Explanation-based Similarity: A Unifying Approach for Integrating Domain Knowledge into Case-based Reasoning for Diagnosis and Planning Tasks

Ralph Bergmann, Gerd Pews, Wolfgang Wilke

University of Kaiserslautern
Dept. of Computer Science
P.O. Box 3049
67653 Kaiserslautern, Germany
E-Mail: {bergmann,pews,wilke}@informatik.uni-kl.de

Abstract. Case-based problem solving can be significantly improved by applying domain knowledge (in opposition to problem solving knowledge), which can be acquired with reasonable effort, to derive explanations of the correctness of a case. Such explanations, constructed on several levels of abstraction, can be employed as the basis for similarity assessment as well as for adaptation by solution refinement. The general approach for explanation-based similarity can be applied to different real world problem solving tasks such as diagnosis and planning in technical areas. This paper presents the general idea as well as the two specific, completely implemented realizations for a diagnosis and a planning task.

1 Introduction and Motivation

The underlying principle of case-based reasoning is the idea to remember solutions to already known problems for their reuse during novel problem solving. The case which is most similar to the current problem is retrieved from a case base and its solution is modified to become a solution to the current problem [Kolodner, 1993; Althoff *et al.*, 1992]. One of the aspired benefits of case-based reasoning is to reduce the need to acquire and explicitly represent general knowledge of the problem domain and thereby to overcome the knowledge acquisition bottleneck [Feigenbaum and McCorduck, 1983]. To avoid the acquisition of explicit domain knowledge, the similarity between two cases is mostly assessed by a numeric computation of selected surface features of the problem description. All knowledge about problem similarity is implicitly encoded into a formula which defines a similarity measure. Besides this, less attention was paid on *case adaptation* since this requires a large amount of knowledge, too. We want to argue that knowledge which can be acquired with reasonable effort should be used for similarity assessments as well as for solution adaptation. Such additional knowledge is required for planning as well as for diagnosis tasks, in order to achieve more powerful and domain-tailored case-based problem solvers.

From the current experience in knowledge acquisition for "traditional" knowledge based systems, we can at least distinguish two different types of knowledge [Newell, 1982]: *Domain knowledge* and *problem solving knowledge* [Wielinga *et al.*, 1992]. Problem solving knowledge describes the *process* of problem solving in terms of steps (i.e. basic inferences or subtasks as in KADS) that should to be executed to (efficiently) derive a solution.

On the other hand, domain knowledge consists of descriptions of the "elements" that are available to construct a problem solution, together with the knowledge about the interaction of these elements within the solution to a problem. Consequently, domain knowledge is sufficient to determine whether a proposed solution really solves a given problem. Moreover, it is much easier to acquire than problem solving knowledge, especially in technical domains. In these domains, systems (e.g. machines) are constructed by human engineers based on a well known functioning and interaction of all components the whole system consists of. Blueprints are generally used to document such a design product and are consequently an appropriate information source for knowledge acquisition.

In the following we want to assume that domain knowledge which is sufficient to derive an explanation of a case can be acquired and formalized. On several levels of abstraction, such an explanation can show the correctness of a solution to a problem. Based on this explanatory information, the similarity between two cases can be assessed and the adaptation process can be focused on the relevant portions of the solution.

In the rest of this paper we will describe the approach in more detail and show its application to diagnosis of faults in technical machines and to production planning in mechanical engineering. The next section demonstrates how explanations can be built and represented, and how similarity can be assessed. In section 3, the approach to solution adaptation by refinement is demonstrated and results are presented. The final sections discuss related work and summarize the characteristics of the proposed approach.

