Case-Based Knowledge Acquisition, Learning and Problem Solving for Diagnostic Real World Tasks[1]

Klaus-Dieter Althoff & Stefan Weß

University of Kaiserslautern
Dept. of Computer Science
P.O. Box 3049, D-6750 Kaiserslautern
Federal Republic of Germany
e-mail: althoff(wess)@informatik.uni-kl.de

Abstract

Within this paper we focus on both the solution of real, complex problems using expert system technology and the acquisition of the necessary knowledge from a case-based reasoning point of view. The development of systems which can be applied to real world problems has to meet certain requirements. E.g., all available information sources have to be identified and utilized. Normally, this involves different types of knowledge for which several knowledge representation schemes are needed, because no scheme is equally natural for all sources. Facing empirical knowledge it is important to complement the use of manually compiled, statistic and otherwise induced knowledge by the exploitation of the intuitive understandability of case-based mechanisms. Thus, an integration of case-based and alternative knowledge acquisition and problem solving mechanisms is necessary. For this, the basis is to define the "role" which case-based inference can "play" within a knowledge acquisition workbench. We will discuss a concrete case-based architecture, which has been applied to technical diagnosis problems, and its integration into a knowledge acquisition workbench which includes compiled knowledge and explicit deep models, additionally.

1 Introduction

Within this paper we focus on both the solution of real, complex problems using expert system technology and the acquisition of the necessary knowledge from a case-based reasoning point of view. The development of systems which can be applied to real world problems has to meet certain requirements. E.g., all available information sources have to be identified and utilized. Normally, this involves different types of knowledge for which several knowledge representation schemes are needed, because no scheme is equally natural for all sources. Facing empirical knowledge it is important to complement the use of manually compiled, statistic and otherwise induced knowledge by the exploitation of the intuitive understandability of case-based mechanisms. Thus, an integration of case-based and alternative knowledge acquisition and problem solving mechanisms is necessary. For

---

this, the basis is to define the "role" which case-based inference can "play" within a knowledge acquisition workbench. Up to now all this problems are at most partially solved. What we need is a complete analysis of the problematic nature of knowledge integration for a complex real world domain. A first systematic approach is, e.g., given by van Someren et al. in [SZP90].

We will discuss a concrete case-based architecture (PATDEX: PATtern Directed EXpert System), which has been applied to technical diagnosis problems, and its integration into a knowledge acquisition workbench (MOLTKE: MOdels, Learning and Temporal Knowledge in Expert Systems for Technical Domains) which includes compiled knowledge and explicit deep models, additionally. For many of the just mentioned unsolved problems the proposed research work offers solutions, from an expert system point of view as well as from a case-based reasoning one. In connection with the first view these are the utilization of all available knowledge sources, a natural representation of the involved knowledge types, the exploitation of the intuitive understandability of case-based mechanisms and its application to diagnosis (including test selection). Research deficits concerning case-based reasoning are, e.g., the handling of real complex applications, the integration of case-based mechanisms with alternative ones for knowledge acquisition and problem solving, the integration of adaptive similarity measures and explicit deep models as well as the definition of a "role" for case-based reasoning.

PATDEX is based on a case representation (*diagnostic cases*) which is simple, natural and easy to acquire (a list of attribute-value-pairs). The applied inference engine can easily be understood because it is controled mainly by a similarity measure (together with two thresholds). Additionally, the classification task depends on the symptom relevances and the test selection task on the average diagnostic cost. The similarity measure is an instance of the *Contrast Model* proposed by [Tve77] and PATDEX acquires adaptively its analogical reasoning abilities for the support of the classification task. Thus, it can be applied to diagnostic problems in general[2]. PATDEX treats the test selection task itself in a case-based manner. This mechanism is based on *strategy cases* which can be automatically generated from the diagnostic cases. This offers the advantage to improve the classification and test selection strategy abilities separately from one another. PATDEX´s task within the workbench is the handling of exceptional cases which can be "temporary" or "absolute"[3]. Therefore PATDEX cooperates with the interpreter of the MOLTKE shell and utilizes the compiled and model-based[4] knowledge of the workbench as background knowledge. PATDEX is an interactive system and learns incrementally. If overgeneralizations are identified they can be corrected directly.

---

[2] PATDEX combines two classification mechanisms, one for the solution process itself and one to acquire the necessary information while searching for a solution.

[3] Exceptional cases are more seldom than typical ones. An exceptional case is called *absolute* if the described exception is related to the structure and range of the expertise. Such a case is called *temporary* if it refers just to the actual knowledge base and a change of its state may be probable.

[4] In this case model denotes a deep functional model of the technical system which is to be diagnosed. It enables the simulation of the behavior of the technical system on a sufficiently precise level of abstraction.

PATDEX is able to handle complex real world applications. E.g., vague cases, default values for symptoms as well as pathologic symptoms and symptom values can be taken into consideration. Because of its efficiency and stability PATDEX can work, besides being integrated in the workbench, as a stand-alone system. PATDEX learns by forgetting and is able to dimension its case base explicitly. The efficiency of its procedure is ensured by a dynamical classification of the cases on different levels of activation as well as a Rete-like case network for the support of the inference engine. The latter one results in a fast exclusion of dissimilar cases, but guarantees that all really similar cases are considered. Thus, PATDEX offers solutions for many of the above mentioned problems which represent the current state of the art both in its combination and in some special aspects as the case-based test selection, the adaptive similarity measure and the integration in the workbench.

The next chapter gives an overview of PATDEX´s integration into the MOLTKE workbench while chapter 3 concentrates on the basic abilities of PATDEX as well as its cooperation with other workbench components. Chapter 4 offers different ways to evaluate and discuss PATDEX: two trade scenarios will be presented which give a distinct idea of the advantages and disadvantages of PATDEX. This is completed by a classification and discussion of PATDEX concerning related research work.

2 Integration of PATDEX into the Workbench

Technical diagnosis usually needs three main sources of information which have to be represented adequately. These are the compiled diagnostic knowledge, the empirical knowledge and the knowledge of the technical system which is to be diagnosed. The MOLTKE workbench allows the integration of all these knowledge types within one knowledge base [cf. Fig. 1 and 2].

