How Mathematicians Prove Theorems

Erica Melis[1]
School of Computer Science[2]
Carnegie Mellon University
Pittsburgh, PA 15213
email: melis@cs.cmu.edu

Abstract

This paper analyzes how mathematicians prove theorems. The analysis is based upon several empirical sources such as reports of mathematicians and mathematical proofs by analogy. In order to combine the strength of traditional automated theorem provers with human-like capabilities, the questions arise: Which problem solving strategies are appropriate? Which representations have to be employed? As a result of our analysis, the following reasoning strategies are recognized: proof planning with partially instantiated methods, structuring of proofs, the transfer of subproofs and of reformulated subproofs. We discuss the representation of a component of these reasoning strategies, as well as its properties. We find some mechanisms needed for theorem proving by analogy, that are not provided by previous approaches to analogy. This leads us to a computational representation of new components and procedures for automated theorem proving systems.

Introduction

Automated theorem proving is a well-established area of Artificial Intelligence, not least because reasoning in mathematics is a potent special case of human reasoning that lends itself particularly well to mechanization and computer support. Automated theorem proving systems have attained a remarkable strength when it comes to pure deductive search. They are, however, still weak with respect to a comprehensible presentation of computer-generated proofs, to long range planning or other global search and control issues. Therefore methods and techniques become more prominent again that more closely follow the reasoning patterns observed in humans, e.g., by Allen Newell [Newell 1981] and, more recently, by Alan Bundy [Bundy 1988]. To combine the strength of traditional automated theorem provers with human-like capabilities, the questions arise: Which problem solving strategies are appropriate? Which representations have to be employed?

This paper addresses these questions by analyzing human mathematical theorem proving, and by drawing conclusions for the design of a new generation of automated and interactive theorem provers. First we present and analyze some reports on human mathematical theorem proving and textbook proofs, then we summarize requirements for ingredients of theorem proving systems, and finally we suggest computational representations which meet these requirements.

[1]This work was supported by the Max Kade Foundation
[2]On leave from University of Saarbrücken, Germany

Empirical Evidence

The hypothesis is that the traditional way of automated theorem proving[3] (as captured in most of todays textbooks, e.g., [Boyer and Moore 1979]) does not reflect how humans find and present mathematical proofs. Also, previous techniques of computational analogy in theorem proving which, in a nutshell, are symbol mapping and transfer of single proof steps, are inadequate.

Reports of Mathematicians

In the early 80ies the German mathematician Gerd Faltings solved a mathematical problem, called Mordell's Conjecture[4]. Mordell's Conjecture has been considered a hard mathematical problem and it took over 60 years to solve it. Faltings gave an interview to a German scientific journal [Faltings and Decker 1983] informally explaining the way he solved the problem. This interview provides several general insights into problem solving and proof mechanisms in mathematics. Hence, it is also a matter of interest for the design of automated theorem proving systems. Faltings reported:

- *"Ich muß sagen, daß ein wesentlicher Teil des Beweises im Prinzip schon da war, den ich nur entsprechend übertragen habe"*
  Translated: I should say that basically an important part of the proof was already there, and I only **transferred this part appropriately**.
- *"Man hat Erfahrungen, daß bestimmte Schlüsse unter bestimmten Voraussetzungen funktionieren. Als erstes überlegt man sich daher, wie der Weg aussehen könnte. Man überlegt sich also im Groben: Wenn ich das habe, könnte ich das zeigen und dann das nächste. Hinterher muß man die Details einfügen und sieht, ob man es auch wirklich so machen kann."*
  Translation: We know from experience that certain inferences are usually successful under certain prerequisites. So first we ponder about a reasonable way to proceed to prove the theorem. In other words, we **roughly plan**: If we get a certain result the next result will follow and then the next etc. Afterwards we have to **fill in the details**, and to **check** whether the plan really works."
- *"Es kommt aber auch durchaus vor, daß man mal da sitzt, nicht mehr weiter weiß und dann probiert, wohin der Weg*

[3]Which is characterized by a stepwise and linear application of basic rules,

[4]Mordell's Conjecture: Algebraic curves of order 2 or more have finitely many rational points.

führt."
Translation: It sometimes happens that there is no other way than trial and error.

