Theorema:
A System for Formal Scientific Training in Natural Language Presentation

Bruno Buchberger, Tudor Jebelean, Daniela Vasaru
RISC–Linz, Austria
last–name@risc.uni–linz.ac.at

Abstract: Lack of formal training is hindering the application of high–level information technology in science and engineering. The *Theorema* project addresses this need by implementing a sophisticated automatic reasoning system with an easy–to–use interface which can be used in various interactive ways – both locally and over the internet. We report on the use of this software for the training of mathematics and engineering students in problem solving and theorem proving.

Background and Motivation

Deep understanding and skilled usage of mathematics and logic are essential ingredients of most scientific and engineering activities, particularly in the fields of Mathematics and Computer Science, but also in many other areas. In Information Technology applications, these skills are becoming crucial as the systems become more and more complex and only the use of formal techniques can guarantee their proper behaviour.

Unfortunately – as the first author has learned over many years of didactic and research activity in mathematical logic – practical training in this field is severely under–emphasized in the vast majority of Math and Computer Science curricula all over the world. In contrast, at RISC–Linz we have offered for over a decade a comprehensive study and training program in mathematical logic and practical problem solving and proving [Buchb–Licht:81]. Based on this experience we are now in the position to develop concrete tools for the practical usage of the working mathematicians and for the students, which can assist them in teaching / learning mathematical logic and proving skills, as well as in mathematical textbook writing, problem solving, and scientific research.

In this paper we present the tools developed in the frame of the Theorema project for the training of students in Computer Science, Mathematics, and Engineering in the language of predicate logic and in establishing proofs in this language. These tools form an interactive theorem prover in high–order logic, written in the computer algebra system Mathematica 3.0 [Wolfram:96], which is presented to the user in the form of an intelligent environment for developing mathematical texts in natural language (e. g. in the style used by scientists in typical text–books).

The main objectives of the formal training in our system are:

– Language training: learn and train the use of mathematical language for concise and exact expressing of models.

– Formal models: learn and train how to build mathematical models for concrete real problems (defining concepts, defining properties, defining and exploring problems).

– Conjecture and prove: learn and train the formulation of conjectures about the properties of mathematical objects, *prove and disprove* conjectures.

The last activity – proving – is one of the most difficult formal activities for the human intellect, and consequently it requires most of the learning and training. Our system offers computing support for all the objectives mentioned above, however we will emphasize in this paper the proving and the proving–assisting capabilities of our software, which took most of the effort to develop.

We believe that providing computer support to the working scientist and to the engineering student opens the way for:

– a new understading of mathematics and computer science;

Published in the proceedings of ED–MEDIA 98 (International Conference on Educational Multimedia), Freiburg, Germany, June 1998. As a courtesy to the publisher, the ditribution of this paper is strictly limited.

– a new level of creativity and productivity in mathematics;
– a new quality of education of teachers and students, both at academic and pre–academic levels.

These aspects are certain to have a "boule–de–neige" effect on the general education in mathematics, computer science, and engineering, with obvious benefic effects on all sectors of human activity.

It is beyond the scope of this paper to document extensively the state–of–the–art in the field of automatic theorem proving, rather we will shortly review this (for more details see [Homann:96], [Buchb–et–al:97]). Currently the most successful and effective computer–assistants for the working scientist are the *computer algebra systems*, which contain a large collection of mathematical knowledge implemented as mathematical models and algorithms (Mathematica, Maple, Macsyma, Scratchpad, etc.). All these systems have quite efficient computing and solving capabilities (over numerical domains), but none of them offers support for constructing mathematical models and for proving (which is a lot more than just being able to compute with logic formulae – facilites recently added to few such systems).

On the other hand, a long–time research in automated reasoning produced a mature theory and many quite powerful systems for proving, however they are rarely used outside academic laboratories and quite unsuitable for use by non–experts. Their user–interface is most of the time quite cryptic, also the presentation of proofs and the way reasoning is performed differ significantly from the way human mathematicians operate on mathematical models.

