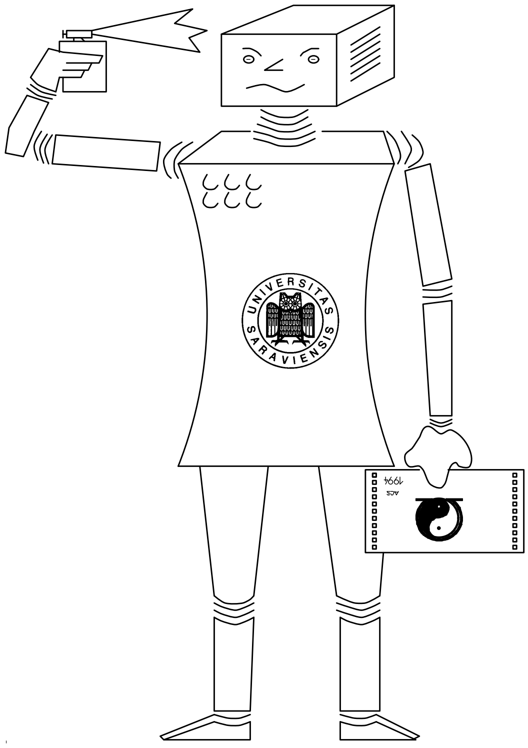
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY
WWW: http://js-sfbsun.cs.uni-sb.de/pub/www/



Reuse of Proofs by Meta-Methods

Xiaorong Huang   Manfred Kerber

Published as: In Jana Köhler, ed., *Proceedings of the IJCAI-Workshop on Formal Approaches to the Reuse of Plans, Proofs, and Programs*, Montreal, Canada, 1995, forthcoming.

Reuse of Proofs by Meta-Methods*

Xiaorong Huang **Manfred Kerber**
Fachbereich Informatik
Universität des Saarlandes
Postfach 151150
D-66041 Saarbrücken
Germany
{huang|kerber}@cs.uni-sb.de

Abstract

This paper describes a declarative approach for encoding the plan operators in proof planning, the so-called methods. The notion of method evolves from the much studied concept of a tactic and was first used by A. Bundy. Significant deductive power has been achieved with the planning approach towards automated deduction; however, the procedural character of the tactic part of methods hinders mechanical modification. Although the strength of a proof planning system largely depends on powerful general procedures which solve a large class of problems, mechanical or even automated modification of methods is necessary, since methods designed for a specific type of problems will never be general enough. After introducing the general framework, we exemplify the mechanical modification of methods via a particular meta-method which modifies methods by transforming connectives to quantifiers.

1 Introduction

Mathematicians learn during their academic training not only facts like definitions or theorems, but also problem-solving *know-how* for proving mathematical theorems. An important part of this know-how can be described in terms of reasoning methods like the diagonalization procedure, the application of a definition, or the application of the homomorphy property.

A similar organization of problem solving knowledge is adopted in the planning paradigm towards automated reasoning [Bundy, 1988]. In such a framework, the methods play the role of plan operators and their executions fill gaps in a partial proof. Bundy views methods essentially as a triple consisting of a tactic, a precondition, and a postcondition. There the tactic is a piece of program code that can manipulate the actual proof in a controlled way. The precondition and the postcondition form a specification of the deductive ability of the tactic, formulating declaratively the applicability condition of the tactic and a description of the proof status after its application. This has been an essential progress compared with a mere tactic language because within this framework it is now possible to develop proof plans with the help of the declarative knowledge in the preconditions and postconditions.

Another main feature contributing to the problem solving competence of mathematicians is their ability to extend the current problem solving repertoire by adapting existing methods to suit novel situations (see [Pólya, 1945] for mathematical reasoning and [VanLehn, 1989] for general problem solving). However, following a one-sided approach relying on procedural knowledge only, the OYSTER-CLAM system [Bundy *et al.*, 1990] developed by Bundy's group still has a severe drawback: the adaption of methods to other problems is almost impossible, because that would require the transformation of programs – tactics are just programs – which is known to be a very hard problem in practice.

To remedy this shortcoming, in previous work we have extended Bundy's notion of a method by separating the procedural and the declarative knowledge in the tactic part of the method [Huang *et al.*, 1994a]. The generalization from Bundy's procedural framework to ours allows to define meta-methods adapting this declarative content in order to create new methods, while the procedural content remains unchanged. Our representation of tactics is simple and intuitive since tactics essentially consist of natural deduction proof schemata possibly containing meta-variables. This direct representation makes a mechanical adaption to new situations feasible in many cases.