These quotations not only characterize ways of proving extraordinarily difficult mathematical theorems, but also highlight some common mathematical theorem proving procedures.

An additional feature of mathematical theorem proving shows up in the report [Leron 1983] of a mathematical journal. There, Uri Leron shows, how proofs are better comprehensible (and easier to find - E.M.) by structuring them into different levels. He describes a common procedure to present proofs of mathematical theorems that can be seen, from our point of view, as a procedure supporting proofs of theorems:
- Start with a top level partial proof that gives the essence of the proof ("proof idea").
- A second level then supplies (partial) proofs for unsubstantiated statements, details for general descriptions, specific constructions for objects, whose existence has been merely asserted etc. If some subproof is itself complicated, we may choose pushing the details further down to lower levels. And so we continue down the hierarchy of subprocedures.
Here is one of his examples:
THEOREM 1: There exist infinitely many triadic primes (i.e., numbers of the form 4k+3).
Proof in the structured style:
Level 1: Suppose the theorem is false and let $p_1, p_2, \ldots p_n$ be all triadic primes. We construct (in level 2) a number M having the following properties:
(a) M as well as all its factors are different from $p_1, p_2, \ldots p_n$,
(b) M has a triadic prime factor.
These two properties clearly produce a contradiction, as we get a triadic prime which is not one of $p_1, p_2, \ldots p_n$.
Level 2: Let $p_1 = 3$ and $M = p_2, \ldots p_n + 3$.
(a) can be proved, since none of $3, p_2, \ldots p_n$ divides M.
(b) can be proved indirectly, assuming that all of M's prime factors were monadic (i.e., of form 4k+1). Then M, as a product of monadic numbers, is itself monadic (which is proved on level 3). This yields a contradiction.
Level 3: Any product of monadic numbers is monadic.
Figure 1 shows the structure of this proof. Leron mentions a theorem that is built by exchanging "triadic" by "monadic" in THEOREM 1, and which can be proved analogously. The two proofs are similar on "top-level", but all lower levels have to be modified, and this modification is not easy to find. On the other hand, he also presents proofs whose analogues are similar down to level 3, which means that only subproofs of level 4 and lower have to be adjusted. From these examples and our experience we *generalize* that proofs can be analogous at different levels of detail and, hence, analogy is executed by transferring partial proofs at different levels.

The analysis of these two sources (other examples can be found in [Polya 1957; Polya 1954; Hadamard19 45; van der Waerden19 64]) suggests a change of the traditional theorem proving paradigm and highlights the following human problem solving strategies:

1. **Proof planning** with partially instantiated methods, where also incomplete proof plans are allowed. Top level methods are refined by lower level methods,