The Theorema Project

Our project aims to close the gap between computing and proving by adding proving support to an existing computer algebra system. A similar approach is pursued by few other research groups: Isabelle [Paulson:96], Analytica [Clarke–Zhao:93], NQTHM [Boyer:88a], Nuprl [Constable:86], Coq [Huet_all:94], HOL [Gordon: 93], Oyster–Clam [Horn:88], [vanHermelen:89], Redlog [Dolzmann_Sturm:96].

The distinctive features of Theorema are:
– integration of an existing rewrite rule language (Mathematica) into our logic;
– imitation of human proof style;
– natural language formulation of proofs;
– combination of functor style programming with proving;
– integration of proving into math textbooks generation and interactive math teaching;
– multi–style proving triggered by special context.

The Theorema system is being built up in the following layers:

– A *mathematical language* as a common frame for both non–algorithmic and algorithmic mathematics. The syntax of this language is implemented by using the syntax extension facilities of Mathematica and it allows to write logic formulae (theorems) containing executable parts (function definitions using induction and bounded quantifiers).

–The concept of "*functor*'' as the general mechanism for building up towers of mathematical domains, which allows the user to construct mathematical models.

– A general *predicate logic prover* of a ''natural style'' introduced in earlier papers by Buchberger (see e.g. [Buchb–Licht:81]).

– Various *special theorem provers* (and / or interactive proof developers) corresponding, in a natural way, to the various functors that build up mathematical domains. All these provers produce proofs that imitate the proof style of human mathematicians. By now, an induction prover for the natural numbers and one over the domain of lists over a given domain are already implemented, a prover for the domain of multivariate polynomials over a given domain of monomials is under way.

– A general facility that allows the *presentation of proofs in the form of ''nested cells''*, which is crucial for being able to read complicated proofs without losing the overview. This facility is based on the manipulation of cells in Mathematica notebooks and can be automatically translated into HTML collections.

For a more detailed description of the components, as well as of the theoretical and algorithmical background of the project, the reader is refered to [Buchb–et–al:97] and [Buchb–Vasaru:97].

Examples

Due to space limitations we can provide in this section only two examples of proofs produced by the predicate logic prover and by the prover by induction on natural numbers. Other examples produced by these provers and by the prover by induction on lists are presented in an extended version of this paper which is available on the internet at "ftp://www.risc.uni–linz.ac.at/pub/techreports/1997/97–34/edm.html".

Example 1: Predicate Logic. Universally quantified formulae are used here in order to define the relation of congruence by bidirectional less–or–equal. Then, one proves that the this congruence is transitive by using the transitivity of the ordering relation. The brakets on the right of the page show the cell structure.

Prove

$$(\sim\ \sim\Rightarrow\ \sim)\quad \forall_{\langle x,y,z\rangle}\ (x\sim y)\wedge(y\sim z)\Rightarrow x\sim z$$

under the assumption(s)

$$(\sim\ :\ \leq\ \leq)\quad \forall_{\langle x,y\rangle}\ x\sim y\Leftrightarrow(x\leq y)\wedge(y\leq x),$$

$$(\leq\ \leq\Rightarrow\ \leq)\quad \forall_{\langle x,y,z\rangle}\ (x\leq y)\wedge(y\leq z)\Rightarrow x\leq z.$$

We prove $(\sim\ \sim\Rightarrow\ \sim)$ by natural deduction.

For proving $(\sim\ \sim\Rightarrow\ \sim)$ we prove, for arbitrary but fixed values,

$$(\sim\ \sim\Rightarrow\ \sim')\quad (x_0\sim y_0)\wedge(y_0\sim z_0)\Rightarrow x_0\sim z_0.$$

We prove $(\sim\ \sim\Rightarrow\ \sim')$ by the deduction rule.