The intention of our work can be compared to Ireland's approach of proof critiques [Ireland, 1992]. For a comparison of the two approaches see [Bundy, 1994]. The work of Giunchiglia and Traverso [Giunchiglia and Traverso, 1993] to represent tactics in a logical meta-language has a similar motivation as our work, namely to represent tactics in a declarative manner. In their approach the whole tactic is represented on a logical meta-level, what enables a full declarative representation. In our approach only parts of the tactic are represented declaratively, what should enable easier transformations in some cases.

In the following we shortly introduce our framework for defining methods and exemplify the modification of methods by reformulating proofs where the connectives

*This work was supported by the Deutsche Forschungsgemeinschaft, SFB 314 (D2)

$\wedge$ and $\vee$ are mapped to the quantifiers $\forall$ and $\exists$, respectively.

2 A Declarative Approach to the Representation of Methods

The work in this paper should be understood in the setting of a computational model that casts the entire process of theorem proving, from the analysis of a problem up to the completion of a proof, as an *interleaving process* of proof planning, method application and verification. In particular, this model ascribes a reasoner's reasoning competence to the existence of methods together with a planning mechanism that uses these methods for proof planning.

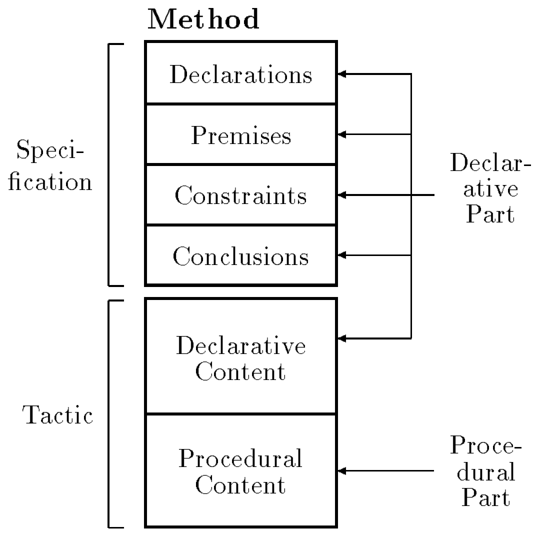
To understand the proof planning process, please remember that the goal of proof planning is to fill gaps in a given partial proof tree by forward and backward reasoning. Thus from an abstract point of view the planning process is the process of exploring the search space of *planning states* that is generated by the *plan operators* in order to find a complete *plan* (that is a sequence of instantiated plan operators) from a given *initial state* to a *terminal state*.

Concretely a *planning state* contains a subset of lines in the current partial proof that correspond to the boundaries of a gap in the proof. This subset can be divided into *open lines* (that must be proved to bridge the gap) and *support lines* (that can be used as premises to bridge it). The initial planning state consists of all lines in the initial problem; the assumptions are the support lines and the conclusion is the only open line. The terminal planning state is reached when there is no more open line in the planning state.

Aimed at modeling a human-oriented reasoning, we want our proof planning mechanism to be able to perform both forward reasoning and backward reasoning. New open lines enter the planning state as subgoals by backward reasoning from existing open lines and new support lines by forward reasoning from existing support lines. In backward reasoning the methods are applied from their conclusions to their premises, that is, some premises become new open lines in the planning state and the open lines matched with the conclusions of the method are deleted in the planning state because they are proved by the method. In forward reasoning the methods are applied from their premises to their conclusions, that is, some conclusions become new support lines in the planning state. In order to achieve this bi-directional reasoning direction with a uni-directional planning mechanism, the planning direction must be independent from the reasoning direction. For this reason we included labels in the premises and conclusions slot of a method [Huang *et al.*, 1994a] by specifying if the method is applied from its premises to its conclusions or vice versa.

In Ω-MKRP, there are mainly two ways of constructing initial methods. Firstly a human users can write them directly. Many typical methods in our framework are written this way, such as the application of a definition or a theorem or the application of a certain property like homomorphy. Secondly initial methods are often extracted from successful proofs as well. This can be easily done since in our approach methods essentially consist of a specification and a *declaratively* represented tactic. Such initial methods are then modified by meta-methods to suit analogous situations.

