `differentiate f`, it attempts to resolve the variable `f` to its definition. Note that this assumption vanishes nicely on backtracking since it is scoped over the current AND-continuation.

Preceding or Postceding Variable Definitions

In mathematics, statements with variable definitions at the beginning or at the end of a formula are both common, e.g., as follows.

```
> given that f is the sum of 'sin(x)' and 'cos(x)'
  what is the derivative of f?
> integrate f with respect to x where f is 'log(x)'
```

This kind of anaphora in mathematics statements can be handled by timeless assumptions [7], which can be consumed after or before being made. The source code [5] shows their (surprisingly concise) implementation in terms of linear implication.

Multiple Uses of Definitions Viewed as Resources

Consider the following examples.

```
> given that f is 'sin(x)' what is the sum of f and f?
> differentiate the product of f and f wrt x where f is 'cos(x)'
```

In these examples, the referent that `f` stands for is referred to more than once. We therefore use intuitionistic implication instead of linear implication, which allows us to consume the referent an indefinite number of times. However, we have implemented our own version of it for this particular research. The source code [5] shows the implementation.

Coordination

Another feature of mathematics statements which is common in natural language is the use of conjunction. Traditionally, one of the most difficult problems in natural language processing, conjunction, is simpler in mathematics statements, although it still exhibits the problem of combinatorial explosion. A proper and general treatment of coordination may involve "meta-grammatical" constructions, but for a restricted, well-defined domain of problems, writing specific rules may suffice [7].

In particular, we recognize that variable definition and usage involves an excessive amount of backtracking. This is due to the fact that any expression can be regarded as a variable, which can be defined either before the usage or after the usage. This interacts with the fact that a conjoined expression is a list of subexpressions separated by `and`.

To greatly reduce backtracking and neatly solve the problem of coordination in mathematics, we restrict the syntax of variable definition as in the predicate `variable`, which can be seen in the source code [5]. Surprisingly, together with some tweaking of rule orderings, the system can handle common conjunctive statements in mathematics such as the following.

```
> graph 'sin(x)' and 'cos(x)' for x
> solve 'x+y=2' and 'x-y=4' for x and y
```

Extensibility and Customizability

Regarding implementation, one major challenge is that Maple, as most closed commercial softwares, does not supply a set of API to control its interface. This severely limits its extensibility and customizability, e.g., we cannot have a plug-in for input in natural language. Fortunately, Maple comes with a command line mode in which plain text can be sent via process control. A simple Emacs Lisp script is written to easily handle the communication of processes in two windows.

4. A SAMPLE SESSION

```
%%-- Command, Query, Digit Name
> add 1 to 3
==> MAPLE: 1+3;

> add one to three
==> MAPLE: 1+3;

> what is the sum of one and three
==> MAPLE: 1+3;

> differentiate 'x^2+1' with respect to x
==> MAPLE: diff(x^2+1,x);

> simplify '(x^2-y^2)/(x-y)'
==> MAPLE: simplify((x^2-y^2)/(x-y));

> solve 'x^2-3*x-4=0' for x
==> MAPLE:solve(x^2-3*x-4=0,x);
```

```
> graph the product of x and x for x
==> MAPLE: plot(x*x,x);
```

Output from Maple is shown in Figure 1.

```
> add the difference of the derivative of 'x^2'
wrt x and the integral of 'y^2' wrt y to 9
==> MAPLE: diff(x^2,x)-int(y^2,y)+9;
```

%%-- *Preceding or Postceding Variable Definitions*

```
> integrate f with respect to x where f is 'log(x^2)+sin(x)'
==> MAPLE: int(log(x^2)+sin(x),x);
```

```
> provided that f is the quotient of 'sin(x)' and
the sum of 'cos(x)' and 'tan(x)' differentiate f with respect to x
==> MAPLE: diff(sin(x)/(cos(x)+tan(x)),x);
```

%%-- *Multiple Uses of Definitions Viewed as Resources*

```
> given that f is 'sin(x)' what is the sum of f and f
==> MAPLE: sin(x)+sin(x);
```

```
> differentiate the product of f and f wrt x where f is 'cos(x)'
==> MAPLE: diff(cos(x)*cos(x),x);
```

