A Proof Presentation Suitable for Teaching Proofs

Erica Melis*
Universität des Saarlandes,
Fachbereich Informatik
66041 Saarbrücken
melis@cs.uni-sb.de

Uri Leron
Dept. of Science Education
Technion Inst. of Technology
Haifa, Israel 32000
uril@tx.technion.ac.il

Abstract. The paper addresses comprehensible proof presentation for teaching and learning that can be provided by an automated proof planner that is a component of the proof development environment ΩMEGA. Starting from empirically discovered requirements for the comprehensibility of proofs we show that, given a proof plan representation of a proof, the problem of automatically presenting a mathematical proof in a comprehensible manner becomes feasible in a way that was impossible before. We show how, based on a proof plan, a structured presentation at the level of proof methods can be automatically constructed

1 Introduction

Computer algebra systems have been successfully used in teaching mathematics (see, e.g. a summary of Dana Scott's courses [12]). Can automated theorem proving systems be useful for teaching mathematical proofs? and under which conditions?

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, for the Theorem: LIM+:
The limit of the sum of two real-valued functions is the sum of their limits.
a so-called ϵ-δ-proof would construct a real number δ dependent on ϵ such that

$$0 < \epsilon \rightarrow (0 < \delta \wedge \forall x(|x - a| < \delta \rightarrow |(f(x) + g(x)) - (L_1 + L_2)| < \epsilon))$$

holds, provided $\lim_{x \to a} f(x) = L_1$ and $\lim_{x \to a} g(x) = L_2$,

whereas its proof as produced by the automated theorem prover OTTER [10] is the following

```
...
20 [binary,12.2,10.3] -LE(0,x)| -LE(abs(F(XS(x))+ -F(A)),HALF(E0))| -LE(abs(G(XS(x))+ -G(A)),HALF(E0)).
126 [binary,20.2,3.3] -LE(0,x)| -LE(abs(G(XS(x))+ -G(A)),HALF(E0))| -LE(0,HALF(E0))| -LE(abs(XS(x)+ -A),D1(HALF(E0))).
699 [binary,126.3,11.2,unit_del,5] -LE(0,x)| -LE(abs(G(XS(x))+ -G(A)),HALF(E0))| -LE(abs(XS(x)+ -A),D1(HALF(E0))).
991 [binary,699.3,8.2] -LE(0,x)| -LE(abs(G(XS(x))+ -G(A)),HALF(E0))| -LE(abs(XS(x)+ -A),MIN(y,D1(HALF(E0)))).
1194 [binary,991.3,6.2,factor_simp] -LE(0,MIN(x,D1(HALF(E0))))| -LE(abs(G(XS(MIN(x,D1(HALF(E0)))))+ -G(A)),HALF(E0)).
1196 [binary,1194.1,9.3] -LE(abs(G(XS(MIN(x,D1(HALF(E0)))))+ -G(A)),HALF(E0))| -LE(0,x)| -LE(0,D1(HALF(E0))).
1231 [binary,1196.3,1.2] -LE(abs(G(XS(MIN(x,D1(HALF(E0)))))+ -G(A)),HALF(E0))| -LE(0,x)| -LE(0,HALF(E0)).
1276 [binary,1231.3,11.2,unit_del,5] -LE(abs(G(XS(MIN(x,D1(HALF(E0)))))+ -G(A)),HALF(E0))| -LE(0,x).
1284 [binary,1276.2,2.2] -LE(abs(G(XS(MIN(D2(x),D1(HALF(E0)))))+ -G(A)),HALF(E0))| -LE(0,x).
1297 [binary,1284.1,4.3] -LE(0,x)| -LE(0,HALF(E0))| -LE(abs(XS(MIN(D2(x),D1(HALF(E0))))+ -A),D2(HALF(E0))).
1303 [factor,1297,1,2] -LE(0,HALF(E0))| -LE(abs(XS(MIN(D2(HALF(E0)),D1(HALF(E0))))+ -A),D2(HALF(E0))).
1413 [binary,1303.1,11.2,unit_del,5] -LE(abs(XS(MIN(D2(HALF(E0)),D1(HALF(E0))))+ -A),D2(HALF(E0))).
```

*This author was supported by the Deutsche Forschungsgemeinschaft, SFB 378.

