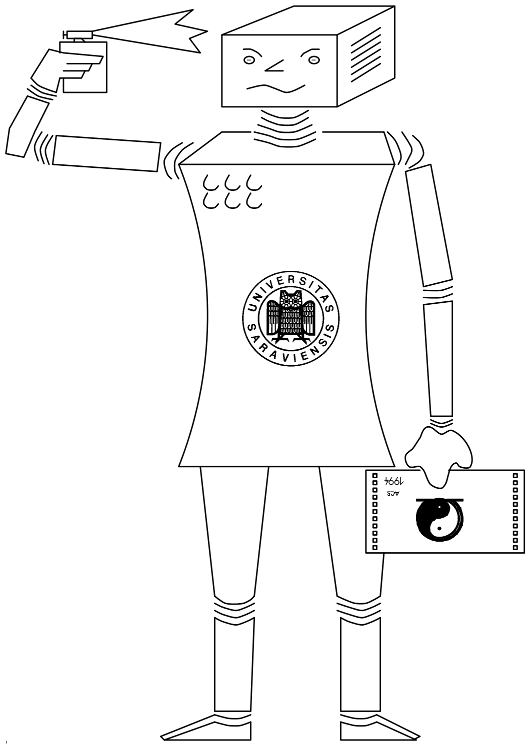
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY
WWW: http://js-sfbsun.cs.uni-sb.de/pub/www/



Decomposition Techniques and their Applications in Automated Theorem Proving

Erica Melis

Published as: Proceedings of AIMSA-94, Sofia, Bulgaria, pages 15-25

DECOMPOSITION TECHNIQUES AND THEIR APPLICATIONS IN AUTOMATED THEOREM PROVING*

ERICA MELIS†

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

email: melis@cs.cmu.edu

ABSTRACT

This paper addresses the decomposition of proofs as a means of constructing methods in plan-based automated theorem proving. It shows also, how decomposition can beneficially be applied in theorem proving by analogy. Decomposition is also useful for human-style proof presentation. We propose several decomposition techniques that were found to be useful in automated theorem proving and give examples of their application.

1 Introduction

The way human experts solve problems often differs from the way computers solve the same problem. For instance, among other differences, human experts can take larger steps [8], while automated systems perform well-defined basic steps towards a solution. This is particularly the case in automated theorem proving systems. These systems usually apply fixed proof calculus rules, e.g., resolution, as basic steps. However, some automated theorem provers have in-built procedures that are specialized to deal with particular subproblems [3],[15], and some employ well-defined larger chunks of steps to search more efficiently for a proof.

These chunks have to be defined by the user or system designer and built into the system, or they are created by the system. Since the design and implementation of complex user-defined chunks is laborious, researchers considered alternatives: Some systems are enriched with a few complex methods which are widely applicable,

---

*AIMSA-94

†On leave from University of Saarbrücken, Germany. This work was supported by the Max Kade Foundation

such as rippling. For instance, Nuprl uses tactics, OYSTER/CLAM [16] employs methods, and INKA [1] works with partial plans. Some interactive approaches tackle the building of complex methods by **method synthesis**, i.e., by combining less complex methods via "high level components". But still, these alternatives do not in general yield all methods used by mathematicians nor all those methods that are generated and needed in theorem proving by analogy. We suggest an **analysis** by decomposing methods as an additional means of constructing methods from other methods.

As the decomposed proof in figure 1 shows, structuring is a very natural activity in finding and understanding proofs: level1 of the proof represents the proof idea, whereas the level2 subproofs supply proofs for statements in level1, specific constructions of objects etc. If one of these subproofs is itself complicated, then the details may be pushed further down to lower levels.