Formally, in our approach a method is a 6-tuple with the components (for details see [Huang *et al.*, 1994a]):



Declarations: A signature that declares the meta-variables used in the method,

Premises: Schemata of proof lines which are used by this method to prove the conclusion,

Constraints: Additional restrictions on the premises and the conclusions, which can not be formulated in terms of proof line schemata,

Conclusions: Schemata of proof lines which this method is designed to prove,

Declarative Content: A piece of declarative knowledge interpreted by the procedural content. This slot is currently restricted to schemata of partial proofs,

Procedural Content: Either a standard procedure interpreting the declarative content, or a special purpose inference procedure.

Methods can be very general, for instance consist essentially of one rule application at the calculus level. More specific methods comprise proof ideas, for instance, to use a homomorphy property, to apply mathematical induction or to use diagonalization. Such methods are typically formulated in terms of proof schemata. The most specific methods consist of full proofs for specific problems. In the rest of the paper we want to illustrate our approach with the help of an example, namely a transformation of a proof (a most specific method) by a meta-method which uses the correspondence between the connectives $\wedge$ and $\vee$ and the quantifiers $\forall$ and $\exists$. Of course this transformation can be done for more general methods too.

Method : `subsemigroup-conjunction`	
Declarations	—
Premises	$\cap$-Def,SubSGrpDef,NonemptyDef
Constraint	—
Conclusions	T
Declarative Content	2 $\text{subsemigrp}(U_0, G_0, \cdot) \wedge \text{subsemigrp}(V_0, G_0, \cdot) \wedge$ $\text{nonempty}(U_0 \cap V_0)$ Hyp 3 $x_0 \in (U_0 \cap V_0) \wedge y_0 \in (U_0 \cap V_0)$ Hyp 5 $x_0 \in U_0 \wedge x_0 \in V_0$ $\cap$-Def(4) 6 $x_0 \in U_0$ And-E(5) 7 $x_0 \in V_0$ And-E(5) 9 $y_0 \in U_0 \wedge y_0 \in V_0$ $\cap$-Def(4) 10 $y_0 \in U_0$ And-E(9) 11 $y_0 \in V_0$ And-E(9) 14 $\forall x_\blacksquare \forall y_\blacksquare x \in U_0 \wedge y \in U_0 \rightarrow x \cdot y \in U_0$ SubSGrpDef(13) 17 $\forall x_\blacksquare \forall y_\blacksquare x \in V_0 \wedge y \in V_0 \rightarrow x \cdot y \in V_0$ SubSGrpDef(16) 20 $x_0 \cdot y_0 \in U_0$ 14(6,10) 21 $x_0 \cdot y_0 \in V_0$ 17(7,11) 22 $x_0 \cdot y_0 \in U_0 \wedge x_0 \cdot y_0 \in V_0$ And-I(20,21) T $\forall G, \cdot, U, V_\blacksquare (\text{subsemigrp}(U, G, \cdot) \wedge \text{subsemigrp}(V, G, \cdot) \wedge$ $\text{nonempty}(U \cap V)) \rightarrow \text{subsemigrp}(U \cap V, G, \cdot)$ Forall-I(32)
Procedural Content	`schema-interpreter`

Figure 1: Method `subsemigroup-conjunction`

3 The Reuse of Proofs by Meta-Methods

After setting up our declarative framework, in the rest of the paper we concentrate on the mechanical modification of methods by meta-methods. A meta-method is essentially a procedure which takes as input a method and some additional parameters and produces a new method. We have already identified a variety of meta-methods: Generalization of methods in order to apply methods in less specific proof states, syntactic adaption of methods to bridge syntactic gaps, for instance, arities of predicates [Huang *et al.*, 1994a; Melis, 1993].

