Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT

Knowledge Adaptation: a Means of Knowledge Acquisition

Alvaro de la Ossa

SEKI Report SR-91-11 (SFB)

Knowledge Adaptation: a Means of Knowledge Acquisition*,+

Alvaro de la Ossa
Research Group on Artificial Intelligence (Expert Systems)
Dept. of Computer Science
University of Kaiserslautern
PO Box 3049
D-6750 Kaiserslautern
Federal Republic of Germany
e-mail: delaossa@informatik.uni-kl.de

ABSTRACT

Knowledge Adaptation is an alternative means of knowledge acquisition from previous experience in similar application domains. Adapting knowledge from an old expert system in some domain means transforming rules into rules useful for problem solving in a new, similar domain. The transformation follows from an analogical mapping of the situation described in a rule to situations in the new domain. This paper describes the Knowledge Adaptation approach to knowledge acquisition. We use diagnosis as the exemplar domain task, MOLTKE[1] as the computational testbed, and CNC-machines[2] as the exemplar application domain.

1. INTRODUCTION

Our research group has developed MOLTKE, a workbench for the construction of expert systems for technical domains. The main domain tasks have been diagnosis and configuration, and components for our workbench on other tasks as process planning, configuration, hypermedia, etc., are being developed. In the domain of diagnosis, exemplar application domains have been CNC-machines, 3D-CNC measurement machines, and heterogenoues computer networks. In this work we use the first application domain for illustration.

The CNC-machine producer configures *different machine versions* using parts from a

---

* Also appeared in: *Proc. of the 6th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop 1991*, Banff, Canada, October, 1991.

+ This work is being developed within project X9 (Learning and Analogy in Technical Expert Systems) of the special research area SFB-314 (Artificial Intelligence — Knowledge-Based Systems) under supervision of Prof. Dr. M. M. Richter. The author is a graduate student holding a fellowship from the German Service for Academical Exchange (DAAD, Deutscher Akademischer Austauschdienst) and support from the University of Costa Rica.

1 For **MO**dels, **L**earning, and **T**emporal **K**nowledge in **E**xpert systems for technical domains

2 For Computerized Numerical Control; see, e.g., (Althoff, Maurer, and Rehbold, 1990)

common library. The knowledge adaptation problem consists in transforming knowledge from an old machine's expert system to be used within the same domain task on a newer, *similar* machine. Adapting knowledge involves reasoning about differences and similarities between both machines and using knowledge adaptation strategies accordingly.

The knowledge adaptation scenario arises from the need of *reducing the effort* invested *in the knowledge acquisition task* during the early stages of expert system development.. Knowledge adaptation can be seen as a means of *making knowledge reusable for solving similar situations within the same domain task.*

The paper is organized as follows. Chapter 2 discusses the knowledge adaptation problem. Chapter 3 gives a brief description of a MOLTKE diagnostic expert system and of the CNC-machines domain. Chapter 4 deeply discusses each step of the knowledge adaptation process. For each step, the expected role of the human experts during knowledge adaptation, as well as the strategic knowledge acquisition techniques used, are described. Chapter 5 presents the acquisition and refinement of strategies for improving the knowledge adaptation process. Finally, chapter 6 discusses the relations of this with other work, and describes the state of our research and our next future plans.

2. THE KNOWLEDGE ADAPTATION PROBLEM

A typical flaw of expert systems, due in part to the intimate dependency on the application domain, is the difficulty to make knowledge *reusable* in other similar domains. An expert system might solve problems in a particular application domain, but when this is modified, even lightly, the system may fail to solve problems with acceptable reliability[3].

Sometimes, totally new knowledge bases must be developed for a new domain. Others, the human expert and the knowledge engineer work hard and long together in the task of extracting from the old expert system knowledge that can be useful for the new domain, adapting it accordingly, and incorporating it into a new knowledge base.

But no solution towards the automatization of this process can be found explicit in the literature. There are attempts to define means of making the domain modeling task more flexible (Morik, 1987, 1988; Schreiber, Wielinga, and Breuker, 1991). In our oppinion, the enormous costs due to knowledge acquisition, normal during early stages of system development, cannot be drastically reduced by means only of flexibility in modeling. A learning system must be able of *explicitly adapting its knowledge to a changing domain.* Without this ability, learning is limited to a static view of the domain.

In technical domains in particular, the application domain is mostly described by a model of the physical world, adequate for a particular domain task, e.g., diagnosis (Struss, 1988). The model allows for causal reasoning about structure and behavior. But models alone lack of means of associating experience in the domain task with causality

3 *Reliability* is used here to mean empirical acceptability criteria set by the human expert

relationships in the model itself. Moreover, models are normally incomplete and too simplifying. Modifying the physical world requires of modifying the model accordingly. The lack of association from experience to causality leads, after a modification, to the loss of experience gained on problem solving for the previous state of the physical world.

On the other side, the empirical acquisition of knowledge in the domain is mostly justified by the human expert's advice, due to the inability of most empirical approaches to provide by themselves, using background knowledge, for well-founded explanations of the acquired knowledge.

Thus, a bridge from the model of the physical world to the experience implicit in empirical knowledge and viceversa must be made available, that can allow for justifying the automatic adaptation of knowledge from the old expert system to cope with the domain task on the new physical world.

2.1. Overview of the Knowledge Adaptation Approach

Given an already developed diagnostic expert system for some machine and a new, functionally *similar* but structurally different machine, we wish to adapt the old system's knowledge to be (re)used for diagnosis on the new machine.

Our approach to knowledge adaptation identifies knowledge suitable for adaptation by using a *similarity-based* approach. Two machine parts are considered similar if they belong to a same parts class and their *functionality* descriptions are similar. A part's functionality and the similarity metric used are described later in section 4.3.1.

A simple *model-based* reasoning mechanism is used to *analogically* map a diagnostic situation from the old machine into a new, *plausible* situation for the new machine. The analogical mapping is approached on two levels. First, following domain-independent mechanisms, a situation is explained on the domain model. Then, domain-dependent features are used to complete the analogical mapping. The plausibility of the new situation is verified through simulation automatically guided by background knowledge[4].

The knowledge adaptation component communicates with the performance element of the expert system to generate a hypothesis for the new, plausible situation. If such a hypothesis is found, the resulting *rule* (i.e., diagnosticSituation → hypothesis) must be incorporated into the new expert system. The rule can be incorporated if it results *consistent* with the current state of the new expert system.

The whole process, from the identification of similar parts in the old machine to the incorporation of an adapted rule in the new expert system is called an *adaptation case*. Adaptation cases are generalized to *adaptation rules*, that represent strategic knowledge for adaptating groups of diagnostic rules sharing a common general diagnostic situation and the same particular hypothesis.