MOLTKE	
Workbench for Technical Diagnosis	
GenRule	MAKE
Case-Compiler	Model-Compiler
PATDEX	MOLTKE-Shell
Case-Based Diagnosis	Heuristic Diagnosis

Fig. 1 – Components of the MOLTKE Workbench

The representation language of the workbench can be considered as a design model [BW89] and forms the basis for the MOLTKE shell (together with the interpreters). A *basic expert system* being automatically generated from the explicit model of the technical

system is the starting-point for the development of the knowledge base[5]. This can be extended and improved by the use of compiled and empirical knowledge. The generation and justification of different kinds of heuristic rules as well as the detection of certain inconsistencies within the knowledge base can be done automatically[6]. PATDEX´s task is the direct interpretation of the empirical knowledge which cannot, or at least not immediately, be integrated into the knowledge base. This kind of knowledge is called *absolute* or *temporary exceptional cases*. At any time there is the possibility of processing the empirical knowledge (exceptional cases) in parallel to the knowledge base. A procedure operating on the case memory for all cases decides which empirical knowledge can be integrated automatically. The actual state of the knowledge base is considered in a snapshot-like manner, but an automatic update to a changed knowledge base is possible[7]. All other cases are available for the PATDEX system in a special case base.
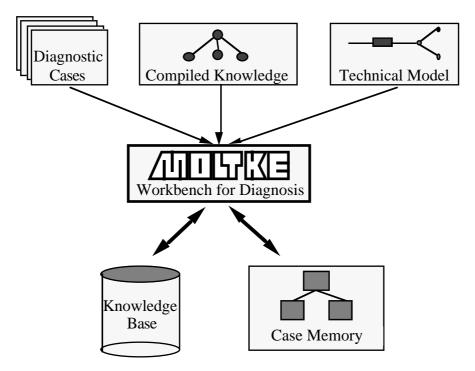

Fig. 2 – Processing of Multiple Knowledge Sources in the Workbench

3 The PATDEX System

In this chapter we describe the classification and the test selection components of PATDEX. They are completed by the description of the underlying similarity measure, the respective learning procedures, and the application of technical background knowledge. In addition, PATDEX´s cooperation with the other workbench components stated. For clarification purposes we begin with a short introduction into the needed terminology.

---

[5]   The basic expert system can be generated by the MAKE system [AMR90 or Reh91], cf. section 3.5.
[6]   This can be carried out by the GenRule system [AFT+89, AMR90, AMW90 or Alt91].
[7]   This is done by the KnowledgeBaseExpander, a subcomponent of the GenRule system [Alt91].

3.1 Terminology and Setting of the Task

We assume a fixed number N of *symptoms* $S_1,...,S_N$. With each symptom S_i a range R_i is associated; in principle symptoms are nothing than attributes. Typically R_i is either a real interval [a,b] or the boolean domain {0,1} or some other finite set. Symptoms may take on values in their range and these values are assumed to be the only source of information. Values of symptoms are obtained by carrying out a *test.* A test can be an observation, a measurement or simply the answer to a question. In some situations certain tests may not be allowed. The information at some stage of the diagnostic process is usually incomplete and is expressed in the form of an *information vector* or a *situation*: A situation is a vector Sit = $(a_{i1},...,a_{ik})$, $1 \le i,j \le N$ such that each $a_{ij} \in R_{ij}$. The components of Sit are the known symptom values whereas the values of the remaining symptoms are unknown. A situation is *complete* if every symptom has a value. A *diagnosis* (or fault description) is a formula of the first order predicate calculus using constants and relations over the ranges R_i; the precise form of these formulae is not of interest here. To avoid technical difficulties we assume always a single fault. This means that the set of complete situations is partitioned into sets representing these faults; a special set is "no fault" and, if wanted, another one is "unknown fault". The applicability of this approach relies on the fact that, at least, the "interesting" faults can be fully described. In the diagnosis of even complex machines this assumption is usually satisfied; in medical diagnostics this sometimes may be doubtful.

In a diagnostic problem some complete situations have occurred but are only partially known, i.e. one is confronted with some incomplete situation Sit. The task is to determine the diagnosis of the unknown complete situation (at least with some certainty).

3.1.1 Cases

A *(diagnostic) case* is the protocol of the real classification behavior of a service technician[8]. It is represented as a list of symptom-value-pairs (the problem description) completed by an empirical justified solution. In our diagnostic context the problem is a situation (Sit) and the solution is the diagnosis (Φ) justified by J, hence a case has the syntactic form C = (Sit,J,Φ). Since real world cases have to be handled three basic possibilities have to be considered:

- Sit did not have sufficiently many symptom values to determine Φ but one had a good guess.
- Sit did determine Φ but contained redundant information, i.e. unnecessary symptom values.
- Sit did determine Φ and no smaller situation would do so.

Since one does not know which possibility has occurred PATDEX has to take care of all of them[9]. The (implicit) subject of all this considerations has been the classification task.

---

[8]   Service technicians are technicians of a manufacturer whose task is to maintain the functionality of sold technical systems.

[9]   Especially we point out here, that the occurrence of incompleteness (1) and redundancy (2) is independent of one another.

For the test selection task we have a similar situation. A *strategy case* is a protocol of the real problem solving strategy of a service technician. The representation of strategy cases is comparable to that of the diagnostic cases whereas the solution is the symptom which is to ascertain next. Hence we have $C_S = (Sit, J, S_i)$.

3.2 Overall Algorithm of PATDEX

PATDEX´s problem solving process is started by the user giving some observed symptom values as input to the system. Then the toplevel algorithm of PATDEX reads as follows (cf. also Fig. 3):
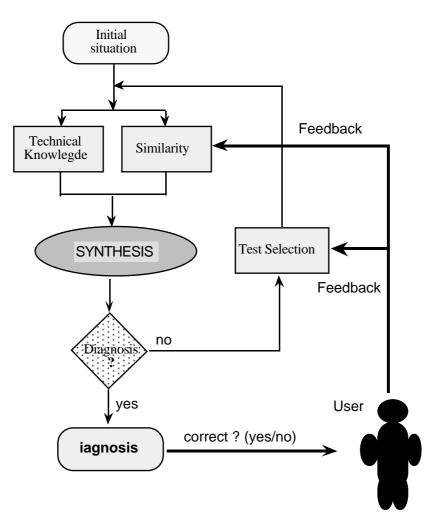

Fig. 3 – Toplevel Algorithm of PATDEX

Input:	The actual situation Sit
Output:	Diagnosis Φ or negative report

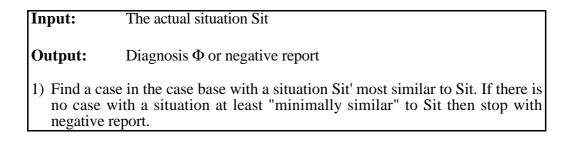1) Find a case in the case base with a situation Sit' most similar to Sit. If there is no case with a situation at least "minimally similar" to Sit then stop with negative report.

