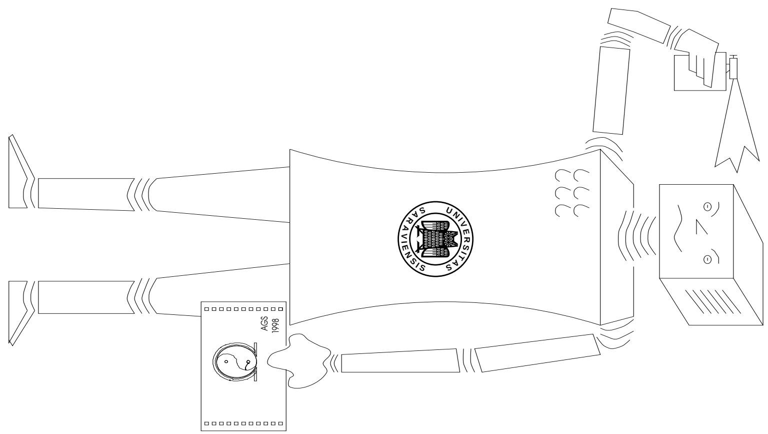
SEKI Report

UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY
WWW: http://www.ags.uni-sb.de/

Proof Presentation Based on Proof Plans

Erica Melis[1]

SEKI Report SR-98-08



Proof Presentation Based on Proof Plans

Erica Melis

Abstract

The paper addresses two problems of comprehensible proof presentation, the hierarchically structured presentation at the level of proof methods and different presentation styles of construction proofs. It provides solutions for these problems that can make use of proof plans generated by an automated proof planner.

1 Introduction

The traditional automated theorem provers' output is at best a presentation of steps representing logic calculus rules.[1] Therefore, this output is hardly readable for an untrained user, let alone understandable as a mathematical proof, even for mathematicians. For instance, the proof of the theorem

THEOREM: *Let K be an ordered field. If $a \in K$, then $1 < a$ implies $0 < a^{-1} < 1$ (and vice-versa).* ■

is

Let $1 < a$. According to lemma 1.10 we then have $a^{-1} > 0$. Therefore $a^{-1} = 1a^{-1} < aa^{-1} = 1$, [10] whereas its proof as provided by the automated theorem prover OTTER [11] is the following

```
1 [] x=x.
2 [] -(x<y)| -(0<z)|x*z<y*z.
3 [] -(x<y)| -(y<z)|x<z.
4 [] -(x<y)| -(y<x).
5 [] x=0| -(0<x)|0<inv(x).
7 [factor,4,1,2] -(x<x).
8 [] -(0<inv(a)| -(inv(a)<1).
9 [] 0<1.
10 [] x=0|x*inv(x)=1.
12,11 [] 1*x=x.
13 [] 1<a.
19 [hyper,3,9,13] 0<a.
27 [para_into,19.1.2,5.1.1,unit_del,7,19] 0<inv(a).
48,47 [para_from,10.1.1,19.1.2,unit_del,7] a*inv(a)=1.
60 [hyper,2,13,27,demod,12,48] inv(a)<1.
65 [hyper,8,27,60] F.
```

This problem was recognized long ago and several attempts have been made to produce proof presentations based on rules of the natural deduction (ND) calculus and otherwise readable proofs. Proof presentation in natural language has recently been realized in ILF [6] and PROVERB [7] which slightly abstract proofs before the presentation. ILF provides a schematic verbalization whereas PROVERB returns a more elaborate proof presentation at the so-called assertion-level[2] that employs linguistic knowledge in order to combine single assertion-level steps.

*The author was supported by the Deutsche Forschungsgemeinschaft Sonderforschungsbereich SFB 378.

[1] such as resolution and paramodulation

[2] An assertion is an axiom or definition the proof refers to.

However a proof verbalization at the assertion-level is not necessarily the most natural and best way to communicate a proof to mathematicians or to students. As texts in mathematics books show, verbalized proof steps can be far *more abstract* than assertion-level steps and may contain explaining text too. For example, in [1], proofs of limit theorems contain phrases like "We need to estimate the magnitude of..".