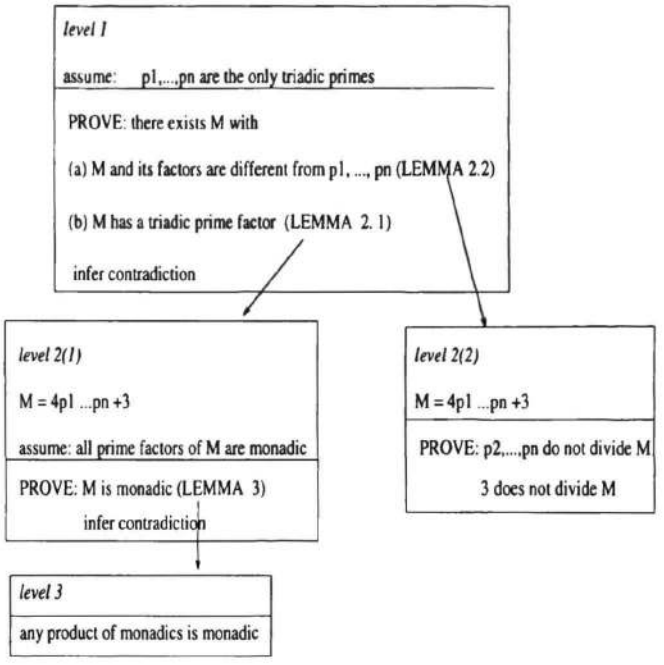

Figure 1: Plan of the proof of THEOREM 1

2. **Structuring** of methods, a process, which has to be embedded into proof planning,
3. **Trial and error**,
4. **Analogy** which is embedded into proof planning. Analogy includes the transfer of proof ideas, partial proofs, and methods.

The presented sources provided details on how mathematicians solve problems. The next examined source is a textbook. Even though we don't believe that textbooks always reflect mathematician's ways of problem solving, the analysis provides insights in what actually is considered by mathematicians to be an analogy. The examination also shows which mechanisms are involved in theorem proving by analogy.

Textbook Analysis

Theorem proving by analogy, as sketched in figure 2, means to find a proof for a target problem on the basis of a given proof of a source problem, which is similar to the target problem.

Traditionally, the analogy between the source proof and the target proof was realized by establishing a mapping from the primitive symbols of the source theorem onto the symbols of the target theorem, and by extending this map such that it provides a proof of the target theorem when applied to the single steps of the source proof. Here, the primitive symbols are those symbols of the signature in which the theorems are expressed. In other words, traditional approaches [Kling19 71; Munyer 1981; Owen 1990] are centered around symbol mapping and the transfer of single proof steps. They do not try to find another representation of, say, the source theorem and are, thus, *highly dependent on the actual representation of the theorems.*

For our research on automated theorem proving by analogy we studied the mathematical textbook "Halbgruppen und

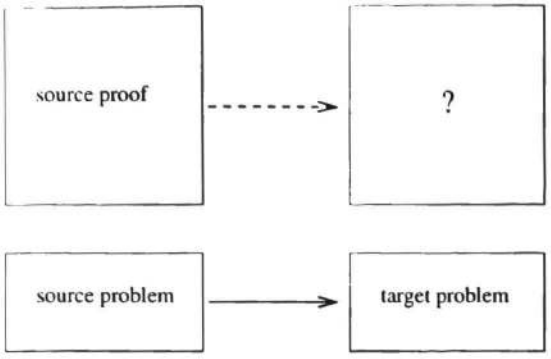

Figure 2: Analogy in theorem proving

Automaten" [Deussen 1971] that is commonly used in undergraduate classes on automata theory in Germany. The book is divided into three chapters:
Semi-groups and Relations
Semi-groups and Semi-moduls
Automata.
In this book many theorems are proved by analogy to theorems in a previous chapter and this gives some of the distinct flavor of this particular textbook. So, we scanned and analyzed all analogies mentioned in this textbook. We found that many analogies do not fit into the framework offered by traditional approaches to analogy in theorem proving (For a detailed analysis of the analogous proofs see [Melis 1993b] and [Melis 1993a]):

- For some analogies (e.g., theorem 5.7 and theorem 5.2[5]) you first have to find the right level of *abstraction* before any transfer of proofs.
- Reformulations are involved in many of these analogies (e.g., theorem 5.7 and theorem 6.9) that are not just symbol- or even term mappings. A striking example for a more complicated reformulation is the change of unary functions to binary functions that additionally requires to add certain proof lines and to change others. We have found several classes of reformulations, namely
  - Normalization
  - Abstraction
  - Direct reformulation
- Many proofs by analogy (e.g., theorem 4.8) result from transferring *parts* of source proofs to parts of the target proof. In order to obtain appropriate subproofs, we have to decompose the original proofs.
- Very often, mathematicians describe their analogy procedure as *applying the same method*, in particular if the method is named, such as the well known Diagonalization method of Cantor. Otherwise they state that the target proof is done *analogously* to the source proof.

