INRECA – A Seamless Integration of Induction and Case-Based Reasoning for Decision Support Tasks

Klaus-Dieter ALTHOFF[1], Stefan WESS[2], Ralph TRAPHÖNER[3]

[1] Centre for Learning Systems and Applications, Dept. of Computer Science, University of Kaiserslautern, PO Box 3049, 67653 Kaiserslautern, Germany; Email: althoff@informatik.uni-kl.de

[2] Inference GmbH, Lise-Meitner-Str. 3, 85716 Unterschleißheim; Email: swess@inference.co.uk

[3] tecInno GmbH, Sauerwiesen 2, 67661 Kaiserslautern, Germany; Email: trappi@tecmath.de

1. Introduction

We propose an integrated approach to decision support and diagnostic problems based on top-down induction of decision trees (TDIDT) and case-based reasoning (CBR). This approach has been implemented within the INRECA[1] Esprit project. While different ways of integration (Auriol, Manago et al., 1994) and evaluation (Althoff, Auriol et al., 1995; Althoff, 1995a; 1995b) have been carried out in the project, in this paper we want to focus on a deep kind of integration of TDIDT and CBR that we call a seamless one. The INRECA system has been influenced by both the KATE system (Manago, 1989) and the MOLTKE workbench (Althoff, 1992; Pfeifer & Richter, 1993), which includes a.o. the PATDEX CBR system (Althoff & Wess, 1991; Richter & Wess, 1991; Wess, 1993; Althoff, 1993). The main idea is to extend the PATDEX indexing mechanism such that it can process also numeric attributes. For this purpose a k-d tree (Friedman, Bentley & Finkel, 1977) has been introduced in INRECA as one basic structure for indexing and retrieval (Althoff, Wess et al. 1994). While this structure allowed the use of numeric attributes and local similarity measures, it had to be extended to preserve the main qualities of the PATDEX system (Althoff, Wess et al. 1994; Wess, Althoff & Derwand, 1994). Within this paper we will call this extended k-d tree an Inreca tree to avoid confusion with the original one.

Based on an Inreca tree both a decision tree and an extended k-d tree can be generated. This allows the use of induced knowledge within a k-d tree to improve the retrieval by the use of generalised knowledge. The more cases are available, the more general knowledge can be extracted and integrated in the Inreca tree for retrieval improvement.

In section 2 we will contrast k-d trees and decision trees, in section 3 an overview of the k-d tree improvements is given. Finally, section 4 summarises the use of the seamless integration of TDIDT and CBR in INRECA.

2. k-d Trees and Decision Trees

We describe the basic characteristics of these data structures and their associated building and search strategies. We summarise their respective advantages and disadvantages and motivate their proposed integration in the INRECA system.

2.1 k-d Trees

The basic idea of k-d tree building (Friedman, Bentley & Finkel, 1977; Wess, Althoff & Derwand, 1994) is to structure the search space based on its observed density (respectively on its classification goal) and to use this pre-computed structure for efficient case retrieval (respectively for efficient diagnosis). It works like a binary fixed indexing structure for the case retrieval. The current state-of-the-art for building such a tree consists in using an inter quartile distance (Koopmans, 1987). Within the k-d tree an incremental best-match search is used to find the K most similar cases (nearest neighbours) within a set of *n* cases with k specified indexing attributes.

The retrieval in a k-d tree is processed by the recursive application of two test procedures: Ball-Overlap-Bounds (BOB) and Ball-Within-Bounds (BWB) (Wess, Althoff & Derwand,

[1] Induction and Reasoning from Cases (Esprit P6322).

1994). While the search is going on, a priority list is defined which contains the ordered list of the current K most similar cases and their similarity to the query. This list is modified when a new case comes in the "top K". The recursive procedure (beginning with the root node) runs as follows:

- If the current node is a final one, the priority list is modified according to the similarity of the cases belonging to this node with the query.
- If the current node is not a final one, the procedure is iterated on the child node specified by the value of the query for the current question.
- Then a test is proceeded to look whether it is reasonable to inspect the other child nodes. This is done through the BOB test. If this test is false, the partition of the other child nodes cannot contain any K nearest neighbours with respect to the query. Therefore, they are not examined further. If this test is true, the procedure is iterated on these nodes.
- At the end of the procedure, the BWB test checks whether all the K nearest neighbours have been found, or not. If it is false, one has to expand the search to previous nodes in the tree.

BOB and BWB are relatively simple geometrical procedures (Wess, 1995). Unfortunately, they are limited with respect to the treatment of ordered symbolic and numeric data (Auriol, Manago et al., 1994).