---

4 See (Dutta, 1988) for a deeper discussion on justifying analogical transformation through the intensive use of background knowledge. Other approaches to verification use *experimentation*, as in (Carbonell and Gil, 1990; and Shen and Simon, 1989)

3. THE MOLTKE DIAGNOSTIC EXPERT SYSTEM

The workbench MOLTKE plays the role of computational testbed for our research. In this chapter, a brief description of the basic terminology and of the diagnostic task in MOLTKE is given, including some details relevant for our knowledge adaptation component. We illustrate with our exemplar application domain. Further details about the conceptual approach to diagnosis and methodological development of our workbench are found in the references mentioned through the description.

3.1. The Development of a MOLTKE Diagnostic Expert System

The diagnostic task is seen in MOLTKE as the combination of *classification* and *test selection*. Classifying means determining a rough or intermediate diagnosis for a given fault. Test selection is used to refine the current diagnosis until a final diagnosis is found. Diagnoses are organized in a *context graph*, where a node, called context, represents a diagnosis, and the links, refinements between contexts.

For knowledge adaptation purposes, the generation of the context graph is incremental, i.e., new contexts are generated only when they are needed for diagnosis or for knowledge adaptation. This has the advantage of reducing the time required for knowledge base generation for a new machine; it also allows the easy changing of the knowledge base in response to light modifications in the structure of the new machine[5].

In our expert system, diagnostic knowledge is structured *vertically*—in a heterarchy of *contexts*, and *horizontally*—separated in *ordering* and *shortcut* knowledge. Contexts represent knowledge about *failures* (*classification*), and ordering and shortcut knowledge, knowledge about the *diagnostic process* (*test selection*).

A *context* has a precondition (conjuntion or disjunction of conjunctions of symptom values) which, when satisfied under the current situation, causes the associated *diagnosis* to become *proven*. To every context a set of ordering rules and a set of shortcut rules are associated, as well as a *context interpreter*. The locality of ordering rules and the presence of a (local) interpreter allow for specialized strategy for test selection.

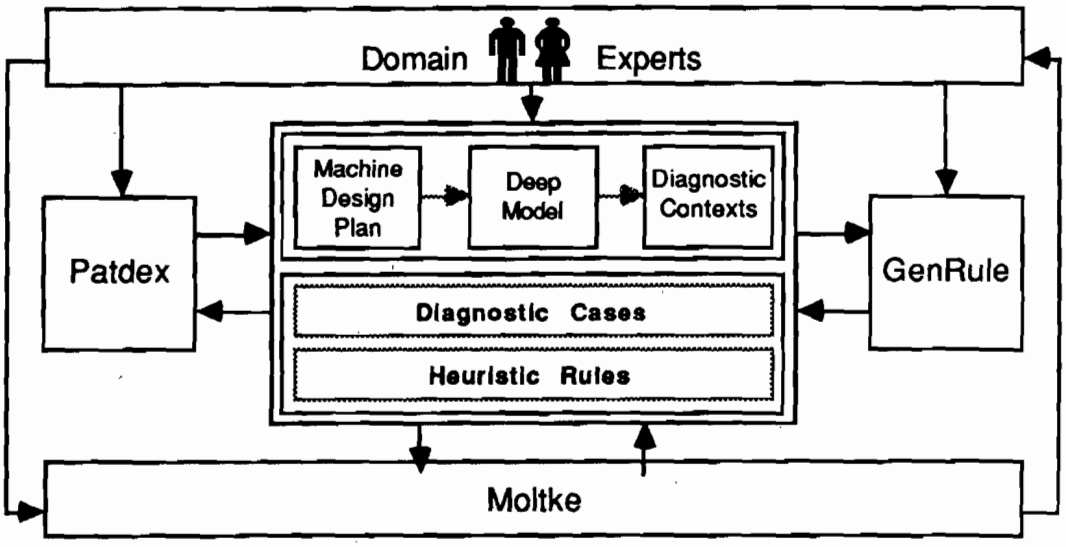
Definitions

Symptom Class — relates a name with a list of possible values (e.g., `Valve` and `(open,closed)`)
Symptom instance — (or symptomvalue, or symptom) describes the state of a machine part (e.g., `(Valve27Y1 closed)`)
Situation — set of all symptoms; (partially) describes the actual state of the machine
Formula — stores the current binding of a symptom (a variable in the predicate calculus with a three-valued logic—`true`, `false`, `unknown`—used to evaluate the formulas in a *formula language*)
Test — determines the current value of one or more symptom instances
Ordering Rules — determine which test to execute next; the order of these rules determines the (potential) execution order of tests

5 For instance, after replacement of defectuous parts

Shortcut Rules — represent a relation between symptom values, and shorten the diagnostic process by deriving one or more (still unknown) symptom values from one or more (known) symptom values, thus eliminating the need of gathering their values through tests

Figure 1 below shows the overall process of expert system construction in MOLTKE.



— Figure 1 — The Construction of the Expert System —

3.1.1. Expert System Generation : Modeling and Simulation

Diagnostic knowledge stored in contexts is extracted automatically and directly from the machine's design plans. The automatic model and diagnostic knowledge compiler MAKE[6] (Rehbold, 1991) builds up a static, deep model of the machine's structure and behavior. The model is checked for consistency and the context heterarchy is built through a simulation process. For each context, its precondition and shortcut rules are generated. MAKE's output is a basic expert system, which may be later refined and extended[7].

A new version of MAKE, called iMAKE, is currently being implemented in the context of this research, which allows the incremental extension and modification of the context graph. iMAKE reduces the effort required for context graph generation by creating a library of subgraphs associated to machine parts according to their functionality.

In the machine's model, specific parts are described by classes of parts, which store mainly its *name*, *ports*, *behavioral* description (input-output rules relating ports), and *typical faulty behavior*[8]. *Primitive* parts cannot be decomposed. *Complex* parts store additionally their *subparts* and *connectivity* (physical connections between subparts). The structure is represented hierarchically. The behavior of complex parts is elaborated

6 For **M**odel-based **A**utomatic **K**nowledge **E**xtractor
7 By, e.g., a *service technician* or a machine's *engineer*, in the case of CNC-machines
8 For simplicity, only the model slots relevant for this discussion have been described

on the basis of their component parts.

¡MAKE extends a part instance's description with its *functionality*. This is defined as the subset of behavioral rules that describe the actual part's behavior in the particular configuration in which it is placed. That is, a part's behavior describes all possible outputs from the part for all possible inputs (what is the part designed to do?), while functionality describes the outputs that are expected from the particular connections of the part in a particular machine (what does the part do in this machine?).

Once the model is built, ¡MAKE simulates it in search for diagnostic information that can be made explicit in MOLTKE's representation formalism. Two tasks are included in this process: definition of symptom and tests, and generation of contexts[9] and their rules. A symptom class is generated for each port of every primitive part instance in the machine. A test is generated for each symptom class. Later in MOLTKE, a symptom will be represented by a symptom instance and the actual value measured on the respective port.

During simulation, for each port on which a deviation from an expected output value can be found, a context is generated, and its name, precondition, and correction are defined. Intermediate diagnostic contexts (those that are not leaves in the graph) additionally contain shortcut rules, refinement links (to other contexts), and ordering rules.