2 Explanation-based Similarity for Diagnosis and Planning

The core idea of our approach is to use domain knowledge – which can be acquired rather easy – to set the similarity assessment and the adaptation on a more profound basis founded on an *explanation* of a case. Such an explanation does not describe how a solution is derived (this would require problem solving knowledge) but that a solutions solves the given problem, i.e. a proof of the correctness of the solution. It is really important to keep this distinction in mind since it makes our approach different from derivational analogy [Carbonell, 1986; Veloso and Carbonell, 1993]. On the other hand, the domain knowledge we employ for explaining a case is stronger than just causal relations like in other approaches [Barletta and Mark, 1988; Koton, 1988]. Moreover, explanations based on strong domain knowledge can be easily derived automatically and do not need to be constructed by case-based methods as for example in [Kass and Leake, 1988].

The similarity of two cases can be judged according to the similarity of their explanations. For this purpose, the relevant domain knowledge must be modeled on several levels of abstraction. This modeling also must allow to switch between those different levels by transforming representational terms from one level to another. An explanation on a lower level of abstraction is more detailed and consequently composed of a larger number of specific rules and facts than an explanation on a higher level of abstraction. Therefore, the explanations of two cases can differ very much on a lower level of abstraction but may be identical on a higher level. This observation leads us to a rating of the similarity of two explanations: The lower the level of abstraction on which two explanations are identical, the higher is the assessment of their similarity.

2.1 Representation of Explanations and Similarity Assessment

Single-level Explanations. In diagnosis as well as in planning, an explanation on an isolated level of abstraction can be represented in a graph structure (see Fig. 1) with two different kinds of labeled nodes: *rule-nodes* and *fact-nodes*. Each rule used in an explanation is represented by a rule-node, labeled with the name of the rule. Fact-nodes represent case specific facts in an explanation which are either given (e.g. from a problem description) or which are derived by a rule. Fact-nodes and rule-nodes are linked by directed edges. Incoming edges into a rule-node (starting at a fact-node) reflect the premises of the rule and outgoing edges, leading to fact-nodes, stand for their conclusions. The problem whose solution is explained, is usually represented by a set of *initial fact-nodes* together with a set of *final fact-nodes*. Initial fact-nodes are the starting point of the explanation graph and are not derived by other rules. The final fact-nodes are the end points of the graph and are not further used as a premise in a rule-node. The solution being explained is always directly linked to the rules that are used in the explanation. So, case explanations can be easily constructed by starting at the initial fact-nodes and applying the rules that are indicated by the solution. If the final fact-nodes of the problem can be reached, the solution is called correct and the respective explanation is found. Note, that this explanation is a proof of the correctness of the solution.

Identity of Explanations. Two single-level explanations are called *identical*, if the graphs are identical except for the labeling of the fact-nodes which can vary. Corresponding rule-nodes must be labeled with the same rule name. However, the instantiations of the rules are not part of the labeling of the rule-nodes and can consequently be different in identical explanations.

Multi-level Explanations. As already introduced, explanations are constructed on several levels of abstraction. On two consecutive levels, the explanation graphs are linked by two different kinds of abstraction mappings. The *fact abstraction*

Fig. 1. Multi-level explanation structure

relates several level-n fact-nodes to a single fact-node on level $n+1$. The required knowledge about the different possibilities of fact abstractions is assumed to be part of the available domain knowledge. The second kind of abstraction (called *rule abstraction*) occurs when a subgraph containing several rule- and fact-nodes is mapped onto a single, more abstract rule-node on the next higher level. Usually, many different abstractions, focusing on different aspects of a case, can be indicated by the domain knowledge.

Similarity of Explanations. The similarity between two complete multi-level explanations can now be determined according to the level of abstraction on which single-level explanations are identical. The higher the level of abstraction on which the respective explanations are identical, the lower is the similarity rating.

