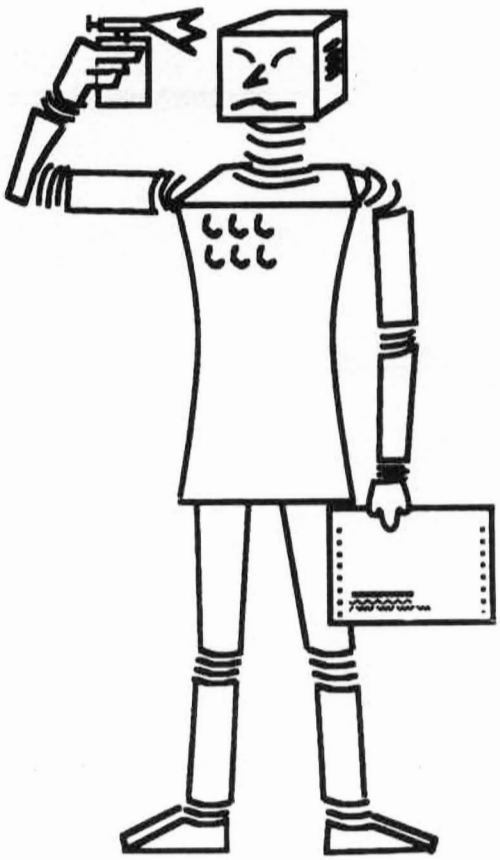
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT



GenRule: Learning of Shortcut-Oriented Diagnostic Problem Solving in the MOLTKE3 Workbench

Klaus-Dieter Althoff, Ralph Traphöner

SEKI Report SR-91-04 (SFB)

GenRule: Learning of Shortcut-Oriented Diagnostic Problem Solving in the MOLTKE$_3$ Workbench[1]

Klaus-Dieter Althoff & Ralph Traphöner
University of Kaiserslautern
Dept. of Computer Science
P.O. Box 3049, D-6750 Kaiserslautern
Germany

Abstract

GenRule is the offline processing component of the MOLTKE$_3$ workbench's learning mechanism. It learns from diagnostic cases, i.e. protocols of the diagnostic behavior of an experienced service technician. The result of a learning step are so called shortcut rules, which allow the derivation of symptom values from other already known values. Furthermore, these rules are used to direct the diagnostic strategy applied by the MOLTKE$_3$ shell. The presented mechanism appears to be well suited for modeling the typical diagnostic behavior of a service technician.

1. Introduction

GenRule[2] is part of the learning component of the MOLTKE$_3$ workbench [1]. The task of it, is the improvement of a given knowledge base. Therefore, the shortcut–oriented problem solving behavior of an experienced service technician is modeled by learning mechanism. This mechanism generates so called shortcut–rules from analogies between new presented diagnostic cases and old ones, already integrated into the knowledge base. This integration is the addition of actual learned shortcut–rules. The application of these rules during the diagnostic process shortens the session by asking the user less questions and improves the systems transparency[3] by focusing the diagnostic behavior to it.

The following chapter gives some examples from the domain of technical diagnosis which illustrates such shortcut–oriented problem solving. The third chapter provides the necessary terminology for the description of the diagnostic background within the MOLTKE$_3$ workbench. Chapter 4 gives a more precise introduction into GenRules task, while Chapter 5 and 6 describe the concrete learning mechanism. Next we present a detailed example and a short overview of other components of GenRule. Finally GenRule is evaluated on the basis of other comparable learning systems.

2. Motivation

Because of his experience, a service technician is able to do fast diagnostics, to avoid wrong diagnoses, to draw conclusions from wrong diagnoses and to reapply successful diagnostic behavior. His ability to do fast diagnostics is based on the art of shortening a solution. This ability is illustrated by the following example:

---

[1] The work presented herein was partially supported by the Deutsche Forschungsgemeinschaft, Sonderforschungsbereich 314: "Artificial Intelligence- Knowledge-Based Systems", projects X6 and X9.

[2] Generator of empirical MOLTKE Rules

[3] *transparent* in the sense more appropriate for validation by the expert

Suppose a machine located in a high temperature area. Also let us suppose, that this high temperature affects 90% of all defects of this machine, located in a special i/o–card. The experienced service technician knowing this machine, recognizes this defect by detecting only a small part of the complete symptomatic. So he shortens the causal necessary path of the solution by applying his special experience-based heuristic.

Another similar scenario is the often found characteristic, that different manufactured series of the same machine type have specific frequent defects. With growing experience, the service technician is able to make use of this knowledge.

Todays expert systems have not such knowledge based on growing experience to their disposal. On the opposite, an experienced service technician knows a lot examples (better called cases) of specific malfunctions. The following example gives such cases from the field of diagnostics of CNC machining centers (fig. 1 and 2).

error code	i59	⎫	**symptoms being ascertained in this order**
i/o-state IN36	logical-1	⎪	
i/o-state OUT7	logical-1	⎪	
valve 5Y1	switched	⎪	
valve 5Y2	not switched	⎪	
piping system	okay	⎪	
chuck	clean	⎪	
i/o-state IN32	logical-1	⎭	
i/o-card	defect	⎫	**diagnosis**

Fig. 1 – A "Case"[1] out of the MOLTKE knowledge base

error code	i59	⎫	**symptoms being ascertained in this order**
i/o-state OUT7	logical-1	⎪	
valve 5Y1	switched	⎪	
valve 5Y2	not switched	⎪	
piping system	okay	⎪	
chuck	clean	⎪	
i/o-state IN32	logical-1	⎭	
i/o-card	defect	⎫	**diagnosis**

Fig. 2 – The Service Technicians's Case

[1] Later we will call these cases "diagnostic paths".

