Interactive Proving in *Theorema*

Florina PIROI and Tudor JEBELEAN (*)

Research Institute for Symbolic Computation
A−4232 Hagenberg, Austria
{Florina.Piroi, Tudor.Jebelean}@risc.uni−linz.ac.at

The *Theorema* system is a software intended to assist the working mathematician during all phases of its mathematical work (proving as well), within one consistent logic and one coherent software system. The system is built upon the computer algebra software *Mathematica* [Wol96] which offers unique facilities for the input/output of logical expressions (including complex graphics), for programming by rewrite rules, and for interactivity. In order to make proofs easy to read and interact with, the provers of *Theorema* follow a *natural style* approach: the inferences imitate the natural steps used by human provers, and the rendering of the proofs is done in natural language. This approach has been pioneered by Buchberger, (see e.g. [Bu96], [BuJeVa98]) who designed and implemented the first versions of the predicate logic prover.

The *Theorema* system contains various provers for general and specific domains: a propositional and a predicate logic prover [BuEAl00], the Prove-Solve-Compute (PCS) prover for predicate logic with equality [VaPhD00], induction provers over natural numbers and over lists [BuVa97], a set-theory prover [Wind01], a Groebner Bases based prover for boolean combinations of polynomial equalities and inequalities, etc.

The provers operate on "proof-situations" consisting of a set of assumptions and a goal (sequents with only one goal). Each prover is composed of a set of inference rules, each rule is typically expressed as a rewrite rule which transforms a proof-situation into one or more proof-situations. The proof-object is a tree representation of the development of the proof: each node corresponds to a proof-situation, while its successors in the tree correspond to the inferred proof-situations. The root of the proof-object corresponds to the initial proof-situation, while the leaves correspond to "proved" (successfully solved), "disproved", "failed" or "pending" (not yet resolved) proof-situations.

The proof-object is used as an internal representation of the proof. The external representation of the proof (which is displayed to the user in a proof-notebook) is produced by a post-processing of the proof-object, which generates a structured cell representation in a *Mathematica* notebook. The structure of the cells reflects the tree structure of the proof-object, and each inference step is typically represented as a phrase in natural language accompanied by some formulae and referring to some other formulae by their labels

Normally, a proof call will try to solve/prove the goal by applying the inferences and the heuristics implemented in the prover (or the combination of provers) which is used. Even for powerful proving methods, many proof-problems arising in the current practice of mathematicians are too difficult to solve, mostly because of the large search involved, but also because the impossibility to anticipate in advance all the necessary facts which the formal proof will need as assumptions. In human proving practice, we usually try various possibilities and rely on our intuition and previous experience in order to find an elegant proof.

One of the advantages of natural-style proving is that the user can easily understand and follow both the inference rules which are available in the prover, as well as the proof situation at any given moment. We capitalize on this advantage and, in order to improve the effectiveness of the *Theorema* provers, we implemented a general mechanism which allows a flexible interaction between the user and the system during the development of the proof. Using this, one can choose between fully automatic (for a predetermined number of steps) and interactive proof-development, one can easily navigate through the proof-object, and one can provide various hints (e.g. suitable instantiations) to the prover.

When calling a prover in interactive mode, the system displays the proof in a special *proof-window*, and shows additionally a *proof-situation-window*, a *log-window* and several *Mathematica* style *menu-palettes*.

(*) Partially supported by the RISC phd scholarship program of the government of the Upper Austria and by the FWF (Austrian Science Foundation) SFB project P1302.

The proof-window displays the current situation of the (possibly unfinished) proof. This is in fact the user-friendly representation of the proof-object, and allows the user to navigate inside the proof-tree in order to continue the proof on a certain branch, introduce new branches, etc.

The proof-situation-window displays the proof situation (current goal and assumptions) associated to the node which is currently selected in the proof-tree. This window gives the user a good overview of the current proof situation, and also allows him to indicate the goal or assumption which is to be manipulated in the next proof step (e.g. for applying a certain inference). Pending (still unresolved) proof situations are shown with gray background, and the result of the most recent inference is shown in framed cells.

The log-window logs the main commands of the user and displays the errors and the warnings.

The menu-palettes contain various interactive commands and settings, each being represented by a button which triggers the respective action. Some of the palettes are general, some of them are specific to the particular prover(s) which is/are used. Some of the commands require arguments (e.g. Execute needs a Prove command, NewGoal needs a formula). In order to correctly use such a command, the user selects first the cell containing the needed information.

In very concise words the main menu-palettes are: the **Prove** palette, containing the main commands for controlling the development of the proof (start/finish execution etc.); the **Proof** palette which contains commands that allow the modification of a proof situation of a certain proof step; the **Simplification** palette that provides "shortcuts" to various proof simplification operations which are mostly used; the **Provers** palette which allows the user to select the domain-specific prover which will be used for the next inferences.

The implementation of the palettes is done in such a way that it is easy to add new palettes and new buttons, and also to arrange them in various ways on the user-screen. We plan to enhance these menus according to the results of the experiments with the users. In particular, we plan to add a special dynamic palette which is shows suggestions for the next inference step.

The work presented here has an experimental flavour: one has to try out various types of simplification and interaction schemes in a real environment in order to take the appropriate decisions, and one has to implement the system in a flexible way in order to be able to easily implement the changes suggested by practical experiments with the users (currently, besides the members of the *Theorema* group, we also use as alpha−testers the PhD students enrolled in a special proof-training program at our institute). The presentation will be accompanied by a live demo of the system.

References

[Bu96] B. Buchberger. Using Mathematica for Doing Simple Mathematical Proofs. InProceedings of the 4th Mathematica Users' Conference, Tokyo, November 2, 1996, pages 80-96, Wolfram Media Publishing, 1996.

[BuEAl00] B. Buchberger and C. Duprè and T. Jebelean and K. Kriftner and K. Nakagawa and D. Vasaru and W. Windsteiger. The Theorema Project: A Progress Report.In M.Kerber and M.Kohlhase, editor, Proceedings of Calculemus 2000: Integration of Symbolic Computation and Mechanized Reasoning,A.K.Peters,Natick,Massatchussets, 2000.

[BuJeVa98] B. Buchberger and T. Jebelean and D. Vasaru. *Theorema:* an Integrated System form Computation and Deduction in Natural Style. In Proceedings of CADE 98 (Satellite workshop on integration of computing and proving).

[BuVa97] B. Buchberger and D. Vasaru. An Induction Prover for Equalities over Lists. In *Proceedings of the First International Theorema Workshop, Hagenberg, Austria, June 9−10,1997*, RISC−Reportsseries No. 97−20,1997.

[VaPhD00] D.Vasaru−Duprè. Automated Theorem Proving by Integrating Proving, Solving and Computing. The Research Institute for Symbolic Computation, Johannes Kepler University, May 2000. RISC report 00−19.

[Wind01] W. Windsteiger. A Set Theory Prover in Theorema: Implementation and Practical Applications. The Research Institute for Symbolic Computation, Johannes Kepler University, May 2001. RISC report 01−07.

[Wol96] S.Wolfram.The Mathematica Book, Wolfram Media and Cambridge University Press, 1996.