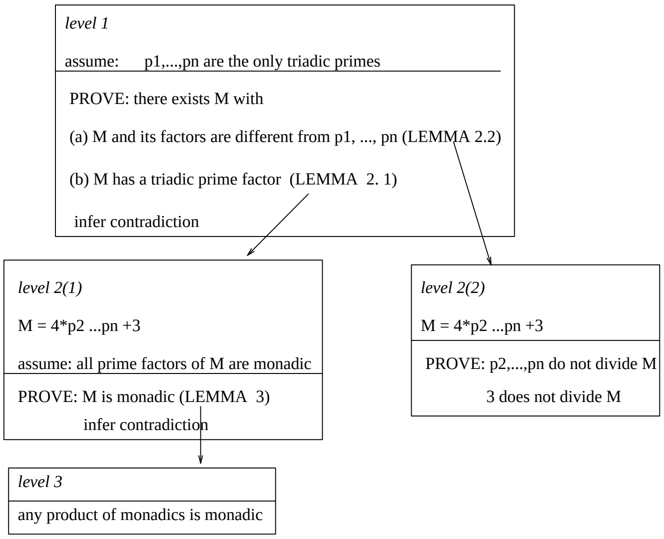


Figure 1: structured proof of theorem1: There exist infinitely many *triadic* primes

This paper addresses heuristically justified break downs of given (partial) proofs into appropriate parts. We introduce techniques for decomposing proofs, and discuss applications in theorem proving by analogy, in debugging of failed analogies, and in proof presentation[a].

[a] Denotational notes: We rely on Natural Deduction proofs applying $\wedge$D-rule etc. For our purpose, we use the notion "method" and "(partial) subproof" interchangeable. A problem is defined as a pair (ass, thm) of a set of assumptions and a theorem. A problem is proved by proving $ass \vdash thm$.

2 Decomposition Techniques

We shall now present several general purpose techniques which are widely applicable for decomposing mathematical proofs. These techniques work not just for proofs but also for partially specified (henceforth partial) proofs.

- The decomposition of proofs by user-supplied *motivations* is introduced in [4]. The range of a motivation is a subproof. Brock et al. give the following example:
  ```
  ...(motive (Eliminate X in 2)
        (chain 1 and 2)
        (eliminate variable X in 3))...
  ```
  This motivation specifies that the purpose of the chaining step is to do variable elimination, which means that the steps included in the range of the motivation form a method that can be treated separately.
- A well known technique in theorem proving is decomposing a proof into subproofs when the theorem is a conjunction or an equivalence (see [2]). It provides a subproof for each conjunct and one consisting of a $\wedge$-introduction step.
- The straightforward decomposition of inductive proofs is well known.
- A decomposition of a case analysis yields $n + 1$ methods, which can be identified in a Natural Deduction proof, namely one for each of the n cases and one for the rest of the proof.
- The decomposition techniques which identify key steps in a proof presume that the proof parts leading from one key step to another constitute subproofs. The following two heuristics for finding key steps were proposed by Stephen Owen in [14]:
  - The application of certain assumptions, such as definitions and axioms, are key steps of a proof.
  - A proof step, where temporarily introduced symbols are removed, often represents the completion of a part of a proof. Hence, the step immediately following the removal is usually a key step.
- The applications of a definition or an axioms forms a subproof M_1 itself, such that a proof M can be decomposed into M_1 and the rest of M. This is particularly useful, if the definition or axiom is applied at the beginning or the end of a proof, since you may just cut the first or the last part of the proof.
- A simple decomposition heuristic is to look for a statement that is used several times in a proof. The derivation of such a statement, which is a well defined part of the proof tree, can likely be considered as a subproof.

- Deduction-Theorem Splitting

  If a hypothesis[b] is introduced into a proof P and eliminated later, then that part of the proof tree which is situated in between (and includes) this introduction and elemination, forms a subproof M_1. M_1 itself can be split into two subproofs M_{11} and M_{12}, where M_{12} consists of the application of the Deduction theorem ($\rightarrow$introduction) and M_{11} contains the rest. In case $\forall$-introductions follow the application of the Deduction theorem, they belong to M_{12} as well. In the example below the decomposition yields just two subproofs, as $P = M_1$.