[1] such as resolution

```
1417 [binary,1413.1,7.2] -LE(abs(XS(MIN(D2(HALF(E0)),D1(HALF(E0))))+ -A),MIN(D2(HALF(E0)),x)).
1418 [binary,1417.1,6.2] -LE(0,MIN(D2(HALF(E0)),D1(HALF(E0)))).
1419 [binary,1418.1,9.3] -LE(0,D2(HALF(E0)))| -LE(0,D1(HALF(E0))).
1422 [binary,1419.1,2.2] -LE(0,D1(HALF(E0)))| -LE(0,HALF(E0)).
1428 [binary,1422.1,1.2,factor_simp] -LE(0,HALF(E0)).
1430 [binary,1428.1,11.2] -LE(0,E0).
1431 [binary,1430.1,5.1] False.
```

The need for better outputs was recognized long ago and several attempts have been made to produce readable proofs. Proof presentation in natural language has recently been realized in ILF [6] and PROVERB [7] which slightly abstract proofs before the presentation. PROVERB returns a proof presentation at the so-called assertion-level.[2] However a proof verbalization at this level is not necessarily the most natural and best way to communicate a proof to mathematicians or to students. In particular, often the proof is not abstract enough and the user cannot go from an abstract level to a more detailed level because a (hierarchical) structure is missing.

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? This paper will give a partial answer to this question.

In this paper, we shall show how 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. Although the idea to use proof plans as a basis for a structured proof presentation might seem obvious, it is new and our systems is the first to implement this idea.

2 Empirical Results on Proof Presentation

Empirical class room studies have provided useful insights about the presentation of proofs for students. In particular, (hierarchically) structured presentations help students to understand and memorize proofs.

Catrambone [4] 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 [5] he showed that even meaningless labels that structure a proof support the problem solving capabilities of people.

Even more impressive is a comparison of a hierarchical presentation of a proof with the standard linear presentation. 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. On the other hand, structured proofs consist of 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

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

of proofs, 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]. Leron demonstrates among others one particular principle for structuring proofs that involve the construction of mathematical objects: At the abstract level the existence of a mathematical object that satisfies certain properties P is postulated and with this assumption the proof (idea) is completed. Only at a hierarchically lower level an object that actually satisfies P is constructed.

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. Since our system ΩMEGA[2] is designed to assist students in proving mathematical theorems, to understand the proofs produced (semi-)automatically, and to improve their capabilities, its proof planner should produce a output comparable to an expert teacher's proof presentation.

3 Proof Planning

Proof planning is a relatively new paradigm in automated theorem proving [3, 11]. As opposed to traditional automated theorem proving which searches for a proof at the level of a logic calculus, proof planning automatically constructs a plan by searching at a *higher level of abstraction* and by using explicit control knowledge. A resulting plan 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 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 may 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.

The proof planner of ΩMEGA automatically produces the plan for LIM+ and the other theorems mentioned below. The graphical user interface of ΩMEGA , LOUI, makes accessible the hierarchical structure of the proof plan data structure (PDS) by first showing the most abstract proof plan in a tree-like graph (top left window in Figure 1). Then the nodes of this abstract proof plan can be expanded and this yields a (hierarchically lower) subplan.

ϵ-δ-Proofs

We shall illustrate the use of proof plans with an example from the class of limit theorems that can be automatically planned by ΩMEGA. The experiments for planning and presenting proofs were mainly conducted with examples from this class. 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.

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

which, after expanding the definition of $\lim_{x \to a}$, becomes
$\forall \epsilon_1 (0 < \epsilon_1 \rightarrow \exists \delta_1 (0 < \delta_1 \wedge \forall x_1 (x_1 \neq a \wedge |x_1 - a| < \delta_1 \rightarrow |f(x_1) - L_1| < \epsilon_1)))$
$\wedge \forall \epsilon_2 (0 < \epsilon_2 \rightarrow \exists \delta_2 (0 < \delta_2 \wedge \forall x_2 (x_2 \neq a \wedge |x_2 - a| < \delta_2 \rightarrow |g(x_2) - L_2| < \epsilon_2)))$
$\rightarrow \forall \epsilon (0 < \epsilon \rightarrow \exists \delta (0 < \delta \wedge \forall x (x \neq a \wedge |x - a| < \delta \rightarrow |(f(x) + g(x)) - (L_1 + L_2)| < \epsilon)))$

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: $\lim_{x \to a} x^2 = a^2$, and other theorems about continuity. The typical way a mathematician proves such a theorem is to (incrementally) construct a real number δ that depends on ϵ.