Similarity Between a Complete Case and a Problem. The definition of explanation-based similarity presented so far requires complete explanations for both cases to be compared. For the similarity assessment within a case-based reasoning process, a complete case from the case-base has to be compared with the current problem description. So, no explanation for the problem is available before the problem is solved. To enable explanation-based similarity assessment nevertheless, the single-level explanations of the case in the case base are attempted to be *mapped* to the current problem description. This mapping can be achieved in two different ways. Starting at the initial facts of the problem description, all relevant rules can be re-applied in the same structure as the explanation in the explained case indicates. If all rules of the original explanation can be applied and if also the final facts from the problem description can be reached, then the mapping is successful at this abstraction level. Alternatively, a whole single-level explanation structure can be compiled – as in more traditional explanation-based learning [Mitchell *et al.*, 1986] – into a set of sufficient conditions over the initial and final facts. This compiled generalization can then be instantiated for the current problem at hand. More details on this topic can be found in [Bergmann, 1992].

The procedure for similarity assessment between a complete case and a problem starts by the attempt to map the explanations at the highest level of abstraction. If the mapping is successful, the process proceeds with the next, more concrete level. The lowest level at which the explanations can still be mapped, indicates the degree of similarity between the case and the current problem.

2.2 An Example from Technical Diagnosis: The MOCAS-System

This general idea will now be applied to the diagnosis of technical systems. MoCAS (<u>Mo</u>del-based <u>C</u>ase <u>A</u>daptation <u>S</u>ystem) [Pews and Wess, 1993] is an already existing, fully implemented realization of this approach for diagnosis of a CNC-machine which consists of 110 components and 356 attributes.

The goal of diagnostic problem solving is to identify a faulty *component* (called *diagnosis*) of a system that shows some unintended behavior. The (partially unintended) system behavior is usually described by a set of *symptoms*. A complete case consists of a collection of known symptoms (the problem description) together with a diagnosis (solution) that is sufficient to explain all of the observed symptoms. However, the diagnosis is usually not determined by the observed symptoms in a case.

In order to explain the diagnosis, domain knowledge about the correct functioning of the system components and their interaction within the system is required. Moreover, complex components (called *compound components*) may also be composed out of several sub-components (parts) which interact in a certain way to achieve the overall functioning. For reasons of simplicity, we want to assume, that each component has a specified interface in which input- and output ports can be clearly distinguished.

Modeling Diagnostic Domain Knowledge. Rules can be used to model the general behavior of components. The precondition of a rule describes certain conditions on the input-ports while the consequences specify the respective components reaction by assigning values to some output-ports. Moreover, the hierarchical *part-of* decomposition of the compound components lead to a natural description of the system behavior on multiple levels of abstraction. So, a behavior can be explained on a high level of abstraction by just using the rules that describe a compound component. On the next more detailed level, the same behavior can be explained by the behavior and interaction of the sub-components the compound component consists of.

Another way of abstracting components is provided by a hierarchical *a-kind-of* structure, all kinds of components are organized in. An abstraction of `motor` as well as of a `light bulb` might be an `electrical machine` (see Fig. 2). Different

Fig. 2. Domain knowledge for technical diagnosis

rules are associated with the motor and the light bulb to describe their specific behavior. But these different behaviors can also be abstracted towards a general rule, valid for all electrical machines.

Example Cases. A simple example from a technical domain is shown in Fig. 3. A generator `G1` supplies a light bulb `L1` via a wire and a relay (see case 1). In this case we want to assume that `wire18` is broken (diagnosis). As a consequence, the lamp stays dark even if the generator supplies voltage and the relay is closed. These symptoms are assumed to be observed. A different case (case 2) appears in a situation, in which a motor `M1` instead of the light bulb is considered. We want to assume that `wire65` is broken which causes the motor to stand still.

Fig. 3. Two cases from a technical diagnostic domain

Explanation-based Similarity. In an explanation-graph, a certain input- or output-value is represented as a fact-node. The actual behavior of the device, transforming input values into output values, will be represented as a rule-node. The diagnosis is indicated by a rule in the explanation structure that describes an unintended behavior (here, the rule for the broken wire). For the two example cases, this modeling leads to explanation structures as shown in Fig. 4. The explanations of case 1 and case 2 turn out to be not identical on the lowest level of abstraction, because different rules describe the behavior of the motor and the bulb. But if we look at these explanation the next higher level, the behavior of the two different components can be abstracted into a rule that reflects a "`doesn't operate`"-behavior of electric machines. So, the explanations of both cases are identical at the second level of abstraction.