  Example-1

  Proof of $\emptyset \vdash \forall x, y((Qxy \rightarrow \forall z(Pxz \rightarrow Pyz)) \wedge Qab \wedge (Paa \vee Pbb) \rightarrow \exists x Pbx)$

NNo	S;D		Formula	Reason
			M_{11}	
1.	; 1	$\vdash$	$\forall x, y(Qxy \rightarrow \forall z(Pxz \rightarrow Pyz)) \wedge Qab \wedge (Paa \vee Pbb)$	(HYP)
⋮	⋮		⋮	⋮
11.	;1;	$\vdash$	$\exists z Pbz$	($\vee D$;8,3,1)
			M_{12}	
12.	;	$\vdash$	$\forall x, y((Qxy \rightarrow \forall z(Pxz \rightarrow Pyz)) \wedge Qab \wedge (Paa \vee Pbb) \rightarrow \exists z Pbz)$	($\rightarrow I$;9)

- Several Splittings for Relevant Assumptions

  Our analysis [11] of the proofs by analogy in a mathematical textbook [5] provided evidence for the fact that only *relevant* proof assumptions are used for analogous proofs. For instance, often just common parts of definitions of the source and the target problem create an analogy. These commonalities are sometimes hidden in the assumptions. For example, the definitions of $left\text{-}congruence(R)$ and $equivalence\text{-}relation(R)$ have, among others, the common component $symmetric(R)$ which is employed in the example below. Relevant-Assumption splitting decomposes a proof into one subproof that extracts it relevant assumptions from the proof assumptions and another subproof that uses the extracted assumptions. As a result, assumptions of different proofs may become comparable.

  The strongest Relevant-Assumptions-decomposition splits a proof M into two subproofs M_1 and M_2, such that M_1 contains all lines with proof assumptions (lines 1 - 4 in example-2) and those lines of M which result from rules that are applied just to lines of M_1 but not to others[c]. M_2 is the rest of the proof. In most cases such a decomposition is far too strong. The weakest Relevant-Assumptions decomposition splits M into M_1 and M_2 such that all assumption lines are in M_1, and all lines of M

[b] which corresponds to "let..." in mathematical proofs
[c] this seems to be similar to the attempts in [18]

resulting by applying $\wedge$-deletion to lines of M_1 are in M_1 and the rest is in M_2. In general, several Relevant-Assumptions splittings exist between the strongest and the weakest decomposition.
A rather practical and well-suited Relevant-Assumptions technique yields an M_1 that contains all lines with proof assumptions and lines of M which result from applying definitions and $\wedge$-deletion to lines of M_1. Again, this technique can yield several options of splittings, as shown in the examples below. The choice of alternatives can be controlled, e.g., in the analogy procedure, by considering the target problem.

Examples-2
Theorem 4[d]: Let ρ and σ be two equivalence relations, then $(\rho \cap \sigma)$ is symmetric.