What is an *appropriate* presentation of proofs for teaching and learning? What kind of system can support the student (and the teacher) to understand (and to present) a proof properly? As we shall show in § 2,

1. a hierarchical presentation of a proof, as compared with the standard linear presentation, is more communicative of the mathematical ideas behind the proof and
2. making explicit the path leading to the construction of mathematical objects helps students to understand proofs.

How can these insights/results be translated to requirements for the representation of a proof in an automated system in order to yield an appropriate proof presentation? We shall demonstrate that in order for automated math proof systems to be more useful in real life teaching and learning, it is necessary (and possible) to generate an output that will be understandable by mathematicians and by students. Moreover, we suggest that automatically generated proof plans can serve as a basis for an appropriate proof presentation that resembles as much as possible 'good' proofs generated by humans.

In this paper, we shall consider two problems whose solutions can make use of the features of proof plans, namely (1) the hierarchically structured presentation at the level of proof methods and (2) different presentation styles.

2 Empirical Results on Proof Presentation

Empirical class room studies have provided useful insights about the presentation of proofs for students.

Structured Proofs

Catrambone [3] proved empirically that people often memorize a sequence of problem solving steps when studying example in mathematics or physics without learning what domain-dependent subgoals or subtasks these steps achieve. As a result, they have trouble solving novel problems that contain the same structural elements. Consequently, the emphasis of subgoal structure yields a more successful transfer to novel problems and teaching this structure helps to solve problems that require novel techniques to solve them. In [4] he showed that even meaningless labels that structure a proof support the problem solving capabilities of people.

Linear proofs, which proceed unidirectionally from hypotheses to conclusion, may be suitable for mechanically checking the validity of the proof, but they leave the student mystified as to what the main ideas of the proof are, why those steps are taken and why in that order. Structured proofs consist of small, independent modules which are arranged hierarchically. First the top level is presented, giving the main idea of the proof, and then the next levels which supply successively finer level of detail till we get to the bottom where all the 'leaks' left at higher levels are plugged and the proof is 'waterproof'. The top level gives a global view of the proof at the cost of leaving some gaps which are then filled at the lower levels. For a comparative analysis of many proofs, elementary as well as advanced, in the two styles, see [8].

Leron has shown that, as opposed to the linear presentation of proofs that merely enables students to check the correctness of a proof, a *hierarchically structured* presentation makes explicit the intuitive and global ideas behind the proof, hence helps students to *understand* and to be able to reproduce a proof rather than just check its correctness [8, 9].

For teachers and textbook authors, who are aware of the deep ideas behind a proof, presenting the proof in a structured format is an easy task. For students or readers structuring a linear proof, amounts to re-discovering those deep ideas from scratch, and may be a very difficult, even impossible task

Construction Proofs

Leron also points out that most non-trivial proofs involve a pivotal act of construction, and that there are two=A0 different ways of organizing the proof around that construction: one, in which the proof starts with the detailed construction 'let us define a function' before the reader or listener has any idea where it is going and why this construction is taken; two, in which the constructed object and its properties are first *postulated* and are then used to show how to the conclusion of the theorem more or less immediately follows, and only then does the proof proceeds to the (often complex) details of how to actually construct the postulated object.

3 Proof Plans

As opposed to classical automated theorem proving which searches for a proof at the level of a logic calculus, proof planning automatically constructs a plan that represents a proof at the more abstract level. A plan consists of methods (which are plan operators). These methods represent (i) common patterns in proofs, e.g. the common structure of an induction proof or (ii) common (procedural) proof techniques such as term simplification or the computation of a derivative. Examples of methods are `Diagonalization`, `Induction`, estimation methods, lifting methods, and the application of an important theorem such as the Hauptsatz of Linear Algebra (each n-dimensional vector space has a basis of n linearly independent vectors), the Hauptsatz of Number Theory (each natural number can be uniquely represented as the product of prime numbers).