It is easily seen, that the service technician did not check the symptom i/o–state IN36 with value logical–1 (fig. 2). It seems to be not necessary. This behavior can be interpreted in two different ways:

- The service technician supposes the symptom to be irrelevant.
- The relevance of the remaining symptoms is supposed to be strong enough to justify an analogical derivation of the missing symptom.

So we have to possible objectives:

- The missing symptom is causal necessary, but the technician supposes it to be spare in most situations.
- The missing symptom is causal unnecessary.

In both cases it can be seen as "partial determined" by the remaining symptoms. It is possible to improve the degree of determination by learning from additional cases. Commonly a technician learns in two ways:

- He learns from specific past cases by adapting their diagnostic to an actual diagnostic behavior.
- He learns from past cases in a whole by improving his general diagnostic behavior.

3. Concept Formation and Knowledge Representation

In the MOLTKE$_3$ workbench's framework, the empirical knowledge of a service technician relevant for diagnosis is modeled and made explicit with the aid of diagnostic cases, which are defined as follows:

Definition 1: Diagnostic Case
A *diagnostic case* (or simply *case*) d is given as a triple (Name(d); Sit(d); D(d)), consisting of the name of the case Name(d), its situation Sit(d), i.e. a set of symptom values, and the corresponding diagnosis D(d).

The aim in MOLTKE is to improve the diagnostic process in the same way as motivated above for the service technician. Moreover, MOLTKE should reflect the service technician's proceeding through an "experience-based" approach. For reaching this to different approaches are combined (fig. 3).