Option (1) for Relevant-Assumptions splitting of the proof of theorem 4

NNo	S;D		Formula	Reason
			M_1	
1.	$;1$	$\vdash$	$\forall R(equivrel(R) \leftrightarrow refl(R) \wedge symm(R) \wedge trans(R))$	(DEF)
2.	$;2$	$\vdash$	$\forall R(symm(R) \leftrightarrow \forall x_1, x_2((x_1, x_2) \in R \rightarrow (x_2, x_1) \in R))$	(DEF)
3.	$;3$	$\vdash$	$\forall R_1, R_2, x((x \in (R_1 \cap R_2) \leftrightarrow (x \in R_1 \wedge x \in R_2))$	(DEF)
4.	$;4$	$\vdash$	$equivrel(\rho)$	(ASS)
5.	$;5$	$\vdash$	$equivrel(\sigma)$	(ASS)
6.	$;4$	$\vdash$	$refl(\rho) \wedge symm(\rho) \wedge trans(\rho)$	(appl-def 4)
7.	$;4$	$\vdash$	$refl(\sigma) \wedge symm(\sigma) \wedge trans(\sigma)$	(appl-def 5)
8.	$;4$	$\vdash$	$symm(\rho)$	($\wedge$D 6)
9.	$;5$	$\vdash$	$symm(\sigma)$	($\wedge$D 7)
			M_2	
10.	$;2,8$	$\vdash$	$\forall x_1, x_2((x_1, x_2) \in \rho \rightarrow (x_2, x_1) \in \rho)$	(appl-def 8)
11.	$;2,9$	$\vdash$	$\forall x_1, x_2((x_1, x_2) \in \sigma \rightarrow (x_2, x_1) \in \sigma)$	(appl-def 9)
12.	12;	$\vdash$	$(f_1, f_2) \in (\sigma \cap \rho)$	(HYP)
$\vdots$	$\vdots$		$\vdots$	$\vdots$
24.	;3,2,9,8	$\vdash$	$symm(\sigma \cap \rho)$	($\forall$D,$\leftrightarrow$D,$\rightarrow$D)

Option (2) for Relevant-Assumptions splitting of the proof of theorem 4

NNo	S;D		Formula	Reason
			M_1	
1.	$;1$	$\vdash$	$\forall R(equivrel(R) \leftrightarrow refl(R) \wedge symm(R) \wedge trans(R))$	(DEF)
2.	$;2$	$\vdash$	$\forall R(symm(R) \leftrightarrow \forall x_1, x_2((x_1, x_2) \in R \rightarrow (x_2, x_1) \in R))$	(DEF)
3.	$;3$	$\vdash$	$\forall R_1, R_2, x((x \in (R_1 \cap R_2) \leftrightarrow (x \in R_1 \wedge x \in R_2))$	(DEF)
4.	$;4$	$\vdash$	$equivrel(\rho)$	(ASS)
5.	$;5$	$\vdash$	$equivrel(\sigma)$	(ASS)
6.	$;4$	$\vdash$	$refl(\rho) \wedge symm(\rho) \wedge trans(\rho)$	(appl-def 4)
7.	$;4$	$\vdash$	$refl(\sigma) \wedge symm(\sigma) \wedge trans(\sigma)$	(appl-def 5)

[d]Example from [5].

8.	; 4	$\vdash$	$symm(\rho)$	($\wedge$D 6)
9.	; 5	$\vdash$	$symm(\sigma)$	($\wedge$D 7)
10.	; 2, 8	$\vdash$	$\forall x_1, x_2((x_1, x_2) \in \rho \rightarrow (x_2, x_1) \in \rho)$	(appl-def 8)
11.	; 2, 9	$\vdash$	$\forall x_1, x_2((x_1, x_2) \in \sigma \rightarrow (x_2, x_1) \in \sigma)$	(appl-def 9)
			M_2	
12.	12;	$\vdash$	$(f_1, f_2) \in (\sigma \cap \rho)$	(HYP)
$\vdots$	$\vdots$		$\vdots$	$\vdots$
24.	;3,2,9,8	$\vdash$	$symm(\sigma \cap \rho)$	($\forall$D,$\leftrightarrow$D,$\rightarrow$D)

- A Rule-independent Heuristic
  Our experience with analogy-driven proof plan construction so far suggests that the backward reasoning part of a proof constitutes the proof idea. This heuristic can be used for structurings required for analogy of proof ideas.

3 Applications

3.1 *Obtaining New Methods*

Most obviously, the above decomposition techniques provide new chunks or methods for theorem provers as discussed in the introduction. Methods which resulted from a decomposition might be stored (maybe generalized), if they appear frequently.