Design Requirements

The above analysis suggests the following strategies for theorem proving:

- a planning framework that employs (partially specified) methods and subproofs
- decomposition of proofs
- analogical transfer of subproofs found by decomposition
- analogical transfer based on some reformulation, rather than just by symbol mapping

Since our aim is to automate theorem proving, we need a computational simulation of the strategies and a computational representation of their ingredients. Planning is a field of Artificial Intelligence and we can use a planning framework and its ingredients, namely operators, for proof planning[6]. Then theorem proving by analogy can, in principle, be modelled as derivational analogy (see [Carbonell 1986]). The model for analogy-driven proof plan construction, given in [Melis 1993b; Melis and Veloso 1994], employs planning operators, defined as *methods*, and requires meta-methods in order to change these operators. Now we present a representation for both operators and meta-methods.

Operators According to the previous sections, the representation of planning operators should meet the following conditions:

- Operators must have pre- and postconditions, as usual in planning, which contain problems to be subgoaled on.
- They should contain constraints to restrict the search for operators.
- Operators should cover mathematical methods, such as the Diagonal method.
- Operators must be able to represent partially unspecified methods, and incomplete proofs, i.e., the language for operators should contain variables for methods and parameters.
- Operators should be ready for reformulation and restructuring.

The last requirement can be satisfied by splitting the representation of an operator into a declarative part, suitable for reformulations, and a procedure that interprets the declarative part. We suggest to represent operators by *methods* which are frame-like structures, where all slots, but procedure, have declarative fillers (see also [Huang, Kerber, and Kohlhase 1992]); the slots of methods are:

- *parameters* which can be instantiated.
- *preconditions* which are inputs that specify the applicability of a method.
- *postconditions* which are outputs of the method application, e.g., a derived problem.
- *constraints* to restrict the search for methods.
- *proof scheme* which is a declarative proof scheme and contains lines of a partial proof[7], maybe with variables for terms, formulas, and even for the justifying method (the most right entry of a line).
- *procedure* which is a schema-interpreting procedure that is applied to the scheme.
- *history* which contains a trace of certain changes of the method, particularly abstractions, for the purpose of their revision.

[5]The numbers follow the original numbering of theorems in [Deussen 1971].

[6]This has also been proposed by Alan Bundy in [Bundy 1988].

[7]In the Natural Deduction calculus.

Method: Diagonal			
parameter	F, c, non: function, $M1, M2$: structures, 1,0: elements		
preconditions	(1)$\forall x_{M2} \exists y_{M1}(F(y) = x)$ (5)$\forall x(x = x)$ (6)$0 \neq 1$ (7)$(H \vee \neg H)$,		
postcondition	$\perp$		
constraints			
proof scheme	1. ;	$\vdash \forall x_{M1}(G(x) = non\ cF(xx) \wedge G \in M2)$	(PLAN1)
	2. ;	$\vdash \exists y_{M1}(F(y) = G)$	($\forall$D,(1) 1)
	3. ;	$\vdash F(x_0) = G$	($\exists$D 2)
	4. ;	$\vdash F(x_0)(x_0) = G(x_0)$	(equ 3)
	5. ;	$\vdash ((cF(x_0x_0) = 0 \rightarrow cG(x_0) = 1)$ $\wedge (cF(x_0x_0) \neq 0 \rightarrow cG(x_0) = 0))$	(PLAN2 1)
	6. ;	$\vdash c(F(x_0x_0)) = 0 \vee c(F(x_0x_0)) \neq 0$	($\forall$D,(7))
	7. ;7	$\vdash cF(x_0x_0) = 0$	(HYP)
	8. ;7	$\vdash cG(x_0) = 0$	(equ. 7 4)
	9. ;7	$\vdash cG(x_0) = 1$	($\forall D, \wedge D \rightarrow$D 5 7)
	10. ;7	$\vdash 0 = 1$	($\wedge$I,equ,(5) 9 8)
	11. ;7	$\vdash \perp$	($\wedge$I,$\perp$I,(6))
	12. ;12	$\vdash cF(x_0x_0) \neq 0$	(HYP)
	13. ;12	$\vdash cG(x_0) = 0$	($\forall$D,$\wedge$D,$\rightarrow$D 12 5)
	14. ;	$\vdash cF(x_0x_0) = 0$	(equ 13 4)
	15. ;12	$\vdash cF(x_0x_0) = 0 \wedge \neg cF(x_0x_0) = 0$	($\wedge$I 14 12)
	16. ;12	$\vdash \perp$	($\perp$I 15)
	17. ;	$\vdash \perp$	($\vee$D 16 11 6)
procedure	schema-interpreter		
history			