2) If Sit and Sit' are "sufficiently similar" then accept the diagnosis Φ of Sit' also for Sit and goto 4).

3) Otherwise select a test in order to obtain an improved situation and goto 1).

4) If the diagnosis is correct then add the case (Sit,J,Φ) to the case base and stop with success.

5) If the diagnosis is not correct then cancel temporarily (i.e. for the actual problem) all cases with diagnosis Φ and goto 3).

Here we need an external teacher who says whether a diagnosis is correct or not. We also have to explain "minimally similar" and "sufficiently similar". For this we need a partition of the case base which is given after the introduction of the similarity measure. Finally, we have to describe the selection of the next test.

3.3 Solving the Classification Task

The goal of the classification task is to find a case which can be used to solve the current problem, i.e. to find a case which can explain the actual situation Sit correctly. Here we can exploit the characteristics of the domain of technical diagnosis that observable similarities, concerning the fault behavior of the technical system being considered, normally have similar causes. Thus, our underlying assumption is that similar cases have similar classifications. The selection of a suitable case uses a similarity measure *sim* which maps any combinations of situations into [0,1]. The basic axioms for *sim* are:

- $sim(x,x) = 1$ (reflexivity)
- $sim(x,y) = sim(y,x)$ (symmetry)

Given Sit, *sim* selects $C' = (Sit',J,\Phi)$ such that Sit' is most similar to Sit with respect to the case base. If the current situation Sit is identical to Sit' then Φ is the correct diagnosis for Sit. Often such an inference is still correct if Sit' differs from Sit "a little bit", i.e. the two situations only have to be "sufficiently similar" with respect to the classification task.

3.3.1 A Static View on Measuring Similarity

A first approach to define "sufficiently similar" is to match the respective situations syntactically where the computed similarity has to exceed a certain threshold $\delta \in [0,1]$, i.e. $sim(Sit,Sit') > \delta$. A very general type of appropriate similarity measures for situations which are used in PATDEX is of the form:

$$sim(Sit_1, Sit_2) = \frac{\alpha\ card(E)}{\alpha\ card(E) + \beta\ card(D) + \eta\ card(U_1) + \gamma\ card(U_2)}$$

where α, β, γ and η are real numbers, card denotes the cardinality of sets, and

E   :=   set of symptoms with the same values for Sit_1 and Sit_2;
D   :=   set of symptoms with different values for Sit_1 and Sit_2;

U_1 := set of symptoms with values for Sit_1 but not for Sit_2;
U_2 := set of symptoms with values for Sit_2 but not for Sit_1.

PATDEX uses, as a first proposal, the parameters $\alpha = 1$, $\beta = 2$, $\gamma = \eta = 1/2$. This special choice of the parameters is mainly motivated by experimental results. The similarity measure has a defensive, pessimistic character. A high negative contribution to the measure is given for conflicting symptom values, i.e. we strongly wish to avoid false diagnoses.

A case becomes disqualified for further use in a particular diagnosis session as soon as all symptoms contained in that case do not hold, given a situation encountered during diagnosis, or if there are no unknown symptom values any more and the specified case does not exceed the diagnosis-threshold δ. Another reason for disqualification is given if the case the system chooses as its hypothesis is refused by the user. For performance purposes a second threshold ϵ is introduced with $0 < \epsilon < \delta < 1$. Both thresholds are locally defined for each case of the case base and enable a dynamic organization of all cases within different levels of activation. Depending on Sit_1 and Sit_2 these levels are called as follows:

- indistinguishable $\Leftrightarrow$ $sim(Sit_1, Sit_2) = 1$;
- sufficiently similar $\Leftrightarrow$ $\delta \leq sim(Sit_1, Sit_2) < 1$;
- probably similar $\Leftrightarrow$ $\epsilon \leq sim(Sit_1, Sit_2) < \delta$;
- at least minimally similar $\Leftrightarrow$ $0 \leq sim(Sit_1, Sit_2) < \epsilon$;
- not minimally similar $\Leftrightarrow$ $sim(Sit_1, Sit_2) < 0$.

3.3.2 Extending the Static View

Within case-based reasoning retrieval can be called "reasonable" only if the solution of the retrieved case can be applied to the current problem. Thus, a diagnostic case is sufficiently similar to a given situation if they have the same diagnosis. The problem with this definition of similarity is that the diagnosis of the given situation is not known at retrieval time (of course). Therefore we have to find a good approximation for this measure. The complexity of the knowledge which is necessary to define a well-suited static similarity measure is comparable to the complexity of the whole diagnostic task. Thus, the definition of a similarity measure for a special domain has to be seen as a learning task.

This extension results in an improved version of the similarity measure, but still does not consider any explicit domain knowledge. Within the domain of technical diagnosis this leads to the following problems:

- the use of compiled diagnostic knowledge
- identification of redundant information
- the handling of pathologic symptom values
- similarity between symptom values

The use of compiled diagnostic knowledge
Real diagnostic situations are characterized by an incomplete information vector, i.e. there exist some symptoms of which the value is unknown. Technical knowledge can be used

to derive such values from known ones. E.g., from "light = on" it can be derived that "switch = closed" holds with a certain probability. Here we can differentiate between causal and heuristic relations. In the MOLTKE workbench both types of knowledge can be automatically generated by specialized components. Thus, this information is utilized by the PATDEX system.

Identification of redundant information
The relevance of a symptom for a certain diagnosis depends on the respective context. An important aspect of our PATDEX approach is to view the relevances of certain symptom values for special situations as a part of the empirical knowledge which shall be learned. In PATDEX we combine the case-based reasoning approach for diagnosis with a connectionist approach for learning this empirical knowledge. These relevances $w_{ij} \in [0,1]$ are represented by means of a *relevance matrix* $\boldsymbol{R} = [w_{ij}]$ where the symptoms S_i and diagnoses Φ_j occur as inscriptions of the rows and columns, respectively. In course of time the weights of the symptoms, i.e. the elements of the relevance matrix, are learned by PATDEX. The strategy for learning the entries of the relevance matrix is similar to the competitive learning mechanism proposed in [RZ85].