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

3.1 Proof Plans for ϵ-δ-Proofs

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.

4 The Use of Proof Plans for Proof Presentation

Hierarchically structured proof plans and the representation of common proof procedures and common structures by planning methods provide information needed for a comprehensible proof presentation and for proof explanation.

Verbalization of Methods

The verbalization of the methods belongs to the domain expertise and therefore has to be designed by a maths knowledge engineer. As experience shows, some methods in proof plans are heavily verbalized, whereas others, such as `QuantifierElemination` and `AndElimination`, are hardly verbalized at all in standard mathematical texts. An example for an always verbalized method in plans of limit theorems is `ComplexEstimate`. `ComplexEstimate` delivers the main idea of the LIM+ proof, whereas other methods will be mentioned in a more detailed description of the proof only.

In the textbook [1] the counterpart of `ComplexEstimate` is, with slight linguistic variations, verbalized as follows.

> In order to estimate the magnitude of $|b|$.
> we rewrite the term to $|k * a\sigma + l|$
> and we use the Triangle Inequality
> $|b| \leq |k * a\sigma| + |l|$
> The goal can be shown in three steps:
>
> - There exists an M such that $|k| \leq M$,
> - $|a| < \epsilon/(2 * M)$, and
> - $|l| < \epsilon/2$.
>
> it follows that $|b| \leq |k| * |a\sigma| + |l| < M * \epsilon/(2 * M) + \epsilon/2 = \epsilon$
> and therefore $|b| < \epsilon$.

In the instantiations of this schematic verbalization, the terms (maths fond) are instantiated with the actual instantiation of the parameters 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 $g(x) - l_2$.)

This schematic verbalization is stored in a slot of the method in order to employ it for the presentation of the proof. Only because the methods represent a relative abstract content, a schematic verbalization is quite appropriate.

Different modi of a method's verbalization are possible and desired, e.g., for different styles of the presentation of a construction proof. All these verbalizations are prefabricated though.

4.1 Automatic Generation of Proof Verbalization

The plan presentation routine *LocalPresent* is integrated into the user interface LOUI [13] and uses the hypertext facility based on the LOUI markup language LML. For the presentation, a presentation window is created, shown at the bottom right window of Figure 1.

The actual input of *LocalPresent* is the proof plan that is graphically presented by LOUI. When the user clicks on a node of the graphical presentation of the proof plan, then *LocalPresent* searches backwardly for the next node whose method can be verbalized (remember that not every methods actually is verbalized). If such a method is found, then its *instantiated* verbalization is shown in the presentation window. If no such method exists, then the verbalization `obviously` is produced.

The instantiated verbalization of a method M may contain a subgoal of M. In this case, a dynamically produced hypertext link points to the proof of this subgoal. If the

user clicks on a marked subgoal g, *LocalPresent* goes to the node in the proof plan that corresponds to g and again searches backwardly for the next method that can be verbalized in order to present the proof of g.

In case the user wants to see more details of a method M that is currently verbalized, then (s)he can click on an expansion link that causes *LocalPresent* to dynamically expand M to a subplan and to show a verbalization of this subplan. Typically, this expansion is the instantiation of a schema that is represented in the *proof schema* slot of a method.

Since only the verbalization of one method is shown in the presentation window, this presentation is a *local* one as opposed to a global verbalization that would present a sequence of methods. This local presentation has the advantage that the user can more easily see the correspondences between the (graphical) plan tree and the verbalization. If (s)he wants to see more of the verbal presentation of the proof (s)he has to click on another node or a subgoal in a verbal presentation.

4.2 Example: LIM+

An understandable presentation of LIM+'s plan can be automatically extracted from the complete proof plan and obtained by first clicking on the node that represents the theorem in the graph. The presentation starts with postulating the answer constraint formula which results from collecting all constraints of δ, ϵ_1, and ϵ_2. This answer constraint formula $\delta \leq \delta_1 \wedge \delta \leq \delta_2 \wedge \ldots$ is the presentation of the `IntroduceCS` method. The next method verbalized is `ComplexEstimate`. Its verbalization contains subgoals that can be clicked on further for obtaining the other parts of the complete verbalization.