An example is the Diagonal method, which in fact covers a mathematical method and is applicable inter alia in the proof of Cantor's theorem, the proof of the uncountability of real numbers, the proof of the unsolvability of the halting problem, as well as in the proof of Gödel's theorem of the incompleteness of arithmetic (this method is discussed in detail in [Melis 1994]). Here, PLANs are variables for unspecified submethods and the first line of the proof scheme expresses, for instance, that a lemma has to be proved somehow, which is also known as the diagonal lemma.

Meta-methods In order to meet the requirements for restructuring and reformulation, to reduce the dependence on the actual representation of the given theorem, and in order to obtain analogies at several levels of abstraction and several levels of detail, we use meta-methods. Meta-methods are procedures which map a method to another method or to several connected methods respectively. The meta-methods employed for analogy-driven proof plan construction have as a parameter the postcondition of the, only partially specified, target method.

Besides normalizing, abstracting, and direct reformulating meta-methods, restructuring meta-methods are defined that split one method into several connected methods. A very simple example is the splitting of methods with a conjunctive postcondition $(F_1 \wedge F_2)$ into two methods, one with postcondition F_1 and another with postcondition F_2.

The restructuring meta-methods are applied to obtain those parts of proofs that can be transferred analogically at once and to keep subproofs, that can not be transferred, small. Normalizing meta-methods are applied, e.g., to make postconditions of somehow analogous methods comparable, by simple reformulations such as replacing a proof assumption which is a conjunction by a set of two proof assumptions. The abstracting meta-methods are used to enable analogies on a more abstract level. An example is the meta-method `Functional-Abstr` (see next page) that was applied to find an analogous proof to theorem 5.2 in the analyzed textbook. This meta-method is applicable to a source method M with a target problem P as parameter, if its precondition is satisfied.

In the representation of this meta-method the x_i are the maximal terms in $\phi(x_1 \ldots, x_n)$. `term` is a metavariable for a reference term that contains only one variable.
`Functional-Abstr` reformulates a method M to a method M' by executing PROCFUNC:

- Replace in M all occurrences of instances of the reference term `term`(t_i) by $f_a(t_i)$, where f_a is a new function variable.
- Delete membership declarations that became superfluous by the introduction of f_a, and delete the corresponding quantifiers.

  For example, if `Functional-Abst` replaces the reference term $(h \cdot x)$ by $f_a(x)$, then the quantifier and membership declaration of h, $(h \in F)$ become superfluous.

Metamethod: `Functional-Abstr`	
parameter	P: problem
preconditions	there exists a formula Φ of the form $\forall x_1, \ldots, x_n, y_1 \ldots y_k$ (membership declaration $\rightarrow$ $\phi(x_1 \ldots, x_n) \rightarrow \phi(\texttt{term}(x_1), \ldots, \texttt{term}(x_n)))$ and $\Phi \in$ postcondition(M) and $\Phi \notin$ P
postconditions	M'=M[$\texttt{term}(x_i)/f_a(x_i)]_i$
procedure	PROCFUNC(see below)
rating	

- Add (Functional-Abst: Φ) to the history slot of M.
- Add the new parameter f_a to the parameter slot of M.

The traditional reformulations, such as symbol mapping, belong to the direct reformulating meta-methods. But we need more of them, for instance, a meta-method that changes unary functions to binary functions. More complicated meta-methods are presented in [Melis 1993b]. The meta-methods correspond to heuristics employed by mathematicians and have to be extracted empirically which we did by analyzing the analogies occurring in the examined textbook.