2.2 Decision Trees

A decision tree is built upon a database of training cases. The partitioning procedure uses traditionally a hill-climbing search strategy and a preference criterion based on the information gain based on Shannon’s entropy (Shannon & Weaver, 1947), like the ID3 system (Quinlan, 1986). At each node in the decision tree, the information gain is evaluated for all the attributes which are relevant and the one is picked which yields the highest increase of the information gain measure (Manago, 1989). The four main steps of a decision tree builder are:

- Test if a node is a final one. A final node is labelled with the name of the most probable class in the node (the probability is evaluated through the relative frequency of each class in the node);
- Develop the space of relevant attributes at each node;
- Select the best partitioning attribute of a node;
- Split the cases in disjoint subsets according to the chosen attribute.

Aspects	Decision tree	k-d tree	INRECA tree
Similarity measure	not necessary	necessary, has to meet monotony requirements	multiple, class-specific similarity measures
nearest neighbour classification using similarity measures	not possible	possible	possible
Result	classification	K nearest neighbours	classification and K nearest neighbours
Backtracking in the tree	not necessary	necessary	possible
Cases during retrieval	not necessary	necessary	possible
Order of tests	fixed	query case has to be completely described w.r.t. indexing slots	arbitrary presentation of query including unknown values
Unordered value ranges	possible	impossible	possible
Ordered value ranges	not necessary	necessary	possible
Unknown attribute values	costly copying of cases	not possible	efficient extension of k-d trees for integrating unknown values

Table 1 - Contrasting k-d trees and decision trees for decision support and diagnosis tasks

2.3 Goals of the integration

Both mechanisms are limited in several ways (Manago, Althoff et al., 1993). We summarise the main points briefly:

- Decision trees: lack of flexibility and incrementallity, sensibility to errors and noise in the data;
- k-d trees: lack of knowledge generalisation, efficiency, and treatment of unknown and un-

ordered symbolic data (because of the BOB and BWB tests).

Table 1 aims at contrasting decision trees and k-d trees and at presenting the improvements gained with integration.

The above contrast calls especially for extending k-d trees with decision-tree-like efficient classification abilities, also for attributes with unordered value ranges. We present such an extension in the next section.

3. Improvements

We present an extension of the k-d tree building and search methods and describe the determination of optimal weight vectors based on the computation of global and local similarity measures. The determination of weights is then extended to cover also disjunctive concepts discovered in the Inreca tree structure. In addition, the achieved similarity measures are improved by the extraction of knowledge from decision trees. This enables the proposed seamless integration of induction and case-based reasoning. The resulting system is a completely integrated one. The same tree can be used simultaneously as a k-d tree in the case-based reasoning process for case indexing and retrieval, or as a decision tree in the induction process for case generalisation. The interactions between both approaches are greatly enhanced. As we will illustrate, by this enhancement we obtain a more flexible and acceptable tool that is easier to use and to maintain.

3.1 Extension of k-d tree building and retrieval methods

This extension aims at creating a single Inreca tree that can be used indifferently as a k-d tree for retrieval and as a decision tree for knowledge and rules extraction. Like in a decision tree and in a k-d tree, the branches of an Inreca tree represent constraints for certain attributes of the cases. Since we need to handle ordered and unordered value ranges as well as unknown attribute values, we introduce different kinds of branches. These branches are shown in figure 2.

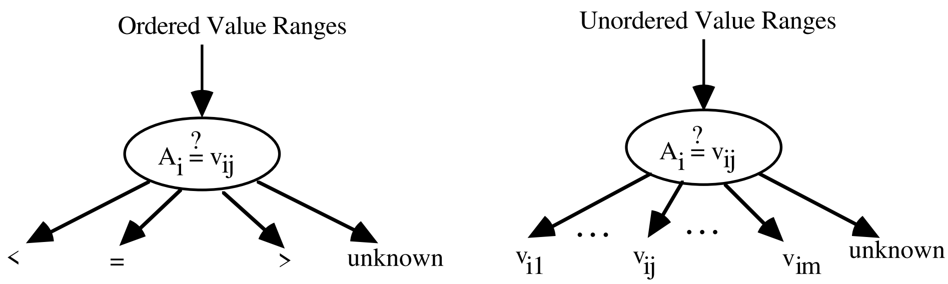

Fig. 2 - Branches of the Inreca tree for ordered and unordered value ranges

As stated previously, the current k-d trees cannot handle unknown and unordered symbolic data. However, let us assume that this kind of data arise often, whatever the attribute choice strategy[2]. Therefore, the retrieval strategy in the k-d tree procedure has to be modified, such that it can deal with cases that contain unordered or unknown attributes as it can be done in decision trees.