If we now consider the case-based diagnosis process involving the mapping (as explained in section 2.1) of the level-2 explanation from case 1 to the problem description of case 2 (generator works, relay is closed, but the bulb stays dark) we can achieve the mapping of the faulty component (`wire18`) from case 1 to the related component (`wire65`). In this situation we can see that a diagnosis adaptation is completely achieved by explanation mapping. However, in general,

Fig. 4. Explanation structures for two diagnostic cases

a mapped abstract solution needs to be refined towards a concrete diagnosis if several rules have been abstracted towards a single abstract rule. This kind of refinement adaptation will be addressed in section 3.1.

2.3 An Example from Planning: The PARIS-System

In the following, we will describe explanation-based similarity as realized in the PARIS-System (Plan Abstraction and Refinement in an Integrated System) [Bergmann, 1993; Wilke, 1993]. In planning, the goal of problem solving is to derive a sequence of actions (or operators), which, when applied, transforms a given *initial state* into a desired *goal state*. Initial state and goal state together constitute the description of a planning problem and the *operator sequence* forms the desired solution.

Production Planning as Example Domain. To demonstrate the application of the explanation-based similarity approach for case-based planning, we present an example from the field of production planning in mechanical engineering adapted from the CaPlan-System [Paulokat and Wess, 1993], a PRODIGY-like approach [Veloso and Carbonell, 1993]. The goal is to generate a process plan for the production of a rotationally-symmetric workpiece on a lathe. The problem description, which may be derived from a CAD-drawing, contains the complete specification (especially the geometry) of the desired workpiece (goal state) together with a specification of the piece of raw material (called mold) it has to be produced from (initial state). Figure 5 shows two examples for rotationally-symmetric workpieces, which both have to be manufactured out of

Fig. 5. Two example cases from production planning of rotationally-symmetric workpieces

a cylindrical mold. To produce the piece in case 1, the mold needs to be chucked on the lathe first. This chucking may only cover a certain part of the workpiece (area `#3` in case 1) so that other parts of the piece remain accessible for the subsequent cutting operations. In the second step of the production plan, the long cylindrical area `#1` must be removed (raw-cut) from the mold. Only when this area is completely processed, the small area `#2` becomes accessible and can be manufactured by a groove-operation. So, these three operations can only be executed in the mentioned order.

Modeling Planning Knowledge. The domain knowledge required for planning is described by the operators that are available, together with the states that are manipulated by them. Planning operators are usually represented in a STRIPS-like manner [Fikes and Nilsson, 1971] with preconditions which refer to state descriptions and effects which describe a state transition function. The operators of a domain can be modeled on several levels of abstraction, an idea already intensively investigated in research on hierarchical planning [Sacerdoti, 1974; Knoblock, 1990]. On the lowest level of abstraction, we require e.g. the description of the `raw-cut(<Area>)` operation. This operation is applicable only if `<Area>` is accessible by the cutting-tool and if `<Area>` specifies a part of the mold which is not already removed. As an effect of this operation, `<Area>` is now removed and additionally, one or more other areas may become accessible.

On a higher level of abstraction, an abstract operator such as `cut(<Range>)` is assumed which has the abstract ability to remove all material on a larger range (e.g. the entire "right side") of the workpiece in one step.