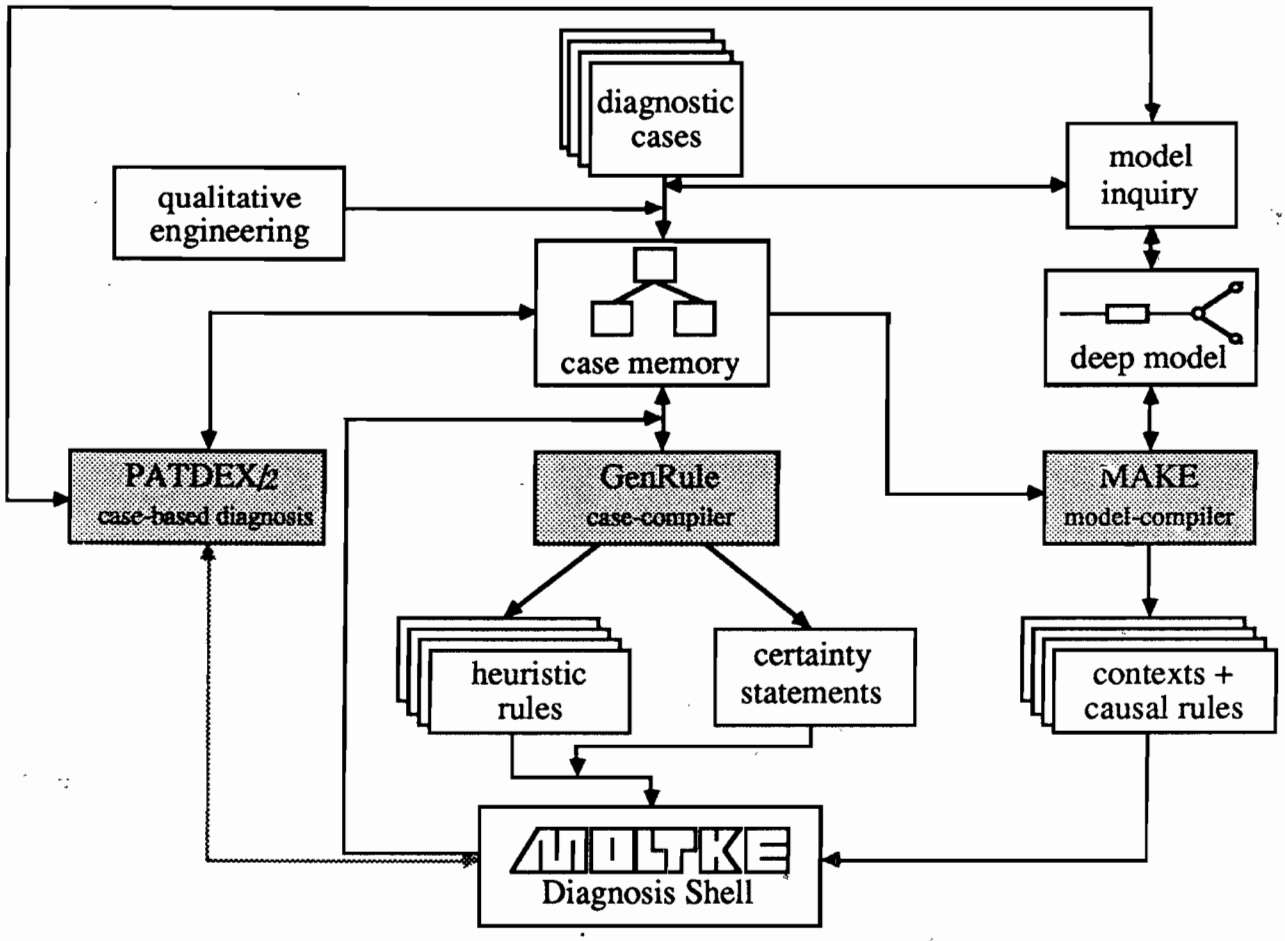


Fig. 3 – The Learning Component in the MOLTKE$_3$ Workbench

- the GenRule system learns from analogies between newly presented diagnostic cases and those already integrated in the knowledge base. These analogical inferences are compiled in "partial shortcut rules" that improve the diagnostic process as motivated above.
- The PATDEX$_2$[1] system carries out case-based reasoning directly on the case base. This is important both for the treatment of exception cases and for supporting KA[2] (direct interpretation of the diagnostic cases, further acquisition of diagnostic cases, etc.).

The main differing feature of the two systems lies in that GenRule applies analogical reasoning offline for the improvement of the knowledge base, while PATDEX$_2$ uses it online as the actual problem solving mechanism.

The subject of this paper is the GenRule system. In relation to the other components of the workbench, it is referred to [1], [2], [3], and [4].

For further comprehension it results indispensable to give a short overview of the used terminology and an introduction in the MOLTKE shell's underlying diagnostic procedure.

---

[1] PATtern Directed EXpert system

[2] knowledge acquisition

3.1. Description Language for Technical Diagnosis

A *symptom* describes a measurable part of the state of the system to diagnose. A symptom has a name, a value range, and a related symptom variable. An actually measured value of a symptom is called a *symptom value*. The default value of a symptom is *unknown*. The set of all symptoms with their corresponding values is called a *situation*. Symptom values are ascertained using *tests* and serve to characterize a *diagnosis*. Formally, a diagnosis is described by *diagnostic formulas*, which are formulas in the first-order predicate logic with the following properties:

- Ground terms are the (possible) symptom values
- Variables are the symptom variables
- Atomic formulas are of the form afb, where a and b are symptom values or symptom variables and f is a comparison operator between symptom values
- Formulas are boolean combinations of atomic formulas
- There are three truth values: TRUE, FALSE, and UNKNOWN
- An atomic formula afb has, for example, the truth value UNKNOWN, if a or b is a symptom variable whose value is unknown in the current situation

An example of a diagnostic formula is:

$$(\text{Switch}_1 = \text{on}) \wedge (\text{Switch}_2 = \text{off}) \wedge (\text{OilPressure} \leq 100)$$

The symptom variables in the formula are Switch_1, Switch_2 and OilPressure; the values "on", "off", and "100" must lie in the corresponding value ranges. Switch_1 and Switch_2 are indeed concrete switches (and "OilPressure" a particular oil pressure). Their positions, however, are first determined by the current situation. Therefore, here they are variables that get their values assigned by an actual situation.

The refinement diagnostic hierarchy is modeled by a graph, the *context graph*. *Contexts* represent rough, intermediate, and final diagnoses. The diagnostic formula characterizing a context is called *context precondition*. In addition, two kinds of rules are associated to a context: *ordering* and *shortcut rules*. The condition part of these rules are diagnostic formulas. The action part consists either of the proposition of a test, in case of a firing ordering rule, or the assignment of a value to a symptom, in case of a shortcut rule.

Because the rules learned by GenRule become integrated in the MOLTKE shell, a short summary of the diagnostic procedure used is necessary.

3.2. Diagnostic Procedure in MOLTKE

Basically, the diagnostic task is conceived as classification plus test selection. The MOLTKE shell's global interpreter starts with the root node of the context graph ("machine defective"); it proposes tests with the aid of the available ordering rules and uses the gathered symptom values to derive further symptom values through shortcut rules. As soon as context preconditions are satisfied, MOLTKE selects the most special context and applies again the same procedure, until a final diagnosis is reached. Final diagnoses are represented by the leaves of the context graph. The classification is effected in this manner using the context preconditions, the test selection with the use of the ordering rules. The latter is merely the standard procedure, which is extended in consideration of the shortcut rules in the sense of shortcut-oriented diagnostic problem solving.

3.3. Influence of GenRule on the Diagnostic Procedure

The most important ways in which GenRule can influence the diagnostic procedure of the shell are the following:

- Symptom values derived by shortcut rules improve the classification process, either by accellerating it as less symptoms are to be collected, or by making it (at least) more transparent because they divide it in the same way the service technician would.
- The shell blocks all tests that determine symptoms for which a value was already derived with a shortcut rule.
- Shortcut rules are useful for focussing the diagnostic procedure of the shell, which tries to fire the most rules possible. This extends the test selection over ordering rules.

The shell processes uncertain information made available by shortcut rules by using a priori and a posteriori estimations. The underlying mechanism was decisively determined by two substantial requirements from the domain of CNC machines:

- MOLTKE must be able to process large knowledge quantities with sufficiently good performance.
- Uncertain knowledge must be used accordingly to the situation, that is, it must be possible to relate the uncertainty of a diagnosis to the effort required for the (practical) validation of the results[1].