We assume

$$(\sim\ \sim\Rightarrow\ \sim'.H)\quad (x_0\sim y_0)\wedge(y_0\sim z_0)$$

and show

$$(\sim\ \sim\Rightarrow\ \sim'.C)\quad x_0\sim z_0.$$

From $(\sim\ \sim\Rightarrow\ \sim'.H.1)$ and $(\sim\ :\ \leq\ \leq)$ we obtain by modus ponens

$$(\sim\ :\ \leq\ \leq.C1)\quad (x_0\leq y_0)\wedge(y_0\leq x_0).$$

From $(\sim\ \sim\Rightarrow\ \sim'.H.2)$ and $(\sim\ :\ \leq\ \leq)$ we obtain by modus ponens

$$(\sim\ :\ \leq\ \leq.C2)\quad (y_0\leq z_0)\wedge(z_0\leq y_0).$$

For proving $(\sim\ \sim\Rightarrow\ \sim'.C)$, by $(\sim\ :\ \leq\ \leq)$, it suffices to prove

$$(\sim\ :\ \leq\ \leq.A)\quad (x_0\leq z_0)\wedge(z_0\leq x_0).$$

We prove the individual conjunctive parts of $(\sim\ :\ \leq\ \leq.A)$:

Proof of $(\sim\ :\ \leq\ \leq.A.1)$ $x_0\leq z_0$:

For proving $(\sim\ :\ \leq\ \leq.A.1)$, by $(\leq\ \leq\Rightarrow\ \leq)$, it suffices to prove

$$(\leq\ \leq\Rightarrow\ \leq.A)\quad (x_0\leq y_0)\wedge(y_0\leq z_0).$$

Formula $(\leq\ \leq\Rightarrow\ \leq.A)$ is true because it is the conjunction of $(\sim\ :\ \leq\ \leq.C1.1)$ and $(\sim\ :\ \leq\ \leq.C2.1)$.

Proof of $(\sim\ :\ \leq\ \leq.A.2)$ $z_0\leq x_0$: *[this cell is closed]*

Example 2: Natural Numbers. Here the symbol $\oplus$ denotes the addition of the natural numbers, x^+ denotes the successor of the natural number x (i.e. $x \oplus 1$) and $\equiv$ denotes the equality predicate.

PROPOSITION:

$(T)\quad \forall_{\langle x\rangle}\ 0 \oplus x \equiv x$

under the assumptions

$(\oplus 0)\quad \forall_{\langle m\rangle}\ m \oplus 0 \equiv m,$

$(\oplus +)\quad \forall_{\langle m,n\rangle}\ m \oplus n^+ \equiv (m \oplus n)^+.$

PROOF:

We try to prove by simplification:

$(T)\quad \forall_{\langle x\rangle}\ 0 \oplus x \equiv x$ *[this cell is closed]*

Now we prove the formula

$(T)\quad \forall_{\langle x\rangle}\ 0 \oplus x \equiv x$

by induction on x.

Induction base for x:

$(T.IB)\quad 0 \oplus 0 \equiv 0$

A proof by simplification works. *[this cell is closed]*

Induction hypothesis for x:

$(T.IH)\quad 0 \oplus x_0 \equiv x_0.$

Induction step for x:

$(T.IS)\quad 0 \oplus (x_0)^+ \equiv (x_0)^+$

A proof by simplification works.

Simplification of the lhs term:

$0 \oplus (x_0)^+ =$ by $(\oplus +)$

$(0 \oplus x_0)^+ =$ by $(T.IH)$

$(x_0)^+$

Simplification of the rhs term:

$(x_0)^+$

Using Theorema for Proof Training

One major use of the Theorema software is for training students how to construct mathematical proofs. The system is already used by us in the clssroom in the frame of a lecture on proof training. The provers can be used at various levels of *interaction*:

Passive Mode: The student studies the proofs produced automatically (eventually embeded in a textbook), in one of the following formats: *printed* on paper (when no computer is available); as *HTML files* (when no

Mathematica software is installed); or as *Mathematica notebooks* (needs only the free version of the software). The later two versions have the advantage that the proofs can be read at various levels of details, and they are also accessible over the internet.