Explanation-based Similarity. For case 1 from Fig. 5, the corresponding 2-level explanation structure is depicted in Fig. 6. Each rule-node in this structure reflects one operation of the solution plan. The fact-nodes represent the states of the workpiece during the execution of the plan. The explanation at the abstract level is composed of two, more abstract rules, each representing an abstract operation. In this explanation, fact abstraction and rule abstraction occur simultaneously. The fact abstraction specifies, for example, that `mold(#1)` together with `mold(#2)` can be abstracted towards `mold(right)`. The rule abstraction, on the other hand, aggregates the `raw-cut` and the `groove` operation and relates it to the more abstract `cut` operation.

If we now want to assess the similarity between case 1 and a second problem as given in case 2, we can see that the explanation given in figure 6 can be completely mapped for the new problem even at the lowest level of abstraction.

Now, consider a third problem in which two grooves (instead of one groove) have to be manufactured on the same side of the workpiece. It turns out that the explanation at the abstract level can be mapped, while the concrete-level explanation cannot be mapped. So, case 1 is more similar to the second problem than to the third problem.

Fig. 6. Explanation structure for case 1

3 Case Adaptation by Refinement

The result of the explanation-based similarity assessment process is not only an assessment of the similarity, but also an adapted solution at some (possibly high) level of abstraction. The goal of the subsequent adaptation is to refine this abstract solutions towards a full solution to the original problem at the required level of detail. This refinement can be achieved by standard hierarchical search based methods, which can employ exactly the same knowledge that was already utilized during similarity assessment. In general, search-based methods are not suitable for solving complex problems on their own. But the abstract solution that is already available, imposes strong constraints on the search space, so that only small sub-problems (the refinement of a single abstract step) have to be solved. The computational cost for the search strictly depends on the number of abstraction levels which have to be bridged and consequently on the degree of similarity between the current problem and the case in the case base. If the similarity is too low, the search space which has to be traversed can even become so large, that no solution can be found in reasonable time. But such a situation can be seen as a strong indication that a new case has to be added to the case base. We can see that the explanation-based similarity assessment estimates adaptation costs.

3.1 Refinement Adaptation in Diagnosis

In case-based diagnosis, the refinement of a diagnosis means specializing a known fault in a compound component to a fault in its sub-components. This only requires a limited search (i.e. by model-based diagnosis techniques [Boblin and

Kashyap, 1992]) in the space of the sub-components the faulty compound component consists of. As an example, we recall Fig. 3 and consider a new, third case which differs from case 1 in that, instead of the wire, a more complex compound component (e.g. in infrared-sender and -receiver) is involved. An explanation mapping at an appropriate level of abstraction will now come up with the mapping of the broken wire to this compound component.

3.2 Refinement Adaptation in Planning

In case-based planning, refinement adaptation means specializing each operator of the abstract solution plan to a sequence of concrete operators. Since this planning task is performed in a limited search space it is assumed to be tractable if the similarity is high enough.

The refinement process for planning tasks is usually more complex than for diagnosis problems. This is because, in planning, all abstract solution steps need to be consistently refined as a whole (see [Bergmann, 1993]), while in diagnosis, only one step of the abstract solution (namely the component which shows unintended behavior) needs to be specialized. For this reason, we have empirically investigated the computational complexity of the refinement process in planning.

Experimental Setting. A case-base of 116 complete cases from the described planning domain has been randomly generated. Then, the case-based planning system was fed in the first run with a random selection of 10% of the cases, and in the second run with all available cases. In both runs, the system was prepared to store only the abstract explanations of the cases. Then, the system was used to solve the problems described in all of the cases. Since no concrete level explanations were available, refinement adaptation was required for each problem solution. During problem solving, the amount of search time required for the refinement was recorded for each problem. Additionally, the problem solving time required by a pure search-based method was determined for comparison.