The consequences of the MOLTKE workbench are the following ones:

- extensive transparency of the uncertain portions of the diagnosis,
- simple representation and processing of uncertainty,
- well-aimed validation possibilities,
- judgement of uncertain diagnoses by the user.

4. Task Specification for GenRule

GenRule should improve the diagnostic procedure, with the aid of diagnostic cases, in the way motivated in chapter 2. GenRule should improve a given knowledge base, which can be produced by MAKE[2] and/or manually. For the shortcut rules that shall be learned, there is an additional interpreter that, whenever possible, uses shortcut rule firing as its main strategy. Ordering rules are only used when they are explicitly given by the expert.

GenRule is not thought of as an interactive component; that task is undertaken by the $PATDEX_2$ system. The direct shortening of context preconditions through shortcut rules learned by GenRule would make no sense, because the shortcut rules are naturally uncertain and the classification ability of the MOLTKE shell may not be deteriorated. The processing of uncertain knowledge by the global interpreter of the shell will be more concretely discussed in the sequel of this paper.

5. Functional Description of GenRule

GenRule learns from analogies between already integrated[3] and newly presented diagnostic cases to derive symptom values from other given ones. The goal is to improve the current diagnostic

---

[1] i.e., the user decides whether she accepts the diagnosis or verifies all or a subset of uncertain symptom values and leaves the shell classify again on the basis of eventually corrected values. In this decision come into account the effort required by the repair action, the effort required to collect a symptom, and the experience of the user.

[2] the model-compiler of the MOLTKE/3 workbench which generates contexts and causal rules out of the deep model of the technical system (fig. 3)

[3] From now on we will call *diagnostic paths* to the cases already integrated in the knowledge base, because one can think of them as a diagnostic course developed in the MOLTKE shell.

procedure, so that the least symptoms have to be gathered. The MOLTKE knowledge base undertakes the role of the background theory, which is to be seen as redundant and/or not transparent. That means that either frequently too many symptoms are ascertained or they are gathered in an order which is difficult to duplicate. GenRule generates partial, i.e. heuristic, shortcut rules that remove present redundancies and give the shell the possibility of applying the same shortcut-oriented diagnostic procedure as motivated in chapter 2.

GenRule takes as examples diagnostic cases from the respective service technician. From a diagnostic case it is possible to "learn" something with the help of an analogy, if it helps to shorten the diagnostic process. The given diagnostic case is compared with the "most similar" diagnostic path. The diagnostic path is similar to the diagnostic case if it has the same diagnosis and contains all symptom values of the case. The shorter the diagnostic path is, the more similar it is to the diagnostic case. The respectively shortest diagnostic paths are called "minimal". On the basis of the analogy modeled by this similarity measure, GenRule tries to "learn symptom values". For that, the given diagnostic path must be "properly shorter" as the diagnostic path compared with it, that is, the diagnostic path must still possess, besides the symptom values of the case, further values.

If all of this is satisfied, the diagnostic case is a positive example. All symptom values contained in the diagnostic path but not in the diagnostic case will be "learned". The learning result is a partial shortcut rule for each "learned" symptom value. The left side of such a rule is constructed on the basis of the situation of the given case. The right side is the respective "learned" symptom value of the most similar diagnostic path.

If the diagnostic case is a negative example, that is, the case is no shortcut, GenRules uses the case merely to update the certainty factors of the already learned shortcut rules.

The interesting specification degree of the learning procedure is the derivation of symptom values and/or the exclusion of tests for symptoms that appear in the most similar diagnostic path.

The results of the learning are applied into the MOLTKE shell through the integration of the learned partial shortcut rules. They are added to the rule set of the respective context. The more these rules conduce to correct shortcuts and the more the way in which they are gained can be duplicated, the more these rules improve the shell's diagnostic procedure. GenRule only gives shortcut rules to the shell when their certainty factor is not under a certain threshold value. That means, that the statistical information accumulated in the certainty factor serves to the evaluation of the rules learned by GenRule. Moreover, the representation of the learning results in form of shortcut rules makes possible a simple retraction of incorrect shortcuts by the user, because the classification ability of the shell over the context preconditions remains untouched.

In order to make the generated shortcut rules as applicable as possible, GenRule computes three different certainty factors that describe the certainty of the proposition of a shortcut rule in dependence with varying basic commonalities of diagnostic cases. These commonalities depend directly on the assumptions regarding the diagnosis reached in a diagnostic session. If no assumptions referring to this are made, the estimation is very cautious (poor). If on the other side it is presupposed that this is the same diagnosis D_i underlying during rule generation, then the estimation is very optimistic because this will not be frequently the case. With the exception of these certainty factors for both basic commonalities, the factors for all other diagnoses will be determined with linear interpolation[1]. All factors are computed on the basis of definition 2.

---

[1] This has sufficient precision and makes possible an efficient computation of the factors [4].

With the purpose of making the processing of uncertain knowledge as simple and transparent as possible, the shell computes during run time in proper sense no certainty factors. The user determines the maximum allowed uncertainty for the shortcut rules attained in the shell for application. Moreover, the user determines how many partial shortcut rules may be fired in a diagnostic session and/or how many (uncertain) symptom values derived by partial shortcut rules may be used to fire (total[1] or partial) shortcut rules.