The context heterarchy reflects the diagnostic expert's proceeding during diagnosis[10]. The search space is shortened by considering only those behavioral rules *relevant* for describing the *intended behavior* of the machine part[11]. A context class is generated for every *relevant* faulty output value (from the faulty behavior description of the part).

In our new conception of this process, ¡MAKE does not need to generate all the contexts for a given machine part if that task was solved before for other part (even from other machine) of the same class and with the same functionality conditions. ¡MAKE stores a newly generated part's context subgraph and associates it to the part's class and its particular functionality. Context subgraphs are later used as patterns for other parts of the same class and with the same functionality conditions, which only need to be consistently inserted in the whole context heterarchy.

3.1.2. Refinement and Extension of the Expert System's Knowledge

The basic MOLTKE-expert system can be refined, e.g., by modification of shortcut rules, inclusion of new ordering rules, etc. This task is carried out in joint work by the expert and the knowledge engineer. The expert elaborates flow diagrams of test execution order, which are traduced by the knowledge engineer into ordering rules and input into the corresponding contexts. The expert can edit the machine model as well, as the context graph, using graphical tools developed specially for that purpose (based on Smalltalk's Browser interface[12]).

---

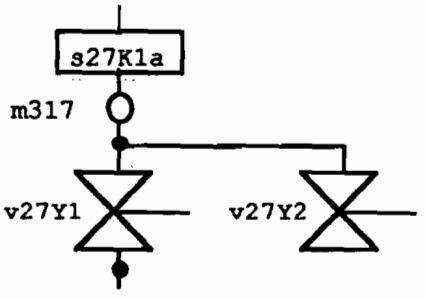
9 Additionally, *corrections* (to the faults) must be also generated, but this is not relevant here
10 We found this to be true for all technical domains modeled so far
11 This is MAKE's terminology, which agrees with our definition of *functionality*
12 Smalltalk is the implementation language of MOLTKE

Moreover, the system acquires, organizes, and refines diagnostic experience through its learning component. Experiential knowledge is case-based modeled. An *ordered set of symptoms* associated to a *hypothesis* defines a *diagnostic case*. Two subsystems cooperate to enhance the diagnostic task. PATDEX/2[13] (Althoff, De la Ossa, Maurer, Stadler, and Weß, 1989; Althoff and Weß, 1991a) is a case-based reasoning system that retrieves the most similar case to the current situation. Similarity is defined in terms of the symptoms in the situation. The hypothesis to the retrieved case is mapped to the current situation. PATDEX/2 acts interactively on-line to enhance the diagnostic process. The other subsystem, GENRULE[14] (Althoff and Traphöner, 1990; Althoff, 1991), acts off-line by compiling heuristic generalization rules from diagnostic cases. Those rules, which describe partial shortcut rules, are given a statistically-determined certainty factor.

3.2. A Sample Application Domain : CNC Machine Centers

A CNC machine center is a production tool machine that gives some raw material a desired form and precise measure through controlled relative movement; it has a magazine of tools from where it can choose and seize one to work with on a piece of material. Some of its operations on the material are drilling, milling, rubbing, tap sharpening and cutting, sawing, etc. The basic components of a machine center are rack, sliding carriage, guides, and drive or propulsion. Attached are other facilities for control, regulation, changing, etc., of tools and material. A large knowledge base for fault diagnosis of a machine center has been developed by our research group[15].

Examples of different configurations leading to functionally similar machines can be drawn ranging from very high abstraction levels in the machine's structure, e.g., using a tool claw arm with finger elements instead of a claw arm with plier claw and rotation, to very low levels, e.g., using one of two different valves in a hydraulic or electrical element. Figure 2 shows a sample machine part design plan. We show its model of structure and behavior and a sample diagnostic context from our knowledge base.



— Figure 2 — Sample Design Plan of a Complex Machine Part —

13 For **PAT**tern-Directed **EX**pert system
14 For **GEN**erator of empirical MOLTKE **RULE**s
15 Some parts (e.g., the tool changer, etc.) have been modeled, not the whole CNC machine, but the modeled domain is complex enough for our purposes; knowledge bases from other technical domains have also been developed, and the results of experiments with them have given fruitful feedback during the development of our workbench

Instance **v2s27Y1-2** of class **Valve2Selection** is entered to the system using a graphic model editor. After modeling, the resulting description for the instance and its subparts looks like follows:

```
ComplexPart Class #Valve2Selection
      ports:          #((lever mechanical in low) (currentIn ac in medium)
                        (currentOut ac in medium) (slide1 mechanical out low)
                        (slide2 mechanical out low))
      subparts:       #((Switch+ s low) (MeasuringPoint m low)
                        (Valve v1 low) (Valve v2 low))
      connections:    #((self currentIn currentIn s) (s currentOut currentIn m)
                        (m currentOut currentIn v1) (m currentOut currentIn v2)
                        (v1 currentOut currentIn self) (v1 slide slide1 self)
                        (v2 currentOut currentIn self) (v2 slide slide2 self)).
Valve2Selection Instance #v2s27Y1-2
      location:       'toolChanger'
      subparts:       #((s s27K1a 214) (m m317 214) (v1 v27Y1 275) (v2 v27Y2 276)).
PrimitivePart Class #Switch+
      ports:          #((current1 dc inOut medium) (current2 dc inOut medium)
                        (lever mechanical in low))
      behavior:       #((lever = unshifted) (current1 = X) -> (current2 = 0))
                        (lever = shifted) (current1 = X) -> (current2 = X))
                        (lever = unshifted) (current2 = X) -> (current1 = 0))
                        (lever = shifted) (current2 = X) -> (current1 = X)))
      failures:       #((noContact (((current1 = X) -> (current2 = 0))
                                    ((current2 = X) -> (current1 = 0))))).
```

During simulation, the context graph is generated. Following is a sample context for **Spindle**, a part containing **v2s27Y1-2** as one of its subparts:

```
#SpindleStopNotReached
precondition:
      (= SpindleStop SpindleStop SpindleStopNotReached)
shortcutRules:
      (if (and (IsPreconditionOf SpindleStopNotReached) (= Valve Valve27Y1 Closed) )
       then IOSstatusOUT32 logic1 factor "total")
      (if (and (IsPreconditionOf SpindleStopNotReached) (= IOStatus IOStatusOUT32 1) )
       then IOStatusIN1112 logic1 fator "total")
orderingRules:
      (if (true) then Valve27Y1 test)
      (if (= Valve27Y1 closed) then OilConductsOrientedSpindleStop test)
      (if (= Valve27Y1 open) then IOStatusOUT32 test)
      (if (= IOStatus IOStatusOUT32 0) then IOStatusIN1112 test)
      (if (= IOStatus IOStatusIN1112 0) then VoltagePotential361362 test)
```

3.2.1. CNC-Machines Diagnosis

A failure in the CNC-machine is reported by the CNC *Input/Output Card* , which sends a *machine failure code*, directly associated to machine parts normally on very high abstraction levels in the machine's structure. A diagnostic situation is initiated when a machine failure code is received from the CNC card. This code has the advantage of restricting the search for a diagnosis on a particular machine part and its subparts. Then, during diagnosis, the expert system enters a *establish-and-refine* cycle of testing for symptoms and refining to more particular diagnostic contexts.