Proof planning chains abstract methods together and then, if desired by the user, recursively expands abstract methods to less abstract subplans. During the expansion of methods, previously hidden subgoals become visible and subject for proof planning. Hiding of subgoals is a form of hierarchical planning (precondition abstraction) and using more abstract methods is another one (operator abstraction). Both yield a hierarchically structured proof plan. For instance, a proof plan for LIM+, as explained below, is depicted in Figure 1, where the dashed methods indicate a lower hierarchical level.

3.1 Class of Examples: Limit Theorems

We shall illustrate the use of proof plans with an example from the class of limit theorems that can be automatically planned by ΩMEGA [2]. The class of limit theorems includes the well-known theorem LIM+ from calculus that states that the limit of the sum of two functions in the real numbers, $\mathbb{R}$, is the sum of their limits.

$$\lim_{x\to a} f(x) = L_1 \wedge \lim_{x\to a} g(x) = L_2 \to \lim_{x\to a} (f(x) + g(x)) = L_1 + L_2,$$

which, after expanding the definition of $\lim\limits_{x\to a}$, becomes

$\forall\epsilon_1\exists\delta_1\forall x_1(0<\epsilon_1 \to 0<\delta_1 \wedge |x_1-a|<\delta_1 \to |f(x_1)-L_1|<\epsilon_1) \wedge$
$\forall\epsilon_2\exists\delta_2\forall x_2(0<\epsilon_2 \to 0<\delta_2 \wedge |x_2-a|<\delta_2 \to |f(x_2)-L_2|<\epsilon_2)$
$\to \forall\epsilon\exists\delta\forall x(0<\epsilon \to 0<\delta \wedge |x-a|<\delta \to |(f(x)+g(x))-(L_1+L_2)|<\epsilon)$

Other class members are, e.g., similar theorems about differences (LIM-) and products (LIM*), Composite that states that the composition of two continuous functions is continuous, ContIfDeriv that states that a function having a derivative at a point is continuous there, Cont+ that states that the sum of two continuous functions is continuous and a similar theorem (Cont*) about products, UNIFcont that says that a uniformly continuous function is continuous, and theorems like LIMsquare: $\lim\limits_{x\to a} x^2 = a^2$. The typical way a mathematician goes about to prove such a theorem is to (incrementally) construct a real number δ that depends on ϵ.

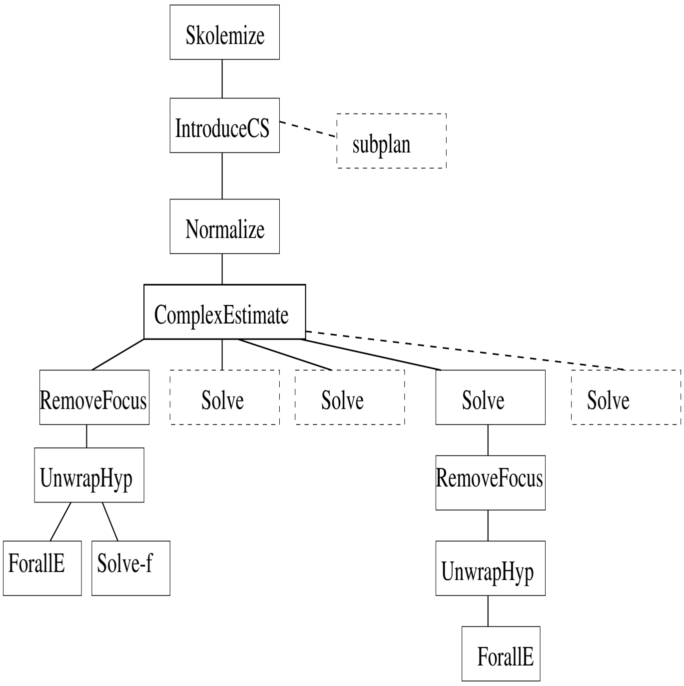


Figure 1: Proof plan of LIM+.