The passive mode is intented to be used in the early stages of the proof−training process, when the teacher introduces the students to the proof−techniques in classroom presentations, and the students study individually the lecture notes distributed either on paper or electronically.

Interactive Mode: The student uses the Mathematica software and the Theorema software in order to produce proofs for a collection of theorems distributed by the teacher.This mode is suitable for the more advanced students, who can now interact with the system by modifying the theorems, adding new assumptions, proving additional lemmas, etc. In this way they get more insight into the proving mechanisms and they begin to train in formal model construction.

It is important to note that this increased level of interaction allows the students to learn individually at their own pace. This is crucial aspect in proof training, because typically the students have different background in formal mathematics, and also in various mathematical fields: some proof techniques are more familiar to ones than to the others because they are more used in the field of their particular background.

Active Mode: In addition to the *Theorema* provers, the student uses a special set of functions allowing a more intimate interaction with the provers. Namely, the proof is displayed in a step−by−step fashion, and the user has the possibility to direct the process in various ways:

– navigate backward and forward through the proof;

– remove redundant assumptions in order to simplify the proof;

– introduce additional subgoals in order to speed−up the process or to give "hints" when the problem is too complicated to be solved automatically;

– indicate which proof technique should be used at which point in the proof, etc.

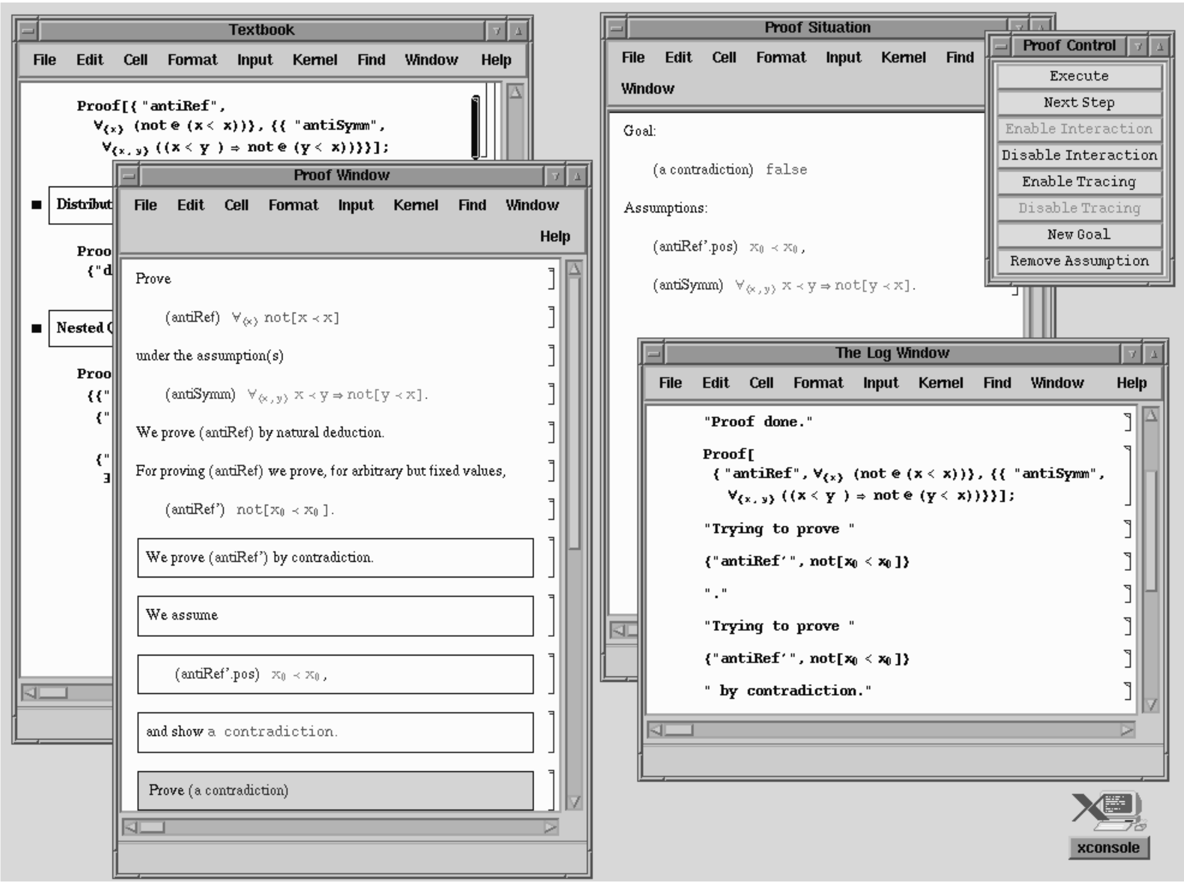

Figure 1: The windows of Theorema in active mode.

Figure 1 presents a typical screen when using the predicate−logic prover in active mode, with several Mathematica notebooks:

− "Proof Control" displays various buttons for controlling the proofs and the level of interaction.

− "Textbook" contains the theorems to prove. By selecting a command of type "Proof[...]" and then pressing "Execute" from the proof control window, the proof is initiated.