We illustrate a concrete modification, which translates connectives into quantifiers. The initial method has been extracted from a proof showing that the intersection of two subsemi-groups remains a subsemi-group:
Let U and V be subsemi-groups of a semi-group F, then the intersection $U \cap V$ is also a subsemi-group of F, if the intersection is not empty.
The informal proof goes as follows:

$$\begin{aligned} u, v \in U \cap V &\rightarrow u, v \in U \wedge u, v \in V \\ &\rightarrow u \cdot v \in U \wedge u \cdot v \in V \\ &\rightarrow u \cdot v \in U \cap V \end{aligned}$$

In a previous work [Kerber, 1989], we have presented informally how this proof can be transformed into a corresponding proof concerning an arbitrary number of subsemi-groups. The proof of the first problem is formalized in the method in figure 1, which consists of a proof generated within the proof development environment Ω-MKRP [Huang *et al.*, 1994b] (to abstract away from details we only show the key steps). In the following we illustrate how this method can be transformed by a meta-method to a method for the case of arbitrarily many subsemi-groups. The new theorem is in some sense a generalization of the previous one:
Let $\{U_i : i \in I\}$ be a family of subsemi-groups of a semi-group F, then the intersection $\bigcap_{i \in I} U_i$ is also a subsemi-group of F, if the intersection is not empty.

Actually, the proof of this theorem is analogous to the one above. The analogy can be derived from the correspondence between "$\wedge$" and "$\cap$" on the one hand and "$\forall$" and "$\bigcap$" on the other hand.

The proof can be sketched as:

$$\begin{aligned} u, v \in \bigcap_{i \in I} U_i &\rightarrow \forall i \in I_\blacksquare u, v \in U_i \\ &\rightarrow \forall i \in I_\blacksquare u \cdot v \in U_i \\ &\rightarrow u \cdot v \in \bigcap_{i \in I} U_i \end{aligned}$$

The corresponding method for this proof is given in figure 2 and can be constructed out of the method `subsemigroup-conjunction` in figure 1 by applying the meta-method `connective-to-quantifier`, which contains the following parts[1]:

- parameters: a (possibly empty) list of pairs of related formula patterns

[1] Meta-variables are indicated by overlining.

Method : `subsemigroup-univ-quantification`			
Declarations	—		
Premises	$\bigcap$-Def,SubSGrpDef,NonemptyDef		
Constraint	—		
Conclusions	T		
Declarative Content	1	$\forall n_\blacksquare \text{subsemigrp}(U_0(n), G, \cdot) \wedge \text{nonempty}(\bigcap U_0)$	Hyp
	2	$x_0 \in \bigcap U_0 \wedge y_0 \in \bigcap U_0$	Hyp
	5	$\forall n_\blacksquare x_0 \in U_0(n)$	$\bigcap$-Def(3)
	6	$x_0 \in U_0(n_0)$	Forall-E(5)
	7	$\forall n_\blacksquare y_0 \in U_0(n)$	$\bigcap$-Def(4)
	8	$y_0 \in U_0(n_0)$	Forall-E(7)
	13	$\forall x, y_\blacksquare (x \in U_0(n_0) \wedge y \in U_0(n_0)) \rightarrow x \cdot y \in U_0(n_0)$	SubSGrpDef(12)
	20	$x_0 \cdot y_0 \in U_0(n_0)$	13(6,8)
	21	$\forall n_\blacksquare x_0 \cdot y_0 \in U_0(n)$	Forall-I(20)
	T	$\forall G, \cdot, U_\blacksquare (\forall n_\blacksquare \text{subsemigrp}(U(n), G, \cdot) \wedge \text{nonempty}(\bigcap U))$ $\rightarrow \text{subsemigrp}(\bigcap U, G, \cdot)$	Forall-I(32)
Procedural Content	`schema − interpreter`		

Figure 2: Method `subsemigroup-univ-quantification`

- transformation rules:
  1. Merge two proof lines with formulae which are different only for one single constant or variable, that is, with formulae $\psi[\overline{U}]$ and $\psi[\overline{V}]$, where $\psi[\overline{U}] = \mathsf{subst}(\psi[\overline{V}], \overline{U}, \overline{V})$. $\psi[\overline{U}]$ denotes that $\overline{U}$ occurs in ψ:
  $$\left.\begin{array}{ll} L_1 & \psi[\overline{U}] \\ L_2 & \psi[\overline{V}] \end{array}\right\} \mapsto L_3 \quad \psi[\overline{U}(\overline{n})]$$
  2. Transform certain conjunctions into universal quantifications and certain disjunctions into existential quantifications in formulae:
  $$\begin{array}{ll} L_1 & \Phi[\varphi[\overline{U}] \wedge \varphi[\overline{V}]] \mapsto L_2 \quad \Phi[\varphi[\forall \overline{n}_\blacksquare \overline{U}(\overline{n})]] \\ L_1 & \Phi[\varphi[\overline{U}] \vee \varphi[\overline{V}]] \mapsto L_2 \quad \Phi[\varphi[\exists \overline{n}_\blacksquare \overline{U}(\overline{n})]] \end{array}$$
  3. Execute the transformation according to the formula patterns in the parameter slot.