3.2 *Theorem Proving by Analogy*

Theorem proving by analogy means to find a proof of a target problem, which is supposed to be similar to a source problem, guided by the known proof of the source problem. Empirical evidence indicates that analogies can best be handled appropriately in a proof planning framework (see [12]). There are at least three reasons why methods have to be restructured within the analogy procedure:

1. Analogies exist at different levels of granularity, such that only some submethods can be transferred appropriately to the target. Consider, for instance, theorem1′ which results from exchanging "triadic" by "monadic" in the theorem1: There exist infinitely many *triadic* primes. of figure 1. The proofs of theorem1 and theorem1′ are similar on "top-level", which means that the level1-method, but not the lower level methods, can be transferred, from proof1 to proof1′. In other proofs by analogy the similarity goes deeper and then lower level methods can be transferred successfully.

2. A condition for the analogical transfer of methods to target proof plan methods is that the problem they prove matches a goal (a problem) of the partial target proof plan. There are, however, different formulations of essentially the same problem. As shown in the examples below, restructuring is one means to

make the assumptions and the theorems of the source and the target problem comparable and thus matching (e.g., by extracting common assumptions).

3. The reformulation of methods which perform the analogical transfer of source methods to target methods may be different for different submethods.

Technically, the frame-like planning-operators, called *methods*, contain a slot with the (partial) subproof scheme they produce. Hence, these methods can be restructured by decomposing partial proofs. The restructuring of methods is done by *meta-methods*, which correspond to the decomposition techniques above[e]. For instance, the meta-method `Deduction-Theorem-Splitting` is applied to a source method S in order to be able to compare (and finally to match) theorems and assumptions of S and a target method T.

Example-3
Assume you like to prove P_T by analogy to the source proof of P_S, where:
P_S: $(\emptyset \vdash \forall x, y((Qxy \to \forall z(Pxz \to Pyz)) \land Qab \land (Paa \lor Pbb) \to \exists x Pbx)$ and
P_T: $(\{\forall x, y((Qxy \to \forall z(Pxz \to Pyz)), Qab, Qbc, (Paa \lor Pcc)\} \vdash \exists x Pcx)$.

As neither the assumptions nor the conclusions of P_S and P_T can be compared directly, even after some reformulation, the source method is restructured by `Deduction-Theorem-Splitting`. It yields two methods S_2 with the postcondition P_S and S_1 with the postcondition $\{\forall x, y((Qxy \to \forall z(Pxz \to Pyz)) \land Qab \land (Paa \lor Pbb)\} \vdash \exists x Pbx$. In the subsequent analogy process, S_1 can be reformulated to a method the postcondition of which matches P_T.

Example-4
Theorem proving by analogy often requires the application of `Relevant-Assumptions` splittings in order to compare the actual assumptions of a problem. The actual splitting may depend on the target problem:
Assume you like to prove theorem 3.1 by analogy to theorem4, where:
Theorem 3.1 Let ρ and σ be two leftcongruences, then $(\rho \cap \sigma)$ is symmetric.
Theorem 4 Let ρ and σ be two equivalence relations, then $(\rho \cap \sigma)$ is symmetric.

An appropriate `Relevant-Assumptions` meta-method splits the proof of theorem4, which can be considered as a method, according to option (1) in example-2.

Assume you like to prove theorem3.2 by analogy to theorem4, where:
Theorem 3.2 Let ρ and σ be two leftcongruences, then
$\forall f(f \in F \to \forall x_1, x_2((x_1 x_2) \in (\rho \cap \sigma) \to (f x_1 f x_2) \in (\rho \cap \sigma)))$.
Theorem 4 Let ρ and σ be two equivalence relations, then $(\rho \cap \sigma)$ is symmetric.

An appropriate `Relevant-Assumptions` meta-method splits the proof of theorem 4, which can be considered as a method, according to option (2) in example-2.

[e] For a more detailed description see [11].

Debugging Analogies

The analysis of failed analogies is an important activity in automated theorem proving by analogy. One necessary component of any such analysis is to find those methods that cannot be transferred adequately to the target proof. Since there are analogies at different levels of granularity (see [9]), the restructuring meta-methods support the debugging process. They isolate those submethods where the failure is located, and which have to be replaced. In the triadic prime number proof (figure 1), for instance, one has to modify all but the level1-methods to get a correct proof plan for the analogous monadic primes proof. In other cases, there is a deeper similarity between the source and the target proof, such that just lower level submethods have to be adjusted.