− "Proof Window" contains the actual text of the proof. In interactive mode, the partial proof obtained at the current stage is displayed with gray cells for the still unsolved subproofs and with framed cells for the most recent proof step which was executed. By pressing "Next Step" in "Proof Control" a further proof step will be executed.

− "Proof Situation" shows the current goal (formula to be proven) and the current assumptions.

− "Log Window" shows a trace of the proof steps and other operations performed.

Acknowledgements

Partially supported by the Austrian Science Fundation − project FO−1302.

References

[Boyer:88] Boyer, R.S., & Moore, J.S. (1988). *A Computational Logic Handbook.* Academic Press Inc.

[Buchb−Licht:81] Buchberger, B., & Lichtenberger, F. (1981). *Mathematics for Computer Science I − The Method of Mathematics* (in German). Second Edition. Springer Verlag.

[Buchb−et−al:97] Buchberger, B., Jebelean, T., Kriftner, F., Marin, M., Tomuta, E. ,& Vasaru, D.(1997). A Survey of the Theorema Project. *Proceedings of ISSAC'97*, *W. Kuechlin (ed)*, ACM Press, 384−391.

[Buchb−Vasaru:97] Buchberger, B., Ida, T., & Vasaru, D. (eds). (1997). *Proceedings of the First International Theorema Workshop, June 1997,* RISC Report 97−20.

[Clarke−Zhao:93] Clarke, E. M., & Zhao, X. (1993). Analytica − A theorem prover for Mathematica. *The Mathematica Journal* 3(1), 56−71.

[Constable:86] Constable, R.L., et al.(1986). *Implementing Mathematics with the Nuprl Proof Development System.*Prentice−Hall.

[Dolzmann−Sturm:96] Dolzmann, A., & Sturm, T. (1996). *REDLOG User Manual. Edition 1.0 for Version 1.0.*Universitaet Passau.

[Gordon:93] Gordon, M., & Melham,T.F. (1993). *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic.*Cambridge University Press.

[Homann:96] Homann, K. (1996) *Symbolisches Loesen mathematischer Probleme durch Kooperation algorithmischer und deduktiver Systeme (in German).* PhD thesis, Universitaet Karlsruhe.

[Horn:88] Horn, C. (1988). *The Oyster Proof Development System.*

[Huet−et−al:94] Huet, G., Kahn, G., & Paulin−Mohring, C. (1994). *The Coq Proof Assistant. A Tutorial. Version 5.10.* INRIA−Rocquencourt, CNRS−ENS Lyon.

[Paulson:96] Paulson, L. C. (1996). *Introduction to Isabelle.* Computer Laboratory, University of Cambridge.

[vanHermelen:89] van Harmelen, F., Ireland, A., Negrete, S., Smaill, A., & Stevens, A. (1989). *The Clam proof planner* .

[Wolfram:96] Wolfram, S. (1996). *The Mathematica Book.* Wolfram Media and Cambridge University Press.