4. KNOWLEDGE ADAPTATION

The knowledge adaptation task is seen as an incremental process in which diagnostic rules and cases from the old expert system are transformed into *rules* for the new system. A rule has in its left-hand side a diagnostic situation (an ordered set of symptom values) and in its right-hand side a hypothesis[16] .

The transformation of rules leads to rules whose situations are *plausible* in the new machine[17] and whose hypotheses shall be proved by the new expert system. The main adaptation strategy in our approach builds an analogy of the situation of a rule from the old machine onto the new machine. The hypothesis for the new situation is obtained by simulating the new situation in the new expert system.

The approach foresees the acquisition, extension, and refinement of adaptation strategies by exploiting the results of successful rule adaptation to improve later performance. Knowledge adaptation is seen as a process of proposing new diagnostic rules and learning from their verification. A case memory for adaptation cases is currently being implemented in the context of this research. Adaptation rules (strategies) are stored associated to particular machine part classes and their faults. A memory architecture inspired in Kolodner's is being used as a model (Kolodner 1983a, 1983b, 1989).

Besides the feedback from the human expert upon accepting or rejecting adapted rules, the expert can also play the role of a tutor by entering manually prepared adaptation cases, considered as positive examples of a particular adaptation strategy. In this case, the expert must provide the system with an indication of how the analogy that leads to the adaptation is originated, in the form of a causal explanation on the model of the machine. This advice can be used later as an alternative strategy to analogy recognition if this is not possible on the machine's model level. A detailed description of the generation of causal explations of a rule is given later.

In the next sections we analyze the kinds of knowledge in our expert system, compel the use of the terms *case* and *rule*, and describe the knowledge acquisition process.

4.1. Causal and Empirical Knowledge in our Expert System

Causal or *background* knowledge describes the application domain (the machine and its diagnosis), while *empirical* knowledge describes experience acquired during diagnosis or directly from the expert. Causal knowledge is acquired directly from the design plans of the machine. This knowledge is stored initially by IMAKE in a *static* knowledge base containing the machine's *model* (knowledge about the particular domain) and the *context graph* (diagnostic knowledge in the particular domain).

Empirical knowledge can be acquired either directly from the *domain expert*, through

16 The use of the terms *rule* and *case* is made clear later in section 4.2.
17 The term *plausibility* is here used to mean that the situation may physically occur in the machine, or in other words, that the symptoms in the situation are not inconsistent with the machine's model

actual *experience* during diagnosis (MOLTKE and PATDEX), or by automatic generation starting from diagnostic cases (GENRULE). Four kinds of empirical knowledge are identified:

Diagnostic cases : successful diagnoses, either actual, or known to the expert and given directly to the system. They are represented as follows:

$$S_1, S_2, \ldots, S_n \rightarrow \mathcal{D}h$$

where S_k ($k = 1, \ldots, n$) are known symptom values, and
$\mathcal{D}h$ is a diagnostic hypothesis (the name of a certain context).

Test selection cases : selection of tests, given a particular diagnostic situation, leading to gathering useful symptom values during actual diagnosis. They are represented as follows:

$$S_1, S_2, \ldots, S_n \rightarrow \mathcal{T}h$$

where S_k ($k = 1, \ldots, n$) are known symptom values, and
$\mathcal{T}h$ is a test hypothesis (the name of a certain test).

Ordering rules : empirical determination of the order in which tests must be executed in order to refine the diagnostic hypothesis for a given context. They are represented the same as test selection cases with the difference that test selection cases are stored in a case memory and ordering rules in the particular expert-given order within a diagnostic context.

Shortcut rules : empirical determination of one or more (still unknown) symptom values given a particular diagnostic situation. This rules can be either directly given by the expert, or generated off-line by GENRULE, and are represented as follows[18]:

$$S_1, S_2, \ldots, S_n \rightarrow S_{n+1}, S_{n+2}, \ldots, S_m$$

where S_k ($k = 1, \ldots, n$) are known symptom values, and
S_l ($l = n+1, \ldots, m$) are (still unknown) symptom values.

4.2. Cases and Rules : What Should Be Adapted?

It is important to note the difference between cases and rules[19]. Cases are examples of real diagnostic situations and their solution. They contain a problem description (the situation), a solution (the hypothesis), and a justification for the solution, which should contain an explanation of how the solution was found, in terms of the situation.

In MOLTKE, the justification for the solution of a diagnostic or test selection case is the temporal order (sequence) in which symptom values were gathered through test execution. A case's situation $S_1, S_2, \ldots, S_n$ is actually given in this order: symptom S_k was gathered before symptom S_{k+1}. A more grounded justification is still missing in MOLTKE. Thus, for knowledge adaptation, we extend a case's justification to a more complete explanation, by abstracting the symptom values in the situation, in their order, on the

18 Shortcut rules are also generated by MAKE. But these rules are justified on the machine's model (they are called *total*); they do not reflect empirical knowledge

19 A more extense discussion on cases and rules can be found in (Althoff and Weß, 1991b)

machine's model. This is done through a simulation process in which parts of the machine involved in the situation and their causal (structural and behavioral) relationships are identified.

Rules, on the other side, differ from cases in that they do not have a justification. Their interpretation is unique, while a case's interpretation depends on the purpose, the similarity with other cases, and so on. Ordering and shortcut rules in MOLTKE have no justification, or in other words, the justification is indeed empirical. Ordering rules are entered into the knowledge base (into particular contexts) by the knowledge engineer, traduced from flow diagrams of test execution order prepared by the human expert. The generation of a model-based justification of ordering rules, and furthermore, of their order, is, for the moment, still not considered, although this will imply a great reduction in knowledge acquisition effort.

We have so far tested the approach only with diagnostic and test selection cases. What makes ordering rules more complex is the fact that their justification must not only take into account each of a context's rules, but also identifying equivalent contexts in the new machine, as well as considering their sequence or order.

Finally, empirical shortcut rules' justification rests on the diagnostic cases from which they were abstracted by GENRULE. We still do not consider shortcut rule adaptation for one reason: after adapting diagnostic and test selection cases, GENRULE can be applied on the new expert system to generate new shortcut rules. It would be however interesting to approach their adaptation and compare the results with GENRULE's.

In the rest of the discussion we refer to diagnostic and test selection cases using the terms *rules* and *cases* without distinction. Following, we describe KALES[20], which is how we have named our knowledge adaptation approach. Along the discussion we describe the intervention of the experts. In chapter 5 we discuss the means of acquisition and refinement of knowledge to improve the adaptation strategy. Due to space limitations, representation and implementation details (particularly algorithms) are left out except where required for the discussion.