This is done through an extension of the BOB and BWB Boolean tests presented in section 2.1, that work primarily on binary trees, towards more general multi-dimensional trees. Beside these two tests the basic retrieval procedure remains unchanged.

2 We investigated several attribute choice strategies for building the INRECA tree [Auriol *et al.*, 1994]: (1) the interquartile distance used in the default *k*-d tree computes the distance between the first and the third quartile of a partition; (2) the maximum distance takes the greatest distance between two consecutive values of an attribute; (3) the average similarity measure estimates the dispersion of cases with respect to a given partition of the database; (4) the information gain measure computes the difference of entropy between a case base and its partition built from a specific attribute. (1) and (2) are restricted to ordered symbolic and numeric attributes, whereas (3) and (4) are more general. All of these measures are implemented in INRECA.

3.1.1. Extension of the BOB Test

The BOB test is executed in order to recognise whether a node may contain some candidates that are more similar to the current query than those that have already been found. Therefore, the geometrical bounds of the node are used to define a test point that is most similar to the current query but still lies within the geometrical bounds of the current node. If this test point is in the ball it means that the ball overlaps with the node and then there may be a candidate to the priority list in this node. The extension of the BOB test requires that the way how these test points are constructed also takes unordered value ranges and unknown values into account.

The definition of such a test point requires the assignment of a value to each attribute used in the case description. The value x_j for each attribute A_j must

a) lie within the geometrical bounds of the dimension defined by the attribute and the node to be investigated and it must be
b) most similar (with respect to the local similarity for this attribute) to the value q_j of the attribute in the query.

For attributes A_j with ordered value ranges, we determine the bounds ($l_j ... u_j$) defined by the attribute and the current node in the tree. The value x_j of the test point is as follows:

$$x_j = \begin{cases} unknown \text{ if the node is in the unknown branch of } A_j \\ * \quad \text{if the node is not in a branch of } A_j \\ q_j \quad \text{if } q_j \geq l_j \text{ and } q_j \leq u_j \\ l_j \quad \text{if } q_j < l_j \\ u_j \quad \text{if } q_j > u_j \end{cases} \tag{1}$$

In this definition, * denotes a new special value assumed in every value range. The local similarity between * and every other value in the value range is defined to be equal to 1. This assignment leads to the most pessimistic assumption about possible values for the attribute A_j in the cases that belong to the current node. If the current node is in the unknown branch of the attribute A_j, then x_j can also be assigned *unknown* because all cases belonging to that node have definitely an unknown value for this attribute.

For attributes with an unordered value range x_j is defined as follows:

$$x_j = \begin{cases} unknown \text{ if the node is in the unknown branch of } A_j \\ v_j \quad \text{if the node is in the } v_j \text{ branch of } A_j \\ * \quad \text{if the node is not in a branch of } A_j \end{cases} \tag{2}$$

In this case, we also have to make the most pessimistic assumption about the possible values of attributes that do not occur already in the path from the root to the current node.

3.1.2. Extension of the BWB Test

The extension of the BWB for correctly handling unknown values and unordered value ranges can be done similar to the extension of the BOB test (Wess, 1995). Unfortunately, this extension significantly increases the computational cost for executing this test. Early experiments demonstrated that, in the case of unordered symbolic attributes, the BWB test can perform poorly under certain circumstances. Let us recall that this test aims at terminating the retrieval process by deciding at a given node of the tree whether all nearest cases have been found, or not. This results in a reduced number of examined cases. Unfortunately, the gain obtained in the case of unordered symbolic attributes is illusory, because the test has to verify each dimension of the attribute space, what can be very high for this type of attribute. However, the execution of this test can nevertheless pay-off, if the access to the cases is very expensive, for example if the cases are stored in some external storage (Manago & Auriol, 1995).

3.2 Determination of Optimal Weight Vectors through Global and Local Similarity Measure Computation

A global similarity measure *SIM* between two cases *a* and *b* can be defined as a weighted sum of local similarity measures sim_j between the *p* attributes that make up the cases. The weights w_{jk} evaluate the relative importance of the attributes for each class.

$$SIM(a,b) = \sum_{j=1}^{p} w_{jk} \times sim_j(A_j(a), A_j(b)) \quad (3)$$

where $A_j(a)$ (resp. $A_j(b)$) stands for the value of case a (resp. b) for attribute A_j.