Figure 1 shows how *LocalPresent* verbalizes this method in the presentation window. The white node in the graph is the one that is currently verbalized. Three subproofs

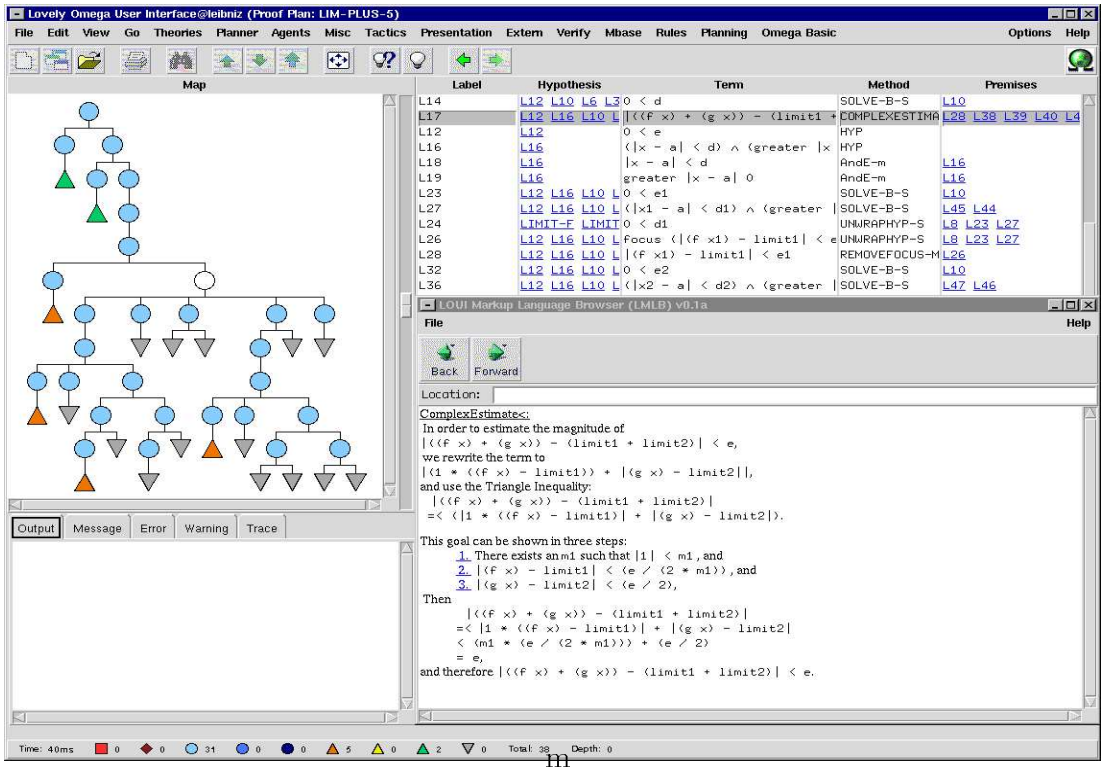

Figure 1: Screen shot of ΩMEGA showing a proof plan of LIM+ and a local verbalization

can be obtained by clicking on the subgoal pointers (1., 2., and 3.) of the verbalization.

For example, the proof of the first subgoal, $1 < M$, of `ComplexEstimate` in the proof of LIM+ is `obvious`, whereas in the proof of LIM* the first subgoal, $|g(x)| \leq M$, 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 $M : |g(x)| < M$'.

A full verbalization of the LIM+ plan is a sequence of method verbalizations and looks as follows.[3]

To show that $f(x) + g(x)$ converges to $L1 + L2$
Let δ be smaller than δ_1 and δ_2, and ϵ_1, ϵ_2 smaller than $\epsilon/2$. (*)
In order to estimate the magnitude of $|f(x) + g(x) - (L1 + L2)|$
we rewrite the term to $|(f(x) - L1) + (g(x) - L2)|$.
and we use the Triangle Inequality
$|f(x) + g(x) - (L1 + L2)| \leq |f(x) - L1| + |g(x) - L2|$.
The goal can be shown in three steps:
1. There exists an M such that $|1| \leq M$,
2. $|f(x) - L1| < \epsilon/(2 * M)$, and
3. $|g(x) - L2| < \epsilon/2$.