The last transformation rule gives the user the possibility to specify additional problem-specific transformations, here ($\overline{U} \cap \overline{V} \mapsto \bigcap \overline{U}$), which results in the transition

$$L_1 \quad \Phi[U \cap V] \mapsto L_2 \quad \Phi[\bigcap U].$$

In other cases this optional transformation is not necessary, e.g. in different formulations of the pigeon-hole principle [Kerber and Präcklein, 1995].

The adaption of the justification is not included in the transformation rules above. In general the elimination rules of logical connectives are transformed into the corresponding rules for the instantiation of quantifications. The justification And-E(5) in the first method becomes Forall-E(5) in the second, for instance.

The transformation above does not formulate an unambiguous characterization of the mappings to be carried out, since the first rule may be applied to the wrong candidate, lines 6 & 10, as well as to right one, lines 6 & 7. Which lines should be merged by this rule, can be determined by traversing the proof tree backwards from the theorem: a line can only be merged with its dual line with respect to the recursive branching of the connectives ($\wedge$ or $\vee$) in concern.

The result of applying this meta-method with the additional transformation pattern as argument is the method in figure 2. The four pairs of lines (6,7), (10,11), (14,17), and (20,21) are merged by rule one to the new lines 6, 8, 13, and 20. For instance

$$\left.\begin{array}{lll} 6 & x_0 \in U_0 & \text{And-E}(5) \\ 7 & x_0 \in V_0 & \text{And-E}(5) \end{array}\right\} \mapsto 6 \;\; x_0 \in U_0(n_0) \;\; \text{Forall-E}(5)$$

Rule three is applied to lines 2, 3, and T, thereby we get from line 3 in the source proof line 2 in the target proof (the other two lines are also manipulated by rule two). By rule two, lines 2, 5, 9, 22, and T are mapped to lines 1, 5, 7, 21, and T, respectively.

This method has been actually used in Ω-MKRP to solve the second problem.

Meta-methods can be incorporated into a planning algorithm. To do this, first it must be decided when to interrupt the planning with methods, in order to create a new method with meta-methods. Second for the current proof situation an adequate pair of a method and a meta-method have to be chosen from the knowledge base. We believe that there can hardly be any general answer to this problem and we have to rely on heuristics. In an interactive proof development environment like Ω-MKRP [Huang *et al.*, 1994b] the user might want to make this choice himself. Therefore our main emphasis lies in the task of providing the user with heuristic support for this choice. Even more challenging would be an automation, of course. A trivial answer would be to apply all existing meta-methods on all existing methods and then choose heuristically the best. Such a procedure can be fairly expensive with a large knowledge base. The first heuristics for choosing a method to adapt we will investigate are listed below:

- Organize methods in a hierarchy of mathematical theories and prefer methods that belong to the same theory as the current problem or whose theory is close to that of the problem in the hierarchy.
- Use general conflict solving strategies like those of OPS5, for instance, favor the methods and meta-methods with the most specific specification.

Of course only successful methods generated in a short-term memory are integrated into the permanent base of methods. Another way to reduce the cost of the operation would be to create only the specification of the methods to be generated, select one for application and create the tactic part by need.

As can be seen from the example above, further automatization is possible by comparing the two theorems: The correspondence of the subformula $\text{subsemigrp}(U, G, \cdot) \wedge \text{subsemigrp}(V, G, \cdot)$ in the first theorem and the subformula $\forall n_{\blacksquare} \text{subsemigrp}(U(n), G, \cdot)$ in the second theorem suggests the application of the meta-method `connective-to-quantifier`. In the same manner the parameter can be constructed from the remaining differences of the two theorems by mapping the term $U \cap V$ to $\bigcap U$.

4 Conclusion

We have proposed an approach to declaratively represent methods. The representation in form of proof schemata is intuitive and supports mechanical adaptions of methods to not directly fitting situations. In this paper, we have examplified our approach by presenting the meta-method `conncective-to-quantifier`. The application of this meta-method is illustrated with the help of a full proof as method, but of course it can also be applied to methods containing proof schemata in a similar way.