The handling of pathologic symptom values
We can differentiate between symptom values which are pathologic or not. Pathologic symptom values are caused by an abnormal system behavior while non-pathologic values reflect the normal functioning of the technical system. For the degree of relevance of a certain symptom value it is important whether it is a consequence of the normal functioning of the technical system or of a fault. E.g., "relais 21K3 switched" is of the first kind while "voltage 214 too high" is a pathological symptom value. The differentiation whether a certain symptom has a pathologic character or not depends on the respective context, e.g. "light≠on" is not automatically a pathologic one. If "switch=closed" is known additionally, the previous symptom becomes pathologic. Thus, functional background knowledge is needed for the handling of such information. Within the PATDEX system every most similar case has to explain all known pathologic symptom values. This defines a kind of "minimal correctness".

Similarity between symptoms
Every really flexible similarity match between the current situation and a case of the case base has to consider the similarity between symptoms and symptom values (Similarity versus Part-Identity [Smi89]). This can be realized by the additional use of local similarity measures $\omega_i(a_{ik},a_{il}) \in [0,1]$, which determine the similarity between possible symptom values $a_{ik},a_{il} \in R_i$ of a symptom S_i, or by a knowledge-based inference (i.e. $\omega_i(a_{ik},a_{il}) \in \{0,1\}$). If one of the symptom values is *unknown* then the similarity ω_i evaluates to 0.

The introduction of $\boldsymbol{R}$ and ω_i leads to the following extended definition of the similarity measure *sim*:

$$sim(\text{Sit}_1, \text{Sit}_2) = \frac{\alpha E}{(\alpha E + \beta D + \eta U_1 + \gamma U_2)}$$

For the exploitation of the above described features we have to extend the definition of the "attribute sets"[10]. Let a_{ik} be the observed symptom value in Sit_1 and a_{il} the defined symptom value in the actual case $C = (Sit_2, \Phi j)$. Then the new definition reads as follows:

$$E := \sum_{S_i \in E} w_{ij} \; \omega_i(a_{ik}, a_{il})$$

$$D = \sum_{S_i \in D} w_{ij} \; (1 - \omega_i(a_{ik}, a_{il}))$$

$$U_1 := \sum_{S_i \in U_1} v_{ij} \; (1 - \omega_i(a_{ik}, a_{il})) = card(U_1) \; \text{with} \; S_i \; \text{pathologic}$$

$$U_2 := \sum_{S_i \in U_2} w_{ij} \; (1 - \omega_i(a_{ik}, a_{il})) = \sum_{S_i \in U_2} w_{ij}$$

We point out here that ω_i is zero for symptoms S_i which belong to one of the attribute sets U_1 or U_2, because the corresponding symptom values are unknown. Additionally, we restrict the representation of redundant symptoms (i.e. $S_i \in U_1$) to pathological ones. Thus, observed redundant symptom values representing the normal behavior of the underlying technical system cannot decrease the value of *sim* any more. Since PATDEX focuses on the learning of symptom relevances only for the respective diagnosis no entries for redundant symptoms S_i can be created. Here we need an alternative weighting v_{ij}. In PATDEX we define $\forall i,j \; v_{ij} = 1$, which is motivated by the above mentioned restriction of U_1.

By the use of these definitions we get a similarity measure *sim* which is depending on the values represented in the relevance matrix. After each erroneous diagnosis the weights of the relevance matrix are changed. Thus, the similarity measure *sim* is the result of an adaptive learning process and the integration of background knowledge available from the MOLTKE workbench.

3.4 Solving the Test Selection Task

For a case-based reasoning system not only its correctness is of importance but also its efficieny. The task of the test selection component is to complete a given incomplete situation as economically as possible, i.e. to avoid redundant tests, such that a diagnosis can be made. Up to now existing case-based reasoning systems concentrate on the classification task only. The test selection task either is not considered[11] or realized in very simple manner. If cases are acquired for the classification task it is on offer to use them for the test selection task, too. In PATDEX the diagnostic cases reflect the whole diagnostic process. Thus, not only the classification process is described but also that of test selection.

---

[10] Here we replace the set cardinalities by the sum of the symptom relevances.
[11] This is based on the unrealistic assumption that all needed information is given.

Therefore the diagnostic cases have to represent the strategy of the respective service technician, i.e. within the case representation we have to assume a temporal order.

For a stepwise optimization of the classification abilities the user´s feedback can be utilized. A correct diagnosis results in a positive feedback, a wrong one in a negative feedback. Here it is clear whether a diagnosis is correct or not. Within the test selection context such a clear definition is not possible, because there do not exist any absolute criterions whether the ascertaining of a symptom is necessary or not. Thus, for the test selection task criterions like fault frequency and diagnostic cost, i.e. the cost which is necessary to ascertain a (special) symptom, come into play which have to be taken under consideration for the feedback. A possible approach for learning a good test selection strategy is to adaptively form a test selection order which is not necessarily optimal for a special case but for the average of all known cases. Of course, it has to be taken into account that special or exceptional strategies normally are more informative than typical ones. Thus, in reality a combination of a case focusing and an average cost minimizing strategy is the best suited approach which is possible.

Of importance for the feedback is the notion of *subsequent cost*. Subsequent cost denotes the cost which still remains to find the correct diagnosis after of a certain strategy case has been applied. E.g., a simple measure for this is the number of symptoms which still have been acquired after a strategy case has been applied. More sensible are domain-specific measures which, additionally, consider diagnostic and temporal cost.

Algorithm for case-based test selection:

Input: Diagnostic case C_k (diagnosis hypothesis), Situation Sit
Output: Symptom S_i which is to ascertain
1) Find a strategy case in the case base with a situation Sit' most similar to Sit.
2) If there is no strategy case with a situation at least "minimally similar" to Sit then use diagnostic case C_k for a case focusing test selection.
3) If several strategy cases are "sufficiently similar" then select that strategy case which has caused the lowest subsequent cost on the average.