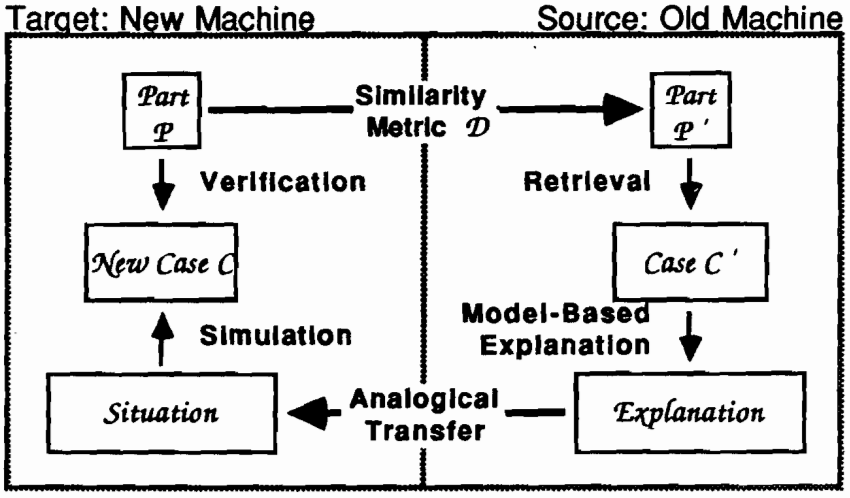
4.3. The Knowledge Adaptation Task

Knowledge adaptation is a source of knowledge acquisition for the new expert system. It is based in the main assumption that knowledge from the expert system for an old machine can be reused for diagnosis on a new machine if it is possible to find a *correspondnce* between both machines.

We assume that both machines are functionally similar, in the sense that they both serve a similar purpose, e.g., drilling, sawing, etc. But they are structurally different, that is, the internal structure, connections among subparts, and internal behavior of the new machine can be seen as a modification of the old machine. Finally, we assume that both machines' subparts are taken from a common library of components.

20 For **K**nowledge **A**daptation in **L**earning **E**xpert **S**ystems

Knowledge adaptation is incremental. First, the process is normally carried out off-line, although on-line, interactive sessions of adapting rules for specific faults can be set with the experts. Second, a step in learning of knowledge adaptation strategies is done from one rule rather than from a set of them, although an incremental definition from sets of rules is possible. Figure 4 below gives a synopsis of the knowledge adaptation task.



— Figure 4 — The Knowledge Adaptation Task —

4.3.1. The First Step : Identification of Knowledge for Adaptation

Every time iMAKE generates the diagnostic contexts for parts of the new machine, KALES tries to adapt empirical knowledge for similar machine parts from the old machine. Thus, the first subtask of KALES is identifying those machine parts. This follows from a similarity-based approach. Recall that the functionality of a machine part $\mathcal{P}$ is defined as the subset of behavioral rules actually used in the particular location of $\mathcal{P}$ in the machine. Two machine parts $\mathcal{P}_1$ and $\mathcal{P}_2$ are *functionally similar* if they

(i) belong to the same parts class,
(ii) have the same number of input and output ports, and
(iii)

$$\mathcal{D}(\mathcal{P}_1, \mathcal{P}_2) := \frac{C_{12} - \mathcal{N}_{12}}{C_{12} + \mathcal{N}_{12}} > \epsilon \quad \text{, where}$$

C_{12} is the number of corresponding and $\mathcal{N}_{12}$ the number of noncorresponding rules in the functionality description of parts $\mathcal{P}_1$ and $\mathcal{P}_2$, and ϵ is some threshold value in [-1,1] (most likely close to 1), that can be interpreted as the minimum (empirically) acceptable similarity.

This is a still simple similarity metric. It can be extended, for instance, by weighting the relevance of rules in the functionality descriptions of both parts. We do not deepen the discussion on similary metrics here, we just propose a metric that can be later improved.

KALES searches for functionally similar parts of the old machine. The search is shortened by restriction (i) in our definition. This avoids the exponential nature of the search in the machine's structure hierarchy. With the list of retrieved old-machine parts, KALES accesses the case memory in search of cases in which the failure code or the diagnostic hypothesis refers to the parts in the list. The retrieved set of cases is called the *adaptation candidates* set. KALES tries to adapt each case in the set for the new machine.

Note that restricting to the machine failure code or the hypothesis is a (domain-dependent) heuristic. The idea behind this is that we try to retrieve from the case memory empirical knowledge *relevant* for diagnosis on the parts for which iMAKE just generated the diagnostic contexts. The experience implicit in a case is described by the symptoms that lead to a fault's recognition. The criterion for case retrieval cannot be the sole appearance of a particular symptom in the case's situation. This would lead to retrieving cases that express the relevance of a machine part for the diagnosis of some other, rather that for the diagnosis of that part itself.

4.3.2. The Second Step : Elaboration of an Adaptation Candidate

Elaborating a candidate means preparing a candidate case for its actual adaptation into the new expert system. This involves mainly producing a justification for the case and analogically transferring the justification to the new machine.

4.3.2.1. Justifying an Adaptation Candidate

Justifying a case means producing an explanation of its situation, that is, of *how* the diagnosed fault was detected and *why* it occurred. Determining how a fault is detected means following the diagnostic situation in the temporal order of symtom gathering and producing in doing this a causal explanation. Detecting why a fault occurred is more complex and we do not intend to provide a conclusive answer to this point now. The cause of a fault might need of the extension of our machine's model with dynamic and geometrical (design) features, which, unfortunately, are still not being modeled.

On the representational level, the explanation of the rule's situation is described by a set of relations between machine parts. The relations can be *structural* (e.g., part $\mathcal{P}_i$ is part of part $\mathcal{P}_k$, or part $\mathcal{P}_i$ is connected to part $\mathcal{P}_k$), *functional* (part $\mathcal{P}_i$'s output is part $\mathcal{P}_k$'s input through port i_k), or *empirical* (denoting that no causal relation—in a strict sense— could be found and that it is the expert who must decide whether a causal relation exists, or is relevant, or none). We record all relations found in the explanation.

On the implementational level, the explanation is a *causal graph* whose nodes are machine parts and whose links are the structural, functional, or empirical relations identified. Identifying the parts involved in the explanation is simple. Each symptom in the situation refers to a machine part (a state) or part's port (an input or output value). For each part referenced by a symptom in the situation, a node is created in the graph.

The creation of the graph follows the order of symptoms in the case's situation. That is,

the graph is first created for the first symptom and the transition to the second one. This is a causal explanation of why a test for the second symptom was chosen. The answer may be, for example, that $\mathcal{P}_2$ is a subpart of $\mathcal{P}_1$, or that a functional relation between $\mathcal{P}_1$ and $\mathcal{P}_2$ is identified in the faulty behavior description of one of those parts, and so on.

The identification of relations between parts in each step of this process follows from a simulation of the (partial) situation on the machine's model and is guided by the context graph. Let $S_1, S_2, \ldots, S_n$ be the diagnostic situation, and $\mathcal{P}_k$ the machine part referenced by symptom S_k. For each pair of symptoms S_k, S_{k+1} in the situation, all possible relations between $\mathcal{P}_k$ and $\mathcal{P}_{k+1}$ are searched, and a link is created between nodes $\mathcal{P}_k$ and $\mathcal{P}_{k+1}$ for each relation found. The link's type is that of the relation. For functional relations, the link is additionally marked with the output value from $\mathcal{P}_k$.