Results. The results of this experiment are depicted in Fig. 7. The average solution time is plotted with respect to the problem complexity (solution length). For the pure search-based methods, the diagram shows that the required solution time increases exponentially with the problem complexity. All problems that require a solution plan longer than 8 steps could not be solved at all. With the presented case-based approach, the required refinement time rises much slower when the problems become harder. Solutions with a length up to 15 operators could be easily generated, but problems that require more than 17 operators could not be solved anymore. We have analyzed this situation and it turned out, that for the longer problems, one or more abstract operators need to be refined towards a sequence of 6 or more concrete operators. Thereby, the subproblems become so large that the exponential nature of the search space comes to the fore. This can be seen as an indication that a finer differentiation in the modeling of the abstraction levels is required.

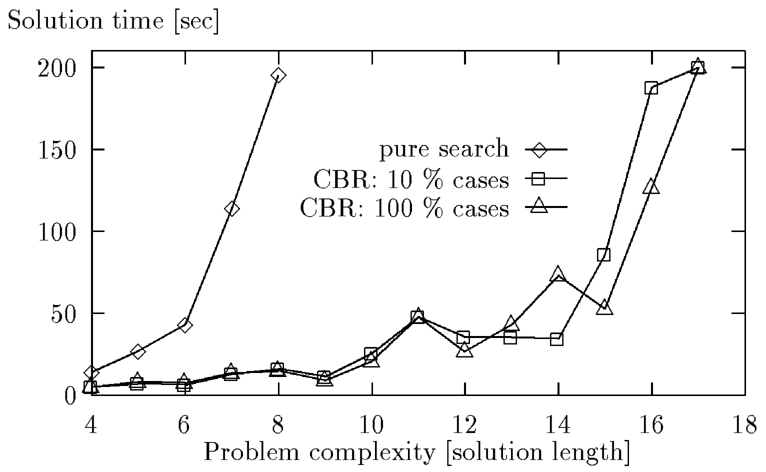

Fig. 7. Empirical results: Planning by search vs. explanation-based similarity

Another observation from this figure is, that the number of cases which were used to train the system does not have a big influence on the refinement time. This is a strong indication, that the number of known cases required to achieve a certain competence of a system can be drastically reduced by the use of the domain knowledge within case-based reasoning.

4 Discussion

Currently, there are some other approaches which favor the integration of additional problem solving knowledge (e.g. [Veloso and Carbonell, 1993]) or more simple causal relationships (e.g. [Barletta and Mark, 1988; Koton, 1988; Janetzko *et al.*, 1992]) into case-based problem solving, while others aim at the integration of different reasoning paradigms (e.g. [Aamodt, 1991]) but mostly in a task-specific manner.

In the following, we want to focus the discussion on related work which also favors the use of general knowledge and explanations for case-based reasoning.

PROTOS and CREEK. In PROTOS [Bareiss, 1989] as well as in CREEK [Aamodt, 1991], general domain knowledge is used to construct explanations which are the basis for similarity assessment. But both approaches focus more on more open domains [Aamodt, 1993] in which only *weak domain theories* are available. Our approach, however, is more appropriate for domains in which *strong domain knowledge* can be acquired such as in technical domains.

Derivational Analogy. In derivational analogy [Carbonell, 1986; Veloso and Carbonell, 1993], general problem solving knowledge is used in case-based reasoning for planning tasks. This approach requires a strong model of the planning process and can only learn from cases that have been solved by the system before. Cases that come from a human expert and that are too complex to be re-solved by PRODIGY's planning component, cannot be used in analogical reasoning. Explanation-based similarity, on the other hand, does not explain the problem solving process and can, therefore, also handle cases that could not be solved before learning.

Explanation-based Learning and Abstraction. There are also relations to a lot of work in explanation-based learning [Mitchell *et al.*, 1986]. Similar to our approach, a strong domain theory is mostly assumed in explanation-based learning. Examples are usually generalized independently, but the generalizations are very often not indexed, which may lead to the utility problem [Minton, 1990]. An additional source of power of the explanation-based similarity approach comes from its ability to abstract explanations on the basis of domain knowledge. Thereby, descriptions are transformed into a completely new abstract language. Other work on abstraction (e.g [Knoblock, 1990]) mostly focuses on abstraction by dropping parts of a description that are not assumed to be relevant on an abstract view.