After a diagnostic session has finished every strategy case is charged with the diagnostic cost it has caused. Strategy cases which encompass redundant symptoms are charged, on the average, with higher cost than cases which include only necessary symptoms. The fault frequences are taken into account implicitly by the computation of the averages. If the system has carried out an unknown strategy then the respective strategy cases are included into the strategy case base. In the worst case all symptom orders have to be represented by strategy cases. Thus, the needed space increases exponentially concerning the number of symptoms. This is not possible for a real world application. In PATDEX the number of representable strategy cases is fixed (e.g. 10000). If more cases have to be represented PATDEX forgets those strategy cases which have not been used for the longest period of time.

The idea behind the case-based test selection of PATDEX is to offer good strategies for frequently occuring situations. Neither the frequency of a fault nor the quality the strategy taken by the system have to be defined a priori. For faults which appear less frequently PATDEX can utilize the strategy of the service technician which is represented by the actual (diagnostic) case, i.e. a case-focusing strategy can be chosen. Of course, such a strategy is not necessarily globally optimal, but exceptional cases normally are more informative than typical ones.

3.5 Cooperation with other Components of the Workbench

Since we have given an overview of the MOLTKE workbench and described PATDEX in more detail we are now able to summarize the cooperation of PATDEX with the other components of the workbench, especially the MOLTKE shell:

- **Cooperation with GenRule** (<u>Gen</u>erator of Empirical MOLTKE <u>Rule</u>s)
  GenRule determines, and regularly updates, the subset of the overall case memory which is available in the PATDEX case base. PATDEX uses heuristic rules generated by GenRule for the derivation of additional symptom values (from already known values).
- **Cooperation with MAKE** (<u>M</u>odel-based <u>A</u>utomatic <u>K</u>nowledge <u>E</u>xtractor)
  PATDEX works in parallel to the basic expert system generated by MAKE which is the first version of the knowledge base. MAKE generates causal rules which are used by PATDEX, in a similar way as the heuristic rules of GenRule. In addition, the MAKE underlying model serves data base-like queries, e.g. for the identification of concrete components of the technical system.
- **Cooperation with the MOLTKE-Shell**
  The MOLTKE-Shell controls the test selection while PATDEX does its case-based classification as a background job. The test selection component of PATDEX is used for clarification purposes only, i.e. if contradictory classifications of the systems are present and cannot be solved immediately.
  In view of the procedure the following situations are of special interest:
  - The shell classifies correctly and PATDEX does not find a sufficiently similar case in its case base:
    - » The diagnosis can be made, because no contradiction has been observed.
  - Both the shell and PATDEX classify correctly:
    - » The diagnosis can be made. There is the possibility to exclude the cases from the case base, because the knowledge base already "encompasses" them.
  - PATDEX classifies correctly and the shell does not find any diagnosis:
    - » There is a gap in the knowledge base. If the most similar case within the PATDEX case base is a temporary exception, then the knowledge base can be extended and the case excluded from the case base.
  - The shell and PATDEX classify differently:
    - » Different conclusions have to be taken depending on the actual situation. It is possible that PATDEX is overgeneralizing and the shell is classifying correctly, or that PATDEX is classifying in a more special way than the shell and an exception can be identified. It is necessary to prove the case chosen by PATDEX. If the case is proven to be sound, then the knowledge base

does not fit to the current situation. It has to be checked if the knowledge base still makes sense taking a more general point of view. If it is not possible to prove the selected case the respective diagnosis threshold has to be increased.

- ○ Other situations, e.g. wrong classifications by PATDEX or the shell:
  - » The correct case can be directly given as input. Afterwards PATDEX can correctly classify the identical situation. If necessary wrong parts of the knowledge base have to be corrected.
- **Case-based test selection within the shell:**
  PATDEX´s test selection mechanism is a component of the shell.

4 Evaluation and Discussion

To enable an intuitive evaluation of the PATDEX system two scenarios are presented[12]. The basic idea is that a software developing company *ExpertSystemsDeveloper ($\mathbb{ESD}$)* tries to sell the PATDEX™ system and/or the complete MOLTKE™ workbench to a company named *TechnicalSystemsManufacturer ($\mathbb{TSM}$)* which might need them for the solution of their diagnostic problems. $\mathbb{TSM}$ makes certain demands on the application of such tools. It takes the advice of $\mathbb{ESD}$ which, additionally, makes certain offers. The subject of the first scenario is PATDEX as a stand-alone system while in the second one PATDEX is only a part of the subject under consideration, namely the MOLTKE workbench. These scenarios are completed with two evaluations, one given by $\mathbb{TSM}$ the other one by $\mathbb{ESD}$. Then PATDEX is classified and discussed and the current state of realization of all involved workbench components is clarified.

4.1 Consultation of the Manufacturer

$\mathbb{TSM}$ wants to use expert systems for the support of its service department. After a thorough analysis of $\mathbb{TSM}$´s current situation $\mathbb{ESD}$ makes an offer which, among others, includes the following items:

- **Possible locations of application:**
  - ○ Usage at the manufacturer
  - ○ Centralized application at a client of $\mathbb{TSM}$
  - ○ Decentralized application at a client of $\mathbb{TSM}$
  - ○ Local usage within a technical system itself
- **Possible types of application:**
  - ○ Support of telephone diagnosis
  - ○ Usage as integrated and/or accompanying diagnostic system
  - ○ Support of training
  - ○ Support of knowledge documentation

$\mathbb{TSM}$ responds affirmatively to $\mathbb{ESD}$´s offer. It decides to support the service of the machines of type MX by the use of a diagnostic expert system. At the same time this should be seen as a test case for the efficiency of $\mathbb{ESD}$´s products. Then $\mathbb{ESD}$ gives an introduction into the problematic nature of developing a knowledge base for a technical

---

[12] Of course, these scenarios are fictitious, though correspondences with real situations are intended.

system like MX. $\mathbb{ESD}$ illustrates this using six different sets of cases which, starting with the smallest set A, are subsets of the respective following set [cf. Fig. 4]:

- **Set of all input cases (A)**
  This are all cases which have been given as input to the knowledge base.
- **Set of all cases which can be correctly classified (B)**
  This set encompasses all cases of which the diagnosis would be selected if the problem description of the respective case is given. If identity is viewed as the best similarity being possible, as done by PATDEX, then all input cases can be correctly classified which results in A ⊆ B.
- **Set of all interesting cases (C)**
  Set C includes all cases which must be correctly classified according to the meaning of the responsible experts. Since only real cases should be given as input and all correctly classified cases are, of course, interesting C is a superset of B: B ⊆ C.
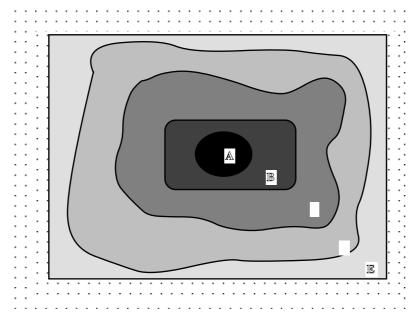

Fig. 4 – Case Scenario for Diagnostic Real World Applications

- **Set of all cases which are possible in view of the functional behavior of the technical system and the most precise level of abstraction which is still being considered (D)**
  This are all cases which are really possible though they would not be considered as interesting (if they do not belong to set C), because the experts do not remember (or know) them, i.e. the experts would not present them as input or test cases[13]. Since all cases being interesting for the experts are possible according to the functional behavior of the technical system, this results in C ⊆ D.
- **Set of all cases which can be described by the chosen case representation (E)**

---

[13] For clarification purposes we want to mention here that *test cases* have to be distinguished from *strategy cases*. While the latter ones support the test selection task, i.e. the ascertaining of symptoms, test cases are a means for carrying out a test of the expert system.

Set E encompasses many cases which do not appear in reality because of technical constraints of the system which is to be diagnosed. The case representation is aimed at being able to describe all cases which belong to D. Thus, we have D ⊆ E.

- **Set of all really possible cases facing the technical system under consideration (F)**
  This set includes all really possible cases according to the behavior of the technical system, especially those of which the level of detail of the case description is more precise than it is necessary for the diagnosis which is to be made. Thus, many of these cases cannot be described using the chosen case representation. Another aspect is that all cases are included which ignore the technical constraints of the underlying system. This results in E ⊆ F.

$\mathbb{ESD}$ calls $\mathcal{TSM}$´s attention to the fact that PATDEX correctly classifies all cases which have been given as input to it. $\mathcal{TSM}$ refers to the point that <u>all</u> interesting cases have to be diagnosed correctly. $\mathbb{ESD}$ replies that this would be the case, if all interesting cases are given as input to the system. $\mathcal{TSM}$ rejects this because of the unjustifiable expenditure and guarantees only a restricted selection of exemplary cases to $\mathbb{ESD}$. In addition, $\mathcal{TSM}$ insists on the carrying out of several exhaustive tests which would be very important concerning the acceptance of the expert system.

4.2 Scenario 1: PATDEX as a Stand-Alone System

$\mathcal{TSM}$´s guideline for the development of a MX knowledge base can be roughly described by the following quotation: "It is better to solve only 70% of the cases with low cost than all cases with high cost". Therefore $\mathbb{ESD}$ decides to develop the diagnostic expert system for the MX only on the basis of PATDEX.

The development of the knowledge base is subdivided into several steps. Within step 1 cases are defined as protocols of the real problem solving behavior of a service technician and represented in PATDEX. Then these cases are acquired by "observing" the service technicians. To reduce the observation expenditure a symptom acquisition blank has been designed which can be used by the service technicians to mark and/or enter the symptom values that have been ascertained during the diagnostic process.

During step 2 an overall test of the knowledge base which has been acquired in step 1 is carried out. $\mathcal{TSM}$ engages different experts who have to check the knowledge base by the use of many test cases being selected for that very purpose. Because of the big gaps within the knowledge base and some wrong diagnoses the tests results in a negative evaluation.

$\mathbb{ESD}$ refers to two facts, namely that within step 1 input cases have been available only in an insufficient number and that step 2 (as well as step 3) have to be seen as a part of the knowledge acquisition process. In the meantime PATDEX would be able to correctly classify all test cases which have been given as input to it, i.e. gaps of the knowledge base can be eliminated easily. In addition, tests could be seen as an appropriate procedure for the explication of the only implicit available cases of set C. Upon request all involved $\mathcal{TSM}$-experts have to admit that they remembered many of the

applied test cases only during the testing of the knowledge base. Moreover, it has been very motivating to search for situations which PATDEX cannot handle. The wrong diagnoses made by PATDEX can, to a big part, be traced back to overgeneralizations. $\mathbb{ESD}$ underlines the fact that every diagnosis, where PATDEX should be in principle able to make it, must be a part of at least one case of the case base. Wrong diagnoses which cannot be traced back to overgeneralizations can be eliminated by PATDEX´s learning component (adaptive learning of the symptom-diagnosis-relevances) or manually because of the transparency of the applied case-based inference.

Thus, the results of step 2 meet, in spite of the negative intermediate evaluations, the requirements of $\mathbb{TSM}$. The main criterion for the acceptability of the solution offered by PATDEX has been the scalability of the validity of the knowledge base which easily enables the fulfilment of $\mathbb{TSM}$´s guideline. $\mathbb{TSM}$ decides to use PATDEX within a third step for a special adaptation process where only the real cases of the respective MX machine (one several selected ones) are given as additional input to the expert system.

4.3 Scenario 2: PATDEX as a Component of the Workbench

Since the MX project has been successful $\mathbb{TSM}$ takes further diagnostic expert system projects into consideration. Especially the problematic nature of determining set C has been realized by $\mathbb{TSM}$ only now. Besides, the use of expert systems for the support of training and knowledge documentation recommends the development of knowledge bases which are as complete as possible, even then if the cost for it is very high. Some critical $\mathbb{TSM}$-cooperators refer to the maintenance and update problem of such high cost software systems. $\mathbb{ESD}$ suggests to analyse which additional information source can be utilized. The result is that two further types of knowledge sources can be exploited, namely compiled diagnostic knowledge and, on a large scale, the technical documentation of the particular machines.

Then $\mathbb{ESD}$ makes the offer to develop a knowledge base on the basis of the whole MOLTKE workbench. as for the MX project this project is aimed at modeling $\mathbb{TSM}$´s knowledge about a special type of machine which in this case has to be as complete as possible. Facing the case sets of Fig. 4 $\mathbb{ESD}$ explains that such a project has two consequences. First, the equality of the sets B and C (as in the MX project) and, additionally, the equality of the sets C and D. Therefore it would be much more difficult to determine set C. $\mathbb{ESD}$ calls $\mathbb{TSM}$´s attention to the fact that the only procedure which makes sense is to represent as many parts of the technical system as possible as a deep explicit model, and, thus, make it available for the expert system. This must be completed with very detailed and exhaustive test steps which enable additional manual improvement of the knowledge base.