Some topics for future investigations will concern the incorporation of meta-methods in the general planning framework of Ω-MKRP. In particular, experience must be accumulated in order to automatically interleave planning with methods with the extension of the method base by meta-methods. Another unsolved problem concerns the robustness of the approach with respect to syntactic variants like $\forall x_{\blacksquare} \neg\varphi$ and $\neg\exists x_{\blacksquare} \varphi$.

Acknowledgement

We would like to thank our colleagues Lassaad Cheikhrouhou, Michael Kohlhase, Erica Melis, Axel Präcklein, Jörn Richts, and Arthur Sehn for many fruitful discussion on proof planning.

References

[Bundy *et al.*, 1990] Alan Bundy, Frank van Harmelen, Christian Horn, and Alan Smaill. The OYSTER-CLAM system. In Mark E. Stickel, editor, *Proceedings of the 10th CADE*, pages 647–648, Kaiserslautern, Germany, 1990. Springer Verlag, Berlin, Germany, LNAI 449.

[Bundy, 1988] Alan Bundy. The use of explicit plans to guide inductive proofs. In Ewing Lusk and Ross Overbeek, editors, *Proceedings of the 9th CADE*, pages 111–120, Argonne, Illinois, USA, May 1988. Springer Verlag, Berlin, Germany, LNCS 310.

[Bundy, 1994] Alan Bundy. A comparison of proof planning methodologies. Blue Book Note 966, University of Edinburgh, Edinburgh, United Kingdom, 1994.

[Giunchiglia and Traverso, 1993] Fausto Giunchiglia and Paolo Traverso. Program tactics and logic tactics. IRST-Technical Report 9301-01, IRST (Istituto per la Ricerca Scientifica e Tecnologica), Trento, Italy, January 1993. Also in Frank Pfenning, editor, Proceedings of LPAR-94, *5th International Conference on Logic Programming and Automated Reasoning*, pages 16–30, Kiev, Ukraine, July 16-21, 1994. Springer Verlag, Berlin, Germany, LNAI 822.

[Huang *et al.*, 1994a] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, and Jörn Richts. Adapting methods to novel tasks in proof planning. In Bernhard Nebel and Leonie Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence – Proceedings of KI-94, 18th German Annual Conference on Artificial Intelligence*, pages 379–390, Saarbrücken, Germany, 1994. Springer Verlag, Berlin, Germany, LNAI 861.

[Huang *et al.*, 1994b] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Dan Nesmith, Jörn Richts, and Jörg Siekmann. Ω-MKRP: A proof development environment. In Alan Bundy, editor, *Proceedings of the 12th CADE*, pages 788–792, Nancy, 1994. Springer Verlag, Berlin, Germany, LNAI 814.

[Ireland, 1992] Andrew Ireland. The use of planning critics in mechanizing inductive proofs. In A. Voronkov, editor, *Proceedings of LPAR*, pages 178–189, St. Petersburg, Russia, 1992. Springer Verlag, Berlin, Germany, LNAI 624.

[Kerber and Präcklein, 1995] Manfred Kerber and Axel Präcklein. Using tactics to reformulate formulae for resolution theorem proving. Annals of Mathematics and Artificial Intelligence, forthcoming, 1995.

[Kerber, 1989] Manfred Kerber. Some aspects of analogy in mathematical reasoning. In Klaus P. Jantke, editor, *Analogical and Inductive Inference; International Workshop AII '89, Reinhardsbrunn Castle, GDR*, pages 231–242. Springer Verlag, Berlin, Germany, LNAI 397, October 1989.

[Melis, 1993] Erica Melis. Analogy between proofs – a case study. SEKI Report SR-93-13, Fachbereich Informatik, Universität des Saarlandes, Im Stadtwald, Saarbrücken, Germany, 1993.

[Pólya, 1945] George Pólya. *How to Solve It.* Princeton University Press, Princeton, New Jersey, USA, also as Penguin Book, 1990, London, United Kingdom, 1945.

[VanLehn, 1989] Kurt VanLehn. Problem solving and cognitive skill acquisition. In Michael I. Posner, editor, *Foundations of Cognitive Science*, chapter 14. MIT Press, Cambridge, Massachusetts, 1989.