(1) obviously, for a real number $1 \leq M$

(2) by hypothesis, if $\epsilon/(2 * M) > 0$, there exists δ_1 such that for all x, if $|x - a| < \delta_1$ then $|f(x) - L1| < \epsilon/(2 * M)$

(3) and by hypothesis , if $\epsilon/2 > 0$, there exists δ_2 such that forall x, if $|x - a| < \delta_2$, then $|g(x) - L2| < \epsilon/2$.

Because of (*) it follows that
$|f(x) + g(x) - (L1 + L2)| \leq M * |f(x) - L1| + |g(x) - L2| < M * \epsilon/(2 * M) + \epsilon/2 = \epsilon$
and therefore $|f(x) + g(x) - (L1 + L2)| < \epsilon$.
This proves the theorem.

5 Benefits for Teaching

We have not yet conducted class room experiments that show the superiority of the plan presentation of ΩMEGA. However, if you compare the above verbalization of the LIM+ proof with the output of the automated theorem prover OTTER as given in the introduction, then the proof plan based presentation clearly is more human-oriented. In particular, the hierarchical structure of a proof can be easily taught to or dicovered by a student.

Since *LocalPresent* allows the user to decide when to look into details, i.e., when to expand a presentation, the presentation becomes rather individualized.

ΩMEGA can not only be used to verbalize the proof of LIM+ but also other ϵ-δ-proofs that we have checked in experiments. More experiments, e.g., with diagonalization proofs will follow shortly. More features will then be integrated into *LocalPresent*, e.g., different presentation styles and dependency on the user model. A *GlobalPresent* will use linguistic knowledge to chain the method verbalizations together and will use structuring features to structure a sequential presentation that is presented independent from the graphical proof tree.

[3]The occurrence of the auxiliary variable M is due to the more general presentation needed for other limit theorems.

6 Conclusion

We have addressed requirements of comprehensible proof presentations, in particular the hierarchically structured presentation at the level of proof *methods*. Proof planning techniques and the proof plan representation is the basis for automatically generating proof presentations in the ΩMEGA system satisfying these requirements.

Sometimes, a change of the underlying knowledge representation totally changes the problem solving horizon. This situation applies to changing from calculus-level proofs of traditional automated theorem proving to proof plans. Now suddenly, problems become simple which were hardly solvable in traditional automated theorem proving since proof plans are hierarchically structured and can be employed for the comprehensible presentation of proofs.

Related the work has been done on XBarnacle [9] which is a proof planner used for proofs by mathematical induction. XBarnacle uses a *flat* proof plan to communicate with the user. It refers to a method by its name only rather than by a natural language description, so the user needs to know what this method is about.

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, *Proc. 14th International Conference on Automated Deduction*, pages 252–255, 1997. Springer.

[3] A. Bundy. The use of explicit plans to guide inductive proofs. In E. Lusk and R. Overbeek, editors, *Proc. 9th International Conference on Automated Deduction*, pages 111–120, 1988.

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

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

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

[7] X. Huang and A. Fiedler. Proof verbalization as an application of nlg. In *Proc. of the 15th 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] H. Lowe and D. Duncan. Xbarnacle: Making theorem provers more acessible. In W. McCune, editor, *Proc. of the Fourteenth Conference on Automated Deduction*, pages 404–408, 1997.

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

[11] E. Melis. AI-techniques in proof planning. In *European Conference on Artificial Intelligence (ECAI-98)*, pages 494–498, Brighton, 1998. Kluwer.

[12] D. Scott. Symbolic computation and teaching. In J. Calmet, J.A. Campbell, and J. Pfalzgraf, editors, *Artificial Intelligence and Symbolic Mathematical Computation, AISMC-3*, Springer, 1996.

[13] J.H. Siekmann, S.M. Hess, C. Benzmüller, L.Cheikhrouhou, D. Fehrer, A. Fiedler, M. Kohlhase, K. Konrad, E. Melis, A. Meier, and V. Sorge. LOUI: A distributed graphical user interface for the interactive proof system ΩMEGA. In *Proc. of the International Workshop on User Interfaces for Theorem Provers*, 1998.