$\mathbb{TSM}$ responds affirmatively to $\mathbb{ESD}$´s offer. $\mathbb{TSM}$ holds out the prospect of another project which deals with the fault diagnosis of the simpler structured MX variant MY. It would be possible to develop a complete, new knowledge base and to do this in very efficient manner, because most of the knowledge being acquired for the MX can be transfered to the MY knowledge base. In addition, this knowledge base is not as extensive as that of MX. The basic condition for starting the project is that there must exist an efficient

strategy how to update an existing knowledge base and how to adapt it for comparable technical systems.

For clarification purposes $\mathbb{ESD}$ gives additional advice concerning the MOLTKE workbench (cf., e.g., Fig. 1 and 2). During the training sessions $\mathbb{TSM}$-cooperators are especially interested in the cooperation between PATDEX (which they already know) and the other components of the workbench (cf. section 3.5). $\mathbb{ESD}$ explains that a basic version of the knowledge base, which is to be developed, can be automatically generated from the technical documentation which has been given as input to the workbench. Then this version has to be manually revised and extended. E.g. one problem which has to be solved is that this basic system does not simulate the real experts´ problem solving behavior. This problem can be partially eliminated by the automatic generation of heuristic rules. Within the workbench PATDEX´s main task is the direct interpretation of the available concrete empirical knowledge. This ensures that as little empirical knowledge as possible is not or in a wrong way represented in the knowledge base. Thus, cases are used for the refinement and correction of the knowledge base as well as for the orientation of the test selection task to the expert´s problem solving behavior. All this is aimed at the development of an integrated general knowledge base which can be processed using the MOLTKE shell interpreter. Exceptional cases stay in the PATDEX case base and are still processed by PATDEX. The shell and the PATDEX system are cooperating dynamically.

One main subject of the training sessions was the maintainability of a developed overall knowledge base (which encompasses the knowledge base and the case base). Minor changes which are needed variants and/or new series of a technical system can be handled without problem using PATDEX. Major changes can be carried out directly in the knowledge base or in the underlying deep model of the technical system. If the latter alternative is chosen a completely new version of the basic knowledge base has to be generated. In addition, all manual changes which have already been done have to be repeated. This produces a certain dissonance among the $\mathbb{TSM}$-cooperators because of the impression that (nearly) the same work has to be done very often. $\mathbb{ESD}$ refers to the fact that this would be in principle a software development problem. Therefore $\mathbb{ESD}$ cooperates with a research group which currently improves the workbench. This relates to the development of a component for knowledge maintenance and consistency checking, the incremental extendability of the basic version of the knowledge base as well as adaptation of the knowledge base to given changes of the underlying deep model and the structural adaptation of the knowledge base to known cases. Because of the positive experiences with the PATDEX system $\mathbb{TSM}$ at last decides to carry out the MY project as another test case.

4.4 Related Work

In view of the classification task the analogy-based inference mechanism of PATDEX can be considered as "transformational analogy", facing the test selection task it can be seen as "derivational analogy" [Car83+86]. In addition, there exist several strong connections to the "Memory-Based Reasoning" approach of Stanfill & Waltz [SW88] and the "Instance-Based Learning" (IBL) algorithms proposed by [AKA91].

Well-known case-based reasoning approaches which can be compared to PATDEX are, among others, the PROTOS [Bar89], CASEY [Kot89], SIZZLE [Off89], CREEK [Aam90], and memory-based reasoning, e.g. MBRtalk [SW88] approaches. Good overviews are given in [KRW89] and [Sla91]. In applying case- and analogy-based reasoning techniques to the problematic nature of test selection and in integrating connectionist methods for adaptive learning, PATDEX contributes to the current state of the art defined by these (and similar) systems, respectively. PATDEX has its own pragmatic view on case representation and similarity [RW91] like SIZZLE. It does not use a case memory to represent the similarities between cases like, e.g., [VC89], but exploits an explicit evaluation function in combination with a partitioning of the case base called levels of activation and a Rete-like case network. This is comparable to the indexing techniques in conceptual memories cf., e.g., [Kol83a+83b]. Since real world applications involve large case bases the retrieval must be very efficient. Our approach is to combine the good performance of a syntactic match like MBRtalk, or the IBL algorithms, with, if necessary, the (expensive) use of explicit background knowledge like, e.g., CREEK and CASEY do. This approach is possible because there exists a defined role for the PATDEX system within the MOLTKE workbench, whereas CREEK and CASEY have to do the whole diagnostic task stand-alone.

PATDEX not only exploits causal background knowledge, as CASEY does, but also functional knowledge. PATDEX is a computational model which is fully implemented. Especially PATDEX/1 [cf. section 4.5] has been tested since early 1989. Because of its adaptive capabilities PATDEX can be easily used for other domains within the field of technical diagnosis. This is not so easily done for PROTOS as many numerical values and the relational structure needed for the explanation facility have to be transferred to the new domain [BBP88].

In the MOLTKE workbench the applied learning strategies have to support the generation of an implementation model for the underlying technical system, based on a design model for technical diagnosis. As compared with, e.g., the BLIP approach [Mor87+89b] this is a simplifying view because no domain model is learned, but we think that our approach is adequate for the field of technical diagnosis.

4.5 State of Realization

Actually, there are two systems, PATDEX/1 and PATDEX/2. PATDEX/2 [Alt91, Weß91] is fully implemented and integrated into the MOLTKE workbench [Ric91]. PATDEX/1 as a stand-alone prototype is already available since early 1989 [AKW+89, AMW+89]. The MOLTKE shell [AMR90], the GenRule system [AMW+90], [Alt91], and the MAKE system [Reh91] are fully implemented, too.

5 Conclusion

PATDEX has been designed for real world applications in the field of technical diagnosis and has its well-defined role within the MOLTKE workbench. PATDEX is fully integrated in an overall view of knowledge acquisition and integration as well as learning.

Having this view in mind PATDEX contributes to the current state of the art for case-based reasoning systems.