When the diagnosis is given as output, the user is informed that uncertain symptom values have been used to establish it. She can then decide to accept the given diagnosis, to verify the uncertain symptom values, to change symptom values, or to input further symptom values. The shell can carry on very efficiently all of the actions here mentioned with the aid of a procedure similar to OPS5's Rete-Algorithm.

6. Architecture of GenRule

The idea of a shortcut rule follows directly from the expert's behavior which has been described in the motivation chapter, namely the ability of directly infering one situation from another without carrying out a test. This ability is based on empirical knowledge which is implicitly included in the expert's diagnostic cases. In this connection shortcut rules are operationalized empirical knowledge. They are justified by second order relations (*partial shortcuts*) which are defined using a frequency interpretation based on all known diagnostic cases. They are a replacement for conditional probabilities[2]. In this connection their role is similar to that of certainty factors for probabilities where the underlying distribution function is not known, too.

Definition 2: Partial Shortcuts (SC_{FC_D})
Let **DC** be the set of all diagnostic cases. Then the *set of partial shortcuts for the failure context FC_D of the diagnosis D* SC_{FC_D} is defined as follows:

$$SC_{FC_D} := \{(FO_1 \}^{\delta} FO_2) \mid FO_1, FO_2 \text{ diagnostic formulas, } FO_2 \text{ atomic}\},$$

where $\}^{\delta}$ is defined (justified) via *determination factor* δ:

$$\delta := \frac{|\{d \in \mathbf{DC};\ FO_1(d) \wedge FO_2(d)\ \}|}{|\{d \in \mathbf{DC};\ FO_1(d)\ \}|}$$

Failure contexts are diagnoses that are not known by name. Instead of this they can be summarized by more general diagnoses which directly correspond to those subparts of the underlying technical system. These subparts are both candidates for being exchanged during a correction and causes for for different failures. To belong to the same failure context (of a diagnosis D) a diagnostic case and a diagnostic path both have to include the diagnosis D. Additionally, contradictory symptom values within their situations are not allowed and the situations must have a minimum number of values in common.

---

[1] Total shortcut rules can be gained with MAKE from the deep model of the system to diagnose. *Total* means here that these rules are certain (in relation to the model).

[2] for lack of more precise information

We now formally introduce when a presented diagnostic case is *(properly) shorter* than an already existing diagnostic path:

Definition 3: Partial Orders on Diagnostic Cases and Paths ($\leq_c$, $<_c$)
Let d_1 and d_2 be arbitrary diagnostic cases and paths for the failure context FC_D of the diagnosis D. Then the (partial) orders $\leq_c$, $<_c$ are defined as follows:

$d_1 \leq_c d_2 :\Leftrightarrow Sit(d_1) \subseteq Sit(d_2)$ ("d_1 is called to be *shorter* than d_2")

$d_1 <_c d_2 :\Leftrightarrow Sit(d_1) \subset Sit(d_2)$ ("d_1 is called to be *properly shorter* than d_2")

Now we can describe what it means for a diagnostic case and a diagnostic path to be similar and when a certain path is more similar to a case than an alternative path.

Definition 4: Similar Diagnostic Paths, Similarity Measure ($\mathbf{SDP_d}$, SM_d)
Let d_1 be an arbitrary diagnostic case and d_2 an arbitrary diagnostic path. d_2 is called *similar to* d_1, if and only if:

- $D(d_1) = D(d_2) = D$,
- $FC_D(d_1) = FC_D(d_2)$,
- $d_1 \leq_c d_2$.

The set of all diagnostic paths being similar to d_1 is denoted by $\mathbf{SDP_{d_1}}$.
The similarity measure SM_d on $\mathbf{SDP_d}$ for an arbitrary diagnostic case d is defined as follows:

- $SM_d: d_i \in \mathbf{SDP_d} \mapsto SM_d(d_i) = |Sit(d_i)|$.

d_i is called *more similar to d then* d_k, if:

- $SM_d(d_i) < SM_d(d_k)$ mit $d_i, d_k \in \mathbf{SDP_d}$.

The shorter a similar diagnostic case is the more similar it is to the corresponding diagnostic case. This interpretation of "more similar" is a direct consequence from the learning of meaningful shortcuts.

Let´s consider an arbitrary diagnostic case d: then the most similar diagnostic path to d is that one which is the minimal element concerning the partial order $<_c$ (or $\leq_c$) for $\mathbf{SDP_d}$. Such paths are called "minimal".

The following definition describes the essential learning step of GenRule: the learning of partial shortcuts.

Definition 5: Learning of Partial Shortcuts (AGPAR)
Let d_1 be an arbitrary diagnostic case and d_2 d_1´s similar minimal diagnostic path (i.e. d_2 is the most similar path for d_1). If d_1 is properly shorter than d_2, then GenRule *learns* the following partial shortcut(s) psc_i:

$$\{psc_i\} := \{ (FO[Sit(d_1)] \xrightarrow{SC}{}^{\delta} FO[x_{S_i}]) ; \quad x_{S_i} \in (Sit(d_2) \setminus Sit(d_1)), i = 1..n, \; n := |Sit(d_2) \setminus Sit(d_1)| \}.$$

$FO[Sit]$ and $FO[x_{S_i}]$ denote the representation of a situation Sit as a diagnostic formula. For achieving this transformation every known symptom value is mapped onto an atomic formula which consists of that value, the "=" operator and the respective symptom variable.
The detailed procedure including both the trivial transformation of an shortcut into a shortcut rule and the computation of the determination factors is described in the next section.