Example:

iMAKE generated the diagnostic contexts for machine part `NewTightenRelease`. Machine part `OldTightenRelease` was found to be similar to the former. This is a part of `OldToolChanger`. Its class is `TightenReleaseDevice`.

Consider the following diagnostic case for `OldTightenRelease`:

```
Name:       #Example1
Situation:  '(and (= FailureCode Code I59)
                  (= Valve Valve5Y1 closed)
                  (= Valve Valve5Y2 open)
                  (= OilConducts OilConductTightenRelease tight)
                  (= Gear AbsorbtionGear clean)
                  (= IoStatus IoStatusIN32 logic-0) )
Diagnosis: EndController5S2atI59Defectuous
```

The causal graph for the situation depicted in this case might look like in figure 4. Once the graph is constructed, KALES proceeds to define an analogical map from the rule's situation to the new machine.

4.3.2.2. Analogical Mapping of the Candidate's Diagnostic Situation

The analogical map of the case's explanation (the causal graph) should lead to producing a new rule (case) for the new system. KALES tries to construct on the new machine a similar explanation graph to that generated in the preceding step.

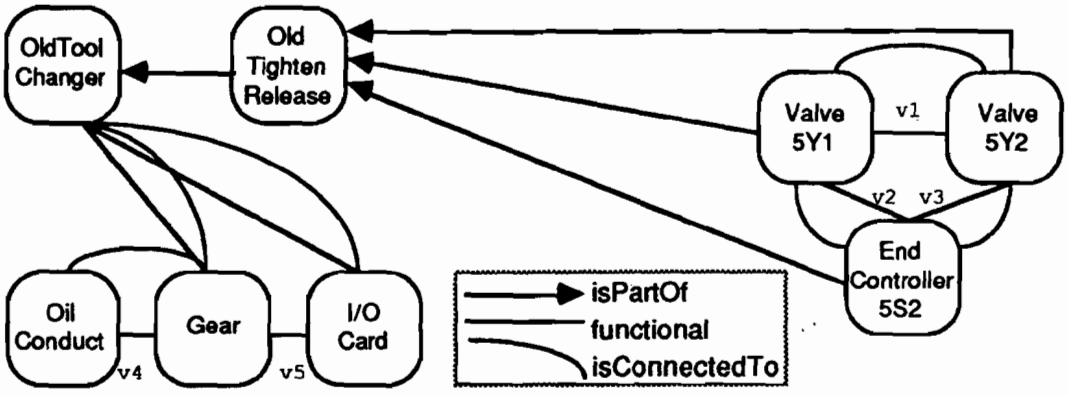
Up to this point, the knowledge adaptation process has been defined independently of the application domain. The same procedure could be applied in other domains provided that they can be modeled and simulated by iMAKE, e.g., other kinds of technical devices. However, completing the analogical process, that is, constructing the situation map, is domain-dependent. To complete the analogical bridge from the old to the new machine we need to search for a graph similar to the explanation graph generated in the preceding step, that can lead to an analogous diagnostic situation.

Deep knowledge from the domain is needed to search for the analogous explanation. We have seen how the machine failure code from the CNC helps constraining the search space to those machine parts referenced by the code. We assume that at least one symptom

in a diagnostic situation contains the failure code received from the CNC. The service technician prepares an association table of machine failure codes of both machines. For instance, if code `159` in the old machine means `ToolChanger`, then that code is associated with, e.g., code `195` corresponding to the tool changer in the new machine.

It is then possible to focus the search for the analogous situation by starting with the first symptom of the new situation being the corresponding machine failure code and its value. The procedure for completing the new situation is an establish-and-refine searching cycle in which the old rule's situation is followed step by step and for each step a new symptom for the new situation is determined.

This determination is guided by the links in the explanation graph. For each symptom processed, the algorithm determines in the new machine's model a corresponding symptom. Finally, a corresponding diagnostic hypothesis must be determined.



— Figure 4 — A Possible Explanation Graph for an Adaptation Candidate —

4.3.3. The Third Step : Incorporation of the New Case in the New Expert System

To complete the generation of the new case, a diagnostic hypothesis must be searched for the produced situation. Furthermore, the hypothesis must then be *tested* before the case can be accepted and incorporated in the new expert system.

To find an hypothesis for the situation, this is simulated through the context graph of the new expert system, by simply letting the expert system search for an appropriate context. This context's name is the new case's solution.

A failure to find an appropriate context may lead to reporting the failure to the service technician, who can either correct the situation, determine the correct context, or reject the situation. A correction of the situation causes KALES to redraw the preceding step's reasoning, by modifying the explanation graph consequently.

The consistent incorporation of the new case in the new system's case memory rests on MOLTKE's Knowledge Base Maintenance and Consistency Checking component (Maurer, 1991). Maurer's component allows the specification of knowledge dependencies on the

definitional level, that is, an object depends on another if its creation, modification, or removal depends on the existence of other objects in the knowledge base.

For knowledge adaptation, dependencies are defined on a different level, namely, on the *justification* of knowledge for its existence. The links on this level, called *justification links*, represent dependencies on a conceptual level. A diagnostic case acquired through adaptation has a justification link poiting to the explanation generated by KALES. The idea behind justification links is to give KALES a means of tracing the knowledge base to the knowledge sources of adapted rules. Thus, knowledge is naturally integrated in a dependency network in which consistency is guaranteed in a TMS-alike fashion[21].

5. STRATEGY ACQUISITION

An *adaptation case* is defined as the procedure through which an old expert system's rule is adapted for the new expert system. It is represented by a triple

$(\mathcal{P}, \mathcal{R}, \mathcal{A})$, where

$\mathcal{P}$ is a pair of parts $(\mathcal{P}.old, \mathcal{P}.new)$, $\mathcal{P}.old$ is an old machine's part and $\mathcal{P}.new$ a new machine's part; $\mathcal{R}$ is a pair of rules $(\mathcal{R}.old, \mathcal{R}.new)$, $\mathcal{R}.old$ is an old expert system's rule and $\mathcal{R}.new$ a new expert system's rule, and $\mathcal{A}$ is the analogy drawn between $\mathcal{R}.old$ and $\mathcal{R}.new$.

During the process of drawing $\mathcal{A}$, paths in the context graph are derived. This means, that for each symptom in $\mathcal{R}.old$, a symptom from the new machine is identified and added to the situation in $\mathcal{R}.new$. This becomes a sequence of symptom determinations, that can be traced on the context graph for the new machine.

KALES abstracts, from a set of adaptation cases relating the same pair of parts $\mathcal{P}.old$ and $\mathcal{P}.new$, and for the same pair of diagnostic hypotheses in $\mathcal{R}.old$ and $\mathcal{R}.new$, an *adaptation strategy*, and associates it to the corresponding diagnostic context.