5 Conclusion

Explanation-based similarity allows an integration of general domain knowledge into the case-based reasoning process for similarity assessment and solution adaptation in an integrated fashion. Similarity can be assessed on the basis of this domain knowledge by comparing and mapping explanations on several levels of abstraction. We have shown, that if the similarity between the new problem and a case in the case base is high enough, refinement adaptation by search is feasible. Furthermore, the scope for which a case can be employed is increased, depending on the amount of domain knowledge that is entered into the system. A knowledge engineer applying this method may decide whether to enter more cases into the case base or whether to spend additional domain knowledge on more elaborated levels of abstraction to achieve the same competence (see also [Holte, 1990]). This general approach has been presented for a diagnosis and a planning task in two real-world domains. Two fully implemented systems accomplish this approach: MoCAS performs a case-based diagnosis task including the described type of solution adaptation for a CNC-machine which consists of over 100 components. PARIS is a domain independent implementation for solving planning tasks. This system works, for example, in the presented domain of mechanical engineering.

Acknowledgements
The authors want to thank Agnar Aamodt, Jaime Carbonell, Manuela Veloso, the reviewers, as well as Michael M. Richter and all members of our research

group for helpful discussions and for remarks on earlier versions of this paper. This research was partially funded by the Commission of the European Communities (ESPRIT contract P6322, the INRECA project). The partners of INRECA are AcknoSoft (prime contractor, France), tecInno (Germany), Irish Medical Systems (Ireland) and the University of Kaiserslautern (Germany).

References

[Aamodt, 1991] Agnar Aamodt. *A Knowledge-Intensive, Integrated Approach to Problem Solving and Sustained Learning.* PhD thesis, University of Trondheim, 1991.

[Aamodt, 1993] A. Aamodt. Explanation-driven retrieval, reuse and learning from cases. In M. M. Richter, S. Wess, K.D. Althoff, and F. Maurer, editors, *Preprints of the First European Workshop on Case-Based Reasoning (EWCBR-93)*, volume II, pages 279–284. University of Kaiserslautern (Germany), 1993.

[Althoff *et al.*, 1992] K.-D. Althoff, Stefan Wess, B. Bartsch-Spörl, D. Janetzko, F. Maurer, and A. Voss. Fallbasiertes Schliessen in Expertensystemen: Welche Rolle spielen Fälle für wissensbasierte Systeme? *KI – Künstliche Intelligenz*, 92(4), December 1992.

[Bareiss, 1989] Ray Bareiss. *Exemplar-Based Knowledge Acquisition: A unified Approach to Concept Representation, Classification and Learning.* Academic Press, 1989.

[Barletta and Mark, 1988] R. Barletta and W. Mark. Explanation-based indexing of cases. In J. Kolodner, editor, *Proceedings of the DARPA Workshop on Case-Based Reasoning*, pages 50–60, San Mateo, California, 1988. Morgan Kaufmann Publishers, Inc.

[Bergmann, 1992] R. Bergmann. Knowledge acquisition by generating skeletal plans. In F. Schmalhofer, G. Strube, and Th. Wetter, editors, *Contemporary Knowledge Engineering and Cognition*, pages 125–133, Heidelberg, 1992. Springer.

[Bergmann, 1993] R. Bergmann. Integrating abstraction, explanation-based learning from multiple examples and hierarchical clustering with a performance component for planning. In Enric Plaza, editor, *Proceedings of the ECML-93 Workshop on Integrated Learning Architectures (ILA-93)*, Vienna, Austria, 1993.

[Boblin and Kashyap, 1992] S. Boblin and R. L. Kashyap. Generating fault hypotheses with a functional model in machine-fault diagnosis. *Applied Artificial Intelligence*, 6:353–382, 1992.