6 Acknowledgement

References

[Aam90]   Aamodt, A. A Computational Model of Knowledge-Intensive Learning and Problem Solving. In: [WBG+90], pp 1-20

[AFT+89]   Althoff K-D, Faupel B, Traphöner R et al. Knowledge Acquisition in the Domain of CNC Machining Centers: the MOLTKE Approach. In: *Proc. EKAW-89*, pp 180-195

[AKA91]   Aha, DW, Kibler, D, Albert, MK. Instance-Based Learning Algorithms. *Machine Learning* 1991, 6: 37-66

[AKW+89]   Althoff K-D, Kockskämper S, Weß S et al. Ein System zur fallbasierten Wissensverarbeitung in technischen Diagnosesituationen. In: Retti J & Leidlmeier K (eds), *5. Österreichische Artificial-Intelligence-Tagung*, Springer Berlin Heidelberg New York, 1989, pp 65-70

[Alt91]   Althoff K-D. *Eine fallbasierte Lernkomponente als integrierter Bestandteil der MOLTKE-Werkbank zur Diagnose technischer Systeme*. Dissertation, University of Kaiserslautern (forthcoming)

[AMR90]   Althoff K-D, Maurer F, Rehbold R. Multiple Knowledge Acquisition Strategies in MOLTKE. In: [WBG+90], pp 21-40

[AMW+89]   Althoff K-D, Maurer F, Weß, S et al. Adaptive Learning in the Domain of Technical Diagnosis. In: *Proc. Workshop on Adaptive Learning*, FAW Ulm, FRG, 1989

[AMW+90]   Althoff K-D, Maurer F, Weß, S et al. Die Lernkomponente der MOLTKE$_3$-Werkbank zur Diagnose technischer Systeme. In: [Mor91], pp 58-65

[Bar89]   Bareiss, R. *Exemplar-Based Knowledge Acquisition*. Academic Press London, 1989

[BBP88]   Bareiss R, Branting K, Porter B. The role of explanation in exemplar-based classification and learning. In: *Proc. Case-Based Reasoning AAAI-88*, 1988

[BW89]   Breuker JA, Wielinga, BJ. Model-Driven Knowledge Acquisition: Interpretation Models. In: Guida P, Tasso, G (eds) *Topics in the Design of Expert Systems*. North Holland Amsterdam, 1989, pp 265-296

[Car83]   Carbonell JG. Learning by analogy: formulating and generalizing plans from experience. In: Michalski RS Carbonell JG, Mitchell TM (eds) *Machine Learning*, vol 1, Tiogo Publishing Company Palo Alto, 1983

[Car86]   Carbonell JG. Derivational Analogy in Problem Solving and Knowledge Acquisition. In: Michalski RS Carbonell JG, Mitchell TM (eds) *Machine Learning*, vol 2, Morgan Kaufmann, Los Altos, 1986

[Ham89]   Hammond K. *Proc. of the 2nd DARPA Workshop on Case-Based Reasoning*. Morgan Kaufmann Palo Alto, 1989

[Kol83a]   Kolodner JL. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science* 1983, 7: 243-280

[Kol83b]   Kolodner JL. Reconstructive Memory: A Computer Model. *Cognitive Science* 1983, 7: 281-328

[Kol88]   Kolodner JL (ed). *Proc. of a DARPA Workshop on Case-Based Reasoning*. Morgan Kaufmann Palo Alto, 1988

[Kot88]   Koton P. Reasoning about evidence in causal explanations. In: *Proc. AAAI-88*, 1988, pp 256-261

[Mor87]   Morik K. Sloppy Modeling. In: [Mor89a], pp. 107-134, 1987

[Mor89a]   Morik K (ed). *Knowledge Representation and Organization in Machine Learning.* Springer Berlin Heidelberg New York, 1989

[Mor89b]   Morik K. Integration Issues in Knowledge Acquisition Systems. *SIGART Newsletter: Special Issue on Knowledge Acquisition* 1989, 108: 124-131

[Mor91]   Morik K (ed). Sonderheft Maschinelles Lernen. *KI* 1991, 5: 1

[Off89]   Offut D. SIZZLE: A case-based-reasoning system. Technical Report, Carnegie Mellon University, Pittsburgh, 1989

[Reh91]   Rehbold R. *Integration modellbasierten Wissens in technische Diagnostik-Expertensysteme. Dissertation*, University of Kaiserslautern (forthcoming), 1991

[Ric91]   Richter MM (ed). *Das △ⅢⅡⅠⅬⅥⅩⅭ◱ Buch.* Springer Berlin Heidelberg New York, 1991 (forthcoming)

[RKW89]   Rissland EL, Kolodner JL, Waltz, D. Case-Based Reasoning. In: [Ham89], pp 1-13

[RW91]   Richter MM, Weß S. Similarity, Uncertainty and Case-Based Reasoning in PATDEX. To appear in: *Festschrift for Woody Bledsoe*, Kluwer Academic Publishers, 1991

[RZ85]   Rumelhart DE, Zipser D. Feature discovery by competitive learning. In: *Cognitive Science* 1985, 9: 75-112

[Sla91]   Slade, S. Case-Based Reasoning: A Research Paradigm. *AI Magazine* Spring 1991

[Smi89]   Smith LB. From global similarities to kinds of similarities: the construction of dimensions in development. In: [VO89], pp. 146-178

[SW88]   Stanfill C & Waltz D. The memory based reasoning paradigm. In: [Kol88], pp. 414-424

[SZP90]   van Someren MW, Zheng LL, Post W. Cases, Models or Compiled Knowledge: a Comparative Analysis and Proposed Integration. In: [WBG+90], pp 339-355

[Tve77]   Tversky A. Features of Similarity. *Psychological Review* 1977, 84: 327-362

[VC89]   Veloso M & Carbonell JG. Learning Analogies by Analogy - The Closed Loop of Memory Organization and Problem Solving. In: [Ham89]

[VO89]   Vosniadou S & Ortony A. *Similarity and Analogical Reasoning.* Cambridge University Press, 1989

[WBG+90]   Wielinga BJ, Boose J, Gaines B et al. Current Trends in Knowledge Acquisition. IOS Press Amsterdam, 1990

[Weß91]   Weß S. PATDEX/ 2: ein System zum adaptiven, fallfokussierenden Lernen in technischen Diagnosesituationen. SEKI Working Paper SWP-91-01, University of Kaiserslautern, 1991