Adaptation strategies can be compared to shortcut rules in MOLTKE. An empirical shortcut rule determines, with some certainty, some symptom value(s) given a set of known symptoms. An adaptation strategy determines some symptom for the new situation (during analogical transformation) given a set of already determined symptoms. The difference rests on the nature and utility of the determination. Symptom value determination by shortcut rules shortens (thereby improving) the diagnostic process. Determinations reflect the relevance of a certain symptom for detecting a given fault. Symptom determination by adaptation strategies is used to shorten the analogical transformation process. Determinations reflect here the relevance of a certain symptom for building an analogy with another diagnostic situation. An adaptation strategy is a rule that determines the next relevant symptom to include in a situation[22].

---

21 A deeper discussion on the relation between (A)TMS and the knowledge base maintenance component in MOLTKE can be found in (Maurer, 1991)

22 See (Davies and Russel, 1987) for a formal discussion on determinations, and (Dutta, 1988) for an extension of Davies and Russel's approach

6. DISCUSSION

6.1. Causation, Functionality, and Explanation

(Shoham, 1988) points out a very important property of causation, namely, its *context-sensitiveness*. The context in which a cause-effect relation occurs is important for reasoning about the differences and similarities in behavior between two structurally different, functionaly similar mechanisms. Our knowledge adaptation approach allows for expressing differences in structure and behavior between two machines through an analogical transfer of diagnostic knowledge guided by background (causal) knowledge.

However, we must restrict our view of causation to a static analysis. Our formalism still does not allow for dynamical and geometrical modelling of the domain. The analysis of a diagnosis is thus reduced to an isolated sequence of test gathering and contex refinement steps. We analyze causation in the structural and behavioral differences between two machines for isolated machine parts. We traduce *dependency associations* between machine parts features *into causal relationships*[23]. Causal relations in KALES's explanation can be seen as feature correlations of the parts involved in an adaptation case.

Using similarity in functionality is, from our view, an appropriate shortcut for the retrieval of adaptation candidates. In the next future we plan the creation of a deep classification of machine parts. IMAKE classifies parts by the expert-given class name (unfortunatelly, still a very shallow schema) and by its functionality (a part's context subgraph stored with functionality as the differentiating key). We will extend this classification in a deeper structure from which it can be possible to assign knowledge adaptation strategies (seeing this as generalization of the analogical mapping constructed by KALES) to a class or to a class's instance.

6.2. Analogy and Case-Based Reasoning for Improving Adaptation

The analogies drawn by KALES are indeed identification of correspondences on three levels. First, on the machine's structure and behavior: parts retrieved for producing an analogy satisfy some (functional) similarity criteria. Second, on the context graph of both expert systems: an analogous case can be generated only if the analogous situation can be simulated on the context graph for the new machine. Third, on empirical knowledge: the analogy describes how a fault in the new machine can be detected, based on the proceeding on a similar device. The first two levels support (explain) the latter.

Very inspiring work has been that of (Carbonell, 1986) and (Hanson, 1990). Our approach uses a kind of *derivational analogy* through the transfer of decision sequences (test selection) and their justifications (on the causal graph) to build a new diagnostic

23 See, e.g., (Geiger, Paz, and Pearl, 1990) for a discussion on the interpretation of dependencies as causal relationships

situation. Carbonell points out several relevant aspects of analogy that can be exploited by the derivation of old solution steps. The most important here is the acquisition of strategical knowledge for knowledge adaptation, discussed in chapter 5. On the other side, Hanson describes an approach to describing feature correlations as a means of supporting induction. Our approach is rather more domain-specific.

6.3. Knowledge Acquisition and Expertise Modeling

We see our workbench as an expertise modeling environment, and analyze this within our CNC-machines scenario. From the point of view of MAKE, the machine's model represents a model of the design expert's expertise on the machine's structure and behavior; the context heterarchy represents a model of the service technician and machine operator's theoretical knowledge about the diagnostic task for that machine. From the point of view of PATDEX, empirical knowledge models the diagnostic experience of a service technician or of a machine operator. From the point of view of GENRULE, empirical knowledge is a source for modeling diagnostic heuristics that can be shared by the design expert and the service technician of the machine. All of them interact and cooperate through MOLTKE with the goal of reflecting the diagnostic expertise of several experts in a CNC-machine center.

From the point of view of KALES, knowledge adaptation strategies model the expertise of a design expert, a service technician, and the knowledge engineer together in the task of knowledge adaptation. Most important, knowledge adaptation becomes an alternative source for knowledge acquisition and refinement for the new expert system.

Several approaches to expertise modeling can be found in the literature. The modeling of a domain theory guided by interaction of the theory with a domain expert is discussed in (Morik, 1987, 1988). Morik presents a system named BLIP designed to compel with the modeling task defined in a three-step cycle: domain model layout, theory extension, and evaluation. This approach presents the interesting advantage of allowing for evaluation and reversability, although evaluation itself is not deeply addressed. A similar approach to ours is Murray's *Knowledge Integration* (Murray, 1991). Murray defines the integration process in three steps: identification, elaboration, and adaptation. Identification is somewhat similar to ours. A fix knowledge representation is used, in which rules associated to a concept are separated into clusters representing its different roles. The elaboration is then carried as an expansion of values through a dependency network until conflicts between old and new information in the knowledge base are identified. After conflicts are resolved, with intervention of the domain expert, the elaboration is incorporated in the knowlede base. This last step is called adaptation. Our use of the knowledge maintenance component in MOLTKE is seen as equivalent of Murray's elaboration and incorporation steps if the justification links defined in the dependency network are complete enough to express all possible sources of inconsistency that might emerge from the insertion of a new rule in the knowledge base.

Another example is (Bardzil and Torgo, 1990). They propose integrating independent knowledge bases by selecting rules using a characterization of the competing theories

under an experimental approach: rules are tested against training case sets generated from available domain data. This is a plausible evaluation strategy for knowledge adaptation. An approach to acquiring strategic knowledge, directed specifically to diagnosis, is described in (Gruber, 1988). Gruber describes an integration of representations to allow for flexible presentation of knowledge to the domain expert and operationalization of the strategic knowledge in the expert system. The aim is giving strategic knowledge a more important role in the knowledge acquisition task. Similarly to our work, Gruber's ASK methodology uses justifications of actions to construct strateic rules.

6.3. Conclusion

Knowledge adaptation by classifying knowledge about structural and behavioral differences and defining a transformation function for rules on the basis of causal dependencies represents an innovative view on (derivational) analogical reasoning. Our approach to knowledge adaptation as a means of knowledge acquisition from past experience intensively exploits background knowledge to justify the acquired knowledge.

ACKNOWLEDGEMENTS

I thank my advisor, Prof. Dr. M.M. Richter, and all my colleagues in the AI Group on Expert Systems at Kaiserslautern, specially Klaus-Dieter Althoff, Frank Maurer, Mike Stadler, Julio Valdés, and Stephan Weß for the fruitful discussions on the topics here presented. The fruits of my work go to my wife, Yalily, for her love and dedication.

