Theorema 2.0: A System for Mathematical Theory Exploration

Wolfgang Windsteiger

RISC / JKU Linz, Austria
Wolfgang.Windsteiger@risc.jku.at
http://www.risc.jku.at/home/wwindste

Abstract. Theorema 2.0 stands for a re-design including a complete re-implementation of the Theorema system, which was originally designed, developed, and implemented by Bruno Buchberger and his Theorema group at RISC. In this talk, we want to present the current status of the new implementation, in particular the new user interface of the system.

Keywords: Theorema, mathematical assistant system, automated theorem proving, theory exploration, user interfaces, GPL

1 Introduction

Theorema 2.0 is—like its predecessor versions—based on Mathematica, which means that it is implemented in the Mathematica programming language and that it uses the Mathematica notebook front end as its user interface. Unlike the command-oriented interaction pattern typically propagated in Mathematica applications, Theorema 2.0 is heavily based on the graphical user interface capabilities supported in recent versions of Mathematica. As a result, the user needs the keyboard only for typing the mathematics (definitions, theorems, explanatory text) into the system, all actions to be performed are guided by the graphical user interface. This approach fosters the convergence of *writing formal mathematics* towards *writing normal mathematics*, because the overhead when *writing a Theorema document* compared to *writing a standard mathematical document* shrinks to almost zero. Moreover, the learning curve for using a mathematical assistant system is considerably flattened and the system will be more attractive, in particular for beginners.

A first version of Theorema 2.0 has already been presented in [Win12], where an emphasis was put on the new graphical user interface. In this presentation, we report on improvements and further extensions, but also on some new developments in the system that are not connected directly to the user interface.

Theorema 2.0 runs on all platforms, on which Mathematica is available. Mathematica is needed to run the system, but the Theorema system itself is open source licensed under GPL and is available at GitHub.

2 How to Use the Theorema System

When using (mathematical) software it is important for the user to exactly understand, for which intended purpose the software has been developed. Of course, there are examples of "legitimate fruitful abuse" (e.g. using a spreadsheet program to illustrate iterative algorithms when teaching mathematics) but in general the user is better off when she uses the software in line with the developers' intentions.

Much of mathematical software falls into the category of *algorithm libraries*, i.e. collections of algorithms for certain more or less well described application areas, like linear algebra, polynomial equations, geometry, differential equations, first order theorem proving, and the like. For each of the algorithms there is an input-output-specification and the systems differ in the range of problems that can be solved, the computational efficiency, or the input/output format. For the Theorema system, the situation is a bit more complex since Theorema tries to be a *mathematical assistant system* that supports the mathematician during all her mathematical activities, from first scratch work on some topic, through giving definitions of mathematical notions, formulating conjectures, proving theorems, formulating algorithms, executing algorithms on concrete input data, organizing the knowledge in order to reuse it in the future, composing lecture notes until finally writing a proper mathematical publication.

Although computer-support for automated or interactive theorem proving is in our main focus, the acceptance of a mathematical assistant system does not depend solely on the power of the prover. The huge variety of different working styles and habits is a major challenge for the user interface. Fig. 1 shows the new interface of Theorema 2.0, which consists of one or more *Theorema notebook documents* (left) and the *Theorema commander* (right).

In addition to standard Mathematica notebook features, a Theorema notebook supports *Theorema environments*, which contain blocks of formal mathematics such as definitions or theorems. The name of the environment ("Facts" in Fig. 1) can be freely chosen, it serves only structuring purposes and carries no semantics. Inside an environment, formal mathematics is written in cells of a particular style defined by the Theorema system, in fact by the *Theorema stylesheet* that is required to be used for Theorema notebooks. Formal mathematics is written in a very rich version of the language of predicate logic in common two-dimensional notation and must be executed (like Mathematica input in a standard Mathematica notebook) in order to become known to Theorema within the current session. It is important to note that the stylesheet does not only define the optical appearance of formal mathematics cells but also their functionality. We use the possibility to define actions to be executed before and after the cell content is processed and only so it is guaranteed that Theorema input is processed correctly. An important consequence of this setting is that Theorema does not interfere Mathematica in any way, the whole functionality of Mathematica can be accessed in standard Mathematica input cells. Every formal math cell carries a label through which the formula can be referenced (e.g. in a