3.3 *Proof Presentation*

Often automated theorem provers generate proofs that are difficult to understand. Usually they differ considerably from human-style mathematical proofs. Even Natural Deduction proofs are far too detailed and hardly readable. In [7] a presentation of proofs similar to proofs in textbooks is investigated. For such a presentation, proofs that were originally generated by an automated theorem prover are reconstructed. One component of the reconstruction is to find appropriate chunks in a proof. This can be done inter alia by decomposition.

Here is a typical example, where steps of the problem rewriting correspond to the application of decomposition techniques yielding subproofs that prove the rewritten problem[f].

Theorem Let $E \subset F$ and let ρ be a leftcongruence in the semi-group F which is compatible with E, then $\rho \subset \pi_E$.

The problem which is to prove is $\vdash \rho \subset \pi_E$ with
$= \{leftcongruence(\rho), compatible(\rho, E), (E \subset F), semigroup(F)\}$.

The problem is reformulated:

1. **Expanding the definition of** $\subset$ yields the subproblem
   $\vdash \forall x, y((x, y) \in \rho \rightarrow (x, y) \in \pi_E)$.
2. **Two applications of the Deduction Theorem** yield the subproblem
   $\cup \{(x_0, y_0) \in \rho\} \vdash (x_0 \in E \leftrightarrow y_0 \in E) \wedge \forall f(f \in F \rightarrow fx_0 \in E \leftrightarrow fy_0 \in E)$.
3. **Expanding the definition of** π_E yields the subproblem
   $\vdash ((x_0, y_0) \in \rho \rightarrow (x_0 \in E \leftrightarrow y_0 \in E) \wedge \forall f(f \in F \rightarrow fx_0 \in E \leftrightarrow fy_0 \in E))$.
4. **Equivalences splitting** yields the subproblems
   - $\cup\{(x_0, y_0) \in \rho\} \vdash (x_0 \in E \leftrightarrow y_0 \in E) \wedge \forall f(f \in F \rightarrow fx_0 \in E \rightarrow fy_0 \in E)$
   - $\cup\{(x_0, y_0) \in \rho\} \vdash (x_0 \in E \leftrightarrow y_0 \in E) \wedge \forall f(f \in F \rightarrow fy_0 \in E \rightarrow fx_0 \in E)$.
5. **Conjunctive Splitting** yields the subproblems
   - $\cup \{(x_0, y_0) \in \rho\} \vdash \forall f(f \in F \rightarrow fx_0 \in E \rightarrow fy_0 \in E)$

[f] The next theorem is theorem 17.6.3 is examined in a case-study on a mathematical textbook [10]

- $\cup \{(x_0, y_0) \in \rho\} \vdash \forall f(f \in F \rightarrow fy_0 \in E \rightarrow fx_0 \in E)$.
- $\cup \{(x_0, y_0) \in \rho\} \vdash (x_0 \in E \leftrightarrow y_0 \in E)$

6. **Application of the Deduction Theorem** yields the subproblems
   1. $\cup \{(x_0, y_0) \in \rho\} \cup \{f_0 \in F, f_0x_0 \in E\} \vdash f_0y_0 \in E$
   2. $\cup \{(x_0, y_0) \in \rho\} \cup \{f_0 \in F, f_0y_0 \in E\} \vdash f_0x_0 \in E$

Some of the subproofs which are produced by decomposition are likely to be omitted in a human-style presentation, since usually they are not even mentioned in a mathematician's proofs. Actually, the proof of the theorem is presented in [5] as follows (note the reverse order of steps):
Let $(x_0, y_0) \in \rho$.
Let $fx_0 \in E$, then $fy_0 \in E$ is shown,
therefore we have, $\forall f(fx_0 \in E \rightarrow fy_0 \in E)$.
The inverse direction $(fy_0 \in E \rightarrow fx_0 \in E)$ is shown analogously and also $(x_0 \in E \leftrightarrow y_0 \in E)$.
Then $(x_0y_0) \in \pi_E$.
Hence we get $\rho \subset \pi_E$, since x_0, y_0 were arbitrary elements of F.
The steps 2, 4, and 6 are not mentioned in the textbook.