It is usual that the system user sets up these weights manually. Afterwards, a normalisation procedure is used such that the sum of the weights is one. This kind of job is error-prone and should be avoided in some real-world applications. Moreover, one usually evaluates only global weights for each attribute, without taking care about the class to which the cases belong. We propose that the system computes automatically the weights used in the similarity measures. For this purpose, we extend the average similarity measure to local intra- and inter-classes distances. The "class" of each case is given by its diagnosis.

3.2.1. Determination of Weights given the Classes

In a diagnostic problem, we know for each case its associated class. Therefore, we can compute the above similarities as follows:

- The local intra-class distance measures to which extent a case a is near to the cases b that belong to its own class with respect to attribute A_j are given by:

$$sim_{j,k}^{\cup} = \frac{1}{|M_k|^2} \sum_{a,b \in M_k} sim(A_j(a), A_j(b)) \quad (4)$$

where A_j(a) (resp. $A_j(b)$) stands for the value of case a (resp. b) for attribute A_j.

- The inter-class distance measures to which extent a case is far from the cases that belong to other classes with respect to attribute A_j are defined as follows:

$$sim_{j,k}^{\cap} = \frac{1}{|M_k||M - M_k|} \sum_{a \in M_k, b \in M - M_k} sim(A_j(a), A_j(b)) \quad (5)$$

The goal of setting optimal weights is to minimise the intra-class distance and to maximise the inter-class distance between cases. For this purpose, the weight relative to each class M_k and each attribute A_j is defined by:

$$w_{jk} = \frac{\max\{0, sim_{j,k}^{\cup} - sim_{j,k}^{\cap}\}}{\sum_{k=1}^{K} \max\{0, sim_{j,k}^{\cup} - sim_{j,k}^{\cap}\}} \quad (6)$$

The knowledge we have about the class of each case is transferred to the similarity measure through the definition of the weights. In the next section, we go a step further by using knowledge extracted from a decision tree.

3.2.2. Determination of Weights given a Decision Tree

The basic idea of this section is the same as the previous one. However, instead of directly using the partition of cases given by the class target (as given in the database), we consider an artificial partition built with a decision tree (i.e., the Inreca tree). A decision tree is a representation of a set of sufficient conditions for a case to fall in some class M_k (the *rules*). Each condition refers to a single attribute A_j contained in the cases. We now want to compile such a set of conditions for the attributes, required to fall in a leaf node of the Inreca tree, into the local similarity measure. Thereby, the user-defined similarity measure is modified on the ground of the induced general knowledge. The process is two-folds:

- Build a decision tree based on the database of cases;
- Extract the weights with respect to the subclasses discovered in the tree.

The main difference with the former method is that we can use disjunctive concepts (i.e., concepts described by a disjunction of conjunctions of features) to refine the classes' weights.

This approach requires that besides the class specific weights introduced in Sect. 3.2.1, class specific local similarity functions sim_j are allowed too. This extension is reflected in the following definition of the global similarity. In this definition, the first argument to SIM refers to the cases of the case bases (which are already classified) and the second argument refers to the query case.

$$SIM(a,b) = \sum_{j=1}^{p} w_{jk} \times sim_j'(A_j(a), A_j(b)) \qquad (7)$$

$$\text{where } sim_j'(x,y) = \begin{cases} 1, \text{ if } x \text{ and } y \text{ are covered by the same rule (cf. Fig. 4)} \\ sim_j(x,y) \text{ else (cf. Fig. 3)} \end{cases} \qquad (8)$$