[Carbonell, 1986] J. G. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise aquisition. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine learning: An artificial intelligence approach*, volume 2, chapter 14, pages 371–392. Morgan Kaufmann, Los Altos, CA, 1986.

[Feigenbaum and McCorduck, 1983] E. Feigenbaum and P. McCorduck. *The fifth generation.* Addison Wesley, Reading MA, 1983.

[Fikes and Nilsson, 1971] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Holte, 1990] R.C. Holte. Commentary to: Protos an exemplar-based learning apprentice. In Y. Kodratoff and R.S. Michalski, editors, *Machine Learning: An Artificial Intelligence Approach*, volume 3, chapter 4, pages 128–139. Morgan Kaufmann Publishers, 1990.

[Janetzko *et al.*, 1992] D. Janetzko, S. Wess, and E. Melis. Goal-driven similarity assessment. In H.J. Ohlbach, editor, *GWAI-92 16th German Workshop on Artificial Intelligence*, volume 671 of *Springer Lecture Notes on AI*, 1992.

[Kass and Leake, 1988] Alex M. Kass and David B. Leake. Case-Based Reasoning Applied to Constructing Explanations. In Janet L. Kolodner, editor, *Proceedings Case-Based Reasoning Workshop*, pages 190–208, San Mateo, California, 1988. Morgan Kaufmann Publishers.

[Knoblock, 1990] C. A. Knoblock. Learning abstraction hierarchies for problem solving. In MIT Press, editor, *Proceedings Eighth National Conference on Artificial Intelligence*, volume 2, pages 923–928, London, 1990. MIT Press.

[Kolodner, 1993] Janet L. Kolodner. *Case-based reasoning.* Morgan Kaufmann, 1993.

[Koton, 1988] P. Koton. Reasoning about evidence in causal explanations. In J. Kolodner, editor, *Proceedings of the DARPA Workshop on Case-Based Reasoning*, pages 260–270, San Mateo, California, 1988. Morgan Kaufmann Publishers, Inc.

[Minton, 1990] S. Minton. Quantitativ results concerning the utility of explanation-based learning. *Artifical Intelligence*, 42:363–391, 1990.

[Mitchell *et al.*, 1986] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[Newell, 1982] Allen Newell. The knowledge level. *Artificial Intelligence*, 18:87–127, 1982.

[Paulokat and Wess, 1993] Jürgen Paulokat and Stefan Wess. Fallauswahl und fallbasierte Steuerung bei der nichtlinearen hierarchischen Planung. In A. Horz, editor, *Beitr"age zum 7. Workshop Planen und Konfigurieren*, number 723 in Arbeitspapiere der GMD, pages 109–120, 1993.

[Pews and Wess, 1993] G. Pews and S. Wess. Combining model-based approaches and case-based reasoning for similarity assessment and case adaptation in diagnositc applications. In M. M. Richter, S. Wess, K.D. Althoff, and F. Maurer, editors, *Preprints of the First European Workshop on Case-Based Reasoning (EWCBR-93)*, volume II, pages 325–328. University of Kaiserslautern, 1993.

[Sacerdoti, 1974] E.D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[Veloso and Carbonell, 1993] M. M. Veloso and J. G. Carbonell. Towards scaling up machine learning: A case study with derivational analogy in PRODIGY. In Steven Minton, editor, *Machine Learning Methods for Planning*, chapter 8, pages 233–272. Morgan Kaufmann, 1993.

[Wielinga *et al.*, 1992] B. Wielinga, W. VandeVelde, G. Schreiber, and H. Akkermans. Towards a unification of knowledge modelling approaches. In *Proceedings of the 7th Banff Knowledge Acquisition for Knowledge-based Systems Workshop*, 1992.

[Wilke, 1993] W. Wilke. Entwurf und Implementierung eines Algorithmus zum wissensintensiven Lernen von Planabstraktionen nach der PABS-Methode. Projektarbeit, Universität Kaiserslautern, 1993.