6.1. Algorithm for the Generation of Partial Shortcut Rules (AGPAR)

Input:

- Set C_{all} of diagnostic cases with diagnoses D_i, i = 1..n
- Set **DC** of all known diagnostic cases
- Set $DP_{min} := \{dp_{min}(D_i; k) \mid i = 1..n, k \in \{1, 2, ..., \#FC_{D_i}{}^{1}\}\}$ of all minimal diagnostic paths of all failure contexts of the diagnoses D_1 to D_n
- Degree **N** for defining the minimal number symptom values which d_{actual} and dp_{min} must have in common to belong tom the same failure context

Output:

- Set AR of all learned shortcut rules

Algorithm:

AR := Ø;
For all $d \in C_{all}$ do
  $d_{actual} := d$;
  $D_{actual} := D(d)$;
  For all $k \in \{1, 2, ..., \#FC_{D_{actual}}\}$ do
    $dp_{min} := dp_{min}(D_{actual}; k)$;
    If $(d_{actual} <_c dp_{min}) \wedge (|Sit(d_{actual})| \geq N * |Sit(dp_{min})|)$
    then AR := AR $\cup$
      $\{(FO[Sit(d_{actual})] \xrightarrow{SR}{}^{\delta} (Sit(dp_{min}) \setminus Sit(d_{actual}))\}$;
      Compute determination factor δ;
    endif
  endfor
endfor

The determination factor δ is computed as follows:

$$\delta := \frac{|\{d \in \mathbf{DC};\ D(d) = D_{actual},\ dp_{min} \leq_c d\}| + 1}{|\{d \in \mathbf{DC};\ D(d) \text{ arbitrary},\ d_{actual} \leq_c d\}| + 1}$$

7. Example

Now we give a simple example for AGPAR using the cases of fig.1 and 2. **c1** is the service technician´s case and **p1** is the (corresponding) minimal diagnostic path. Additionally, the following inputs are given: $C_{all} := \{\mathbf{c1}\} =: \mathbf{DC}$, $DP_{min} := \{\mathbf{p1}\}$, and $\mathbf{N} := 3$.

For AGPAR being able to generate a partial shorcut rule three conditions must be fulfilled:

---

[1] number of different failure contexts of the respective actual diagnosis D_i.

- **p1** must be similar to **c1**,
- **p1** must be minimal,
- **c1** must be properly shorter than **p1**: $\mathbf{c1} <_c \mathbf{p1}$.

The conditions for similarity are given in definition 4:

- $D(\mathbf{c1}) = D(\mathbf{p1})$,
- $FC_{D(\mathbf{c1})}(\mathbf{c1}) = FC_{D(\mathbf{c1})}(\mathbf{p1})$,
- $\mathbf{c1} \leq_c \mathbf{p1}$.

The first and the third condition are obviously fulfilled. For **c1** and **p1** being in the same failure context the following requirements must be met:

- **c1** must not include a symptom with a certain value (≠ unknown) which has a different value (≠ unknown) in **p1** √
- $3 * |Sit(\mathbf{c1})| \geq |Sit(\mathbf{p1})| \quad \Leftrightarrow$
  $3 * 7 \geq 8$ √

Thus, **c1** und **p1** are similar. As there exists only one diagnostic path in this example it is, of course, a minimal one. Additionally, **c1** is properly shorter than **p1**. Therefore AGPAR can generate a partial shortcut rule:

AR = {((?CODE[1]=i59 ∧
?OUT7=logical-1 ∧
?5Y1=switched ∧
?5Y2=notSwitched ∧
?PIPES=okay ∧
?CHUCK=clean ∧
?IN32=logical-1)
$\xrightarrow{SR, \delta}$
(?IN36 ← logical-1))}.

The corresponding determination factor is computed as follows:

$$\delta = \frac{|\{d \in \mathbf{DC};\ D(d) = \text{i/o-card},\ \mathbf{p1} \leq_c d\}| + 1}{|\{d \in \mathbf{DC};\ D(d) = \text{i/o-card},\ \mathbf{c1} \leq_c d\}| + 1}$$

$$= \frac{|\emptyset| + 1}{|\{\mathbf{c1}\}| + 1} = \underline{\mathbf{1/2}}$$

8. Further Components of GenRule

GenRule uses a conceptual memory for the representation and processing of the diagnostic cases. The memory enables an efficient computation of the determination factors. Initial diagnostic cases can be automatically generated out of a given knowledge base. The generated shortcut rules are made "more

---

[1] error code

applicable" by restricting the basic commonality of cases and by generalizing the conditions of the rules. Shortcuts having a very small determination factor are interpreted as irrelevant knowledge and treated in a special way. For a detailed description of these components and mechanisms see [4].

9. Discussion and Evaluation

We give a short evalution of the GenRule approach and state its relation to other ones. Finally, we describe the state of implementation of GenRule and all affected components of the MOLTKE$_3$ workbench (Oct. 1990).