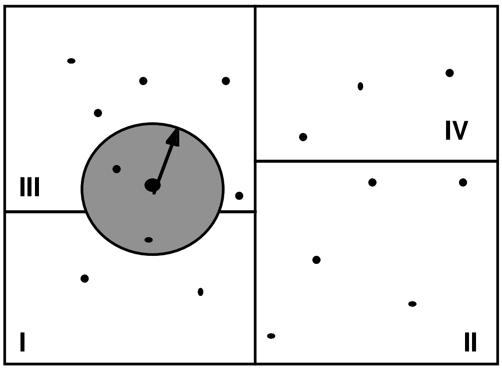

Fig. 3 - Classification based on cases

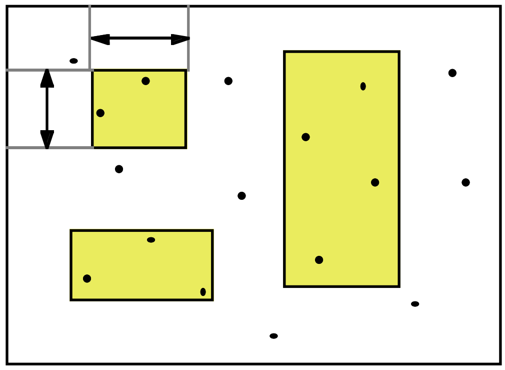

Fig. 4 - Classification based on inductively learned concepts

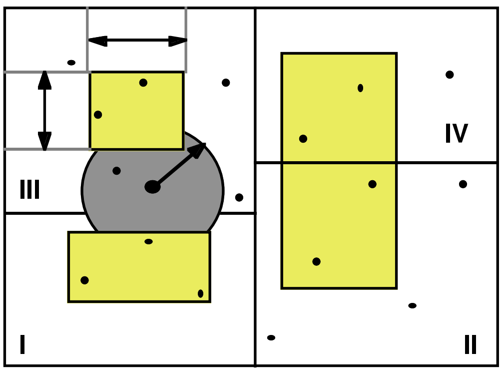

Fig. 5 - Combining case-based and inductive classification

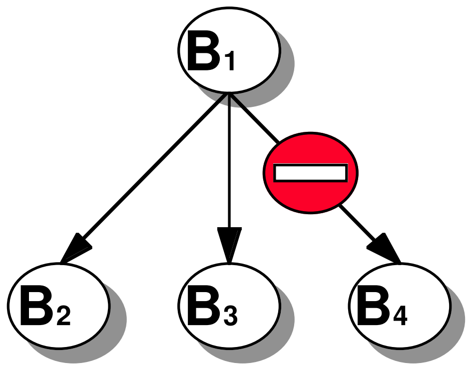

Fig. 6 - Reducing the amount of backtracking through inductively learned knowledge

4. Results

The Inreca tree meets all the requirements presented in table 1. The introduced improvements allow the implementation of a CBR system (Fig. 3) and an inductive system (Fig. 4). The integrated system combines these approaches in a seamless way (Fig. 5) that is completely transparent for the user. If a general concept has been learned that can answer a given query, this is the result of the retrieval process and no more backtracking has to be carried out (Fig. 6). The usual k-d tree search is applied only if the query lies not within the "region" of an induced concept. In the course of time, more and more generalised concepts can be induced based on an increasing number of cases. Thus, the INRECA system can "smoothly" evolve from a (more or less) pure CBR system to a system that (more or less) purely reasons based on inductively learned knowledge (Fig. 7).

Acknowledgement

Funding for INRECA has been provided by the Commission of the European Communities (ESPRIT contract P6322) to which the authors are greatly indebted. The partners of INRECA are AcknoSoft (prime contractor, France), tecInno (Germany), Irish Multimedia Systems (Ireland), the University of Kaiserslautern (Germany). The authors are indebted to Guido Derwand who implemented the core parts of the integrated INRECA system.

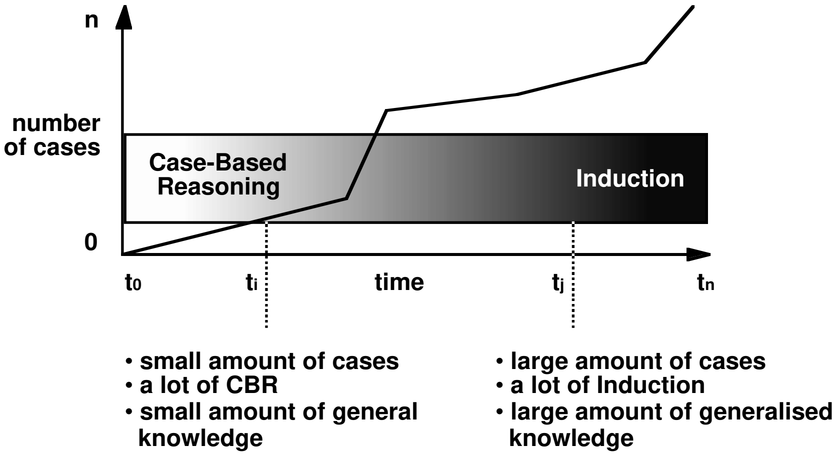

Fig. 7 - Evolution of the Seamlessly Integrated System

References

Althoff, K.-D. (1992). *Eine fallbasierte Lernkomponente als integrierter Bestandteil der MOLTKE-Werkbank zur Diagnose technischer Systeme*. Doctoral Dissertation, University of Kaiserslautern; also: Sankt Augustin, infix Verlag