In the remainder, $/, *, +, -, ||$ denote the division, multiplication, addition, subtraction, and absolute value function in $\mathbb{R}$, respectively. F_σ denotes the result of applying a substitution σ to an expression F.

3.2 Proof Plans for Limit Theorems

In proof planning limit theorems some of the methods to be used are the estimation methods `ComplexEstimate` and `Solve` and other methods such as `UnwrapHypothesis`, `Skolemize`, and `IntroduceCS`. `ComplexEstimate` proves the estimation of a complicated term b by representing b as a linear combination $k * a_\sigma + l$ of a term a whose estimation is already known and by reducing the goal $|b| < \epsilon$ to three simpler subgoals that contain $\mathbf{M}$ – a real number whose existence is postulated by `ComplexEstimate`:

1. $|k| \leq \mathbf{M}$,
2. $|a_\sigma| < \epsilon/(2 * \mathbf{M})$,
3. $|l| < \epsilon/2$.

For instance, in planning LIM+, at some point the goal
$|f(x) + g(x) - (l_1 + l_2)| < \epsilon$ is reduced by `ComplexEstimate` to

(1) $|1| \leq \mathbf{M}$,

(2) $|f(X_1) - l_1| < \epsilon/(2*\mathbf{M})$,

(3) $|g(x) - l_2| < \epsilon/2$.

Another estimation method, `Solve`, handles simple inequality goals such as $x < a$. `Solve` does not produce any subgoal but only removes the particular goal.

3.3 Constraint Solving in Proof Planning

The `Solve` method passes an inequality goal g to an external *constraint solver* in case g is consistent with the constraints collected so far during proof planning. The consistency check, the propagation of constraints, and finally the computation of instances that satisfy the collected constraints is carried out by the constraint solver.

The constraint solver is dependent on the theory in which the problem is stated. For theories such as linear arithmetic in the $\mathbb{R}$, set theory, finite domain, term algebra other constraint solvers exists. These are very efficient programs that implicitly use specific data structures and theory-specific algorithms for their computation. Their typical functionalities are consistency and entailment checks, propagation of constraints, and the final computation of a so-called *answer constraint* that describes the properties of postulated objects.

The tasks of a constraint solver in proof planning is to determine the applicability of certain methods by consistency and entailment checks, and to provide an answer constraint. For instance, the incremental restriction of possible values of instantiations in the proof of limit theorems is realized by a constraint solver for linear arithmetic in $\mathbb{R}$ collects constraints in a constraint storeand adds new constraints when they are consistent with the store.

EXAMPLE: In planning a proof of LIM+, the constraint solver receives the following constraints in a row $0 < D$, $|1| \leq \mathbf{M}$, $0 < \mathbf{M}$ $E_1 \leq \epsilon/(2 * \mathbf{M})$, $x = X_1$, $E_2 \leq \epsilon/2$, $x = X_2$ $D \leq \delta_2$, $(0 < \delta_2)$, $D \leq \delta_1$, $(0 < \delta_1)$, $0 < E_1$, $0 < E_2$. From the constraints $1 \leq \mathbf{M}$ and $E_1 \leq \epsilon/2{*}\mathbf{M}$ the new upper bound $\epsilon/2$ of E_1 is propagated.
From the final constraint store

$$\begin{array}{ccccc}
0 & < & E_2 & \leq & \epsilon/2; \\
0 & < & D & \leq & \delta_2, \delta_1; \\
0 & < & E_1 & \leq & \epsilon/(2 * \mathbf{M}), \epsilon/2; \\
1 & \leq & \mathbf{M} & < & \epsilon/(2 * E_1); \\
-\infty & < & X_1 = x = X_2 & < & +\infty
\end{array}$$

the answer constraint $E_1 \leq \epsilon/2 \wedge E_2 \leq \epsilon/2 \wedge D \leq \delta_1 \wedge D \leq \delta_2$ is computed. ∎

4 The Use of Proof Plans

Hierarchical planning and the representation of proof techniques and common structures by planning methods provide a (hierarchically structured) proof plan. Since proof plans are abstract and hierarchical representations of proofs, they are well suited for a comprehensible proof presentation, proof explanation, and theorem proving by analogy.