9.1. Evaluation

GenRule can not be evaluated independent of the MOLTKE$_3$ workbench. It is not expected that GenRule can carry out "arbitrary" learning tasks within a technical diagnosis situation. Whereas the characteristics for the attractiveness of the GenRule approach are especially its specilization, its supplementation with the case-based problem solver PATDEX$_2$, and the integration of the learning results together with the results of the manual and (deep-)model-based KA within the shell.

The MOLTKE shell is the result of a several years' cooperation with a globally acknowledged mechanical engineering institute. It meets all the requirements that have been posted by the institute [5]. The shell is also based on preliminary (theoretical) studies [6] on the modeling of diagnostic problem solving which have been adapted for our practical needs [7].

This all has resulted in the MOLTKE$_3$ workbench. The original application (diagnosis of CNC machining centers, cf. [8]) has been used as a natural medium for the integration of different KA and ML[1] mechanisms[2]. One fundamental result of the cooperation is that it is sensible to routinely develop a knowledge base and to adapt it to "real situations", later, by the use of additional knowledge. Therefore shortcut rules have already been introduced by the engineer who has been engaged in our project. During further project work it turns out that shortcuts basically influence the diagnostic strategy of the respective experts. Thus, the learning of shortcut rules offers a natural possibility to integrate manual KA and ML. The starting point for GenRule is a redundant and not-transparent knowledge base which has been generated by MAKE and/or developed by a knowledge engineer.

For the considered CNC machine about 5.000 failures (and some more symptoms) can be identified. Because of the decomposability of the domain the design of a conceptual memory has been possible which enables the efficient computation of the determination factors. As the shell has been strongly modularized using the context graph the diagnostic paths can be automatically generated out of a given knowledge base. For a detailed estimation of the complexity of the involved algorithms see [4].

Up to now three different knowledge bases for complex technical systems have been successfully implemented using the MOLTKE workbench, a fourth is currently under development. GenRule has been designed for the diagnosis of the MC600 of the Maho company. The usefulness of GenRule has been validated during the implementation of the knowledge base for the driving machine in mining PAURAT-E200. One important result was that it is sensible to combine shortcut-oriented problem solving with manually edited ordering rules.

The quintessence of all the projects being realized so far is that the shell can be considered as successful. GenRule completes this shell in a natural way. Additionally, the applicability of GenRule

---

[1] machine learning

[2] case- and analogy-based as well as adaptive learning

is increased by first the fact that the shell's classification ability is not affected but only the order of tests, and second the possibility that the diagnostic procedure can terminate "earlier" if the user accepts this (but need not, of course). GenRule can be applied to other domains (of technical diagnosis) if it is similar decomposable as that of CNC machining centers (which is typical for comparable technical systems).

We must point out that a sensible evaluation of a workbench for real, complex applications like MOLTKE is extremely difficult and still a hot research topic. Naturally, this is also the case for a subcomponent like GenRule. Sufficient evidence can only be collected via product development and "commercial" application.

Thus, in this connection it must be taken into account that the utilization of ML tools depends on both the comfort of the user interface and the reusability of their mechanisms (e.g. in the context of an overall application).

9.2. Related Work

As it is sufficient for our purposes here we restrict our discussion concerning GenRule's relations to other approaches to the aspects of *analogy*, *knowledge base refinement*, and *knowledge integration*.

GenRule learns rules for a diagnossis shell based on a comparison between presented cases and in the knowledge base already available (compiled) knowledge. The learning procedure aims at "improving" the knowledge base, i.e. to shorten the diagnostic procedure and/or to make it more transparent.

Because of its inductive character and the explicit representation of the learning results GenRule can be viewed as an EBL-system. In [1] this has been motivated in more detail. Inspite of this, a comparison with Kedar-Cabelli's "Purpose-directed Analogy" [10] seems to be more helpful, because both two cases are compared with one another (whereas only "one" case is considered in an EBL-system) and *purpose* plays a fundamental role in MOLTKE's whole learning component (e.g. during the case-based KA and treatment of exceptions).

All analogical inferences within the learning component are based on similarity measures[1]. Therefore they can be carried out automatically and need no "analogical hints". To a high degree GenRule bases its analogical inferences on dissimilarities. Via the definition of the failure contexts contradictory situations are explicitly excluded and not considered further. The importance of dissimilarities for analogy has already been pointed out by [11].

Kodratoff [12] introduces two dimensions for the evalution of analogies: complexity and relevance. In his terminology GenRule's analogies are "good" analogies of "full" complexity. As the causality between the symptom values (situations) and the respective diagnosis is closely interweaved with the defined notion of similarity the complexity is not "trivial". Similarity and causality also can't be "simply" combined and then applied to an arbitrary target. The degree of generality is the same for cases as for diagnostic paths. Overgeneralizations with GenRule are possible, but only in a restricted way. Especially, the shell's treatment of such shortcut rules is comparatively uncritical, because its classification ability is not affected.

For the field of knowledge base refinement we vicariously mention the systems KRUST [13] and INDE+ [14]. These systems try to improve the classification ability of their knowledge base which is a significant difference to GenRule. The MOLTKE shell acquires its classification ability from the deep model of the technical system which can be completed via manual KA. GenRule's novel view of

---

[1] GenRule's similarity measure is a kind of meta criterion for the preparation of appropriate diagnostic paths.

knowledge base refinement is enabled by the view of diagnostic problem solving within the MOLTKE project:

Diagnosis = Classification + Test Selection.