As shown in [9] different levels of proof presentation may be appropriate depending on the presentation purpose. For a general understanding of a proof, a high level presentation is most useful. For such a presentation you have to find the part of the proof that encodes the proof idea. This is again a question of decomposition. For instance, the level1-partial proof in figure 1 encodes the global view of the proof, while level1 together with level2 provides a lower level presentation.

4 Conclusion

The use of decomposition is not limited to theorem proving. Basically the same necessity of restructuring solutions as in our analogy-driven proof plan construction appears, e.g., in merging multiple cases in the analogical replay of the problem solving system Prodigy, as described by Manuela Veloso in [17]. We claim that often **temporary** macro-operators are sufficient to handle the commonly encountered cases of analogy in problem solving. This is contrary to the common view in planning, (see [13]), which regards macro-operators as repeatedly used sequences of operators and suggests to keep them permanently. The idea behind decomposition is to introduce temporary macro-operators regardless of whether they are repeatedly employed.

This paper has focused on applications in automated theorem proving and in particular on the analogy of proofs. The decomposition techniques presented have been tailored for theorem proving and contribute to analogy, discovery of methods, and presentation of mathematical proofs. The techniques have been tested on exapmles from [5] and applied in the analogy-driven proof plan construction of the pumping lemma for cfl.

Related work was reported in [6] where outlines of proofs, which are somehow comparable to proof ideas, are constructed by abstractions. An open problem is the control of the application of decompositions, as well as of abstractions.

5 References

1. S. Biundo, B. Hummel, D. Hutter, and Ch. Walther. The Karlsruhe Induction Theorem Proving System. In *Proc. 8th International Conference on Automated Deduction (CADE)*, LNCS 230. Springer, 1986.

2. W. Bledsoe. Splitting heuristics in automatic theorem proving. *Artificial Intelligence*, 2:55–77, 1971.

3. R.S. Boyer and J.S. Moore. Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic. *Machine Intelligence*, 11:83–124, 1988.

4. B. Brock, S. Cooper, and W. Pierce. Some experiments with analogy in proof discovery. Tech.Rep. AI-347-86, Microelectronics and Computer Technology Corporation, Austin, TX, 1986.

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

6. F. Giunchiglia and T. Walsh. Tree subsumption: Reasoning with outlines. In *Proceedings of 11th ECAI*, pages 77–81, Vienna, 1992.

7. X. Huang. Reconstructing proofs at the assertion level. In *Proc. 12th International Conference on Automated Deduction (CADE)*, Nancy, 1994.

8. J.Larkin, J. McDermott, D. Simon, and H. Simon. Expert and novice performance in solving physics problems. *Science*, 208:1335–1342, 1980.

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

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

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

12. E. Melis. How mathematicians prove theorems. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, Georgia U.S.A., 1994.

13. S Minton. Selectively generalizing plans for problem solving. In *Proceedings of AAAI-85*, 1985.

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

15. M.K. Stickel. Automated deduction by theory resolution. In *Proceedings of the 9th IJCAI*, pages 1181–1186, 1985.

16. F. van Harmelen, A. Ireland, S.Negrete, A. Stevens, and A. Smail. The CLAM proof planner, user manual and programmers manual. Technical Report version 2.0, University of Edinburgh, Edinburgh, 1993.

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

18. Ch. Walther and Th. Kolbe. Optimizing proof search by machine learning techniques. In *Workshop on Automated Theorem Proving IJCAI-93*, Chambery, 1993.