BIBLIOGRAPHY

Althoff, K.-D. (1991). Eine fallbasierte Lernkomponente als integrierter Bestandteil der MOLTKE-Werkbank zur Diagnose technischer Systeme, (in english: A Case-Based Learning Component as Integral Part of the MOLTKE Workbench for the Diagnosis of Technical Systems). Doctoral Dissertation, University of Kaiserslautern, FRG, to appear.

Althoff, K-D., De la Ossa, A., Maurer, F., Stadler, M., and Weß, S. (1989). Case-Based Reasoning for Real-World Applications. In: *Proceedings of the FAW Workshop on Adaptive Learning and Neural Networks 1989*, Ulm, Federal Republic of Germany. (Also appeared as SEKI-Report SR-90-22 SFB, Dept. of Computer Science, University of Kaiserslautern, FRG, 1989).

Althoff, K-D., Maurer, F., and Rehbold, R. (1990). Multiple Knowledge Acquisition Strategies in MOLTKE. In: B. Wielinga, J. Boose, B. Gaines, G. Schreiber, M. van Someren (Eds.), *Current Trends in Knowledge Acquisition*, Amsterdam: IOS Press (Proc. of the 4th European Knowledge Acquisiton Workshop, EKAW-90), pp. 21-40.

Althoff, K-D., and Weß, S. (1991a). Case-Based Knowledge Acquisition, Learning, and Problem Solving for Diagnostic Real World Tasks. In: *Proceedings of the fifth European Knowledge Acquisition Workshop,. EKAW-91*, Pre-print (to appear)

Althoff, K.-D., and Weß, S. (1991b). Case-Based Reasoning and Expert Systems Development. In: *Proceedings of the Knowledge Engineering and Cognition Workshop, KECOG*, University of Kaiserslautern, February, 1991 (to appear).

Althoff, K-D., and Trapphöner, R.: GENRULE (1990). Learning of Shortcut-Oriented Diagnostic Problem-Solving in the MOLTKE/3 Workbench. Technical Report, University of Kaiserslautern, Federal Republic of Germany, October 1990.

Bradzil, P.B., and Torgo, L. (1990). Knowledge Acquisition via Knowledge Integration. In: B. Wielinga, J. Boose, B. Gaines, G. Schreiber, M. van Someren (Eds.), *Current Trends in Knowledge Acquisition*, Amsterdam: IOS Press (Proceedings of the 4th European Knowledge Acquisition Workshop, EKAW-90), pp.90-104.

Carbonell, J.G. (1986). Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In: R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), *Machine Learning, An Artificial Intelligence Approach*, Vol. II, Los Altos, California: Morgan Kaufmann.

Carbonell, J.G., and Gil, Y. (1990). Learning by Experimentation: The Operator Refinement Method. In: Y. Kodratoff and R.S. Michalski, R.S. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Volume III. Los Altos, California: Morgan Kaufmann, Chap. 7.

Davies, T.R., and Russel, S.J. (1987). A Logical Approach to Reasoning by Analogy. In: J.W. Shavlik and T.G. Dietterich, *Readings in Machine Learning* (1990), San Mateo, California: Morgan Kaufmann, pp.657-663.

Dutta, S. (1988). Learning of Background Knowledge for Making Reliable Analogical Inferences. n: J.S. Gero (Ed.), *Artificial Intelligence in Engineering: Diagnosis and Learning*, Amsterdam: Elsevier, and Southampton: Computational Mechanics Publications, pp. 305-331.

Geiger, D., Paz, A., and Pearl, J.: Learning Causal Trees from Dependence Information. In: *Proceedings of the 1990 AAAI Conference*, pp.770-776.

Gruber, T.R. (1988). Acquiring Strategic Knowledge From Experts. In: J. Boose and B. Gaines (Eds.), *Knowledge-Based Systems Vol. 4: The Foundations of Knowledge Acquisition*, Cambridge: Academic Press, pp.115-133.

Hanson, S.J. (1990). Conceptual Clustering and Categorization: Bridging the Gap between Induction and Causal Models. In: Y. Kodratoff and R.S. Michalski (Eds.), *Machine Learning: An Artificial Intelligence Approach. Volume III.* Los Altos, California: Morgan Kaufmann, Chap. 9, pp.235-268.

Kolodner, J.L. (1983a). Maintaining Organization in a Synamic Long-Term Memory. In: *Cognitive Science*, Vol. 7, pp. 243-280.

Kolodner, J.L. (1983b). Reconstructive Memory: A Computer Model. In: *Cognitive Science*, Vol. 7, pp. 281-328.

Kolodner, J.L. (1989). The Mediator: Analysis of an Early Case-Based Problem Solver. In: *Cognitive Science*, Vol. 13, pp. 507-549.

Maurer, F. (1991). Knowledge Base Maintenance and Consistency Checking in MOLTKE. In: *Proceedings of the Sixth Knowledge Acquisition for Knowledge-Based Systems Workshop, KAW-91*, Banff, Canada, October 1991, to appear.

Mitchell, T.M., Utgoff, P.E., and Banerji, R.B.: Learning by Experimentation: Acquiring and Refining Problem Solving Heuristics. In: R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), *Machine Learning, An Artificial Intelligence Approach*, Berlin: Springer-Verlag, pp. 163-190.

Morik, K. (1987). Sloppy Modeling. In: K. Morik (Ed.), *Knowledge Representation and Organization in Machine Learning*, Lecture Notes in Artificial Intelligence 347, Berlin: Springer-Verlag, pp.107-134.

Morik, K. (1988). Acquiring Domain Models. In: J. Boose and B. Gaines (Eds.), *Knowledge-Based Systems Vol. 4: The Foundations of Knowledge Acquisition*, Cambridge: Academic Press, pp. 245-256.

Murray, K.S. (1991). Learning as Knowledge Integration. Ph.D. Dissertation Extended Abstract, Dept. of Computer Sciences, University of Texas at Austin, February 1991.

Rehbold, R. (1991). Die Integration von modellbasiertem Wissen in technische Diagnose-expertensystemen (in english: The Integration of Model-Based Knowledge in Technical Diagnosis Expert Systems). Doctoral Dissertation, University of Kaiserslautern, FRG, 1991.

Schreiber, A.Th., Wielinga, B.J., and Breuker, J.A. (1991). The KADS Framework for Modelling Expertise. In: *Proceedings of the fifth European Knowledge Acquisition Workshop,. EKAW-91*, Pre-print (to appear)

Shen, W.M., and Simon, H.A. (1989). Rule Creation and Rule Learning Through Environmental Exploration. In: *Proceedings of the International Joint Conferenfe on Artificial Intelligence 1989*, Morgan Kaufmann.

Shoham, Y. (1988). Reasoning About Change. Time and Causation from the Standpoint of Artificial Intelligence. Cambridge: MIT Press.

Struss, P. (1988). Extensions to ATMS-Based Diagnosis. In: J.S. Gero (Ed.), *Artificial Intelligence in Engineering: Diagnosis and Learning*, Amsterdam: Elsevier, and Southampton: Computational Mechanics Publications, pp. 3-28.