Important systems within the field of knowledge integration are BLIP [15] and DISCIPLE [16]. The latter integrates different learning strategies for the support of the KA process. BLIP considers the KA task as modeling a domain. ML is then viewed as automatic modeling which enables a natural way of integration of both fields. The MOLTKE$_3$ workbench bases on the same view.

An (excellent) overview over the problematic nature of knowledge integration is given in [17]. Their results and integration proposals impressively underline the quality of the MOLTKE$_3$ workbench. Thus, it can be seen as the state of the art in knowledge integration, with the restriction to the domain of technical diagnosis. In their final discussion concerning the applicability of ML to KA the authors give prominence to the refinement of explicit knowledge bases using cases, i.e. exactly to the task being covered by GenRule.

9.3. State of Implementation (Oct. 1990)

GenRule, as it is described here, is fully implemented and integrated into the MOLTKE$_3$ workbench[1]. The implementation of PATDEX$_2$ will be finished soon [3]. A stand-alone prototype is already available since early 1989 [18]. The MOLTKE shell and the MAKE[2] system are fully implemented, too.

10. Acknowledgement

Thanks go to Frank Maurer and our ML research group at Kaiserslautern for many very engaged discussions which have been an important step for the development of GenRule as it is described here. The essential parts of it have been implemented by Wolfgang Wernicke and Stephan Boecher.

11. References

[1] Althoff, K.-D., Maurer, F., Rehbold, R.: Multiple KA Strategies in MOLTKE, in: [22], pp. 21-40

[2] Rehbold, R.: Integration modellbasierten Wissens in technische Diagnostikexpertensysteme, doctoral dissertation, University of Kaiserslautern, 1991 (forthcoming)

[3] Weß, S.: PATDEX2: ein System zum adaptiven, fallfokussierenden Lernen in technischen Diagnosesituationen, diploma thesis, University of Kaiserslautern, 1990

[4] Althoff, K.-D.: Eine fallbasierte Lernkomponente als integrierter Bestandteil der MOLTKE3-Werkbank zur Diagnose technischer Systeme, (forthcoming)

[5] Richter, M. M., Pfeifer, T., Althoff, K.-D., Faupel, B., Nökel, K., Rehbold, R.: final report project X6, Sonderforschungsbereich "Artificial Intelligence - Knowledge-Based Systems", Kaiserslautern: 1990

[6] Wetter, T.: Ein modallogisch beschriebenes Expertensystem, ausgeführt am Beispiel von Ohrenerkrankungen, doctoral dissertation, Technical University of Aachen, 1984

[7] Maurer, F.: FOMEX: ein fehlerorientiertes, modulares Expertensystem zur Diagnose von CNC-Bearbeitungszentren, diploma thesis, University of Kaiserslautern, 1989

---

1 [19], [9]

2 [20]

[8] Althoff, K.-D., Faupel, B., Kockskämper, S., Traphöner, R., Wernicke, W.: KA in the Domain of CNC Machining Centers: the MOLTKE Approach, in: Proc. EKAW 1989, pp. 180-195

[9] Boecher, S.: Integration automatischer Wissensakquisitonswerkzeuge für MOLTKE am Beispiel einer Wissensbasis für Vortriebsmaschinen, diploma thesis, University of Kaiserslautern, 1990

[10] Kedar-Cabelli, S.: Toward a Computational Model of Purpose-Directed Analogy, in: (A. Prieditis (ed.): Analogica, London: Pitman, 1988), pp. 89-107

[11] Vrain, C. , Kodratoff, Y.: The Use of Analogy in Incremental SBL, in: [21], pp. 231-246

[12] Kodratoff, Y.: Combining Similarity and Causality in Creative Analogy, in: Proc. ECAI 1990, pp. 398-403

[13] Craw, S., Sleeman, D.: Automating the Refinement of Knowledge-Based Systems, in: Proc. ECAI 1990, pp. 167-172

[14] Aben, M., van Someren, M.: Heuristic Refinement of Logic Programs, in: Proc. ECAI 1990, pp. 7-12

[15] Morik, K.: Sloppy Modeling, in: [21], pp. 107-134

[16] Kodratoff, Y., Tecuci , G.: The Central Role of Explanations in DISCIPLE, in: [21], pp. 135-147

[17] van Someren, M. W., Zheng, L. L., Post, W.: Cases, Models or Compiled Knowledge; a Comparative Analysis and Proposed Integration, in: [22], pp. 339-355

[18] Althoff, K.-D., De la Ossa, A., Maurer, F., Stadler, M., Weß, S.: Adaptive Learning in the Domain of Technical Diagnosis, Proc. Workshop on Adaptive Learning, FAW Ulm, 1989

[19] Wernicke, W.: Ein System zur Verarbeitung von Erfahrungswissen in MOLTKE, diploma thesis, University of Kaiserslautern, 1990

[20] Rehbold, R.: Model-Based KA from Structure Descriptions in a Technical Diagnosis Domain, Proc. Avignon 1989

[21] Morik, K. (ed.): Knowledge Representation and Organization in ML, Springer Verlag: Berlin, 1989

[22] Wielinga, B., Boose, J., Gaines, B., Schreiber, G., van Someren, M. W. (eds.): Current Trends in KA, Proc. EKAW90, IOS Amsterdam, 1990