Verbalization of Proof Plans

The separation into

logic + control

in proof planning, that is into

methods + control-rules,

allows for separately presenting proof steps and the reasons for their choice. Therefore, a verbalization of a proof plan can rely on the verbalization of its methods (steps) and on explanations generated from control-rules.

In addition, proof plans provide a (hierarchical) structure that can be employed for the comprehensible presentation of proofs. For instance, `ComplexEstimate` delivers the idea of the whole LIM+ proof, whereas other methods will be mentioned in a more detailed description of the proof only. Methods that occur in a plan at a lower level of abstraction are usually not verbalized in a first, not too detailed, presentation.

The verbalization of the methods belongs to the domain expertise. As experience shows, some methods in proof plans are heavily verbalized, whereas others, such as quantifier elemination and and-elimination, are hardly verbalized at all in standard mathematical texts.

An example for an always verbalized method in plans of limit theorems is `ComplexEstimate`$_<$. In the textbook [1] the counterpart of `ComplexEstimate` is, with slight linguistic variations, verbalized as follows.

> we need to estimate the magnitude of $|b|$. Since
> $|b| = |k * a + l|$
> we use the Triangle Inequality and obtain
> $|b| \leq |k * a| + |l|$
> This goal can be shown in three steps:
>
> - There exists an M such that $|k| < M$ and
> - $|a| < \epsilon/(2 * M)$ to be shown from the hypothesis $|a| < \epsilon_1$ for $\epsilon_1 = \epsilon/(2 * M)$, and
> - $|l| < \epsilon/2$.
>
> Hence, if Δ, then $|b| \leq |k| * |a| + |l| < M * \epsilon/(2 * M) + \epsilon/2 = \epsilon$
> and therefore $|b| < \epsilon$.

In the instances of this schematic verbalization the terms (maths fond) are instantiated with the actual instantiation of the parameters Δ, a, b, k, l in the particular `ComplexEstimate` step in the proof plan. (For LIM+, a is $f(X_1) - l_1$, b is $f(x) + g(x) - (l_1 + l_2)$, k is 1, l is $g(x) - l_2$.)

Again, the three subproofs are then verbalized according to their plans. For example, in the textbook [1], the proof of LIM+ does not provide a proof of the first subgoal, $1 < \mathbf{M}$ because this belongs to a lower abstract level, whereas in the proof of LIM* the first subgoal, $|g(x)| < M$, is proven by a whole subproof that is captured in a lemma: 'According to Theorem 3.2.2 there exists a real number $M > 0$ such that $M : |g(x)| < M$'.

Currently, in ΩMEGA this verbalization at the method-level is realized in popup windows only. In the near future we shall use hypertext explanation as a tool. For a more elaborate presentation, linguistic knowledge has to be employed to combine the verbalization of single methods to a presentation of the proof plan clear from the full verbalization of LIM+'s proof.

4.1 Presentation Styles

Relying on proof planning, we can make first steps towards two different styles of proof presentation:

1. *textbook style* that starts with postulating the existence of mathematical objects with certain properties and may prove these later in a detailed proof only. The textbook style corresponds to the first way characterized in §2;
2. *construction style* that captures the detailed construction/the incremental restriction of an object right away when it occurs in the proof planning/ proof discovery process. This kind of presentation provides the constructed object only at the end of the planning.