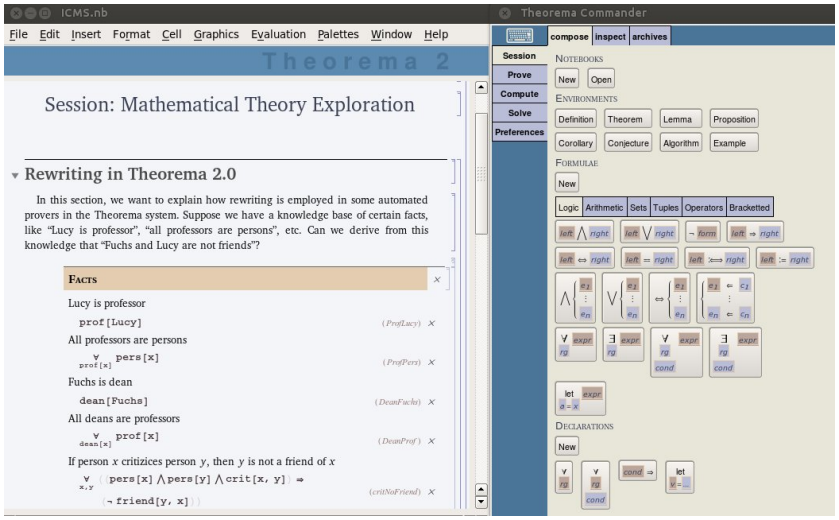

Fig. 1. Theorema 2.0 user interface

proof). In order to accommodate common practice, formal mathematics can be intermixed with plain informal text as shown in Fig. 1 also.

The Theorema commander is responsible to guide the user through all sorts of *activities* (to be selected in the left column in the commander) to be performed on the formal mathematics written in the notebooks. For every activity chosen, the commander opens a wizard that guides the user through the concrete *actions* in that activity. The "Session"-activity, for instance, has the actions "compose", "inspect", and "archives" helping to compose notebook content, inspect the formulas available in the current session, and setting up of knowledge archives, respectively.

Example 1 (How to prove a theorem). In order to prove a theorem, the system needs the theorem (= the proof goal), the knowledge base available, and the proving method to be applied (since Theorema is a multi-method system). The prove-activity with its actions "goal", "knowledge", "built-in", "prover", "submit", and "inspect" guides the user through this process.

1. The goal is specified by simply selecting the cell containing the theorem in the notebook.

2. The knowledge base consists of a) user-defined knowledge contained in some environments possibly spread over several notebooks and b) Theorema built-in knowledge that can be added to the knowledge base, e.g. on built-in arithmetic operations. For composing user-defined knowledge, Theorema 2.0 provides the *knowledge browser*, which contains for each Theorema notebook available in the current session a structured outline, in which all (groups of) formulas that should go into the knowledge base can be checked by mouse-click. A similar mechanism is used to select built-ins.

3. A prover in Theorema 2.0 is a collection of inference rules. Rules are grouped into categories (e.g. quantifier rules) that are displayed in the *rule browser*. In analogy to the knowledge browser (groups of) rules can be activated or deactivated by mouse-click. In addition, rule priorities for their application during the proof search and the granularity of the resulting proof can be adjusted by the user.

4. When all settings are finished the collected data is submitted to Theorema by mouse-click and the answer of the system is printed into the notebook directly underneath the environment containing the goal. In addition to a summary of all settings the answer contains most importantly a button to display the proof and a button to regenerate the proof using the original settings. The proof displays in a separate window with natural language explanation and the "inspect"-panel in the commander shows the corresponding proof tree as an alternative representation. Clicking the mouse in one of the representations will reposition the cursor in the other representation for quick navigation through a proof.

3 System Highlights

All actions to be performed in the system are mouse-driven, there is no need for the user to call complicated functions with lots of parameters in order to initiate some action. The interaction pattern should be more like using a web shop in the internet.

The proof methods are highly configurable through the Theorema commander. It should be easy for the user to adjust the behavior of the system as appropriate for a concrete problem.

Computation is an integral component in the provers. Every formula is silently simplified by computation as soon as it enters a proof. The computational knowledge applied is subject to user configuration, no user is forced to use Theorema built-in knowledge in a proof.

Formula input is supported both through palettes with mouse-click and through keyboard shortcuts. Structural input of formulas following their tree structure is supported, "invisible parentheses" guarantee correct grouping without any need for operator precedences.

Theorema notebooks can be setup to contain *theories*, which can be exported and stored in a format to be later imported and re-used in other notebooks or theories. The namespaces are separated such that naming collisions between theories are avoided.

References

[Win12] W. Windsteiger. Theorema 2.0: A Graphical User Interface for a Mathematical Assistant System. In Cezary Kaliszyk and Christoph Lth, editors, *Proceedings 10th International Workshop On User Interfaces for Theorem Provers, Bremen, Germany, July 11th 2012*, volume 118 of *Electronic Proceedings in Theoretical Computer Science*, pages 72–82. Open Publishing Association, 2012.