%%-- *Coordination*

```
> graph 'sin(x)' and 'cos(x)' for x
==> MAPLE: plot(sin(x),cos(x),x);
```

```
> graph the derivative of 'x^2' wrt x and the product of
'sin(x)' and 'tan(x)' for x
==> MAPLE: plot(diff(x^2,x),sin(x)*tan(x),x);
```

```
> solve 'x+y=10' and 'y+z=20' and 'x+z=30' for x and y and z
==> MAPLE: solve(x+y=10,y+z=20,x+z=30,x,y,z);
```

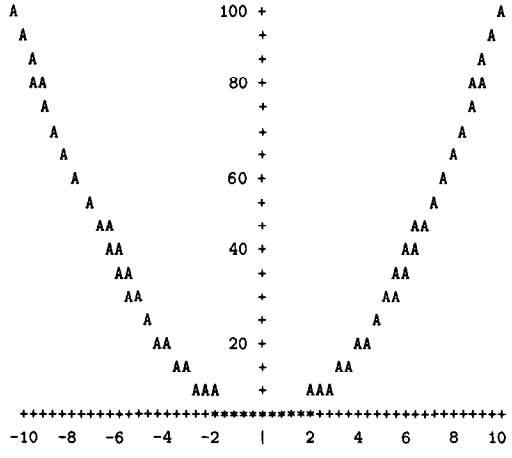

Figure 1.

5. RELATED WORK

To the best of our knowledge, there are no similar systems providing natural language interfaces to mathematics software. The closest we have come to in the literature is standardizing efforts for cross-platform representation and communication of mathematics, such as MathML [2] and OpenMath [1], but these are not designed to act as user interfaces to a mathematics system.

The most related other work would be that of tutorials. There are many, for instance, for learning Maple, but none of them is as direct as ours, which allows users to input commands in natural language directly to the system. Online help in Maple, which lists syntax format and usage examples, can be useful sometimes but functions rather like a reference manual—too formal and restrictive for learning.

6. CONCLUSION, EXTENSIONS

We have provided a proof-of-concept, in the form of an implemented English interface to Maple, that natural language is a viable option for providing a natural way or learning and using mathematics software. Implemented in BinProlog to make use of the unique expressiveness of the different types of assumptions needed (linear, intuitionistic, timeless), the system successfully and concisely addresses in particular two crucial problems in NLP for mathematics, namely, definitions and coordination. The result is an interestingly rich set of user interactions with Maple.

Future work includes the extension of our ideas into other mathematics software, and the provision of support for multiple back-ends simultaneously, giving users a uniform interface for various softwares. It would also be interesting to integrate the NLP system into an open source mathematics system such that the NLP system has more powerful and complete control of the interface. To gain more publicity and a wider audience, a web-ready applet version of the system will testify to the friendliness of the interface.

REFERENCES

1. J. Abbott, A. van Leeuwen and A. Strotmann, Objectives of OpenMath, Technical Report 12, RIACA, (1996); `http://www.openmath.org`.
2. P. Ion and R. Miner, Mathematical markup language, W3C Recommendation REC-MathML-19980407, World Wide Web Consortium, (1998); `http://www.w3.org/TR/REC-MathML`.
3. P. Wang, Design and protocol for internet accessible mathematical computation, Technical Report, ICM/Kent State University, (1999).
4. B.W. Char, K.O. Geddes, G.H. Gonnet, B. Leong, M.B. Monagan and S.M. Watt, *Maple V: Language Reference Manual*, Springer-Verlag, (1991); `http://www.maplesoft.com`.
5. V. Dahl and S. Tse, Documentation to accompany the paper: Learning and using mathematics software the natural way, Technical Report, Simon Fraser University, (2001); `http://www.cs.sfu.ca/people/Faculty/Dahl/learn_math`.
6. P. Tarau, Professional edition: User guide, Technical Report, BinNet Corp, (1998); `http://www.binnetcorp.com/binprolog`.
7. V. Dahl, P. Tarau and R. Li, Assumption grammars for processing natural language, In *Proceedings International Conference on Logic Programming*, (1997).