Textbook Style Presentation

The first (textbook style) can be extracted from the complete proof plan which contains the answer constraint formula as a hypothesis, (§3.3). The answer constraint formula is the starting assumption for a textbook style proof without explaining how this was found. For instance, the textbook style presentation of LIM+'s plan starts with postulating the answer constraint:

Let δ be smaller than δ_1 and δ_2, and ϵ_1, ϵ_2 smaller than $\epsilon/2$. (*)

EXAMPLE: A full verbalization of the LIM+ plan that uses a schematic verbalization of `ComplexEstimate`, is the following (The occurrence of M is due to the more general presentation needed for other limit theorems. It is actually not necessary for the LIM+ verbalization.):
To show that $f(x)+g(x)$ converges to $L1+L2$

1. Let δ be smaller than δ_1 and δ_2, and ϵ_1, ϵ_2 smaller than $\epsilon/2$. (*)
2. we need to estimate the magnitude of $|f(x)+g(x)-(L1+L2)| = |(f(x)-L1)+(g(x)-L2)|$.
3. To do this, we use the Triangle Inequality and obtain $|f(x)+g(x)-(L1+L2)| \leq |f(x)-L1|+|g(x)-L2|$.
   This goal can be shown in three steps: (1) There exists an M such that $|1| < M$ and (2) $|f(x)-L1| < \epsilon/(2*M)$, and (3) $|g(x)-L2| < \epsilon/2$.
   - For a real number $M \geq 1$
   - by hypothesis, if $\epsilon/(2*M) > 0$, there exists δ_1 such that for all x, if $|x-a| < \delta_1$ then $|f(x)-L1| < \epsilon/(2*M)$
   - and (by hypothesis too) if $\epsilon/2 > 0$, there exists δ_2 such that forall x, if $|x-a| < \delta_2$, then $|g(x)-L2| < \epsilon/2$.
4. Because of 1, it follows that
5. if $|x-a| < D$, then $|f(x)+g(x)-(L1+L2)| \leq M*|f(x)-L1|+|g(x)-L2| < M*\epsilon/(2*M)+\epsilon/2 = \epsilon$ and therefore $|f(x)+g(x)-(L1+L2)| < \epsilon$.
6. this proves the theorem.

Compared with the proof plan of LIM+, the items 2, 3, and 5 belong to the verbalization of `ComplexEstimate`. Step 3 and 3 verbalize (very short) subproofs of `ComplexEstimate`'s subgoals. 1 provides the answer constraint formula by a verbalization of the method `IntroduceCS`. ■

When verbalized at a lower (more detailed) level, the `Solve` method that passed a constraint goal to the constraint solver must refer to the answer constraint formula as a justification.

Construction Style Presentation

In such a textbook style proof presentation the information on how instances have been constructed is not present any more, whereas the second (construction style) presentation follows the incremental restriction of instantiations, i.e., the construction of mathematical objects.

That is, in a construction style presentations, the answer constraint formula (that is known at the end of the automated proof only) is ignored or mentioned at the end of the proof only. The verbalization of some methods also differs from their verbalization in the textbook style presentation. For example, the method `IntroduceCS` that introduces (a place holder for) the answer constraint, is not verbalized at all in construction style presentations and the `Solve` method must state that we have to further restrict the instantiation of a postulated object. Moreover, the verbalization refers to lower-level parts of the plan too, for instance, the `Solve` method applications.

A construction style presentation of LIM+ may look as follows

EXAMPLE: To show that $f(x)+g(x)$ converges to $L1+L2$

1. we need to estimate the magnitude of $|f(x)+g(x)-(L1+L2)| = |(f(x)-L1)+(g(x)-L2)|$.

2. To do this, we use the Triangle Inequality and obtain
$|f(x) + g(x) - (L1 + L2)| \leq |f(x) - L1| + |g(x) - L2|$.

3. For proving $|f(x) - L1| + |g(x) - L2| < \epsilon$, it suffices to show that
for some $M \geq 1$,
$|F(x) - L1| < \epsilon/(2 * M)$ and
$|g(x) - L2| < \epsilon/2$.