Conclusions

By analyzing empirical sources we examined real proof strategies and methods. The results challenge traditional automated theorem proving on the basis of reports by mathematicians of how they solve problems. We revealed several strategies and components of human mathematical reasoning that are interesting for automated and computer-supported reasoning. The insights into how mathematical analogies compare to current models of theorem proving by analogy include that analogical reasoning may require non-trivial reformulations and restructuring.

Following the analysis, we discussed some requirements that the empirical results impose on components of a computational system, which have to be considered for the implementation of human-style theorem proving system. This led us to a computational representation of a structure used in proof planning and procedures employed by an analogy-driven proof plan construction. The actual analogy-driven proof-plan construction, that is an extended and modified analogical replay (see [Veloso 1992]), is presented in [Melis 1993b] and [Melis and Veloso 1994].

References

Boyer, R.S. and Moore, J.S. (1979). *A Computational Logic*. Academic Press, London.

Bundy, A. (1988). The use of explicit plans to guide inductive proofs. In E. Lusk and R. Overbeek, editors, *Proc. 9th International Conference on Automated Deduction (CADE)*, volume 310 of *Lecture Notes in Computer Science*, pages 111–120, Argonne. Springer.

Carbonell, J.G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R.S. Michalsky, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 371–392. Morgan Kaufmann Publ., Los Altos.

Deussen, P. (1971). *Halbgruppen und Automaten*, volume 99 of *Heidelberger Taschenbücher*. Springer.

Faltings, G. and Deker, U. (1983). Interview: Die Neugier, etwas ganz genau wissen zu wollen. *bild der wissenschaft*, (10):169–182.

Hadamard, J. (1945). *The Psychology of Invention in the Mathematical Field*. Princeton Univ. Press, Princeton.

Huang, X., Kerber, M., and Kohlhase, M. (1992). Methods - the basic units for planning and verifying proofs. SEKI-Report SR-92-20 (SFB), Universität des Saarlandes.

Kling, R.E. (1971). A paradigm for reasoning by analogy. *Artificial Intelligence*, 2:147–178.

Leron, U. (1983). Structuring mathematical proofs. *The American Mathematical Monthly*, 90:174–185.

Melis, E. (1993a). Analogies between proofs – a case study. SEKI-Report SR-93-12, Universität des Saarlandes, Saarbrücken.

Melis, E. (1993b). Change of representation in theorem proving by analogy. SEKI-Report SR-93-07, Universität des Saarlandes, Saarbrücken.

Melis, E. (1994). A representation of the diagonal method. Technical report, Carnegie Mellon University, Dept. of Computer Science, Pittsburgh, Pennsylvania, U.S.A..

Melis, E. and Veloso, M.M. (1994). Analogy makes proofs feasible. In D. Aha, editor, *AAAI-workshop on Case Based Reasoning*, Seattle.

Munyer, J.C. (1981). *Analogy as a Means of Discovery in Problem Solving and Learning*. PhD thesis, University of California, Santa Cruz.

Newell, A. (1981). The Heuristic of George Polya and its Relation to Artificial Intelligence. Technical Report CMU-CS-81-133, Carnegie-Mellon-University, Dept. of Computer Science, Pittsburgh, Pennsylvania, U.S.A..

Owen, S. (1990). *Analogy for Automated Reasoning*. Academic Press.

Polya, G. (1954). *Mathematics and Plausible Reasoning*. Princeton University Press, NJ.

Polya, G. (1957). *How to Solve it*. 2nd ed. Doubleday, New York.

van der Waerden, B.L. (1964). Wie der Beweis der Vermutung von Baudet gefunden wurde. *Abh.Math.Sem.Univ.Hamburg*, 28.

Veloso, M.M. (1992). *Learning by Analogical Reasoning in General Problem Solving*. PhD thesis, Carnegie Mellon University, CMU, Pittsburgh, USA. CMU-CS-92-174.