Althoff, K.-D. (1993). Lernverfahren in MOLTKE In: *Pfeifer & Richter (1993)*, 173-200

Althoff, K.-D. (1995a). Evaluating Case-Based Reasoning Systems. *Proc. Unicom Workshop on CBR*

Althoff, K.-D. (1995b). Description, Analysis, and Development of Integrated Problem Solving and Learning Architectures for Decision Support and Diagnostic Reasoning. *forthcoming*

Althoff, K.-D., Auriol, E., Barletta, R. & Manago, M. (1995). *A Review of Industrial Case-Based Reasoning Tools*. Oxford, UK: AI Intelligence

Althoff, K.-D. & Wess, S. (1991). Case-Based Knowledge Acquisition, Learning, and Problem Solving for Diagnostic Real World Tasks. In: D. Smeed (ed.), *Proc. EKAW-91*

Althoff, K.-D., Wess, S., Manago, M., Bergmann, R., Maurer, F., Auriol, E., Conruyt, N., Traphöner, R., Bräuer, M. & Dittrich, S. (1994). Induction and Case-Based Reasoning for Classification Tasks. In: H.-H.. Bock, W. Lenski & M. M. Richter (eds.), *Proc. 17th Conference of the GfKl,* Springer Verlag, 3-16

Auriol, E., Manago, M., Wess, S., Althoff, K.-D. & Dittrich, S. (1994). Integrating Induction and Case-Based Reasoning: Methodological Approach and First Evaluations. In: M. Keane, J. P. Haton & M. Manago (eds.), *Proc. EWCBR-94*, 145-156

Friedman, J. H., Bentley, J. L. & Finkel, R. A. (1977). An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Software 3*, 209-226

Koopmans, L. H. (1987). *Introduction to Contemporary Statistical Methods*. Second Edition, Duxbury, Boston

Manago, M. (1989). Knowledge Intensive Induction. *Proc. ML-89*, Morgan Kaufmann

Manago, M., Althoff, K.-D., Auriol, E., Traphöner, R., Wess, S., Conruyt, N., Maurer, F. (1993). Induction and Reasoning from Cases. In: *Richter, Wess et al. (1993)*, 313-318

Manago, M. & Auriol, E. (1995). Integrating Induction and Case-Based Reasoning for Troubleshooting CFM-56 Aircraft Engines. In: B. Bartsch-Spörl, D. Janetzko & S. Wess (eds.), *Fallbasiertes Schließen - Grundlagen & Anwendungen*, XPS-95 Workshop, Zentrum für Lernende Systeme und Anwendungen, LSA-95-02, 73-80

Pfeifer, T. & Richter, M. M. (eds.) (1993). *Diagnose von technischen Systemen - Grundlagen, Methoden und Perspektiven der Fehlerdiagnose*. Deutscher Universitäts-Verlag.

Quinlan, R. (1986). Induction of Decision Trees. *Machine Learning 1,* 81-106

Richter, M. M. & Wess, S. (1991). Similarity, Uncertainty and Case-Based Reasoning in PATDEX. Automated Reasoning - Essays in Honor of Woody Bledsoe, Kluwer Academic Publishers.

Richter, M. M., Wess, S., Althoff, K.-D. & Maurer, F. (eds.) (1993). *Proc. 1st European Workshop on Case-Based Reasoning (EWCBR-93).* SEKI-REPORT SR-93-12, University of Kaiserslautern

Shannon & Weaver (1947). *The Mathematical Theory of Computation*. University of Illinois Press, Urbana

Wess, S. (1993). PATDEX - Inkrementelle und wissenbasierte Verbesserung von Ähnlichkeitsurteilen in der fallbasierten Diagnostik. In: F. Puppe & A. Günter (eds.), *Proc. 2. Deutsche Expertensystemtagung*

Wess, S. (1995). *Fallbasiertes Schließen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnose*. Doctoral Dissertation, University of Kaiserslautern (in press)

Wess, S., Althoff, K.-D. & Derwand, G. (1994). Using k-d Trees to Improve the Retrieval Step in CBR. In: S. Wess, K.-D. Althoff & M. M. Richter (eds.), *Topics in Case-Based Reasoning,* Springer, 167-181

Wess, S., Althoff, K.-D. & Richter, M. M. (eds.) (1994). *Topics in Case-Based Reasoning.* Springer-Verlag