4. We restrict $M \geq 1$.

5. Since by hypothesis, for any ϵ_1 there exists a δ_1 such that if $\epsilon_1 > 0$, there exists δ_1 such that for all x, if $|x - a| < \delta_1$, then $|f(x) - L1| < \epsilon_1$,

   (a) we must restrict $\epsilon_1 \leq \epsilon/(2 * M)$ and $\delta \leq \delta_1$ in order to prove $|f(x) - L1| < \epsilon/(2 * M)$.

6. Since by hypothesis, for any ϵ_2 there exists a δ_2 such that if $\epsilon_2 > 0$, there exists δ_2 such that for all x, if $|x - a| < \delta_2$, then $|g(x) - L2| < \epsilon_2$,

   (a) we must restrict $\epsilon_2 \leq \epsilon/2$ and $\delta \leq \delta_2$. in order to prove $|g(x) - L2| < \epsilon/2$.

7. From the collected restrictions follows
if $|x - a| < D$ then
$|f(x) + g(x) - (L1 + L2)| \leq M * |f(x) - L1| + |g(x) - L2| < M * \epsilon/(2 * M) + \epsilon/2 = \epsilon$ and therefore $|f(x) + g(x) - (L1 + L2)| < \epsilon$.

8. This proves the theorem.

■

The example shows how the subproofs by the `Solve` method in 5a and 6a are mentioned as opposed to the textbook style presentation that, instead, needed a verbalization of the answer constraint formula.

5 Conclusion

We have addressed two problems of comprehensible proof presentation, the hierarchically structured presentation at the level of proof methods and different presentation styles of construction proofs. The proof planning techniques and proof presentations offer solutions for these problems that can be supported by an automated tool. This is because the (automatically) generated proof plans have a hierarchical structure and their methods can be schematically verbalized and because two kinds of presentations can be developed from the same proof plan of construction proofs.

An additional feature of our proof plans, the stored justifications for the particular application of a method, seems to support learning: Catrambone suggests from his psychological experiments [5] that if the conditions for applying a method are highlighted in examples, learners are more likely to appropriately adapt that method to novel problems, where the conditions do not fully match those required for the old methods.

6 Acknowledgement

I thank Uri Leron for helpful discussions.

References

[1] R.G. Bartle and D.R. Sherbert. *Introduction to Real Analysis*. John Wiley& Sons, New York, 1982.

[2] C. Benzmueller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, A. Meier, E. Melis, W. Schaarschmidt, J. Siekmann, and V. Sorge. OMEGA: Towards a mathematical assistant. In W. McCune, editor, *Proceedings 14th International Conference on Automated Deduction (CADE-14)*, pages 252–255, Townsville, 1997. Springer.

[3] R. Catrambone. Improving examples to improve transfer to novel problems. *Memory & Cognition*, 22(5):606–615, 1994.

[4] R. Catrambone. Generalizing solution procedures learned from examples. *Journal of Experimental Psychology, Learning, Memory, and Cognition*, 22(4):1020–1031, 1996.

[5] R. Catrambone and K.J. Holyoak. Learning subgoals and methods for solving probability problems. *Memory and Cognition*, 18(6):593–603, 1990.

[6] B.I. Dahn and A. Wolf. Natural language presentation and combination of automatically generated proofs. In *Proceedings of FroCos'96*, pages 175–192. Kluwer, 1996.

[7] X. Huang and A. Fiedler. Proof verbalization as an application of nlg. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence*, pages 965–970. Morgan Kaufmann, 1997.

[8] U. Leron. Structuring mathematical proofs. *The American Mathematical Monthly*, 90:174–185, 1983.

[9] U. Leron. Heuristic presentations: the role of structuring. *For the Learning of Mathematics*, 5(3):7–13, 1985.

[10] Heinz Lüneburg. *Vorlesungen über Analysis*. BI Wissenschaftsverlag, 1981.

[11] W.W. McCune. Otter 2.0 users guide. Technical Report ANL-90/9, Argonne National Laboratory, Maths and CS Division, Argonne, Illinois, 1990.